50th International Colloquium on Automata, Languages, and Programming

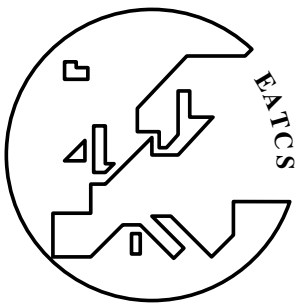
ICALP 2023, July 10–14, 2023, Paderborn, Germany

Edited by

Kousha Etessami

Uriel Feige

Gabriele Puppis



Editors

Kousha Etessami
University of Edinburgh, UK
kousha@inf.ed.ac.uk

Uriel Feige
Weizmann Institute of Science, Rehovot, Israel
uriel.feige@weizmann.ac.il

Gabriele Puppis
University of Udine, Italy
gabriele.puppis@uniud.it

ACM Classification 2012
Theory of computation

ISBN 978-3-95977-278-5

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at https://www.dagstuhl.de/dagpub/978-3-95977-278-5.

Publication date
July, 2023

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at https://portal.dnb.de.

Digital Object Identifier: 10.4230/LIPIcs.ICALP.2023.0

ISBN 978-3-95977-278-5 **ISSN 1868-8969** **https://www.dagstuhl.de/lipics**

LIPIcs – Leibniz International Proceedings in Informatics

LIPIcs is a series of high-quality conference proceedings across all fields in informatics. LIPIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

ISSN 1868-8969

https://www.dagstuhl.de/lipics

Contents

Track B: Automata, Logic, Semantics, and Theory of Programming

Preface

This volume contains the papers presented at the *50th EATCS International Conference on Automata, Languages and Programming (**ICALP 2023**)*, held in Paderborn, Germany, during July 10–14, 2023. ICALP is a series of annual conferences of the *European Association for Theoretical Computer Science (EATCS)*, which first took place in 1972.

This year, the ICALP program consisted of two tracks:

Track A: Algorithms, Complexity, and Games

Track B: Automata, Logic, Semantics, and Theory of Programming

In response to the call for papers, a total of 443 eligible, anonymous submissions were received: 346 for Track A and 97 for Track B. The committees decided to accept 132 papers for inclusion in the scientific program: 103 papers for Track A and 29 for Track B. The selection was made by the program committees based on originality, quality, and relevance to theoretical computer science. The quality of the submissions was very high, and many deserving papers could not be selected.

The EATCS sponsored awards for both a best paper and a best student paper in each of the two tracks, selected by the program committees.

The **best paper awards** were given to the following papers:

Track A: Tsun-Ming Cheung, Hamed Hatami, Pooya Hatami, and Kaave Hosseini. *Online Learning and Disambiguations of Partial Concept Classes.*

Track A: Miguel Bosch Calvo, Fabrizio Grandoni, and Afrouz Jabal Ameli. *A* 4/3 *Approximation for* 2*-Vertex-Connectivity.*

Track B: Marvin Künnemann, Filip Mazowiecki, Lia Schütze, Henry Sinclair-Banks, and Karol Węgrzycki. *Coverability in VASS Revisited: Improving Rackoff's Bound to Obtain Conditional Optimality.*

The **best student paper awards**, for papers that are solely authored by students, were given to the following papers:

Track A: Manuel Cáceres. *Minimum Chain Cover in Almost Linear Time.*

Track B: Ruiwen Dong. *The Identity Problem in* $\mathbb{Z} \wr \mathbb{Z}$ *is decidable.*

Apart from the contributed talks, ICALP 2023 included invited presentations by

- Anna Karlin, University of Washington, USA,
- Rasmus Kyng, ETH Zurich, Switzerland,
- Rupak Majumdar, Max Planck Institute for Software Systems, Germany,
- Thomas Vidick, California Institute of Technology, USA, and Weizmann Institute of Science, Israel,
- James Worrell, University of Oxford, UK.

This volume contains all the contributed papers presented at the conference, and an abstract or paper accompanying each of the invited talks by Anna Karlin, Rasmus Kyng, Rupak Majumdar, Thomas Vidick, and James Worrell.

For this special 50th anniversary of ICALP 2023, the conference program also included a special session with two invited talks by

- Kurt Mehlhorn, Max Planck Institute for Computer Science, Germany,
- Thomas A. Henzinger, Institute of Science and Technology, Austria.

50th International Colloquium on Automata, Languages, and Programming (ICALP 2023).
Editors: Kousha Etessami, Uriel Feige, and Gabriele Puppis
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Although they did not provide abstracts for the proceedings, we acknowledge their involvement and contribution.

The program of ICALP 2023 also included presentations of the EATCS Award 2023 to Amos Fiat (Tel Aviv University), the Presburger Award 2023 to Aaron Bernstein (Rutgers University) and to Thatchaphol Saranurak (University of Michigan), the Alonzo Church Award 2023 to the following group of papers:

- Ralf Jung, David Swasey, Filip Sieczkowski, Kasper Svendsen, Aaron Turon, Lars Birkedal, Derek Dreyer: "Iris: Monoids and Invariants as an Orthogonal Basis for Concurrent Reasoning". POPL 2015.
- Ralf Jung, Robbert Krebbers, Lars Birkedal, Derek Dreyer: "Higher-order ghost state". ICFP 2016.
- Robbert Krebbers, Ralf Jung, Aleš Bizjak, Jacques-Henri Jourdan, Derek Dreyer, Lars Birkedal: "The Essence of Higher-Order Concurrent Separation Logic". ESOP 2017.
- Ralf Jung, Robbert Krebbers, Jacques-Henri Jourdan, Aleš Bizjak, Lars Birkedal, Derek Dreyer: "Iris from the ground up: A modular foundation for higher-order concurrent separation logic". J. Funct. Program. 28 (2018).

The EATCS Distinguished Dissertation Award 2023 was awarded jointly to the following PhD dissertations:

- Kuikui Liu (University of Washington): "Spectral Independence: A New Tool to Analyze Markov Chains" (supervisor Shayan Oveis Gharan).
- Alex Lombardi (MIT, Department of Electrical Engineering and Computer Science): "Provable Instantiations of Correlation Intractability and the Fiat-Shamir Heuristic" (supervisor Vinod Vaikuntanathan).
- Lijie Chen (MIT, Department of Electrical Engineering and Computer Science): "Better Hardness via Algorithms, and New Forms of Hardness versus Randomness" (supervisor Ryan Williams).

There was also the announcement of the new EATCS Fellows for 2023, who are:

- Michael A. Bender (Stoney Brook University),
- Leslie Ann Goldberg (University of Oxford),
- Claire Mathieu (CNRS, IRIF, Université de Paris).

The following workshops were held as satellite events of ICALP 2023 on July 10, 2023:

- Combinatorial Reconfiguration
- Graph Width Parameters: from Structure to Algorithms (GWP 2023)
- Algorithmic Aspects of Temporal Graphs VI
- Adjoint Homomorphism Counting Workshop (ad hoc)
- Congestion Games
- Workshop On Reachability, Recurrences, and Loops '23 (WORReLL'23)
- Workshop on Recent Trends in Online Algorithms
- Quantum Computing with Qiskit, and why Classical Algorithms still matter!
- Algebraic Complexity Theory
- Computer Science for CONTINUOUS Data

We wish to thank all authors who submitted extended abstracts for consideration, the program committees for their scholarly effort, and all the reviewers who assisted the program committees in the evaluation process.

We are also grateful to the Conference General Chair, Sevag Gharibian, his colleagues from Paderborn University, and EATCS, for organizing ICALP 2023.

Finally, we would like to thank Anca Muscholl, the Chair of the ICALP Steering Committee, for her continuous support, Artur Czumaj, the president of EATCS, for his generous advice on the organization of the conference, as well as Michael Wagner, Michael Didas, and the entire editorial office of LIPIcs for their support in editing these proceedings.

July 2023

Kousha Etessami
Uriel Feige
Gabriele Puppis

Organization

Program Committees

Track A

Amir Abboud	Weizmann Institute, Israel
Mikkel Abrahamsen	University of Copenhagen, Denmark
Sepehr Assadi	Rutgers University, USA
Aditya Bhaskara	University of Utah, USA
Arnab Bhattacharyya	National University of Singapore, Singapore
Greg Bodwin	University of Michigan, USA
Karl Bringmann	Saarland University, Germany
Clément Canonne	University of Sydney, Australia
Vincent Cohen Addad	Google Research, Zurich
Amin Coja Oghlan	TU Dortmund, Germany
Michael Dinitz	Johns Hopkins University, USA
Uriel Feige (Chair)	Weizmann Institute, Israel
Moran Feldman	University of Haifa, Israel
Sebastian Forster	University of Salzburg, Austria
Sumegha Garg	Stanford University, USA
Parikshit Gopalan	Apple, USA
Karthik C.S.	Rutgers University, USA
Yin Tat Lee	University of Washington, USA
Stefano Leonardi	Sapienza Universita di Roma, Italy
Sepideh Mahabadi	MSR Redmond, USA
Giulio Malavolta	Max Planck Institute for Security and Privacy, Germany
Jesper Nederlof	Utrecht University, Netherlands
Vianney Perchet	Ensae and Criteo AI Lab, France
Will Perkins	Georgia Institute of Technology, USA
Marcin Pilipczuk	University of Warsaw, Poland, and IT University of Copenhagen, Denmark
Aviad Rubinstein	Stanford University, USA
Barna Saha	University of California San Diego, USA
Rahul Santhanam	University of Oxford, UK
Thatchaphol Saranurak	University of Michigan, USA
Igor Shinkar	Simon Fraser University, Canada
Mohit Singh	Georgia Institute of Technology, USA
David Steurer	ETH Zurich, Switzerland
Ola Svensson	EPFL, Switzerland
Inbal Talgam-Cohen	Technion, Israel
Kavitha Telikepalli	Tata Institute of Fundamental Research, Mumbai, India
Vera Traub	University of Bonn, Germany
Salil Vadhan	Harvard University, USA
David Wajc	Google Research, USA
Henry Yuen	Columbia University, USA
Meirav Zehavi	Ben-Gurion University, Israel

Track B

Shaull Almagor	Technion, Israel
Albert Atserias	Universitat Politecnica de Catalunya, Barcelona, Spain
Christel Baier	TU Dresden, Germany
Véronique Bruyère	University of Mons, Belgium
Thomas Colcombet	IRIF / CNRS / Université Paris Cité, France
Andrei Bulatov	Simon Fraser University, Canada
Wojciech Czerwiński	University of Warsaw, Poland
Kousha Etessami (Chair)	University of Edinburgh, UK
John Fearnley	University of Liverpool, UK
Dana Fisman	Ben-Gurion University, Israel
Rob van Glabbeek	University of New South Wales, Australia
Chris Heunen	University of Edinburgh, UK
Justin Hsu	Cornell University, USA
Stefan Kiefer	University of Oxford, UK
Kohei Kishida	University of Illinois Urbana-Champaign, USA
Jan Kretinsky	Technical University of Munich, Germany
Karoliina Lehtinen	CNRS, Université Aix Marseille et Université de Toulon, LIS, France
Anthony Widjaja Lin	TU Kaiserslautern & MPI-SWS, Germany
Wim Martens	University of Bayreuth, Germany
Joanna Ochremiak	CNRS, University of Bordeaux, France and University of Warsaw, Poland
Daniela Petrisan	Université Paris Cité, IRIF, France
Sam Staton	University of Oxford, UK
Ashutosh Trivedi	University of Colorado - Boulder, USA
Takeshi Tsukada	Chiba University, Japan
Mahesh Viswanathan	University of Illinois Urbana-Champaign, USA

Organizing Committee

Sevag Gharibian, *chair*
Johannes Blömer
Friedhelm Meyer auf der Heide
Christian Scheideler
Ulf-Peter Schroeder

Steering Committee

Nikhil Bansal	University of Michigan, US
Artur Czumaj	Warwick University, UK
Javier Esparza	TUM Munich, Germany
Simon Gay	University of Glasgow, UK
Leslie Ann Goldberg	Oxford University, UK
Thore Husfeldt (co-chair)	Lund University, Sweden and IT University of Copenhagen, Denmark
Giuseppe Italiano	Luiss University, Italy
Emanuela Merelli	University of Camerino, Italy
Anca Muscholl (chair)	Bordeaux University, France
Yuval Rabani	Hebrew University, Israel
Paul Spirakis	University of Liverpool, UK and University of Patras, Greece
James Worrell	University of Oxford, UK

Financial Sponsors

GOLD SPONSOR

SILVER SPONSOR

BRONZE SPONSOR

External Reviewers

Upon their request, a small number of people who served as external reviewers do not appear in this list.

Hugo Aaronson
Jayadev Acharya
Shyan Akmal
Dylan Altschuler
Aditya Anand
Alexandr Andoni
Kazuyuki Asada
Nathalie Aubrun
Tanvi Bajpai
Nikhil Bansal
Khashayar Barooti
Ulrich Bauer
Paul Bell
Ziyad Benomar
Benjamin Bergougnoux
Sebastian Berndt
Siddharth Bhandari
Sujoy Bhore
Andreas Björklund
Jannis Blauth
Achim Blumensath
Benedikt Bollig
Joshua Brakensiek
Samuel Braunfeld
Jop Briet
Niv Buchbinder
Jean Cardinal
Ruoxu Cen
Scott Aaronson
Akanksha Agrawal
Hannaneh Akrami
Kazuyuki Amano
Konrad Anand
Marcelo Arenas
Vikrant Ashvinkumar
Per Austrin
Ainesh Bakshi
Leonid Barenboim
James Bartusek
Soheil Behnezhad
Rémy Belmonte
Suman Kalyan Bera
Pascal Bergstraesser
Aaron Bernstein
Sayan Bhattacharya
Georgios Birmpas
Eric Blais
Joakim Blikstad
Hans L. Bodlaender
Édouard Bonnet
Pedro Branco
Anton Braverman
Joshua Brody
Jaroslaw Byrka
Alejandro Cassis
Keren Censor-Hillel
Behzad Abdolmaleki
Michal Ajdarow
Gorjan Alagic
Dong An
Nima Anari
Elena Arseneva
Hilal Asi
Muqsit Azeem
Etienne Bamas
Corentin Barloy
Dorian Baudry
Michael Bekos
Aleksandrs Belovs
Ioana Bercea
Pascal Bergsträßer
Nayel Bettache
Rishiraj Bhattacharyya
Csaba Biro
Jaroslaw Blasiok
Ivan Bliznets
Levente Bodnár
Christian Borgs
Simina Branzei
Léonard Brice
Adam Brown
Benjamin Böhm
Federica Cecchetto
Arghya Chakraborty

Parinya Chalermsook
T-H. Hubert Chan
Timothy Chan
Timothy M. Chan
Karthekeyan Chandrasekaran
Yi-Jun Chang
Bernadette Charron-Bost
Arnab Chatterjee
Arkadev Chattopadhyay
Ho-Lin Chen
Zongchen Chen
Maria Cherifa
Vincent Cheval
Leroy Chew
Nai-Hui Chia
Andrew Childs
Suryajith Chillara
Rajesh Chitnis
Eden Chlamtac
Davin Choo
Keerti Choudhary
Anirban Chowdhury
Valerio Cini
Lorenzo Clemente
Richard Cleve
Christian Coester
Gil Cohen
Ilan Cohen
Alessio Conte
Arjan Cornelissen
Martin Costa
Matthew Coudron
Stefano Crespi Reghizzi
Andrés Cristi
Maxime Crochemore
Lorenzo Croissant
Emilio Cruciani
David Cui
Radu Curticapean
Artur Czumaj
Yuval Dagan
Anatole Dahan
Debarati Das
Ewan Davies
Sami Davies
Nabarun Deka
Mahsa Derakhshan
Palash Dey
Yann Disser
Taylor Dohmen
Jinshuo Dong
Florian Dorfhuber
Anne Driemel
Marina Drygala
Guillaume Ducoffe
Aditi Dudeja
Jesko Dujmovic
Talya Eden
Charilaos Efthymiou
Marco Eilers
Hafedh El Ferchichi
Marek Elias
Ehsan Emamjomeh-Zadeh
Alina Ene
David Eppstein
Leah Epstein
Thomas Erlebach
Meryem Essaidi
Alexandros Evangelidis
Tomer Ezra
Amin Falah
Ashkan Norouzi Fard
Vitaly Feldman
Andreas Emil Feldmann
Guillaume Fertin
Hendrik Fichtenberger
Côme Fiegel
Yuval Filmus
Nick Fischer
Bailey Flanigan
Henry Fleischmann
Noah Fleming
Jacob Focke
Fedor Fomin
Dimitris Fotakis
Kyle Fox
Cody Freitag
Dominik D. Freydenberger
Tobias Friedrich
Zachary Friggstad
Honghao Fu
Federico Fusco
Daniel Gabric
Jakub Gajarský
Rishikesh Gajjala
Andreas Galanis
Moses Ganardi
Ankit Garg
Mohit Garg
Felipe Garrido-Lucero
Peter Gartland
Surya Teja Gavva
Pawel Gawrychowski
Loukas Georgiadis
Timo Gervens
Abheek Ghosh
Suprovat Ghoshal
Rohan Ghuge
George Giakkoupis
Amin Shiraz Gilani
Jacob Gilbert
Niv Gilboa
Uma Girish
Marc Glisse
Aarushi Goel
Shay Golan
Emmanuel Goldberg
Petr Golovach
Louis Golowich
Timothy Gomez
Rica Gonen
Junqing Gong
Sivakanth Gopi
Barun Gorain
Gramoz Goranci
Spencer Gordon
Mayank Goswami
Themistoklis Gouleakis
Mohak Goyal
Fabrizio Grandoni
Marta Grobelna
Carla Groenland
Martin Grohe
Kush Grover
Shibashis Guha
He Guo
Heng Guo
Meghal Gupta
Tom Gur
Jens Oliver Gutsfeld
Roland Guttenberg
Thorsten Götte
Michel Habib
Amar Hadzihasanovic

Maximilian Hahn-Klimroth
Yassine Hamoudi
Qizheng He
Dorothee Henke
D. Ellis Hershkowitz
Alexandros Hollender
Gary Hoppenworth
Shang-En Huang
Tanmay Inamdar
Noa Izsak
Meena Jagadeesan
Tomáš Jakl
David N. Jansen
Matthew Jenssen
Ce Jin
Chris Jones
Dominik Kaaser
Naoyuki Kamiyama
Panagiotis Kanellopoulos
Adam Karczmarz
Alexander Kelley
George Kenison
Kamyar Khodamoradi
Robbie King
Peter Kiss
Kim-Manuel Klein
Yusuke Kobayashi
Shimon Kogan
Leszek Kolodziejczyk
Ama Koranteng
Guy Kortsarz
Kishore Kothapalli
Dariusz Kowalski
Lena Krieg
Piotr Krysta
Janardhan Kulkarni
Mrinal Kumar
Alexander Kurz
William Kuszmaul
Arnaud Labourel
Alexander Lam
John Lapinskas
Thomas Lavastida
Van Bang Le
Engel Lefaucheux
Roie Levin
Ray Li
Yi Li

Shai Halevi
Tesshu Hanaka
Klaus Heeger
Léo Henry
John M. Hitchcock
Jacob Holm
Ross Horne
Kasper Høgh
Stavros Ioannidis
Hugo Jacob
Palak Jain
Arun Jambulapati
Klaus Jansen
Mark Jerrum
Zhengzhong Jin
Vincent Jugé
Praneeth Kacham
Frank Kammer
Mamadou Moustapha Kanté
Juhani Karhumaki
Esther Kelman
Thomas Kesselheim
Sandra Kiefer
Valerie King
Susumu Kiyoshima
Nathan Klein
Caleb Koch
Pascal Koiran
Christian Komusiewicz
Tuukka Korhonen
Peter Kostolányi
Michal Koucky
Laszlo Kozma
Ravishankar Krishnaswamy
Ariel Kulik
Pooja Kulkarni
Nikhil Kumar
Eyal Kushilevitz
David Kutner
Bundit Laekhanukit
Michael Lampis
Chris Laskowski
Hoang-Oanh Le
Chin Ho Lee
Johannes Lengler
Jason Li
Shi Li
Yinan Li

Thekla Hamm
Nathaniel Harms
Brett Hemenway
Monika Henzinger
Piotr Hofman
Max Hopkins
Lingxiao Huang
Jacob Imola
Yuval Ishai
Riko Jacob
Pallavi Jain
Bart M.P. Jansen
Rajesh Jayaram
Xinrui Jia
Philips George John
Raphaël Jungers
Iden Kalemaj
Anthimos-Vardis Kandiros
Mong-Jen Kao
Matthew Katz
David Kempe
Arindam Khan
Eun Jung Kim
Sándor Kisfaludi-Bak
Pieter Kleer
Dušan Knop
Tomasz Kociumaka
Benedikt Kolbe
Tsvi Kopelowitz
Andre Kornell
Egor V. Kostylev
Matt Kovacs-Deak
Jan Kratochvil
Paul Krogmeier
Alexander Kulikov
Akash Kumar
Rajendra Kumar
Dietrich Kuske
Marvin Künnemann
Abhiruk Lahiri
Frédéric Lang
Silvio Lattanzi
Hung Le
Euiwoong Lee
Shoham Letzter
Minming Li
Wenzheng Li
Yingying Li

Lyuben Lichev
Moritz Lichter
Thomas Lidbetter
Bingkai Lin
Daogao Liu
Jiahui Liu
Jin-Peng Liu
Kuikui Liu
Mike Liu
Mingmou Liu
Quanquan Liu
Yang Liu
Leo Lobski
William Lochet
Sylvain Lombardy
Dimitrios Los
Florian Luca
Josep Lumbreras
Xin Lyu
Christof Löding
Weiyun Ma
Marten Maack
Andreas Maggiori
James C.A. Main
Monosij Maitra
Mikhail Makarov
Frederik Mallmann-Trenn
Malhar Managoli
Nikhil Mande
Peter Manohar
Bodo Manthey
Alberto Marchetti-Spaccamela
Jay Mardia
Andrea Marino
Oliver Markgraf
Dan Marsden
Simon Martiel
Tomáš Masařík
Umang Mathur
Simon Mauras
Theo McKenzie
Audra McMillan
Nicole Megow
Uri Meir
Raghu Meka
Alexander Melnikov
Darya Melnyk
Nadav Merlis
Marcus Michelen
Martin Milanič
Jason Milionis
Antoine Miné
Majid Mirzanezhad
Pranabendu Misra
Slobodan Mitrović
Valia Mitsou
Parth Mittal
Divyarthi Mohan
Sidhanth Mohanty
Stefanie Mohr
Mathieu Molina
Hendrik Molter
Benjamin Monmege
Yoàv Montacute
Shay Moran
Benjamin Moseley
Tamer Mour
Ramin Mousavi
Shay Mozes
Loay Mualem
Anish Mukherjee
Tamalika Mukherjee
Sagnik Mukhopadhyay
Adithya Murali
Vishnu Murali
Tobias Mömke
Noela Müller
Viswanath Nagarajan
Tassio Naia
Vasileios Nakos
Seffi Naor
Meghana Nasre
Sivaramakrishnan Natarajan
Yasamin Nazari
Jelani Nelson
Daniel Neuen
Stefan Neumann
Alantha Newman
Ilan Newman
Dung Nguyen
Matthias Niewerth
Reino Niskanen
André Nusser
Zeev Nutov
Martin Nägele
Corentin Odic
Timm Oertel
Pierre Ohlmann
Yoshio Okamoto
Rafael Oliveira
Dan Olteanu
Lukáš Ondráček
Yota Otachi
Benedikt Pago
Soumyabrata Pal
Katarzyna Paluch
Ioannis Panageas
Debmalya Panigrahi
Fahad Panolan
Charles Paperman
Dmitry Paramonov
Louis Parlant
Nikos Parotsidis
Salman Parsa
Merav Parter
Paweł Parys
Viresh Patel
Subhasree Patro
Michał Pawłowski
Anurudh Peduri
Pan Peng
Guillermo Perez
Mateo Perez
Asaf Petruschka
Seth Pettie
Michał Pilipczuk
Vladimir Podolskii
Tristan Pollner
Aleksandr Popov
Danny Bøgsted Poulsen
Lionel Pournin
Thomas Powell
Nicola Prezza
Eric Price
Siddharth Pritam
Maximilian Probst
Maximilian Prokop
Kirk Pruhs
Ioannis Psarros
Manish Purohit
Edward Pyne
Pengyu Qian
Youming Qiao
Kent Quanrud

Yuval Rabani
Ahmadreza Rahimi
Ninad Rajgopal
Cyrus Rashtchian
Abhishek Rathod
Felix Reidl
Nicolas Resch
Kilian Risse
Mohammad Roghani
Maurice Rolvien
Marc Roth
Davide Rucci
Paweł Rzążewski
Sushant Sachdeva
Ken Sakayori
Ignasi Sau
Saket Saurabh
Christian Scheideler
Martin Schirneck
Steffen Schuldenzucker
Pascal Schweitzer
Masood Seddighin
Flore Sentenac
Hadas Shachnai
Liren Shan
Suhail Sherif
Xinkai Shu
Sahil Singla
Bart Smeulders
Shay Solomon
Akshayaram Srinivasan
Frank Staals
Tatiana Starikovskaya
Miltiadis Stouras
Elina Sudit
Céline Swennenhuis
Zihan Tan
Sébastien Tavenas
Théophile Thiery
Jacobo Torán
Ohad Trabelsi
Andrea Turrini
Jakub Tětek
Seeun William Umboh
Arsen Vasilyan
Rahul Vaze
Santosh Vempala
Radu Vintan

Tomasz Radzik
Saladi Rahul
Michael Rao
Mikhail Raskin
Gaurav Rattan
Rebecca Reiffenhäuser
David Richerby
Cristian Riveros
Daniel Rogozin
Gregory Rosenthal
Thomas Rothvoss
Matteo Russo
Heiko Röglin
Ron Safier
Sukolsak Sakshuwong
Thomas Sauerwald
Saurabh Sawlani
Sven Schewe
Niklas Schlomberg
Mark Schultz
Chris Schwiegelshohn
Pavel Semukhin
C. Seshadhri
Vihan Shah
Roohani Sharma
Abhishek Shetty
Sebastian Siebertz
Makrand Sinha
Siani Smith
Tasuku Soma
Srikanth Srinivasan
Georgios Stamoulis
Rafał Stefański
Hsin-Hao Su
Scott Summers
Prafullkumar Tale
Ewin Tang
Soeren Terziadis
Mads Toftrup
Patrick Totzke
Thorben Tröbst
Iddo Tzameret
Jonathan Ullman
Jalaj Upadhyay
Sergei Vassilvitskii
Nate Veldt
Moritz Venzin
Lukas Vogl

Prasad Raghavendra
Justin Raizes
Christoforos Raptopoulos
Nidhi Rathi
Luca Reggio
Hanlin Ren
Sabine Rieder
David Roberson
Lars Rohwedder
Neil Ross
Sanjukta Roy
Andrew Ryzhikov
Sagi Saadon
Prakash Saivasan
Sai Sandeep
David Saulpic
Olga Scheftelowitsch
Aaron Schild
Sylvain Schmitz
Roy Schwartz
Adam Sealfon
Sayantan Sen
Martin P. Seybold
Omer Shahar
Adrian She
Mahsa Shirmohammadi
Harsimran Singh
Rene Sitters
Marek Sokołowski
Manuel Sorge
Piyush Srivastava
Kevin Stangl
Uri Stemmer
Sathyawageeswar Subramanian
Janani Sundaresan
Navid Talebanfard
Jakub Tarnawski
Guillaume Theyssier
Szymon Toruńczyk
Noam Touitou
Ta-Wei Tu
Ilay Tzarfati
Chris Umans
Ali Vakilian
Yadu Vasudev
Santhoshini Velusamy
N. V. Vinodchandran
Mikhail Volkov

Tjark Vredeveld
Anil Kumar Vullikanti
Thuy Duong Vuong
Jens Vygen
László Végh
Magnus Wahlström
Erik Waingarten
Hendrik Waldner
Tomasz Walen
Nathan Wallheimer
Chen Wang
Haitao Wang
Quanlong Wang
Justin Ward
Julian Wargalla
Thomas Watson
Chana Weil-Kennedy
Oren Weimann
Maximilian Weininger
Jennifer Welch
Philip Wellnitz
Sebastian Wiederrecht
Andreas Wiese
Lisa Wilhelmi
Sarah Winkler
Sarah Winter
Michal Wlodarczyk
Alexander Wolff
Damien Woods
James Worrell
John Wright
David Wu
Yuchen Wu
Christian Wulff-Nilsen
Karol Węgrzycki
Ning Xie
Yinzhan Xu
Zhou Xu
Taisuke Yasuda
Christopher Ye
Di-De Yen
Asaf Yeshurun
Yuichi Yoshida
Haifeng Yu
Jing Yu
Qian Yu
Weiqiang Yuan
Peter Yuen
Igor Zablotchi
Carol Zamfirescu
Rico Zenklusen
Wei Zhan
Chihao Zhang
Minjian Zhang
Rachel Zhang
Tianyi Zhang
Xinzhi Zhang
Yuhao Zhang
Da Wei Zheng
Hang Zhou
Rudy Zhou
Samson Zhou
Zixin Zhou
Marius Zimand
Martin Zimmermann
Sebastian Zur
Goran Zuzic
Paloma de Lima
Tijn de Vos
Erik Jan van Leeuwen
Jan van den Brand
Ivor van der Hoog
Tom van der Zanden
Stanislav Živný

List of Authors

Anders Aamand (6)
MIT, Cambridge, MA, USA

Bader Abu Radi (109)
School of Computer Science and Engineering, Hebrew University, Jerusalem, Israel

Peyman Afshani (7)
Aarhus University, Denmark

Ishan Agarwal (8)
New York University, NY, USA

Daniel Agassy (9)
Tel Aviv University, Israel

Amirreza Akbari (10)
Aalto University, Espoo, Finland

Shyan Akmal (11)
MIT EECS and CSAIL, Cambridge, MA, USA

Prashanth Amireddy (12)
Harvard University, Cambridge, MA, USA

Yossi Azar (13)
School of Computer Science, Tel-Aviv University, Israel

Amir Azarmehr (14)
Northeastern University, Boston, MA, USA

Ishan Bansal (15)
Operations Research and Information Engineering, Cornell University, Ithaca, NY, USA

Siddharth Barman (16)
Indian Institute of Science, Bangalore, India

Aniket Basu Roy (7)
Aarhus University, Denmark

Pascal Baumann (3, 110)
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Paul Beame (17)
Computer Science & Engineering, University of Washington, Seattle, WA, USA

Bartosz Bednarczyk (111)
Computational Logic Group, Technische Universität Dresden, Germany; Institute of Computer Science, University of Wrocław, Poland

Soheil Behnezhad (14)
Northeastern University, Boston, MA, USA

Michael Benedikt (112)
Department of Computer Science, University of Oxford, UK

Petra Berenbrink (18)
Universität Hamburg, Germany

Benjamin Aram Berendsohn (19)
Institut für Informatik, Freie Universität Berlin, Germany

Thiago Bergamaschi (20)
Department of Computer Science, University of California, Berkeley, CA, USA

Christoph Berkholz (113)
Technische Universität Ilmenau, Germany

Rajarshi Bhattacharjee (21)
Manning College of Information and Computer Sciences, University of Massachusetts, Amherst, MA, USA

Sudatta Bhattacharya (22)
Computer Science Institute of Charles University, Prague, Czech Republic

Therese Biedl (23)
David R. Cheriton School of Computer Science, University of Waterloo, Canada

Davide Bilò (24)
Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Italy

Fabian Birkmann (114)
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Hadley Black (25)
Department of Computer Science, University of California at Los Angeles, CA, USA

Guy E. Blelloch (26)
Carnegie Mellon University, Pittsburgh, PA, USA

Michael Blondin (115)
Department of Computer Science, Université de Sherbrooke, Canada

Manuel Bodirsky (116)
Institut für Algebra, TU Dresden, Germany

Hans L. Bodlaender (27)
Department of Information and Computing Sciences, Utrecht University, The Netherlands

50th International Colloquium on Automata, Languages, and Programming (ICALP 2023).
Editors: Kousha Etessami, Uriel Feige, and Gabriele Puppis
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Andrej Bogdanov (28)
University of Ottawa, Canada

Mikołaj Bojańczyk (117)
Institute of Informatics, University of Warsaw, Poland

Miguel Bosch-Calvo (29)
IDSIA, USI-SUPSI, Lugano, Switzerland

Patricia Bouyer (118)
Université Paris-Saclay, CNRS, ENS Paris-Saclay, Laboratoire Méthodes Formelles, 91190, Gif-sur-Yvette, France

Samuel Braunfeld (119)
Computer Science Institute of Charles University (IUUK), Prague, Czech Republic

Vladimir Braverman (30)
Rice University, Houston, TX, USA

Manuel Cáceres (31)
Department of Computer Science, University of Helsinki, Finland

Chris Cade (32)
QuSoft and University of Amsterdam (UvA), The Netherlands

Jin-Yi Cai (33)
Department of Computer Sciences, University of Wisconsin-Madison, WI, USA

Titouan Carette (120)
Centre for Quantum Computer Science, Faculty of Computing, University of Latvia, Riga, Latvia

Olivier Carton (121)
Université Paris Cité, CNRS, IRIF, F-75013, France; Institut Universitaire de France, Paris, France

Antonio Casares (122)
LaBRI, Université de Bordeaux, France

Diptarka Chakraborty (123)
National University of Singapore, Singapore

Sourav Chakraborty (123)
Indian Statistical Institute, Kolkata, India

Timothy M. Chan (34)
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA

Yi-Jun Chang (35)
National University of Singapore, Singapore

Chandra Chekuri (36)
Department of Computer Science, University of Illinois, Urbana-Champaign, Urbana, IL, USA

Lijie Chen (39)
Miller Institute for Basic Research in Science at University of California at Berkeley, CA, USA

Yanlin Chen (38)
QuSoft and CWI, Amsterdam, The Netherlands

Yu Chen (37)
EPFL, Lausanne, Switzerland

Kuan Cheng (41)
Department of Computer Science, Peking University, Beijing, China

Pingan Cheng (7)
Aarhus University, Denmark

Siu-Wing Cheng (40)
Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China

Joseph Cheriyan (15)
Department of Combinatorics and Optimization, University of Waterloo, Canada

Tsun-Ming Cheung (42)
McGill University, Montreal, Canada

Dmitry Chistikov (112)
Centre for Discrete Mathematics and its Applications (DIMAP) & Department of Computer Science, University of Warwick, Coventry, UK

Keerti Choudhary (24)
Department of Computer Science and Engineering, Indian Institute of Technology Delhi, India

Omer Cohen Sidon (44)
Tel Aviv University, Israel

Ilan Reuven Cohen (43)
Faculty of Engineering, Bar-Ilan University, Ramat Gan, Israel

Sarel Cohen (24)
School of Computer Science, Tel-Aviv-Yaffo Academic College, Israel

Richard Cole (8)
New York University, NY, USA

Spencer Compton (45)
Stanford University, CA, USA

Sam Coy (46)
University of Warwick, Coventry, UK

Artur Czumaj (46)
University of Warwick, Coventry, UK

Peter Davies (46)
Durham University, UK

Anuj Dawar (119)
Department of Computer Science and Technology, University of Cambridge, UK

Ronald de Wolf (38)
QuSoft and CWI, Amsterdam, The Netherlands; University of Amsterdam, The Netherlands

Gregory Dexter (21)
Department of Computer Science, Purdue University, West Lafayette, IN, USA

Yann Disser (47)
TU Darmstadt, Germany

Magdalen Dobson (26)
Carnegie Mellon University, Pittsburgh, PA, USA

Ruiwen Dong (124)
Department of Computer Science, University of Oxford, UK

Dani Dorfman (9)
Tel Aviv University, Israel

Andrzej Dorobisz (48)
Theoretical Computer Science Department, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland

Ilan Doron-Arad (49)
Computer Science Department, Technion, Haifa, Israel

Gaëtan Douéneau-Tabot (121)
Université Paris Cité, CNRS, IRIF, F-75013, France; Direction générale de l'armement – Ingénierie des projets, Paris, France

Jan Dreier (125)
TU Wien, Austria

Lukas Drexler (50)
Heinrich-Heine Universität Düsseldorf, Germany

Petros Drineas (21)
Department of Computer Science, Purdue University, West Lafayette, IN, USA

Shaddin Dughmi (51)
University of Southern California, Los Angeles, CA, USA

Talya Eden (52)
Bar Ilan University, Ramat Gan, IL

Klim Efremenko (53)
Ben-Gurion University, Beer Sheva, Israel

Charilaos Efthymiou (54, 55)
Computer Science, University of Warwick, Coventry, UK

Ioannis Eleftheriadis (119)
Department of Computer Science and Technology, University of Cambridge, UK

David Eppstein (56)
Department of Computer Science, University of California, Irvine, CA, USA

Navid Eslami (10)
Aalto University, Espoo, Finland; Sharif University of Technology, Tehran, Iran

Javier Esparza (126)
Technische Universiät München, Germany

Louis Esperet (57)
Laboratoire G-SCOP, Grenoble, France

Jan Eube (50)
Universität Bonn, Germany

Austen Z. Fan (127)
Department of Computer Sciences, University of Wisconsin-Madison, WI, USA

Michal Feldman (58)
Blavatnik School of Computer Science, Tel Aviv University, Israel; Microsoft Research, Herzliya, Israel

Weiming Feng (54)
School of Informatics, University of Edinburgh, Edinburgh, UK

Robert Ferens (59)
University of Wrocław, Poland

Nathanaël Fijalkow (118)
CNRS, LaBRI and Université de Bordeaux, France; University of Warsaw, Poland

Emmanuel Filiot (121)
Université libre de Bruxelles & F.R.S.-FNRS, Brussels, Belgium

Marten Folkertsma (32)
QuSoft and CWI, Amsterdam, The Netherlands

Fedor V. Fomin (60, 61)
Department of Informatics,
University of Bergen, Norway

Tobias Friedrich (24, 62)
Hasso Plattner Institute,
Universität Potsdam, Germany

Zachary Friggstad (63)
Department of Computing Science,
University of Alberta, Canada

Daniel Frishberg (56)
Department of Computer Science,
University of California, Irvine, CA, USA

Honghao Fu (64)
CSAIL, Massachusetts Institute of Technology,
Cambridge, MA, USA

Federico Fusco (58)
Department of Computer, Control and
Management Engineering "Antonio Ruberti",
Sapienza University of Rome, Italy

Jakub Gajarský (128)
University of Warsaw, Poland

Moses Ganardi (3, 110)
Max Planck Institute for Software Systems
(MPI-SWS), Kaiserslautern, Germany

Ankit Garg (12)
Microsoft Research, Bangalore, India

Mohit Garg (65)
Department of Computer Science and
Automation, Indian Institute of Science,
Bengaluru, India

Sevag Gharibian (32)
Paderborn Universität, Germany

Badih Ghazi (66)
Google, Mountain View, CA, US

Alexandru Gheorghiu (67)
Department of Computer Science and
Engineering, Chalmers University of Technology,
Göteborg, Sweden;
Institute for Theoretical Studies, ETH Zürich,
Switzerland

Ashish Goel (70)
Stanford University, CA, USA

Leslie Ann Goldberg (68)
Department of Computer Science,
University of Oxford, UK

Ishay Golinsky (19)
Blavatnik School of Computer Science,
Tel Aviv University, Israel

Petr A. Golovach (60, 61)
Department of Informatics,
University of Bergen, Norway

Gramoz Goranci (69)
Faculty of Computer Science,
Universität Wien, Austria

Mohak Goyal (70)
Stanford University, CA, USA

Vincent P. Grande (126)
RWTH Aachen University, Germany

Fabrizio Grandoni (29)
IDSIA, USI-SUPSI, Lugano, Switzerland

Carla Groenland (27)
Mathematical Institute, Utrecht University,
The Netherlands

Logan Grout (15)
Operations Research and Information
Engineering, Cornell University,
Ithaca, NY, USA

Andreas Göbel (62)
Hasso Plattner Institute,
Universität Potsdam, Germany

Daniel Hader (71)
Department of Computer Science and Computer
Engineering, University of Arkansas,
Fayetteville, AR, USA

Nathaniel Harms (57)
EPFL, Lausanne, Switzerland

David G. Harris (72)
Department of Computer Science, University of
Maryland, College Park, MD, USA

Hamed Hatami (42)
McGill University, Montreal, Canada

Pooya Hatami (42)
Ohio State University, Columbus, OH, USA

Ishay Haviv (73)
School of Computer Science, The Academic
College of Tel Aviv-Yaffo, Israel

Ryu Hayakawa (32)
Kyoto University, Japan

Qizheng He (34)
Department of Computer Science, University of
Illinois at Urbana-Champaign, IL, USA

Monika Henzinger (69, 74)
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Thomas A. Henzinger (129)
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Lukas Hintze (18)
Universität Hamburg, Germany

Petr Hliněný (75)
Masaryk University, Brno, Czech republic

Felix Hommelsheim (65)
Faculty of Mathematics and Computer Science, Universität Bremen, Germany

Kaave Hosseini (42)
University of Rochester, NY, USA

Hamed Hosseinpour (18)
Universität Hamburg, Germany

Jakob Bæk Tejs Houen (76)
BARC, Department of Computer Science, University of Copenhagen, Denmark

Jun-Ting Hsieh (77, 78)
Carnegie Mellon University, Pittsburgh, PA, USA

Haoqiang Huang (40)
Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China

Dylan Hyatt-Denesik (79)
Eindhoven University of Technology, The Netherlands

Sharat Ibrahimpur (15, 80)
Department of Mathematics, London School of Economics and Political Science, UK

Takehiro Ito (81, 82)
Graduate School of Information Sciences, Tohoku University, Sendai, Japan

Yuni Iwamasa (81)
Graduate School of Informatics, Kyoto University, Japan

Siddharth Iyer (83)
University of Washington CSE, Seattle, WA, USA

Afrouz Jabal Ameli (29, 79)
TU Eindhoven, The Netherlands

Rhea Jain (36)
Department of Computer Science, University of Illinois, Urbana-Champaign, Urbana, IL, USA

Jan Jedelský (75)
Masaryk University, Brno, Czech republic

Ce Jin (11)
MIT EECS and CSAIL, Cambridge, MA, USA

Zhengzhong Jin (41)
Massachusetts Institute of Technology, Cambridge, MA, USA

Dominik Kaaser (18)
TU Hamburg, Germany

Naonori Kakimura (81, 82)
Faculty of Science and Technology, Keio University, Yokohama, Japan

Yusuf Hakan Kalayci (51)
University of Southern California, Los Angeles, CA, USA

Iden Kalemaj (25)
Department of Computer Science, Boston University, MA, USA

Pritish Kamath (66)
Google, Mountain View, CA, US

Naoyuki Kamiyama (82)
Institute of Mathematics for Industry, Kyushu University, Fukouka, Japan

Haim Kaplan (9, 19)
Tel Aviv University, Israel

Tobias Kappé (136)
Open Universiteit, Heerlen, The Netherlands; ILLC, University of Amsterdam, The Netherlands

Adam Karczmarz (6, 84)
University of Warsaw, Poland; IDEAS NCBR, Warsaw, Poland

Anna R. Karlin (1)
Paul G. Allen School of Computer Science and Engineering, University of Washington, Seattle, WA, USA

Maximilian Katzmann (62)
Karlsruhe Institute of Technology, Germany

Neeraj Kayal (12)
Microsoft Research, Bangalore, India

Pavol Kebis (129)
University of Oxford, UK

George Kenison (130)
Institute of Logic and Computation,
TU Wien, Austria

Sanjeev Khanna (37)
University of Pennsylvania,
Philadelphia, PA, USA

Max Klimm (47)
TU Berlin, Germany

Simon Knäuer (116)
Institut für Algebra, TU Dresden, Germany

Yusuke Kobayashi (81, 82)
Research Institute for Mathematical Sciences,
Kyoto University, Japan

Shimon Kogan (85)
Weizmann Institute of Science, Rehovot, Israel

Daumantas Kojelis (111)
Department of Computer Science,
University of Manchester, UK

Gillat Kol (53)
Princeton University, NJ, USA

Vladimir Kolmogorov (72)
Institute of Science and Technology Austria,
Klosterneuburg, Austria

Niels Kornerup (17)
Computer Science, University of Texas,
Austin, TX, USA

Pravesh K. Kothari (77, 78)
Carnegie Mellon University,
Pittsburgh, PA, USA

Michal Koucký (22)
Computer Science Institute of Charles
University, Prague, Czech Republic

Paraschos Koutris (127)
Department of Computer Sciences,
University of Wisconsin-Madison, WI, USA

Dexter Kozen (136)
Department of Computer Science,
Cornell University, Ithaca, NY, USA

Jakub Kozik (48)
Theoretical Computer Science Department,
Faculty of Mathematics and Computer Science,
Jagiellonian University, Kraków, Poland

László Kozma (19)
Institut für Informatik, Freie Universität Berlin,
Germany

Robert Krauthgamer (30)
Weizmann Institute of Science, Rehovot, Israel

Aditya Krishnan (30)
Pinecone, San Francisco, CA, USA

Simon Krogmann (24)
Hasso Plattner Institute,
Universität Potsdam, Germany

Ariel Kulik (49)
CISPA Helmholtz Center for Information
Security, Saarbrücken, Germany

Pooja Kulkarni (16)
University of Illinois at Urbana-Champaign, IL,
USA

Gunjan Kumar (123)
National University of Singapore, Singapore

Ravi Kumar (66)
Google, Mountain View, CA, US

Orna Kupferman (109)
School of Computer Science and Engineering,
Hebrew University, Jerusalem, Israel

Rasmus Kyng (2)
ETH Zürich, Switzerland

Marvin Künnemann (131)
RPTU Kaiserslautern-Landau, Germany

François Ladouceur (115)
Department of Computer Science,
Université de Sherbrooke, Canada

Michael Lampis (132)
Université Paris-Dauphine, PSL University,
CNRS, LAMSADE, 75016, Paris, France

François Le Gall (32)
Nagoya University, Japan

Shi Li (86)
State Key Laboratory for Novel Software
Technology, Nanjing University, China;
Department of Computer Science and
Engineering, University at Buffalo, NY, USA

Xiantao Li (87)
Department of Mathematics, Pennsylvania State
University, University Park, PA, USA

Xin Li (41)
Department of Computer Science,
Johns Hopkins University, Baltimore, MD, USA

Moritz Lichter (133)
TU Darmstadt, Germany

Henrik Lievonen (10)
Aalto University, Espoo, Finland

Paul Liu (74)
Stanford University, CA, USA

Quanquan C. Liu (52)
Northwestern University, Evanston, IL, US

S. Cliff Liu (88)
Carnegie Mellon University,
Pittsburgh, PA, USA

Shu Liu (89)
The National Key Laboratory on Wireless Communications, University of Electronic Science and Technology of China, Chengdu, China

Markus Lohrey (134)
Universität Siegen, Germany

Daniel Lokshtanov (90)
University of California Santa Barbara, CA, USA

Kelin Luo (50)
Universität Bonn, Germany

Xin Lyu (39)
University of California at Berkeley, CA, USA

Shun-ichi Maezawa (81, 82)
Department of Mathematics,
Tokyo University of Science, Japan

Rupak Majumdar (3, 110)
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Alessio Mansutti (112)
IMDEA Software Institute, Madrid, Spain

Pasin Manurangsi (66)
Google, Bangkok, Thailand

Claire Mathieu (91)
CNRS Paris, France

Simon Mauras (58)
Blavatnik School of Computer Science,
Tel Aviv University, Israel

Filip Mazowiecki (131)
University of Warsaw, Poland

Nicolas Mazzocchi (129)
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Rose McCarty (128)
Princeton University, NJ, USA

Kuldeep S. Meel (123)
National University of Singapore, Singapore

Nicole Megow (65)
Faculty of Mathematics and Computer Science, Universität Bremen, Germany

Konstantina Mellou (92)
Microsoft Research, Redmond, WA, USA

Darya Melnyk (10)
Aalto University, Espoo, Finland;
TU Berlin, Germany

Tony Metger (67)
Institute for Theoretical Physics, ETH Zürich, Switzerland

Stefan Milius (114)
Friedrich-Alexander-Universität
Erlangen-Nürnberg, Germany

Gopinath Mishra (46)
University of Warwick, Coventry, UK

Slobodan Mitrović (45)
University of California Davis, CA, USA

Marco Molinaro (92)
Microsoft Research, Redmond, WA, USA;
PUC-Rio de Janeiro, Brazil

Laure Morelle (93)
LIRMM, Université de Montpellier, CNRS, France

Tomoyuki Morimae (32)
Kyoto University, Japan

Ramin Mousavi (63)
Department of Computing Science, University of Alberta, Canada

Etienne Moutot (120)
CNRS, I2M, Aix-Marseille Université, Marseille, France

Karthik Murali (23)
School of Computer Science, Carleton University, Ottawa, Canada

Cameron Musco (21)
Manning College of Information and Computer Sciences, University of Massachusetts, Amherst, MA, USA

Nikolas Mählmann (125, 128)
Universität Bremen, Germany

Lê Thành Dũng (Tito) Nguyễn (117)
Laboratoire de l'informatique du parallélisme (LIP), École normale supérieure de Lyon, France

Joris Nieuwveld (130)
Max Planck Institute for Software Systems, Saarland Informatics Campus, Saarbrücken, Germany

Yuta Nozaki (81, 82)
Faculty of Environment and Information Sciences, Yokohama National University, Japan; SKCM, Hiroshima University, Japan

Pierre Ohlmann (122, 128, 135)
University of Warsaw, Poland

Yoshio Okamoto (81, 82)
Graduate School of Informatics and Engineering, The University of Electro-Communications, Tokyo, Japan

Kazusato Oko (94)
Department of Mathematical Informatics, The University of Tokyo, Japan;
Center for Advanced Intelligence Project, RIKEN, Tokyo, Japan

Rotem Oshman (95)
Tel-Aviv University, Israel

Joël Ouaknine (130)
Max Planck Institute for Software Systems, Saarland Informatics Campus, Saarbrücken, Germany

Kenta Ozeki (81)
Faculty of Environment and Information Sciences, Yokohama National University, Japan

Debmalya Panigrahi (43)
Department of Computer Science, Duke University, Durham, NC, USA

Aris Papadopoulos (119)
School of Mathematics, Univesity of Leeds, UK

Dmitry Paramonov (53)
Princeton University, NJ, USA

Nikos Parotsidis (6)
Google Research, Zürich, Switzerland

Merav Parter (85)
Weizmann Institute of Science, Rehovot, Israel

Neel Patel (51)
University of Southern California, Los Angeles, CA, USA

Matthew J. Patitz (71)
Department of Computer Science and Computer Engineering, University of Arkansas, Fayetteville, AR, USA

Pan Peng (96)
School of Computer Science and Technology, University of Science and Technology of China, Hefei, China

Thomas Perez (120)
Université de Lyon, ENS de Lyon, France

Michał Pilipczuk (27, 128, 135)
Institute of Informatics, University of Warsaw, Poland

Alexander Poremba (67)
Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA, USA

Aaron Potechin (78)
University of Chicago, IL, USA

Ian Pratt-Hartmann (111)
Department of Computer Science, University of Manchester, UK; Institute of Computer Science, University of Opole, Poland

Wojciech Przybyszewski (128, 135)
University of Warsaw, Poland

Manish Purohit (80)
Google Research, USA

Minglong Qin (97)
State Key Laboratory for Novel Software Technology, Nanjing University, China

Rajmohan Rajaraman (98)
Northeastern University, Boston, MA, USA

Mickael Randour (118)
F.R.S.-FNRS & UMONS – Université de Mons, Belgium

Sofya Raskhodnikova (25, 52)
Department of Computer Science, Boston University, MA, USA

Peter M. R. Rasmussen (6)
BARC, University of Copenhagen, Denmark

Malin Rau (18)
Universität Hamburg, Germany

Archan Ray (21)
Manning College of Information and Computer Sciences, University of Massachusetts, Amherst, MA, USA

Rebecca Reiffenhäuser (58)
Institute for Logic, Language and Computation, University of Amsterdam, The Netherlands

Nicolas Resch (99)
Informatics' Institute, University of Amsterdam, The Netherlands

Eric Rivals (100)
LIRMM, Université Montpellier, CNRS, France

David E. Roberson (101)
Department of Applied Mathematics and Computer Science, Technical University of Denmark, Copenhagen, Denmark;
QMATH, Department of Mathematical Sciences, University of Copenhagen, Denmark

Dana Ron (44)
Tel Aviv University, Israel

Alon Rosen (28)
Bocconi University, Milano, Italy;
Reichman University, Herzliya, Israel

Andreas Rosowski (134)
Universität Siegen, Germany

Marc Roth (68)
Department of Computer Science, University of Oxford, UK

Tal Roth (95)
Tel-Aviv University, Israel

Ronitt Rubinfeld (45)
MIT, Cambridge, MA, USA

Ittai Rubinstein (102)
Qedma Quantum Computing, Tel Aviv, Israel

Wojciech Różowski (136)
Department of Computer Science, University College London, UK

Heiko Röglin (50)
Universität Bonn, Germany

Danil Sagunov (60)
St. Petersburg Department of V.A. Steklov Institute of Mathematics, Russia

Chandan Saha (12)
Indian Institute of Science, Bangalore, India

Shinsaku Sakaue (94)
Department of Mathematical Informatics, The University of Tokyo, Japan

Sukolsak Sakshuwong (70)
Stanford University, CA, USA

Laura Sanità (79)
Bocconi University, Milano, Italy

Piotr Sankowski (84)
University of Warsaw, Poland;
IDEAS NCBR, Warsaw, Poland

Shay Sapir (30)
Weizmann Institute of Science, Rehovot, Israel

N. Ege Saraç (129)
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Sahasrajit Sarmasarkar (70)
Stanford University, CA, USA

Ignasi Sau (61, 93)
LIRMM, Université de Montpellier, CNRS, France

Thomas Sauerwald (103)
University of Cambridge, UK

Saket Saurabh (90)
The Institute of Mathematical Sciences, HBNI, Chennai, India;
University of Bergen, Norway

Raghuvansh R. Saxena (53)
Microsoft Research, Cambridge, MA, USA

Kevin Schewior (47)
University of Southern Denmark, Odense, Denmark

Leon Schiller (62)
Hasso Plattner Institute, Universität Potsdam, Germany

Martin Schirneck (24)
Faculty of Computer Science, Universität Wien, Austria

Todd Schmid (136)
Department of Computer Science, University College London, UK

Melanie Schmidt (50)
Heinrich-Heine Universität Düsseldorf, Germany

Lia Schütze (131)
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Tim Seppelt (101)
RWTH Aachen University, Germany

Hadas Shachnai (49)
Computer Science Department, Technion, Haifa, Israel

Sebastian Siebertz (125, 128)
Universität Bremen, Germany

Alexandra Silva (136)
Department of Computer Science, Cornell University, Ithaca, NY, USA

Kirill Simonov (60)
Hasso Plattner Institute, Universität Potsdam, Germany

Henry Sinclair-Banks (131)
Centre for Discrete Mathematics and its Applications (DIMAP) & Department of Computer Science, University of Warwick, Coventry, UK

Adam Smith (52)
Boston University, MA, US

Marek Sokołowski (128)
University of Warsaw, Poland

Zhao Song (88)
Adobe Research, San Jose, CA, USA

David Stalfa (98)
Northeastern University, Boston, MA, USA

Giannos Stamoulis (61, 93)
LIRMM, Université de Montpellier, CNRS, France

He Sun (103)
University of Edinburgh, UK

Jukka Suomela (10)
Aalto University, Espoo, Finland

Vaishali Surianarayanan (90)
University of California Santa Barbara, CA, USA

Zoya Svitkina (80)
Google Research, USA

Michelle Sweering (100)
CWI, Amsterdam, The Netherlands

Marek Szykuła (59)
University of Wrocław, Poland

Joona Särkijärvi (10)
Aalto University, Espoo, Finland

Avishay Tal (39)
University of California at Berkeley, CA, USA

Zihan Tan (37)
DIMACS, Rutgers University, NJ, USA

Shin-ichi Tanigawa (94)
Department of Mathematical Informatics, The University of Tokyo, Japan

Tatsuya Terao (104)
Research Institute for Mathematical Sciences, Kyoto University, Japan

Bhargav Thankey (12)
Indian Institute of Science, Bangalore, India

Dimitrios M. Thilikos (61, 93)
LIRMM, Université de Montpellier, CNRS, France

Ramanathan S. Thinniyam (3, 110)
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Mikkel Thorup (6, 76)
BARC, University of Copenhagen, Denmark

Szymon Toruńczyk (125, 128, 135)
University of Warsaw, Poland

Noam Touitou (105)
Amazon, Tel Aviv, Israel

Henning Urbat (114)
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Frits Vaandrager (137)
Radboud University, Nijmegen, The Netherlands

Danny Vagnozzi (103)
University of Edinburgh, UK

Danny Vainstein (13)
School of Computer Science, Tel-Aviv University, Israel

Pierre Vandenhove (118)
F.R.S.-FNRS & UMONS - Université de Mons, Belgium; Université Paris-Saclay, CNRS, ENS Paris-Saclay, Laboratoire Méthodes Formelles, 91190, Gif-sur-Yvette, France

Erik Vee (80)
Google Research, USA

Thomas Vidick (4)
Weizmann Institute of Science, Rehovot, Israel; California Institute of Technology, Pasadena, CA, USA

Renaud Vilmart (120)
Université Paris-Saclay, ENS Paris-Saclay, Inria, CNRS, LMF, 91190, Gif-sur-Yvette, France

Harry Vinall-Smeeth (113)
Humboldt-Universität zu Berlin, Germany

Jan Vondrák (74)
Stanford University, CA, USA

Chunhao Wang (87)
Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA

Daochen Wang (64)
QuICS, University of Maryland, College Park, MD, USA

Joshua R. Wang (80)
Google Research, USA

Pengfei Wang (100)
LIRMM, Université Montpellier, CNRS, France

Yuyang Wang (96)
School of Computer Science and Technology, University of Science and Technology of China, Hefei, China

Julian Wargalla (50)
Heinrich-Heine Universität Düsseldorf, Germany

David Weckbecker (47)
TU Darmstadt, Germany

Jordi Weggemans (32)
QuSoft and CWI, Amsterdam, The Netherlands

Zhewei Wei (7)
Renmin University of China, Beijing, China

Zhide Wei (41)
Department of Computer Science, Peking University, Beijing, China

Michael Whitmeyer (83)
University of Washington CSE, Seattle, WA, USA

Sarah Winter (121)
Université libre de Bruxelles & F.R.S.-FNRS, Brussels, Belgium

Thorsten Wißmann (137)
Radboud University, Nijmegen, The Netherlands;
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

James Worrell (5, 130)
Department of Computer Science, University of Oxford, UK

Hongxun Wu (39)
University of California at Berkeley, CA, USA

Kewen Wu (66)
University of California at Berkeley, CA, US

Karol Węgrzycki (131)
Saarland University and Max Planck Institute for Informatics, Saarbrücken, Germany

Michał Włodarczyk (106)
Ben-Gurion University, Beer Sheva, Israel

Chaoping Xing (89)
School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China

Jeff Xu (78)
Carnegie Mellon University, Pittsburgh, PA, USA

Sheng Yang (98)
Shanghai, CN

Penghui Yao (97)
State Key Laboratory for Novel Software Technology, Nanjing University, China;
Hefei National Laboratory, 230088, China

Ben Young (33)
Department of Computer Sciences, University of Wisconsin-Madison, WI, USA

Yuancheng Yu (34)
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA

Chen Yuan (89, 99)
School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China

Viktor Zamaraev (57)
University of Liverpool, UK

Or Zamir (107)
Princeton University, NJ, USA

Kostas Zampetakis (55)
Computer Science, University of Warwick, Coventry, UK

Georg Zetzsche (3, 110)
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Hengjie Zhang (88)
Columbia University, New York, NY, USA

Lichen Zhang (88)
Massachusetts Institute of Technology, Cambridge, MA, USA

Ruizhe Zhang (108)
The University of Texas at Austin, TX, USA

Xinzhi Zhang (108)
University of Washington, Seattle, WA, USA

Yihan Zhang (99)
Institute of Science and Technology Austria, Klosterneuburg, Austria

Hangdong Zhao (127)
Department of Computer Sciences, University of Wisconsin-Madison, WI, USA

Qi Zhao (64)
QuICS, University of Maryland, College Park, MD, USA;
QICI, The University of Hong Kong, China

Da Wei Zheng (74)
University of Illinois Urbana-Champaign, IL, USA

Yu Zheng (41)
Meta Platforms Inc

Hang Zhou (91)
École Polytechnique, Institut Polytechnique de Paris, France

Rudy Zhou (92)
Carnegie Mellon University, Pittsburgh, PA, USA

Tianyi Zhou (88)
University of California San Diego, CA, USA

Jakub Łącki (6)
Google Research, New York, NY,USA

A (Slightly) Improved Approximation Algorithm for the Metric Traveling Salesperson Problem

Anna R. Karlin ✉
Paul G. Allen School of Computer Science and Engineering,
University of Washington, Seattle, WA, USA

Abstract

We describe recent joint work with Nathan Klein and Shayan Oveis Gharan showing that for any metric TSP instance, the max entropy algorithm studied by [1] returns a solution of expected cost at most $\frac{3}{2} - \epsilon$ times the cost of the optimal solution to the subtour elimination LP and hence is a $\frac{3}{2} - \epsilon$ approximation for the metric TSP problem. The research discussed comes from [1], [2] and [3].

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Traveling Salesperson Problem, approximation algorithm

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.1

Category Invited Talk

Related Version *Full Version*: `https://arxiv.org/pdf/2007.01409.pdf`

Funding *Anna R. Karlin*: Research supported by Air Force Office of Scientific Research grant FA9550-20-1-0212 and NSF grant CCF-1813135.

References

1 Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved approximation algorithm for metric TSP. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 32–45. ACM, 2021. `doi:10.1145/3406325.3451009`.

2 Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved bound on the integrality gap of the subtour LP for TSP. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 – November 3, 2022*, pages 832–843. IEEE, 2022. `doi:10.1109/FOCS54457.2022.00084`.

3 Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved deterministic approximation algorithm for metric TSP. *CoRR*, abs/2212.06296, 2022. `doi:10.48550/arXiv.2212.06296`.

An Almost-Linear Time Algorithm for Maximum Flow and More

Rasmus Kyng
ETH Zürich, Switzerland

Abstract

In this talk, I will explain a new algorithm for computing exact maximum and minimum-cost flows in almost-linear time, settling the time complexity of these basic graph problems up to subpolynomial factors.

Our algorithm uses a novel interior point method that builds the optimal flow as a sequence of approximate minimum-ratio cycles, each of which is computed and processed very efficiently using a new dynamic data structure.

By well-known reductions, our result implies almost-linear time algorithms for several problems including bipartite matching, optimal transport, and undirected vertex connectivity. Our framework also extends to minimizing general edge-separable convex functions to high accuracy, yielding the first almost-linear time algorithms for many other problems including entropy-regularized optimal transport, matrix scaling, p-norm flows, and isotonic regression.

This talk is based on joint work with Li Chen, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva [1]. Our result appeared in FOCS'22 and won the FOCS best paper award.

2012 ACM Subject Classification Theory of computation → Network flows; Theory of computation → Sparsification and spanners; Theory of computation → Dynamic graph algorithms

Keywords and phrases Maximum flow, Minimum cost flow, Data structures, Interior point methods, Convex optimization

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.2

Category Invited Talk

Funding *Rasmus Kyng*: The research leading to these results has received funding from the grant "Algorithms and complexity for high-accuracy flows and convex optimization" (no. 200021 204787) of the Swiss National Science Foundation.

References

1 Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 612–623, 2022. doi:10.1109/FOCS54457.2022.00064.

Context-Bounded Analysis of Concurrent Programs

Pascal Baumann
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Moses Ganardi
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Rupak Majumdar
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Ramanathan S. Thinniyam
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Georg Zetzsche
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Abstract

Context-bounded analysis of concurrent programs is a technique to compute a sequence of under-approximations of all behaviors of the program. For a fixed bound k, a context bounded analysis considers only those runs in which a single process is interrupted at most k times. As k grows, we capture more and more behaviors of the program. Practically, context-bounding has been very effective as a bug-finding tool: many bugs can be found even with small bounds. Theoretically, context-bounded analysis is decidable for a large number of programming models for which verification problems are undecidable. In this paper, we survey some recent work in context-bounded analysis of multithreaded programs.

In particular, we show a general decidability result. We study context-bounded reachability in a language-theoretic setup. We fix a class of languages (satisfying some mild conditions) from which each thread is chosen. We show context-bounded safety and termination verification problems are decidable iff emptiness is decidable for the underlying class of languages and context-bounded boundedness is decidable iff finiteness is decidable for the underlying class.

2012 ACM Subject Classification Theory of computation → Concurrency; Software and its engineering → Software verification

Keywords and phrases Context-bounded analysis, Multi-threaded programs, Decidability

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.3

Category Invited Talk

Funding This research was partially funded by the DFG project 389792660 TRR 248–CPEC and by the European Union (ERC, FINABIS, 101077902). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

1 Introduction

Algorithmic verification of shared-state multithreaded programs is one of the main motivations for research in theoretical computer science. The general problem is undecidable, even when the class of programs is restricted in different ways. Thus, one direction of research has focused on finding decidable models that *over-approximate* the problem and another on finding *under-approximations*. An over-approximate model captures more behaviors than the original program; thus, if we find that the over-approximation has no bad behaviors, we

can be certain that neither does the original program. An under-approximation, conversely, captures fewer behaviors. In this case, if we find a bad behavior in the approximation, we know that the bad behavior is also possible in the original program.

We consider a particular type of under-approximation: *context bounding*. Context-bounding is a technique to construct a parameterized sequence of under-approximations [46, 37]. For a fixed parameter k, a k-context-bounded analysis considers only those behaviors of the program in which an individual thread is interrupted by the scheduler at most k times. As k increases, more and more behaviors of the original program fall into the purview of the analysis. In the limit, all behaviors are covered.

Context-bounding has become a popular technique because of two reasons. For a wide class of programming models and verification questions, context-bounded analyses become decidable, even though the unrestricted problems are undecidable. Moreover, in practice, context-bounded analysis has had success as a bug finding tool, since many bugs in practical instances can be discovered even with small values of k [46, 44, 36, 34].

We focus on decidability questions. In order to avoid "trivial" encodings of Turing machines, we restrict programs to be *finite data* – that is, we assume each program variable to take on finitely many values. Even with this restriction, depending on the model of programs, decidability can be non-immediate because the state space of a program can be infinite in other respects, such as the stack of an individual thread or the number of pending threads.

Properties of concurrent programs. For the moment, we focus on three decision problems: context-bounded *reachability* ("is there a k-bounded execution that reaches a specific global state?"), context-bounded *termination* ("all all k-bounded executions terminating?"), and context-bounded *boundedness* ("is there a bound on the number of pending threads along every k-bounded execution?"). We shall come back to other problems later.

Context-bounded analysis is a family of problems, depending on the model of concurrent programs as well as on the correctness properties considered. Qadeer and Rehof's original paper [46], that introduced context-bounding, stipulated that there is a fixed number of *recursive* threads that read or write shared variables but these threads do not spawn further threads. They showed that the reachability problem is NP-complete. Note that even with two threads, the reachability problem for finite-data programs is already undecidable: for example, we can encode the intersection non-emptiness problem for pushdown automata. On the other hand, if threads are not recursive, then the reachability problem is decidable without any context bounding restrictions, even if threads can spawn further threads: this can be shown by a reduction to the coverability problem for vector addition systems with states (VASS). Subsequently, Atig, Bouajjani, and Qadeer [11] extended decidability for context-bounded reachability when threads can spawn further threads. They showed an upper bound of 2EXPSPACE and a matching lower bound was shown by Baumann et al. [14]. Similar techniques show the same complexity for termination and boundedness.

A special case: Asynchronous programs. The special case of $k = 0$ of context-bounded analysis is important enough to have its own name: *asynchronous programs*. In an asynchronous program, threads are executed atomically to completion (that is, never interrupted by the scheduler). Many software systems based on cooperative scheduling implement this model. Sen and Viswanathan [47] studied the model and showed reachability is decidable by reducing to a well-structured transition system. Ganty and Majumdar [28] showed that reachability, termination, and boundedness are all EXPSPACE-complete, by again reducing to coverability problems for VASS.

Majumdar, Thinniyam, and Zetzsche [40] proved decidability results for asynchronous programs in a general language-theoretic setting. They fix a class of languages $\mathcal{C}$, and consider asynchronous programs in which each individual thread is a language from the class $\mathcal{C}$ over the alphabet of thread names as well as a transformer over the global states. That is, each thread is a language (from $\mathcal{C}$) of words of the form dwd', where d and d' are global states and w is a sequence of thread names. The intent is that an atomic execution of the thread takes the global state from d to d' and also spawns new instances of all the threads in w.

They show that for all classes $\mathcal{C}$ satisfying a mild language-theoretic assumption (the class $\mathcal{C}$ is a *full trio*), safety and termination are decidable if and only if the underlying language class $\mathcal{C}$ has a decidable emptiness problem. Similarly, boundedness is decidable if and only if finiteness is decidable for $\mathcal{C}$. As a consequence, they get decidability results for asynchronous programs over context-free languages, higher-order recursion schemes, as well as other language classes studied in infinite-state verification.

Contribution. Our starting point is the general approach of Majumdar, Thinniyam, and Zetzsche [40]. We show their general decidability results can be extended to context-bounded analysis (any $k \geq 0$). We define concurrent programs over a language class $\mathcal{C}$ and show analogous decidability results: (i) context-bounded reachability and context-bounded termination for programs are decidable if and only if $\mathcal{C}$ has a decidable emptiness problem, and (ii) context-bounded boundedness is decidable if and only if $\mathcal{C}$ has a decidable finiteness problem. As a consequence, we get a uniform proof for decidability for these problems for programs over context-free languages and for programs over higher-order recursion schemes.

The key argument in both settings is that of *downclosures* of languages under the subword ordering. Safety, termination, and boundedness are preserved if we "lose" some spawned threads, as long as the sequence of global state changes (and there are at most k of them for the fixed context bound k) is maintained. Since downclosures (even when maintaining a bounded number of distinguished letters) are always regular languages, this implies: If our concurrent program satisfies one of the above properties, then each thread can be over-approximated by a regular language so that the property is still satisfied. The decision procedure for reachability then runs two semi-decision procedures: one enumerates executions (to check for reachability) and the other enumerates regular languages and checks that (1) the thread languages are contained in the regular languages and (2) uses known decidability results for context-bounded reachability with regular thread languages.

The decision procedure does not, in particular, need to *construct* an explicit description of the downclosure. In fact, it even shows decidability for language classes for which downclosures cannot be constructed. On the flip side, we do not get complexity bounds.

Other properties. What about other properties? Ganty and Majumdar showed fair termination for context-free asynchronous programs is decidable (by reduction to Petri net reachability) [28]. Majumdar, Thinniyam, and Zetzsche generalized the result to show that fair termination is equivalent to configuration reachability in the general setting [40]. On the other hand, decidability of fair termination implies the decidability of checking the "equal letters problem": deciding if a language in $\mathcal{C}$ has an equal number of as and bs. Thus, fair termination is undecidable for indexed languages. The undecidability is inherited by context-bounded fair termination. On the other hand, somewhat surprisingly, fair termination is decidable for context-bounded runs of context-free multithreaded programs [15].

2 Preliminaries

An *alphabet* is a finite non-empty set of *symbols*. For an alphabet Σ, we write Σ^* for the set of finite sequences of symbols (also called *words*) over Σ. A set $L \subseteq \Sigma^*$ of words is a *language*. By $\mathsf{pref}(L) = \{u \in \Sigma^* \mid \exists v \in \Sigma^* \colon uv \in L\}$ we denote the set of prefixes of words in L.

The *subword* order $\sqsubseteq$ on Σ^* is defined as follows: for $u, v \in \Sigma^*$ we have $u \sqsubseteq v$ if and only if u can be obtained from v by deleting some of v's letters. For example, *abba* $\sqsubseteq$ *bababa*, but *abba* $\not\sqsubseteq$ *baaba*. The *downclosure* (or *downward closure*) $\downarrow w$ of a word $w \in \Sigma^*$ with respect to the subword order is defined as $\downarrow w := \{w' \in \Sigma^* \mid w' \sqsubseteq w\}$. The downclosure $\downarrow L$ of a language $L \subseteq \Sigma^*$ is given by $\downarrow L := \{w' \in \Sigma^* \mid \exists w \in L \colon w' \sqsubseteq w\}$. An important fact is that the subword ordering $\sqsubseteq$ is a *well-quasi ordering* (Higman's lemma). A consequence is that the downclosure $\downarrow L$ of *any* language L is a regular language [32]. However, a representation for the downclosure of a language may not be effectively constructible.

The projection of a word $w \in \Sigma^*$ onto some alphabet $\Gamma \subseteq \Sigma$, written $\mathrm{Proj}_\Gamma(w)$, is the word obtained by erasing from w each symbol which does not belong to Γ. For a language L, define $\mathrm{Proj}_\Gamma(L) = \{\mathrm{Proj}_\Gamma(w) \mid w \in L\}$. We write $|w|_\Gamma$ for the number of occurrences of letters $a \in \Gamma$ in w, and similarly $|w|_a$ if $\Gamma = \{a\}$.

A *multiset* $\mathbf{m} \colon X \to \mathbb{N}$ over a set X maps each symbol of X to a natural number. The size $|\mathbf{m}|$ of a multiset $\mathbf{m}$ is given by $|\mathbf{m}| = \sum_{x \in X} \mathbf{m}(x)$. The set of all multisets over X is denoted $\mathbb{M}[X]$. We identify subsets of X with multisets in $\mathbb{M}[X]$ where each element is mapped to 0 or 1. We write $\mathbf{m} = [\![a, a, c]\!]$ for the multiset $\mathbf{m} \in \mathbb{M}[\{a, b, c, d\}]$ such that $\mathbf{m}(a) = 2$, $\mathbf{m}(b) = \mathbf{m}(d) = 0$, and $\mathbf{m}(c) = 1$. The Parikh image $\mathsf{Parikh}(w) \in \mathbb{M}[\Sigma]$ of a word $w \in \Sigma^*$ is the multiset such that for each letter $a \in \Sigma$ we have $\mathsf{Parikh}(w)(a) = |w|_a$.

Given two multisets $\mathbf{m}, \mathbf{m}' \in \mathbb{M}[X]$ we define $\mathbf{m} \oplus \mathbf{m}' \in \mathbb{M}[X]$ to be the multiset such that for all $a \in X$, we have $(\mathbf{m} \oplus \mathbf{m}')(a) = \mathbf{m}(a) + \mathbf{m}'(a)$. If $\mathbf{m}(a) \geq \mathbf{m}'(a)$ for all $a \in X$, we also define $\mathbf{m}' \ominus \mathbf{m} \in \mathbb{M}[X]$: for all $a \in X$, we have $(\mathbf{m} \ominus \mathbf{m}')(a) = \mathbf{m}(a) - \mathbf{m}'(a)$. For $X \subseteq Y$ we regard $\mathbf{m} \in \mathbb{M}[X]$ as a multiset in $\mathbb{M}[Y]$ where undefined values are mapped to 0.

Language Classes and Full Trios. A *language class* is a collection of languages, together with some finite representation. Examples are the regular languages (e.g. represented by finite automata) or the context-free languages (e.g. represented by pushdown automata). A relatively weak and reasonable assumption on a language class is that it is a *full trio*, that is, it is closed under *rational transductions*. Equivalently, a language class is a full trio if it is closed under each of the following operations: taking intersection with a regular language, taking homomorphic images, and taking inverse homomorphic images [16].

We assume that all full trios $\mathcal{C}$ considered in this paper are *effective*: Given a language L from $\mathcal{C}$, a regular language R, and a homomorphism h, we can compute a representation of the languages $L \cap R$, $h(L)$, and $h^{-1}(L)$ in $\mathcal{C}$.

Many classes of languages studied in formal language theory form effective full trios. These include the regular and the context-free languages [33], the indexed languages [2, 25], the languages of higher-order pushdown automata [42], higher-order recursion schemes [31, 24, 40], Petri nets [29, 35], and lossy channel systems. However, the class of deterministic context-free languages is not a full trio: this class is not closed under rational transductions.

3 A Language-Theoretic Model of Concurrent Programs

Intuitively, a concurrent program consists of a shared global state and a finite number of thread names. Instances of thread names are called threads. A configuration of such a program consists of the current value of the global state and a multiset of partially-executed

threads. A non-deterministic scheduler picks a partially-executed thread and runs it for some number of steps. An executing thread can change the global state. It can also spawn new threads – these can be picked and executed by the scheduler (in any order) in the future. When a scheduler swaps a running thread for another one, we say that there is a context switch. In our formal model, we keep the global state explicit and we model the execution behavior of threads as languages. The language of a thread captures the new threads it can spawn, as well as the effect of the execution on the global state.

3.1 Model

Let $\mathcal{C}$ be an (effective) full trio. A *concurrent program* (CP) over $\mathcal{C}$ is a tuple $\mathfrak{P} = (D, \Sigma, (L_a)_{a\in\Sigma}, d_0, \mathbf{m}_0)$, where D is a finite set of *global states*, Σ is an alphabet of *thread names*, $(L_a)_{a\in\Sigma}$ is a family of languages from $\mathcal{C}$ over the alphabet $\Sigma_D = D \cup \Sigma \cup (D \times D)$, $d_0 \in D$ is an *initial state*, and $\mathbf{m}_0 \in \mathbb{M}[\Sigma]$ is a multiset of *initial pending thread instances*. We assume that each L_a, $a \in \Sigma$, satisfies the condition $L_a \subseteq aD\big(\Sigma \cup (D \times D)\big)^* D$ (we provide the intuition behind this condition below).

A *configuration* $c = (d, \mathbf{m}) \in D \times \mathbb{M}[\Sigma_D^*]$ consists of a global state $d \in D$ and a multiset $\mathbf{m}$ of strings representing pending threads instances and partially executed threads. Given a configuration $c = (d, \mathbf{m})$, we write $c.d$ and $c.\mathbf{m}$ to denote the elements d and $\mathbf{m}$, respectively. The size of a configuration c is $|c.\mathbf{m}|$, i.e. the number of threads in the task buffer. We distinguish between threads that have been spawned but not executed (*pending threads*) and threads that have been partially executed (but swapped out). The pending thread instances are represented by single letters $a \in \Sigma$ (which corresponds to the name of the thread) while the partially executed threads of "type" $a \in \Sigma$ are represented by strings in $\mathsf{pref}(L_a)$ which end in a letter from $D \times D$.

Before presenting the formal semantics, let us provide some intuition. Suppose the current configuration is $(d, \mathbf{m})$. A non-deterministic scheduler picks one of the outstanding threads (either a pending thread $a \in \mathbf{m}$ or a partially executed thread $w \in \mathbf{m}$) and executes it for some time, until it terminates or until the scheduler decides to interrupt it. The execution of a thread a is abstractly modeled by the language L_a. A word $ad_1w_1(d'_1, d_2)w_2(d'_2, d_3)\dots(d'_{k-1}, d_k)w_{k+1}d_{k+1} \in L_a$ represents a run of an instance of the thread a. The run starts executing in global state d_1. It spawns new threads $w_1 \in \Sigma^*$, then gets interrupted at global state d'_1 by the scheduler. At some future point, the scheduler starts executing it again at global state d_2, when new threads w_2 are spawned before it is interrupted again at d'_2. The execution continues in this way until the thread terminates in global state d_{k+1}. Thus, the jump from one global state to another (from the perspective of the thread) when a context switch is made is represented by a letter from $D \times D$. The part of a run starting at global state d_i, spawning threads w_i and interrupted at d'_i is called a *segment*. Each interruption is called a *context switch*; the above word has k context switches.

Formally, the semantics of $\mathfrak{P}$ are given as a labelled transition system over the set of configurations with the transition relation $\Rightarrow \subseteq (D \times \mathbb{M}[\Sigma_D^*]) \times (D \times \mathbb{M}[\Sigma_D^*])$. The initial configuration is given by $c_0 = (d_0, \mathbf{m}_0)$.

The transition relation is defined using rules of four different types shown below. All four types of rules are of the general form $d \xrightarrow{[\![w]\!], \mathbf{n}'} d'$. A rule of this form allows the program to move from a configuration $(d, \mathbf{m})$ to configuration $(d', \mathbf{m}')$, i.e., $(d, \mathbf{m}) \Rightarrow (d', \mathbf{m}')$, iff $d \xrightarrow{[\![w]\!].\mathbf{n}'} d'$ matches a rule and $(\mathbf{m} \ominus [\![w]\!]) \oplus \mathbf{n}' = \mathbf{m}'$. Note that due to the definition of $\ominus$, $\mathbf{m}$ has to contain w for the rule to be applicable. We also write $\overset{w}{\Longrightarrow}$ to specify the particular w used in the transition. As usual, the reflexive transitive closure of $\Rightarrow$ is denoted by $\Rightarrow^*$. A configuration c is said to be *reachable* if $c_0 \Rightarrow^* c$.

(R1) $d \xrightarrow{[\![a]\!],\mathsf{Parikh}(w)\oplus[\![adw(d',d'')]\!]} d'$ if $\exists w \in \Sigma^* : adw(d',d'') \in \mathsf{pref}(L_a)$.
Rule (R1) allows us to pick some thread a from $\mathbf{m}$ and atomically execute it until the point it is switched out by the scheduler. Note that the final letter (d', d'') of the thread indicates that it has been switched out at global d' and can be resumed when the global state is d''.

(R2) $d \xrightarrow{[\![a]\!],\mathsf{Parikh}(w)} d'$ if $\exists w \in \Sigma^* : adwd' \in L_a$.
Rule (R2) allows us to pick some thread a from $\mathbf{m}$ and atomically execute it to completion.

(R3) $d \xrightarrow{[\![aw'(d'',d)]\!],\mathsf{Parikh}(w)\oplus[\![aw'(d'',d)w(d',d''')]\!]} d'$ if $\exists w \in \Sigma^* : aw'(d'',d)w(d',d''') \in \mathsf{pref}(L_a)$
Rule (R3) allows us to pick some partially executed thread and execute it atomically until the point it is switched out by the scheduler.

(R4) $d \xrightarrow{[\![aw'(d'',d)]\!],\mathsf{Parikh}(w)} d'$ if $\exists w \in \Sigma^* : aw'(d'',d)wd' \in L_a$
Rule (R4) allows us to pick some partially executed thread and execute it to completion.

3.2 Runs and Context-bounded Runs

A *prerun* of a concurrent program $\mathfrak{P} = (D, \Sigma, (L_a)_{a\in\Sigma}, d_0, \mathbf{m}_0)$ is a finite or infinite sequence $\rho = (e_0, \mathbf{n}_0), w_1, (e_1, \mathbf{n}_1), w_2, \ldots$ of alternating elements of configurations $(e_i, \mathbf{n}'_i) \in D \times \mathbb{M}[\Sigma_D^*]$ and strings $w_i \in \Sigma^*$.

The set of preruns of $\mathfrak{P}$ will be denoted $\mathsf{Preruns}(\mathfrak{P})$. Note that if two concurrent programs $\mathfrak{P}$ and $\mathfrak{P}'$ have the same global states D and alphabet Σ, then $\mathsf{Preruns}(\mathfrak{P}) = \mathsf{Preruns}(\mathfrak{P}')$. The length $|\rho|$ of a finite prerun ρ is the number of configurations in ρ.

A *run* of a CP $\mathfrak{P} = (D, \Sigma, (L_a)_{a\in\Sigma}, d_0, \mathbf{m}_0)$ is a prerun $\rho = (d_0, \mathbf{m}_0), w_1, (d_1, \mathbf{m}_1), w_2, \ldots$ starting with the initial configuration $(d_0, \mathbf{m}_0)$, where for each $i \geq 0$, we have $(d_i, \mathbf{m}_i) \xrightarrow{w_{i+1}} (d_{i+1}, \mathbf{m}_{i+1})$. The set of runs of $\mathfrak{P}$ is denoted $\mathsf{Runs}(\mathfrak{P})$.

For a number k, the run ρ is said to be k-context-bounded (k-CB for short) if for each $c_i = (d_i, \mathbf{m}_i) \in \rho$ and for each $w \in \mathbf{m}_i$, we have $|w|_{D\times D} \leq k$. The set of k-context-bounded runs of $\mathfrak{P}$ is denoted by $\mathsf{Runs}_k(\mathfrak{P})$. In the case of finite runs which reach a certain configuration c, We say a configuration c is *k-reachable* if there is a finite k-CB run ρ ending in c.

3.3 Decision Problems

We study the following decision problems.

▶ **Definition 1.**

- **CB Safety (Global state reachability):**
  Instance: A concurrent program $\mathfrak{P}$, a context-bound k and a global state $d_f \in D$.
  Question: Is there a k-reachable configuration c such that $c.d = d_f$? If so, d_f is said to be k-reachable *(in $\mathfrak{P}$) and* k-unreachable *otherwise.*
- **CB Boundedness:**
  Instance: A concurrent program $\mathfrak{P}$ and a context-bound k.
  Question: Is there an $N \in \mathbb{N}$ such that for every k-reachable configuration c we have $|c.\mathbf{m}| \leq N$? If so, the concurrent program $\mathfrak{P}$ is k-bounded*; otherwise it is* k-unbounded*.*
- **CB Termination:**
  Instance: A concurrent program $\mathfrak{P}$, a context-bound k.
  Question: Is $\mathfrak{P}$ k-terminating, that is, is every k-CB run of $\mathfrak{P}$ finite?

3.4 Orders on Runs and Downclosures

Intuitively, k-safety, k-termination, and k-boundedness are preserved when the multiset of pending threads is "k-lossy": pending threads can get lost and we only consider runs where each thread makes at most k context switches. This loss corresponds to these pending threads never being scheduled by the scheduler. However, if a run demonstrates reachability of a global state, or non-termination, or unboundedness, in the k-lossy version, it corresponds also to a k-CB run in the original problem (and conversely). We make this intuition precise by introducing an ordering on runs and defining the downclosure.

Let $w, w' \in \Sigma D\big(\Sigma \cup (D \times D)\big)^* \big(D \cup (D \times D)\big)$ be words with $w = adw_1e_1w_2e_2 \dots w_le_l$ and $w' = a'd'w'_1e'_1w'_2e'_2 \dots w'_le'_l$, where $a, a' \in \Sigma$, $d, d' \in D$, $e_l, e'_l \in D \cup (D \times D)$, $w_i, w'_j \in \Sigma^*$ for $i, j \in [1, l]$, and $e_i, e'_j \in D \times D$ for $i, j \in [1, l-1]$. We define the state-preserving order $\sqsubseteq_D$ by $w \sqsubseteq_D w'$ iff $a = a'$, $d = d'$, $e_i = e'_i$ for each $i \in [1, l]$, and $w_i \sqsubseteq w'_i$, that is, w_i is a subword of w'_i, for each $i \in [1, l]$. We denote the corresponding notion of state-preserving downclosure under this order by $\Downarrow$. Intuitively, the $\sqsubseteq_D$ relation is a subword ordering on words that preserves the initial letter in Σ and all occurrences of $D \cup (D \times D)$, but potentially loses letters from each segment – that is, newly spawned threads can be lost.

We use the order $\sqsubseteq_D$ to naturally define the order $\preceq_D$ on $\mathbb{M}[\Sigma_D^*]$ by induction: for $\mathbf{m}, \mathbf{m}' \in \mathbb{M}[\Sigma_D^*]$ with $|\mathbf{m}|, |\mathbf{m}'| \geq 1$, we have $\mathbf{m} \preceq_D \mathbf{m}'$ iff there are $\mathbf{n}, \mathbf{n}' \in \mathbb{M}[\Sigma_D^*]$, $w, w' \in \Sigma_D^*$ with $\mathbf{m} = \mathbf{n} \oplus [\![w]\!]$ and $\mathbf{m}' = \mathbf{n}' \oplus [\![w']\!]$ such that $\mathbf{n} \preceq_D \mathbf{n}'$ and $w \sqsubseteq_D w'$. Furthermore, for all $\mathbf{m} \in \mathbb{M}[\Sigma_D^*]$, we have $\emptyset \preceq_D \mathbf{m}$.

We define an order $\trianglelefteq$ on preruns as follows: For preruns $\rho = (e_0, \mathbf{n}_0), w_1, (e_1, \mathbf{n}_1), w_2, \dots$ and $\rho' = (e'_0, \mathbf{n}'_0), w'_1, (e'_1, \mathbf{n}'_1), w'_2, \dots$, we have $\rho \trianglelefteq \rho'$ iff $|\rho| = |\rho'|$, $e_i = e'_i, w_i \sqsubseteq_D w'_i$ and $\mathbf{n}_i \preceq_D \mathbf{n}'_i$ for each $i \geq 0$. The downclosure $\downarrow R$ of a set R of preruns of $\mathfrak{P}$ is defined as $\downarrow R = \{\rho \in \mathsf{Preruns}(\mathfrak{P}) \mid \exists \rho' \in R.\ \rho \trianglelefteq \rho'\}$.

We write $\Downarrow\mathsf{Runs}(\mathfrak{P})$ for the downclosure with respect to $\trianglelefteq$ restricted to valid runs.

Some properties of a concurrent program $\mathfrak{P}$ only depend on the downclosure $\Downarrow\mathsf{Runs}_k(\mathfrak{P})$ of the set $\mathsf{Runs}_k(\mathfrak{P})$ of k-CB runs of the program $\mathfrak{P}$. For these properties, we may transform the program $\mathfrak{P}$ to a program $\Downarrow_k\mathfrak{P}$ such that the latter is easier to analyze but retains the properties of the former.

▶ **Definition 2.** *For a language L_a of a* CP*, let*

$$\Downarrow_k L_a = \left\Downarrow \left(L_a \cap \left(\bigcup_{i=0}^{k} \left(aD(\Sigma^* D \times D)^i \Sigma^* D \right) \right) \right) \right.$$

For any CP $\mathfrak{P} = (D, \Sigma, (L_a)_{a\in\Sigma}, d_0, \mathbf{m}_0)$ *and number k, we define the* CP $\Downarrow_k\mathfrak{P} = (D, \Sigma, (\Downarrow_k L_a)_{a\in\Sigma}, d_0, \mathbf{m}_0)$*. In other words, $\Downarrow_k\mathfrak{P}$ is the program obtained by taking the state-preserving downclosure of those words in L_a which contain at most k context switches.*

Note that, by well-quasi-ordering arguments, for any fixed k, the languages L_a of $\Downarrow_k\mathfrak{P}$ are all *regular*.

▶ **Proposition 3.** *Let $\mathfrak{P} = (D, \Sigma, (L_c)_{c\in\mathfrak{C}}, d_0, \mathbf{m}_0)$ be a concurrent program. Then $\Downarrow\mathsf{Runs}_k(\mathfrak{P}) = \Downarrow\mathsf{Runs}(\Downarrow_k\mathfrak{P})$. In particular,*

1. *For every $d \in D$, $\mathfrak{P}$ can k-reach d if and only if $\Downarrow_k\mathfrak{P}$ can k-reach d.*
2. *$\mathfrak{P}$ is k-terminating if and only if $\Downarrow_k\mathfrak{P}$ is k-terminating.*
3. *$\mathfrak{P}$ is k-bounded if and only if $\Downarrow_k\mathfrak{P}$ is k-bounded.*

Clearly, every run in $\mathsf{Runs}_k(\mathfrak{P})$ is also in $\mathsf{Runs}(\Downarrow_k\mathfrak{P})$. Conversely, we can show by induction on the length of the run that for every run $\rho \in \mathsf{Runs}(\Downarrow_k\mathfrak{P})$ there is a run $\rho' \in \mathsf{Runs}(\mathfrak{P})$ such that $\rho \trianglelefteq \rho'$. The result follows.

4 Decidability Results

We now characterize full trios $\mathcal{C}$ for which decision problems for concurrent programs over $\mathcal{C}$ are decidable. We shall make use of the following decidability results about regular languages.

▶ **Theorem 4.**
1. [28, 10] *CB Safety is decidable for concurrent programs over regular languages.*
2. [28, 15] *CB Boundedness and CB termination are decidable for concurrent programs over regular languages.*

In fact, the above problems are decidable even if there is no bound on the number of context switches. The result in [10] is stated for a model called *Dynamic networks of Concurrent Finite-state Systems* (DCFS), but it is easy to see that there is a polynomial time reduction for the problems of safety, termination and boundedness for CP over regular languages to the corresponding problems for DCFS. The paper [15] shows decidability of CB termination and CB boundedness for the model of dynamic networks of concurrent pushdown systems, of which DCFS is a special case. There is also a simple reduction of these problems to the corresponding results for the model of asynchronous programs [28].

Our first decidability result is the following.

▶ **Theorem 5.** *Let $\mathcal{C}$ be a full trio. The following are equivalent:*
- **(i)** *CB Safety is decidable for concurrent programs over $\mathcal{C}$.*
- **(ii)** *CB Termination is decidable for concurrent programs over $\mathcal{C}$.*
- **(iii)** *Emptiness is decidable for $\mathcal{C}$.*

The implications "(i)⇒(iii)" and The implications "(ii)⇒(iii)" are immediate from corrsponding results for asynchronous programs [40], since context bounded analysis problems generalize the corresponding analysis for asynchronous programs.

Before we prove the next implication, let us introduce a bit of notation. For each $i \in \mathbb{N}$, let R_i be the regular language $R_i = \Sigma D \Sigma^* ((D \times D)\Sigma^*)^i D$, $R'_i = \Sigma D \Sigma^* ((D \times D)\Sigma^*)^i (D \times D)$, for each $l \in \mathbb{N}$ we define $\mathcal{R}_l = \bigcup_{i=0}^{l} (R_i \cup R'_i)$. For any language L and $k \in \mathbb{N}$, the language $L \cap \mathcal{R}_k$ captures those words in L that contain at most k context switches.

For the implication "(iii)⇒(i)", we construct two semidecision procedures (Algorithm 1): the first one searches for regular over-approximations A_a of each language L_a such that the program $\mathfrak{P}'$ obtained by replacing each L_a by the corresponding A_a is safe. We can check whether our current guess for $\mathfrak{P}'$ is safe using Theorem 4. By Proposition 3, we know that in case $\mathfrak{P}$ is safe, then there must exist such a safe regular over-approximation. Concurrently, the second procedure searches for a k-CB run reaching the target global state d which witnesses the negation. Clearly, one of the two procedures must terminate. Note that we use an emptiness check to ensure that our current guess for A_a includes the set $L_a \cap \mathcal{R}_k$.

To show "(iii)⇒(ii)", we need an algorithm for termination of concurrent programs. As in the case of safety, it consists of two semi-decision procedures. The one for termination works just like the one for safety: It enumerates regular over-approximations and checks if one of them terminates. The procedure for non-termination requires some terminology:

Predictions. We will use a notion of *prediction*, which assigns to each configuration $(e, \mathbf{n})$ of a run a multiset of strings that encode not only the *past* of each thread (as is done in $\mathbf{n}$), but also its *future*. To do this, we define the alphabet $\Gamma_D = \Sigma_D \cup \{\#\}$ that extends Σ_D a fresh letter $\#$. We shall encode predictions using strings of the form $au\#v$, which encode a thread with name a, past execution au, and future execution v. Additionally, we extend the order $\preceq_D$ to strings of the form $au\#v$ by treating $\#$ as a letter from $D \times D$ which is to be preserved. Let us make this precise.

Algorithm 1 Checking CB Safety.

Input: Concurrent program $\mathfrak{P} = (D, \Sigma, (L_a)_{a\in\Sigma}, d_0, \mathbf{m}_0)$ over $\mathcal{C}$, context bound $k \in \mathbb{N}$, state $d \in D$

run concurrently

begin `/* find a safe over-approximation */`

foreach *tuple* $(A_a)_{a\in\Sigma}$ *of regular languages* $A_a \subseteq \Sigma^*$ **do**

if $(L_a \cap \mathcal{R}_k) \cap (\Sigma_D^* \setminus A_a) = \emptyset$ *for each* $a \in \Sigma$ **then**

if $\mathfrak{P}' = (D, \Sigma, (A_a)_{a\in\Sigma}, d_0, \mathbf{m}_0)$ *does not* k*-reach* d **then**

return d *is not reachable.*

begin `/* find a run reaching` d `*/`

foreach *prerun* ρ *of* $\mathfrak{P}$ **do**

if ρ *is a* k*-CB run that reaches* d **then**

return d *reachable.*

Suppose ρ is a (finite or infinite) prerun $(e_0, \mathbf{n}_0), w_1, (e_1, \mathbf{n}_1), \ldots$. An *annotation for* ρ is a sequence $\mathbf{f}_0, \mathbf{f}_1, \ldots \in \mathbb{M}[\Gamma_D^*]$ of multisets of strings such that the sequence has the same length as ρ. If ρ is a run, then we say that the annotation $\mathbf{f}_0, \mathbf{f}_1, \ldots$ is a *prediction* if

1. each string occurring in $\mathbf{f}_0, \mathbf{f}_1, \ldots$ is of the form $au\#v$ such that $auv \in \Sigma_D^*$ and $auv \in \mathsf{pref} L_a \cap \big(\Sigma \cup (D \times D)\big)^* \big(D \cup (D \times D)\big)$
2. for each $i \geq 0$, the multisets $\mathbf{n}_i$ and $\mathbf{f}_i$ have the same cardinality and there is a bijection between $\mathbf{n}_i$ and $\mathbf{f}_i$ so that (i) each word au in $\mathbf{n}_i$ is in bijection with some word $au\#v$ in $\mathbf{f}_i$ and (ii) if au is the active thread when going from $(e_i, \mathbf{n}_i)$ to $(e_{i+1}, \mathbf{n}_{i+1})$ and $au\#v$ is its corresponding string $au\#v$ in $\mathbf{f}_i$, then the system executes the next segment in v.

Note that then indeed, for each thread, its string in $\mathbf{n}_i$ records its past spawns, whereas the corresponding string in $\mathbf{f}_i$ contains all its future spawns (and possibly an additional suffix).

Of course, for each (finite or infinite) run, there exists a prediction: Just take the sequence of actions of each thread in the future. Moreover, taking a prefix of both a run and some accompanying prediction will yield a (shorter) run with a shorter prediction.

Self-covering runs. Recall that for each alphabet Θ, we have an embedding relation $\preceq_D$ on the set $\mathbb{M}[\Theta_D^*]$, and in particular on $\mathbb{M}[\Gamma_D^*]$. We say that a finite run $(e_0, \mathbf{n}_0), w_1, (e_1, \mathbf{n}_1), \ldots, w_m, (e_m, \mathbf{n}_m)$, together with a prediction $\mathbf{f}_0, \ldots, \mathbf{f}_m$ is *k-self-covering* if for some $i < m$, we have $e_i = e_m, \mathbf{f}_i \preceq_D \mathbf{f}_m$, and also, all words in $\mathbf{f}_0, \mathbf{f}_1, \ldots$ contain at most k context-switches. As the name suggests, self-covering runs are witnesses for non-termination:

▶ **Lemma 6.** *For every $k \in \mathbb{N}$, a concurrent program has an infinite k-CB run if and only if it has a k-self-covering run.*

Here, it is crucial that for each $k \in \mathbb{N}$, the ordering $\sqsubseteq_D$ is a WQO on the set of words with at most k context-switches (on all of Σ_D^*, $\sqsubseteq_D$ is not a WQO).

We can now decide termination (Algorithm 2): the algorithm either (i) exhibits a k-self-covering run, which shows the existence of a k-bounded infinite run by Lemma 6, or (ii) finds a regular over-approximation that terminates, which means the original program is terminating. We can check termination of the regular over-approximation using Theorem 4. The algorithm also terminates: If there is an infinite k-bounded run, then Lemma 6 yields the existence of a k-self-covering run. Moreover, if the concurrent program does terminate, then Proposition 3 ensures the existence of a terminating regular over-approximation. This concludes our proof of Theorem 5.

Algorithm 2 Checking CB Termination.

Input: Concurrent program $\mathfrak{P} = (D, \Sigma, (L_a)_{a\in\Sigma}, d_0, \mathbf{m}_0)$ over $\mathcal{C}$ and context bound $k \in \mathbb{N}$
run concurrently
  begin `/* find a terminating over-approximation */`
    foreach *tuple* $(A_a)_{a\in\Sigma}$ *of regular languages* $A_a \subseteq \Sigma_D^*$ **do**
      if $(L_a \cap \mathcal{R}_k) \cap (\Sigma^* \setminus A_a) = \emptyset$ *for each* $a \in \Sigma$ **then**
        if $\mathfrak{P}' = (D, \Sigma, (A_a)_{a\in\Sigma}, d_0, \mathbf{m}_0)$ *is k-terminating* **then**
          return $\mathfrak{P}$ *is k-terminating.*
  begin `/* find a self-covering run */`
    foreach *prerun* ρ *of* $\mathfrak{P}$ *and an annotation* σ **do**
      if ρ *with* σ *is a k-self-covering run* **then**
        return $\mathfrak{P}$ *is not k-terminating.*

Our second theorem is as follows.

▶ **Theorem 7.** *Let $\mathcal{C}$ be a full trio. The following are equivalent:*
(i) *CB Boundedness is decidable for concurrent programs over $\mathcal{C}$.*
(ii) *Finiteness is decidable for $\mathcal{C}$.*

The implication "(i)⇒(ii)" follows from the special case of asynchronous programs [40]. It was also observed in [40] that decidability of finiteness for $\mathcal{C}$ implies decidability of emptiness for $\mathcal{C}$. Further, by Theorem 5, we may assume that CB safety is decidable for CP over $\mathcal{C}$.

We now show the implication "(ii)⇒(i)". For a language $L \subseteq \Sigma_D^*$ and $n \in \mathbb{N}$, let $L|_n = L \cap \Sigma_D^{\leq n}$ be the language restricted to strings of length at most n and, in addition, for $k \in \mathbb{N}$, let $L'_a|_n = L_a|_n \cap \mathcal{R}_k$. Moreover, for an alphabet Θ, a language $L \subseteq \Theta^*$, and a word $w \in \Theta^*$, we define the left quotient of L by w as $w^{-1}L := \{u \in \Theta^* \mid wu \in L\}$. Our algorithm is based on the following characterization of unboundedness.

▶ **Lemma 8.** *The program $\mathfrak{P}$ is k-unbounded iff one of the two following conditions hold:*
(P1) *Either there exists some number n such that $\mathfrak{P}_n = (D, \Sigma, (L'_a|_n)_{a\in\Sigma}, d_0, \mathbf{m}_0)$ is unbounded, or*
(P2) *for some $a \in \Sigma$, there exists some word $w \in \mathsf{pref}(L_a)$ ending in a letter $(d, d') \in D \times D$ such that $\mathsf{pref}(w^{-1}L_a) \cap \Sigma^*$ is infinite and there exists a run ρ reaching a configuration c with $w \in c$ and $c.d = d'$.*

Essentially, (P1) captures the case where each thread spawns a finite number of other threads and (P2) the case that there is some reachable configuration at which a single thread can spawn an unbounded number of new threads. The above characterization allows us to implement Algorithm 3, which interleave three semidecision procedures: Checking properties (P1) and (P2) for positive certificates of unboundedness, as well as looking for certificates of boundedness by looking for bounded regular over-approximations. Here we can check boundedness for the latter by Theorem 4. Note that while checking for (P1), it is possible to compute each language $L'_a|_n$ explicitly since these languages are all finite. This is because, given any finite language $F \in \mathcal{C}$ and an explicitly given finite language A, we know $F = A$ iff $F \cap (\Sigma_D^* \setminus A) = \emptyset$ and for all $w \in A$, $F \cap \{w\} \neq \emptyset$, where the first condition checks if $F \subseteq A$ and the second if $A \subseteq F$. Therefore, by enumerating all strings w, we can build A iteratively.

A Remark on Complexity. Our procedures show decidability, but do not provide complexity results. For particular classes of languages, precise complexity bounds are known. For example, CB Safety, CB Termination, and CB Boundedness for concurrent programs over

Algorithm 3 Checking CB Boundedness.

Input: Concurrent program $\mathfrak{P} = (D, \Sigma, (L_a)_{a\in\Sigma}, d_0, \mathbf{m}_0)$ over $\mathcal{C}$ and context bound $k \in \mathbb{N}$
run concurrently
begin `/* (P1): Check if finite under-approximation is unbounded */`
foreach $n \in \mathbb{N}$ **do** `/* Explicitly find strings in` $L'_a|_n$ `*/`
foreach $a \in \Sigma$ **do**
$X_a \leftarrow \emptyset, L'_a|_n \leftarrow L_a \cap \Sigma_D^{\leq n} \cap \mathcal{R}_k$
foreach $w \in \Sigma_D^{\leq n}$ **do**
if $L'_a|_n \cap \{w\} \neq \emptyset$ **then**
$X_a \leftarrow X_a \cup \{w\}$
if $L'_a|_n \cap (\Sigma_D^* \setminus X_a) = \emptyset$ **then**
break
if $\mathfrak{P}_n = (D, \Sigma, (X_a)_{a\in\Sigma}, d_0, \mathbf{m}_0)$ *is unbounded* **then**
return $\mathfrak{P}$ *unbounded.*
begin `/* (P2): Check if unbounded segment can be reached */`
foreach *prerun* ρ *of* $\mathfrak{P}$, $a \in \Sigma$, $w \in aD\Sigma^*(D \times D\Sigma^*)^{\leq k-2}(D \times D) \cup \{a\}$ **do**
if ρ *is a* k*-run that reaches* c *with* $w \in c$, $w = w'(d, d')$ *where* $d' = c.d$, *and* $\mathsf{pref}(w^{-1}L_a) \cap \Sigma^*$ *is infinite* **then**
return $\mathfrak{P}$ *unbounded.*
if ρ *is a* k*-run that reaches* c *with* $w \in c$, $w = a$ *where* $d' = c.d$, *and* $\mathsf{pref}((wd')^{-1}L_a) \cap \Sigma^*$ *is infinite* **then**
return $\mathfrak{P}$ *unbounded.*
begin `/* Find a bounded over-approximation */`
foreach *tuple* $(A_a)_{a\in\Sigma}$ *of regular languages* $A_a \subseteq (a\Sigma_D^* \cap \mathcal{R}_k)$ **do**
if $(L_a \cap \mathcal{R}_k) \cap (\Sigma_D^* \setminus A_a) = \emptyset$ *for each* $a \in \Sigma$ **then**
if $\mathfrak{P}' = (D, \Sigma, (A_a)_{a\in\Sigma}, d_0, \mathbf{m}_0)$ *is bounded* **then**
return $\mathfrak{P}$ *bounded.*

regular languages are all EXPSPACE-complete [28], and over context-free languages are 2EXPSPACE-complete [11, 14]. These bounds use explicit constructions of the downclosure. In particular, our results show decidability of the same problems for concurrent programs over higher-order recursion schemes. However, we do not get an explicit complexity bound. While there is an explicit construction of the downclosure of these languages [53, 30, 20], a precise complexity bound for the construction remains open.

5 Further Results

Other Decision Problems. While we focus on safety, termination, and boundedness, there are decidability results for other properties and other classes of systems. The *fair termination* problem is a variant of termination, where we require that the scheduler is *fair*. Intuitively, a scheduler is fair if it schedules each partially executed thread that is infinitely often ready to execute. Context-bounded fair termination is decidable (but non-elementary) for context-free concurrent programs [15]. The problem is equivalent to Petri net reachability already for asynchronous programs [28]. It is undecidable for indexed languages.

Context-bounded analysis has also been studied for non-regular specifications. Lal et al. [38] showed decidability for context-bounded analysis for a subclass of weighted pushdown systems. Recently, Baumann et al. [13] studied the *context-bounded refinement problem* for

non-regular specifications. In their setting, there is a fixed number of recursive (context-free) threads which also generate a language over a set of events. The specification is given by a Dyck language. They show that checking containment in the specification is coNP-complete, the same complexity as that of context-bounded safety verification, albeit requiring very different techniques. An analogous result was shown for the setting of asynchronous programs, but the complexity is EXPSPACE-complete [12].

Tools and Sequentialization. A practical motivation for studying context-bounded reachability was that, empirically, many bugs in concurrent programs could be found with a small number of context switches. This led to the development of several academic and industrial tools, such as CHESS [44] and CSeq [27]. CHESS incorporated context bounding in an enumerative search. CSeq and several other tools implemented *sequentialization*: a preprocessing step that compiles the original concurrent program into a sequential program that preserves all k-context bounded runs, an idea going back to Lal and Reps [37]. Context-bounding was integrated with other exploration heuristics such as abstract interpretation and partial-order reduction [45, 21, 41].

Context-Bounded Analysis of Related Models. Context-bounding was studied for other models of concurrency, such as parameterized state machines communicating through message-passing over a given topology [18], concurrent queue systems [49], programs over weak memory models [9, 1], abstract models such as valence automata [43], etc. In each case, the notion of "context" has to be refined based on the model.

Similar Restrictions. The theory of context-bounding has inspired other natural bounds in the analysis of concurrent systems. For example, a well-studied restricion is *scope-bounding*: In a *k-scope-bounded* run, there can be an unbounded number of context-switches, but during the time span of a single function call (i.e. between a push and its corresponding pop), there can be at most k interruptions [52]. This covers more executions than context-bounding, which comes at the cost of PSPACE-completeness of safety verification [52]. Scope-boundedness has also been studied in terms of timed systems [4, 17], temporal-logic model-checking [6], resulting formal languages [51], and as an under-approximation for infinite-state systems beyond multi-pushdown systems [48].

Similarly, a *k-phase-bounded* run consists of k phases, in each of which at most one stack is popped [50, 8]. Another variant is *k-stage-bounded* runs: They consist of k stages, each of which allows only one thread to write to the shared memory, whereas the other threads can only read from it [7]. Further restrictions are *ordered multi-pushdown systems* [19, 5] and *delay-bounded scheduling* [26].

Powerful abstract notions of under-approximate analysis (which explain decidability of several concrete restrictions described above) are available in the concepts of *bounded tree-width* [39] and *bounded split-width* [3, 23, 22].

In conclusion, *context-bounding* is an elegant idea that has been very influential both in practice and in theory. In practice, it has been incorporated in several tools for automatic analysis of programs. Theoretically, it has led to a wealth of new models and analysis algorithms. At this point, the theory has marched ahead of implementations: it is an interesting open challenge to see how far the new algorithms can also lead to practical tools.

References

1 Parosh Aziz Abdulla, Mohamed Faouzi Atig, Ahmed Bouajjani, and Tuan Phong Ngo. Context-bounded analysis for POWER. In Axel Legay and Tiziana Margaria, editors, *Tools and Algorithms for the Construction and Analysis of Systems – 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part II*, volume 10206 of *Lecture Notes in Computer Science*, pages 56–74, 2017. doi:10.1007/978-3-662-54580-5_4.

2 Alfred V. Aho. Indexed grammars – An extension of context-free grammars. *J. ACM*, 15(4):647–671, 1968. doi:10.1145/321479.321488.

3 C. Aiswarya, Paul Gastin, and K. Narayan Kumar. Verifying communicating multi-pushdown systems via split-width. In Franck Cassez and Jean-François Raskin, editors, *Automated Technology for Verification and Analysis – 12th International Symposium, ATVA 2014, Sydney, NSW, Australia, November 3–7, 2014, Proceedings*, volume 8837 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2014. doi:10.1007/978-3-319-11936-6_1.

4 S. Akshay, Paul Gastin, Shankara Narayanan Krishna, and Sparsa Roychowdhury. Revisiting underapproximate reachability for multipushdown systems. In *Tools and Algorithms for the Construction and Analysis of Systems – 26th International Conference, TACAS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings, Part I*, volume 12078 of *Lecture Notes in Computer Science*, pages 387–404. Springer, 2020. doi:10.1007/978-3-030-45190-5_21.

5 Mohamed Faouzi Atig, Benedikt Bollig, and Peter Habermehl. Emptiness of ordered multi-pushdown automata is 2ETIME-complete. *Int. J. Found. Comput. Sci.*, 28(8):945–976, 2017. doi:10.1142/S0129054117500332.

6 Mohamed Faouzi Atig, Ahmed Bouajjani, K. Narayan Kumar, and Prakash Saivasan. Linear-time model-checking for multithreaded programs under scope-bounding. In Supratik Chakraborty and Madhavan Mukund, editors, *Automated Technology for Verification and Analysis – 10th International Symposium, ATVA 2012, Thiruvananthapuram, India, October 3–6, 2012. Proceedings*, volume 7561 of *Lecture Notes in Computer Science*, pages 152–166. Springer, 2012. doi:10.1007/978-3-642-33386-6_13.

7 Mohamed Faouzi Atig, Ahmed Bouajjani, K. Narayan Kumar, and Prakash Saivasan. On bounded reachability analysis of shared memory systems. In Venkatesh Raman and S. P. Suresh, editors, *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15–17, 2014, New Delhi, India*, volume 29 of *LIPIcs*, pages 611–623. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014. doi:10.4230/LIPIcs.FSTTCS.2014.611.

8 Mohamed Faouzi Atig, Ahmed Bouajjani, K. Narayan Kumar, and Prakash Saivasan. Parity games on bounded phase multi-pushdown systems. In Amr El Abbadi and Benoît Garbinato, editors, *Networked Systems – 5th International Conference, NETYS 2017, Marrakech, Morocco, May 17–19, 2017, Proceedings*, volume 10299 of *Lecture Notes in Computer Science*, pages 272–287, 2017. doi:10.1007/978-3-319-59647-1_21.

9 Mohamed Faouzi Atig, Ahmed Bouajjani, and Gennaro Parlato. Context-bounded analysis of TSO systems. In Saddek Bensalem, Yassine Lakhnech, and Axel Legay, editors, *From Programs to Systems. The Systems perspective in Computing – ETAPS Workshop, FPS 2014, in Honor of Joseph Sifakis, Grenoble, France, April 6, 2014. Proceedings*, volume 8415 of *Lecture Notes in Computer Science*, pages 21–38. Springer, 2014. doi:10.1007/978-3-642-54848-2_2.

10 Mohamed Faouzi Atig, Ahmed Bouajjani, and Shaz Qadeer. Context-bounded analysis for concurrent programs with dynamic creation of threads. In *Proceedings of TACAS 2009*, pages 107–123, 2009.

11 Mohamed Faouzi Atig, Ahmed Bouajjani, and Shaz Qadeer. Context-bounded analysis for concurrent programs with dynamic creation of threads. *Log. Methods Comput. Sci.*, 7(4), 2011. doi:10.2168/LMCS-7(4:4)2011.

12 Pascal Baumann, Moses Ganardi, Rupak Majumdar, Ramanathan S. Thinniyam, and Georg Zetzsche. Checking refinement of asynchronous programs against context-free specifications. In *ICALP '23*, LIPIcs. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.

13 Pascal Baumann, Moses Ganardi, Rupak Majumdar, Ramanathan S. Thinniyam, and Georg Zetzsche. Context-bounded verification of context-free specifications. *Proc. ACM Program. Lang.*, 7(POPL):2141–2170, 2023. doi:10.1145/3571266.

14 Pascal Baumann, Rupak Majumdar, Ramanathan S. Thinniyam, and Georg Zetzsche. The complexity of bounded context switching with dynamic thread creation. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8–11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 111:1–111:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.111.

15 Pascal Baumann, Rupak Majumdar, Ramanathan S. Thinniyam, and Georg Zetzsche. Context-bounded verification of liveness properties for multithreaded shared-memory programs. *Proc. ACM Program. Lang.*, 5(POPL):1–31, 2021. doi:10.1145/3434325.

16 Jean Berstel. *Transductions and context-free languages*. Springer-Verlag, 1979.

17 Devendra Bhave, Shankara Narayanan Krishna, Ramchandra Phawade, and Ashutosh Trivedi. On timed scope-bounded context-sensitive languages. In *Developments in Language Theory – 23rd International Conference, DLT 2019, Warsaw, Poland, August 5–9, 2019, Proceedings*, volume 11647 of *Lecture Notes in Computer Science*, pages 168–181. Springer, 2019. doi:10.1007/978-3-030-24886-4_12.

18 Benedikt Bollig, Paul Gastin, and Jana Schubert. Parameterized verification of communicating automata under context bounds. In Joël Ouaknine, Igor Potapov, and James Worrell, editors, *Reachability Problems – 8th International Workshop, RP 2014, Oxford, UK, September 22–24, 2014. Proceedings*, volume 8762 of *Lecture Notes in Computer Science*, pages 45–57. Springer, 2014. doi:10.1007/978-3-319-11439-2_4.

19 Luca Breveglieri, Alessandra Cherubini, Claudio Citrini, and Stefano Crespi-Reghizzi. Multi-push-down languages and grammars. *Int. J. Found. Comput. Sci.*, 7(3):253–292, 1996. doi:10.1142/S0129054196000191.

20 Lorenzo Clemente, Paweł Parys, Sylvain Salvati, and Igor Walukiewicz. The diagonal problem for higher-order recursion schemes is decidable. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5–8, 2016*, pages 96–105. ACM, 2016. doi:10.1145/2933575.2934527.

21 Katherine E. Coons, Madan Musuvathi, and Kathryn S. McKinley. Bounded partial-order reduction. In Antony L. Hosking, Patrick Th. Eugster, and Cristina V. Lopes, editors, *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications, OOPSLA 2013, part of SPLASH 2013, Indianapolis, IN, USA, October 26–31, 2013*, pages 833–848. ACM, 2013. doi:10.1145/2509136.2509556.

22 Aiswarya Cyriac. *Verification of communicating recursive programs via split-width. (Vérification de programmes récursifs et communicants via split-width)*. PhD thesis, École normale supérieure de Cachan, France, 2014. URL: https://tel.archives-ouvertes.fr/tel-01015561.

23 Aiswarya Cyriac, Paul Gastin, and K. Narayan Kumar. MSO decidability of multi-pushdown systems via split-width. In Maciej Koutny and Irek Ulidowski, editors, *CONCUR 2012 – Concurrency Theory – 23rd International Conference, CONCUR 2012, Newcastle upon Tyne, UK, September 4–7, 2012. Proceedings*, volume 7454 of *Lecture Notes in Computer Science*, pages 547–561. Springer, 2012. doi:10.1007/978-3-642-32940-1_38.

24 Werner Damm. The IO-and OI-hierarchies. *Theoretical Computer Science*, 20(2):95–207, 1982.

25 Werner Damm and Andreas Goerdt. An automata-theoretical characterization of the OI-hierarchy. *Information and Control*, 71(1):1–32, 1986.

26 Michael Emmi, Shaz Qadeer, and Zvonimir Rakamaric. Delay-bounded scheduling. In Thomas Ball and Mooly Sagiv, editors, *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, TX, USA, January 26–28, 2011*, pages 411–422. ACM, 2011. doi:10.1145/1926385.1926432.

27 Bernd Fischer, Omar Inverso, and Gennaro Parlato. Cseq: A sequentialization tool for C – (competition contribution). In Nir Piterman and Scott A. Smolka, editors, *Tools and Algorithms for the Construction and Analysis of Systems – 19th International Conference, TACAS 2013*, volume 7795 of *Lecture Notes in Computer Science*, pages 616–618. Springer, 2013. doi:10.1007/978-3-642-36742-7_46.

28 Pierre Ganty and Rupak Majumdar. Algorithmic verification of asynchronous programs. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 34(1):6, 2012.

29 Sheila A. Greibach. Remarks on blind and partially blind one-way multicounter machines. *Theoretical Computer Science*, 7(3):311–324, 1978. doi:10.1016/0304-3975(78)90020-8.

30 Matthew Hague, Jonathan Kochems, and C.-H. Luke Ong. Unboundedness and downward closures of higher-order pushdown automata. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20–22, 2016*, pages 151–163. ACM, 2016. doi:10.1145/2837614.2837627.

31 Matthew Hague, Andrzej S. Murawski, C.-H. Luke Ong, and Olivier Serre. Collapsible pushdown automata and recursion schemes. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24–27 June 2008, Pittsburgh, PA, USA*, pages 452–461, 2008. doi:10.1109/LICS.2008.34.

32 Leonard H Haines. On free monoids partially ordered by embedding. *Journal of Combinatorial Theory*, 6(1):94–98, 1969.

33 John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation, 3rd Edition*. Pearson international edition. Addison-Wesley, 2007.

34 Omar Inverso, Ermenegildo Tomasco, Bernd Fischer, Salvatore La Torre, and Gennaro Parlato. Bounded verification of multi-threaded programs via lazy sequentialization. *ACM Trans. Program. Lang. Syst.*, 44(1):1:1–1:50, 2022. doi:10.1145/3478536.

35 Matthias Jantzen. On the hierarchy of Petri net languages. *RAIRO – Theoretical Informatics and Applications – Informatique Théorique et Applications*, 13(1):19–30, 1979. URL: http://www.numdam.org/item?id=ITA_1979__13_1_19_0.

36 Salvatore La Torre, P. Madhusudan, and Gennaro Parlato. Reducing context-bounded concurrent reachability to sequential reachability. In Ahmed Bouajjani and Oded Maler, editors, *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 – July 2, 2009. Proceedings*, volume 5643 of *Lecture Notes in Computer Science*, pages 477–492. Springer, 2009. doi:10.1007/978-3-642-02658-4_36.

37 Akash Lal and Thomas W. Reps. Reducing concurrent analysis under a context bound to sequential analysis. *Formal Methods Syst. Des.*, 35(1):73–97, 2009. doi:10.1007/s10703-009-0078-9.

38 Akash Lal, Tayssir Touili, Nicholas Kidd, and Thomas W. Reps. Interprocedural analysis of concurrent programs under a context bound. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008*, volume 4963 of *Lecture Notes in Computer Science*, pages 282–298. Springer, 2008. doi:10.1007/978-3-540-78800-3_20.

39 P. Madhusudan and Gennaro Parlato. The tree width of auxiliary storage. In Thomas Ball and Mooly Sagiv, editors, *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, TX, USA, January 26–28, 2011*, pages 283–294. ACM, 2011. doi:10.1145/1926385.1926419.

40 Rupak Majumdar, Ramanathan S. Thinniyam, and Georg Zetzsche. General decidability results for asynchronous shared-memory programs: Higher-order and beyond. *Log. Methods Comput. Sci.*, 18(4), 2022. doi:10.46298/lmcs-18(4:2)2022.

41 Iason Marmanis, Michalis Kokologiannakis, and Viktor Vafeiadis. Reconciling preemption bounding with DPOR. In Sriram Sankaranarayanan and Natasha Sharygina, editors, *Tools and Algorithms for the Construction and Analysis of Systems – 29th International Conference, TACAS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Paris, France, April 22–27, 2023, Proceedings, Part I*, volume 13993 of *Lecture Notes in Computer Science*, pages 85–104. Springer, 2023. doi:10.1007/978-3-031-30823-9_5.

42 AN Maslov. The hierarchy of indexed languages of an arbitrary level. *Doklady Akademii Nauk*, 217(5):1013–1016, 1974.

43 Roland Meyer, Sebastian Muskalla, and Georg Zetzsche. Bounded context switching for valence systems. In Sven Schewe and Lijun Zhang, editors, *29th International Conference on Concurrency Theory, CONCUR 2018, September 4–7, 2018, Beijing, China*, volume 118 of *LIPIcs*, pages 12:1–12:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CONCUR.2018.12.

44 Madanlal Musuvathi and Shaz Qadeer. Iterative context bounding for systematic testing of multithreaded programs. In *Proceedings of the ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation, PLDI 2007, San Diego, CA, USA, June 10–13, 2007*, pages 446–455. ACM, 2007. doi:10.1145/1250734.1250785.

45 Truc L. Nguyen, Bernd Fischer, Salvatore La Torre, and Gennaro Parlato. Concurrent program verification with lazy sequentialization and interval analysis. In Amr El Abbadi and Benoît Garbinato, editors, *Networked Systems – 5th International Conference, NETYS 2017, Marrakech, Morocco, May 17–19, 2017, Proceedings*, volume 10299 of *Lecture Notes in Computer Science*, pages 255–271, 2017. doi:10.1007/978-3-319-59647-1_20.

46 Shaz Qadeer and Jakob Rehof. Context-bounded model checking of concurrent software. In Nicolas Halbwachs and Lenore D. Zuck, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 11th International Conference, TACAS 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005, Proceedings*, volume 3440 of *Lecture Notes in Computer Science*, pages 93–107. Springer, 2005. doi:10.1007/978-3-540-31980-1_7.

47 Koushik Sen and Mahesh Viswanathan. Model checking multithreaded programs with asynchronous atomic methods. In *CAV '06: Proc. 18th Int. Conf. on Computer Aided Verification*, volume 4144 of *LNCS*, pages 300–314. Springer, 2006.

48 Aneesh K. Shetty, Shankara Narayanan Krishna, and Georg Zetzsche. Scope-bounded reachability in valence systems. In Serge Haddad and Daniele Varacca, editors, *32nd International Conference on Concurrency Theory, CONCUR 2021, August 24–27, 2021, Virtual Conference*, volume 203 of *LIPIcs*, pages 29:1–29:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CONCUR.2021.29.

49 Salvatore La Torre, P. Madhusudan, and Gennaro Parlato. Context-bounded analysis of concurrent queue systems. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29 – April 6, 2008. Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, pages 299–314. Springer, 2008. doi:10.1007/978-3-540-78800-3_21.

50 Salvatore La Torre, Parthasarathy Madhusudan, and Gennaro Parlato. A robust class of context-sensitive languages. In *22nd IEEE Symposium on Logic in Computer Science (LICS 2007), 10–12 July 2007, Wroclaw, Poland, Proceedings*, pages 161–170. IEEE Computer Society, 2007. doi:10.1109/LICS.2007.9.

51 Salvatore La Torre, Margherita Napoli, and Gennaro Parlato. Scope-bounded pushdown languages. *Int. J. Found. Comput. Sci.*, 27(2):215–234, 2016. doi:10.1142/S0129054116400074.

52 Salvatore La Torre, Margherita Napoli, and Gennaro Parlato. Reachability of scope-bounded multistack pushdown systems. *Inf. Comput.*, 275:104588, 2020. doi:10.1016/j.ic.2020.104588.

53 Georg Zetzsche. An approach to computing downward closures. In *ICALP 2015*, volume 9135, pages 440–451. Springer, 2015. Full version: arXiv:1503.01068.

Quantum Codes, Local Testability and Interactive Proofs: State of the Art and Open Questions

Thomas Vidick ✉
Weizmann Institute of Science, Rehovot, Israel
California Institute of Technology, Pasadena, CA, USA

Abstract

The study of multiprover interactive proof systems, of locally testable codes, and of property testing are deeply linked, conceptually if not formally, through their role in the proof of the PCP theorem in complexity theory. Recently there has been substantial progress on an analogous research programme in quantum complexity theory. Two years ago we characterized the power of multiprover interactive proof systems with provers sharing entanglement, showing that MIP* =RE [4], a hugely surprising increase in power from the classical result MIP=NEXP of [2]. The following year Panteleev and Kalachev gave the first construction of quantum low-density parity-check codes (QLDPC) [5], thus marking a major step towards the possible realization of good quantum locally testable codes – the classical analogue of which was only constructed quite recently [3]. And finally, less than a year ago Anshu, Breuckmann and Nirkhe used facts evidenced in the construction of good decoders for the new QLDPC codes to resolve the NLTS conjecture [1], widely viewed as a crucial step on the way to a possible quantum PCP theorem.

In the talk I will survey these results, making an effort to motivate and present them to the non-expert. I will explain the connections between them and point to where, in my opinion, our understanding is currently lacking. Along the way I will highlight a number of open problems whose resolution could lead to further progress on one of the most important research programmes in quantum complexity theory.

2012 ACM Subject Classification Theory of computation → Quantum complexity theory

Keywords and phrases quantum interactive proofs, quantum codes

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.4

Category Invited Talk

References

1 Anurag Anshu, Nikolas Breuckmann, and Chinmay Nirkhe. NLTS Hamiltonians from good quantum codes. *arXiv preprint*, 2022. arXiv:2206.13228.
2 László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational complexity*, 1:3–40, 1991.
3 Irit Dinur, Shai Evra, Ron Livne, Alexander Lubotzky, and Shahar Mozes. Locally testable codes with constant rate, distance, and locality. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 357–374, 2022.
4 Zhengfeng Ji, Anand Natarajan, Thomas Vidick, John Wright, and Henry Yuen. MIP*=RE. arxiv e-prints, page. *arXiv preprint*, 2020. arXiv:2001.04383.
5 Pavel Panteleev and Gleb Kalachev. Asymptotically good quantum and locally testable classical LDPC codes. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 375–388, 2022.

The Skolem Landscape

James Worrell ✉
Department of Computer Science, University of Oxford, UK

Abstract

The Skolem Problem asks to determine whether a given integer linear recurrence sequence (LRS) has a zero term. This decision problem arises within a number of different topics in computer science, including loop termination, weighted automata, formal power series, and probabilistic model checking, among many other examples. Decidability of the problem is notoriously open, despite having been the subject of sustained interest over several decades [2]. More specifically, the problem is known to be decidable for recurrences of order at most 4 – a result obtained some 40 years ago [4, 5] – while decidability is open already for recurrences of order 5.

In this talk we take a wide-ranging view of the Skolem Problem. We survey its history and context, starting with the theorem of Skolem-Mahler-Lech characterising the set of zeros of a LRS over fields of characteristic zero. Here we explain the non-effective nature of the existing proofs of the theorem. Among modern developments, we overview versions of the Skolem-Mahler-Lech theorem for non-linear recurrences and for fields of non-zero characteristic. We also describe two recent directions of progress toward showing decidability of the Skolem Problem subject to classical number theoretic conjectures.

The first new development concerns a recent algorithm [1] that decides the problem on the class of simple LRS (those with simple characteristic roots) subject to two classical conjectures about the exponential function. The algorithm is self-certifying: its output comes with a certificate of correctness that can be checked unconditionally. The two conjectures alluded to above are required for the proof of termination of the algorithm.

A second new development concerns the notion of Universal Skolem Set [3]: a recursive set $\mathcal{S}$ of positive integers such that it is decidable whether a given non-degenerate linear recurrence sequence has a zero in $\mathcal{S}$. Decidability of the Skolem Problem is equivalent to the assertion that $\mathbb{N}$ is a Universal Skolem Set. In lieu of this one can ask whether there exists a Universal Skolem Set of density one. We will present a recent a construction of a Universal Skolem Set that has positive density unconditionally and has density one subject to the Bateman-Horn conjecture in number theory. The latter is a far-reaching generalisation of Hardy and Littlewood's twin primes conjecture.

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics; Computing methodologies → Algebraic algorithms

Keywords and phrases Automata, Formal Languages, Linear Recurrence Sequences

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.5

Category Invited Talk

Funding *James Worrell*: Supported by UKRI Frontier Research Grant EP/X033813/1.

References

1 Y. Bilu, F. Luca, J. Nieuwveld, J. Ouaknine, D. Purser, and J. Worrell. Skolem meets Schanuel. In *47th International Symposium on Mathematical Foundations of Computer Science*, volume 241 of *LIPIcs*, pages 20:1–20:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

2 V. Halava, T. Harju, M. Hirvensalo, and J. Karhumäki. Skolem's problem – On the border between decidability and undecidability. Technical report, Turku Centre for Computer Science, 2005.

3 F. Luca, J. Ouaknine, and J. Worrell. A Universal Skolem Set of positive lower density. In *47th International Symposium on Mathematical Foundations of Computer Science*, volume 241 of *LIPIcs*, pages 73:1–73:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

4 M. Mignotte, T. Shorey, and R. Tijdeman. The distance between terms of an algebraic recurrence sequence. *Journal für die Reine und Angewandte Mathematik*, pages 63–76, 1984.
5 N. Vereshchagin. Occurrence of zero in a linear recursive sequence. *Mathematical notes of the Academy of Sciences of the USSR*, 38(2):609–615, August 1985.

Optimal Decremental Connectivity in Non-Sparse Graphs

Anders Aamand
MIT, Cambridge, MA, USA

Adam Karczmarz
University of Warsaw, Poland
IDEAS NCBR, Warsaw, Poland

Jakub Łącki
Google Research, New York, NY,USA

Nikos Parotsidis
Google Research, Zürich, Switzerland

Peter M. R. Rasmussen
BARC, University of Copenhagen, Denmark

Mikkel Thorup
BARC, University of Copenhagen, Denmark

Abstract

We present a dynamic algorithm for maintaining the connected and 2-edge-connected components in an undirected graph subject to edge deletions. The algorithm is Monte-Carlo randomized and processes any sequence of edge deletions in $O(m + n \operatorname{poly} \log n)$ total time. Interspersed with the deletions, it can answer queries whether any two given vertices currently belong to the same (2-edge-)connected component in constant time. Our result is based on a general Monte-Carlo randomized reduction from decremental c-edge-connectivity to a variant of fully-dynamic c-edge-connectivity on a sparse graph.

For non-sparse graphs with $\Omega(n \operatorname{poly} \log n)$ edges, our connectivity and 2-edge-connectivity algorithms handle all deletions in optimal linear total time, using existing algorithms for the respective fully-dynamic problems. This improves upon an $O(m \log(n^2/m) + n \operatorname{poly} \log n)$-time algorithm of Thorup [J.Alg. 1999], which runs in linear time only for graphs with $\Omega(n^2)$ edges.

Our constant amortized cost for edge deletions in decremental connectivity in non-sparse graphs should be contrasted with an $\Omega(\log n / \log \log n)$ worst-case time lower bound in the decremental setting [Alstrup, Husfeldt, and Rauhe FOCS'98] as well as an $\Omega(\log n)$ amortized time lower-bound in the fully-dynamic setting [Patrascu and Demaine STOC'04].

2012 ACM Subject Classification Theory of computation → Dynamic graph algorithms; Mathematics of computing → Paths and connectivity problems; Mathematics of computing → Graph algorithms

Keywords and phrases decremental connectivity, dynamic connectivity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.6

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2111.09376

Funding *Anders Aamand*: Supported by Thorup's VILLUM Investigator Grant 16582 and by a DFF-International Postdoc Grant 0164-00022B from the Independent Research Fund Denmark.
Adam Karczmarz: Partially supported by the ERC CoG grant TUgbOAT no 772346.
Peter M. R. Rasmussen: Supported by Thorup's VILLUM Investigator Grant 16582.
Mikkel Thorup: Supported by Investigator Grant 16582, Basic Algorithms Research Copenhagen (BARC), from the VILLUM Foundation.

1 Introduction

In this paper, we present Monte Carlo randomized decremental dynamic algorithms for maintaining the connected and 2-edge-connected components in an undirected graph subject to edge deletions. Starting from a graph with n vertices and m edges, the algorithm can process any sequence of edge deletions in $O(m + n \operatorname{polylog} n)$ total time while answering queries whether a pair of vertices is currently in the same (2-edge-)connected component. Each query is answered in constant time. The algorithm for decremental 2-edge-connectivity additionally reports all bridges as they appear.

Putting our results in perspective, we say a graph is *non-sparse* if it has $n \log^{\omega(1)} n$ edges. Large areas of algorithmic research are devoted to non-sparse graphs, e.g., the generic goal of sparsifying graphs to $O(n \operatorname{polylog} n)$ edges [6], or semi-streaming algorithms that aim to sketch graphs using $O(n \operatorname{polylog} n)$ space [11]. Our result states that for dynamic connectivity and 2-edge-connectivity, we can get down to amortized constant time per edge deletion if the initial input graph is non-sparse. Prior to this work, such a result was only known in the case where the initial input graph is very dense with $\Omega(n^2)$ edges, and in the case of some special classes of sparse graphs.

Achieving constant update and query time is generally the ideal target in data structures. What makes amortized constant time for decremental connectivity particularly interesting is that the most closely related problems have near-logarithmic cell-probe lower bounds. This concerns the problem of getting worst-case time bounds or getting a fully-dynamic algorithm (supporting both insertions and deletions of edges). The decremental setting and the fact that we allow for amortization is therefore just enough assumptions to barely push us into the world of constant update and query time (removing any of these assumptions, the polylogarithmic lower bounds would kick in) and as such, our result draws a fine line between the possible and the impossible. We shall discuss this further with precise references in Section 1.1. It is worth noting that for some dynamic graph problems related to maintaining (approximate) maximum matchings and colorings, constant amortized update bounds have been shown, see, e.g., [7, 20, 21, 37].

Our algorithms are Monte Carlo randomized and answer all queries correctly with high probability[1]. We note that since the correct answer to each query is *uniquely* determined from the input, the algorithms work against adaptive adversaries, that is, each deleted edge may depend on previous answers to queries and (in the case of decremental 2-edge-connectivity) on the alleged bridges reported by the algorithm[2]

Furthermore, our algorithms offer a *self-check* capability. At the end, after all updates and queries have been processed online, each algorithm can deterministically check if it might have made a mistake. If the self-check passes, it is *guaranteed* that no incorrect answer was given. Otherwise, the algorithm *may have* made a mistake. Given the self-check is deterministic, the probability that the self-check passes following the execution of the algorithm only

[1] We define high probability as probability $1 - O(n^{-\gamma})$ for any given γ.

[2] To be precise, with unique correct answers, for any adaptive adversary $\mathcal{A}_{\text{ad}}$, there is a non-adaptive adversary $\mathcal{A}_{\text{non-ad}}$ which provides the same sequence of edge deletions up to the first point in time where the algorithm potentially reports an incorrect answer. $\mathcal{A}_{\text{non-ad}}$ is simply defined to provide the same edge-deletions as $\mathcal{A}_{\text{ad}}$ would conditioned on it receiving the *unique* correct answers to every query. Intuitively, the adaptivity of the adversary only becomes relevant once the algorithm has already made a mistake. Illustrating the issue of non-uniqueness in the case of decremental connectivity, suppose we augmented our algorithm to report a (non-unique) *path* between queried pairs of vertices in the same component. The choice of path could reveal information about the random bits employed by our algorithm and this could be very problematic if $\mathcal{A}_{\text{ad}}$ decided to delete the reported path edges.

depends on the correctness of the algorithm execution. However, as we show in the following, the self-check passes with high probability. This feature implies that we can obtain Las Vegas algorithms for certain *non-dynamic* problems whose solutions employ decremental (2-edge-)connectivity algorithms as subroutines: we simply repeat trying to solve the static problem from scratch, each time with new random bits, until the final self-check is passed. With high probability, we are done already after the first round. A nice concrete example is the algorithm of Gabow, Kaplan, and Tarjan [15] for the static problem of deciding if a graph has a unique perfect matching. The algorithm uses a decremental 2-edge-connectivity algorithm as a subroutine. With our decremental 2-edge-connectivity algorithm, repeating until the self-check is passed, we obtain a Las Vegas algorithm for the unique perfect matching problem that is always correct, and which terminates in $O(m + n \operatorname{polylog} n)$ time with high probability.

The tradition of looking for linear time algorithms for non-sparse graphs goes back at least to Fibonacci heaps, which can be used for solving single source shortest paths in $O(m + n \log n)$ time [14]. Our results show that another fundamental graph problem can be solved in linear time in the non-sparse case.

The previous best time bounds for the decremental connectivity and 2-edge-connectivity problems were provided by Thorup [39]. His algorithms run in $O(m \log(n^2/m) + n \operatorname{polylog} n)$ total time. This is amortized constant time per edge deletion only for very dense graphs starting with $\Omega(n^2)$ edges. For graphs with $O(n^{1.99})$ edges, this is $O(\log n)$ amortized time per edge deletion.

Both our algorithm and the previous one by Thorup are based on a general reduction from decremental c-edge-connectivity to fully-dynamic c-edge-connectivity on a sparse c-certificate graph with $\tilde{O}(cn)$ updates.

The contribution of this paper is a new type of sparse c-certificate that is much more efficient to maintain during edge deletions, reducing amortized time per deletion from $O(\log(n^2/m))$ to the optimal $O(1)$. We hope that this new sparse c-certificate will inspire other applications. We shall discuss it further in Section 2.

It should be noted that [39] used Las Vegas randomization, that is, correctness was guaranteed, but the running time bound only held with high probability. Our algorithms are Monte Carlo randomized, but offer the final self-check. Another difference is that our new algorithms need only a polylogarithmic number of random bits, whereas the ones from [39] used $\Theta(m)$ random bits.

We will now give a more detailed discussion of our results in the context of related work.

1.1 Connectivity

Dynamic connectivity is the most fundamental dynamic graph problem. The fully dynamic version has been extensively studied [8, 9, 12, 22, 23, 26, 28, 31, 34, 35, 36, 40, 42, 43] from both the lower and upper bound perspective, even though close to optimal amortized update bounds have been known since the 90s [22, 23, 40]. Currently, the best known amortized update time bounds are $O(\log^2 n / \log\log n)$ deterministic [43] and $O(\log n \cdot (\log\log n)^2)$ expected time [26].

Note that Thorup's $O(\log(n^2/m))$ bound for decremental connectivity is essentially only a $(\log\log n)^2$ factor better than the latter of these bounds for fully-dynamic connectivity, while our new bound brings the decremental cost down to a constant (for non-sparse graphs). Getting down to a constant is particularly interesting when we compare with related lower bounds as discussed below.

Connectivity Lower Bounds. Our result implies that decremental connectivity is provably easier than fully-dynamic connectivity for a wide range of graph densities. Specifically, let t_u be the update time of a fully dynamic connectivity algorithm and let t_q be its query time. Pătraşcu and Demaine [35] showed a lower bound of $\Omega(\log n)$ on $\max(t_u, t_q)$ in the cell-probe model. Pătraşcu and Thorup [36] also showed that $t_u = o(\log n)$ implies $t_q = \Omega(n^{1-o(1)})$. These lower bounds hold for all graph densities and allow for both amortization and randomization. As a result, no fully-dynamic connectivity algorithm can answer connectivity queries in constant time and have an amortized update time of $o(\log n)$.

In sharp contrast, assuming that $m = \Omega(n \operatorname{poly} \log n)$ edges are deleted, our algorithm shows that one can solve decremental connectivity handling both queries and updates in constant amortized time.

We note that such a result is possible only because we allow for amortization, as any decremental connectivity algorithm with *worst-case* update time $O(\operatorname{polylog} n)$ must have worst-case query time $\Omega\left(\frac{\log n}{\log \log n}\right)$ [3]. This lower bound holds even for trees supporting restricted connectivity queries of the form "are u and v connected?" for a fixed "root" u. This lower bound also holds for dense graphs, as we can always add a large static clique to the problem.

An optimal incremental connectivity algorithm has been known for over 40 years. Namely, to handle $m \geq n$ edge insertion and q connectivity queries, one can use the union-find data structure [38] with $n - 1$ unions and $2(m + q)$ finds. The total running time is $\Theta((m+q)\alpha((m+q), n))$, which is linear for all but very sparse graphs (since $\alpha(\Omega(n \log n), n) = O(1)$). It was later shown that this running time is optimal for incremental connectivity [13]. Interestingly, incremental connectivity can be solved in optimal linear time in the case of forests provided that the final shape of the forest is known in advance [16].

Similarly to the decremental case, one cannot hope to obtain an analogous result with a worst-case update time in the incremental setting: Pătraşcu and Thorup [36] showed that any incremental connectivity data structure with $o\left(\frac{\log n}{\log \log n}\right)$ worst-case update time must have worst-case $\Omega(n^{1-o(1)})$ query time in the cell-probe model.

Other cases of optimal decremental connectivity. There is much previous work on cases where decremental connectivity can be supported in $O(m)$ total time. Alstrup, Secher, and Spork [5] showed that decremental connectivity can be solved in optimal $O(m)$ total time on forests, answering queries in $O(1)$ time.[3] This was later extended to other classes of sparse graphs: planar graphs [32], and minor-free graphs [24]. All these special graph classes are sparse with $m = O(n)$ edges.

For general graphs, we only have the previously mentioned work by Thorup [39], yielding a total running time of $O(m)$ for very dense graphs with $m = \Omega(n^2)$ edges. We now obtain the same linear time bound for all non-sparse graphs with $m = \Omega(n \operatorname{poly} \log n)$ edges.

1.2 General reduction for c-edge-connectivity

Our algorithm for decremental connectivity is based on a general randomized reduction from decremental c-edge-connectivity (assuming all m edges are deleted) to fully-dynamic c-edge-connectivity on a sparse graph with $\tilde{O}(cn)$ updates. The reduction has a polylogarithmic cost per vertex as well as a constant cost per edge. The previous decremental connectivity

[3] The general *word encoding* trick behind [5, 16] that brings the update time to amortized constant has been even shown to have practical relevance [4].

algorithm of Thorup [39] was also based on such a general reduction, but the cost per edge was $O(\log(n^2/m))$ which is $O(1)$ only for very dense graphs with $m = \Omega(n^2)$. Below we will describe the format of the reductions in more detail.

Because there are different notions of c-edge-connectivity, we first need to clarify our definitions. We say that two vertices u, v are c-edge-connected iff there exist c edge-disjoint paths between u and v in G. It is known that c-edge-connectivity is an equivalence relation; we call its classes the *c-edge-connected classes*. However, for $c \geq 3$, a c-edge-connected class may induce a subgraph of G which is not connected, so it also makes sense to consider *c-edge-connected components*, i.e., the maximal c-edge-connected induced subgraphs of G.[4] It is important to note that the c-edge-connected components and the c-edge-connected classes are *uniquely defined* and both induce a natural partition of the vertices of the underlying graph. Moreover, each c-edge-connected component of G is a subset of some c-edge-connected class of G. For $c = 1, 2$, the c-edge-connected classes are c-edge-connected, so the two notions coincide. To illustrate the difference, let us fix $c \geq 3$ and consider a graph with $c + 2$ vertices $v_s, v_t, v_1, \ldots, v_c$ and edges $\{v_s, v_t\} \times \{v_1, \ldots, v_c\}$; while all c-edge-connected components in this graph are singletons, there is one c-edge-connected class, which is not a singleton, namely $\{v_s, v_t\}$.

We define a *c-certificate* of G to be a subgraph H of G that contains all edges not in c-edge-connected components, and also contains a c-edge-connected subgraph of each c-edge-connected component. Both Thorup's and our reduction maintains a c-certificate H of G. Then, for any $c' \leq c$, we have that the c'-edge-connected equivalence classes and the c'-edge-connected components are the same in G and H. As the edges from G are deleted, we maintain a c-certificate with $\widetilde{O}(cn)$ edges undergoing only $\widetilde{O}(cn)$ edge insertions and deletions in total.

The (uniquely defined) c-edge-connected components of a graph can be found using the following algorithm: while the graph contains a cut of size at most $c - 1$, remove all edges of this cut. For the reductions, we need algorithms that can help us in this process. We therefore define the *fully dynamic c-edge-cut problem* as follows. Suppose a graph G is subject to edge insertions and/or deletions. Then, a fully dynamic c-edge-cut data structure should report, after each update, some edge e that belongs to some cut of size less than c. A typical application of such a data structure is to repeatedly remove such edges e belonging to cuts of size less than c, which splits G into its c-edge-connected components. For each $c \geq 1$, denote by $T_c(n)$ the amortized time needed by the data structure to find an edge belonging to a cut of size less than c. For example, for $c = 1$ we have $T_1(n) = O(1)$ since we do not have to maintain anything. For $c = 2$, the data structure is required to maintain some *bridge* of G and it is known that $T_2(n) = O((\log n \cdot \log\log n)^2)$ [25]. For $c \geq 3$, in turn, we have $T_c(n) = O(n^{1/2} \operatorname{poly}(c))$ [41].

Given a fully dynamic c-edge-cut data structure, whose update time for a graph on n vertices is $T_c(n)$, Thorup's [39] reduction maintains, in $O(m \log(n^2/m)) + \widetilde{O}(c \cdot n \cdot T_c(n))$ total time, a c-certificate H of the decremental graph G starting with n vertices and m edges. The certificate undergoes only $\widetilde{O}(cn)$ edge insertions and deletions throughout any sequence of deletions issued to G. We reduce here the total time to $O(m) + \widetilde{O}(c \cdot n \cdot T_c(n))$.

Combining our reduction with the polylogarithmic fully-dynamic connectivity and 2-edge-connectivity algorithm of Holm, de Lichtenberg, and Thorup [23], we can now solve decremental connectivity and 2-edge-connectivity in $O(m) + \widetilde{O}(n)$ time.

[4] There is no consensus in the literature on the terminology relating to c-edge-connected components and classes. Some authors (e.g., [17, 18]) reserve the term c-edge-connected components for what we in this paper call c-edge-connected classes.

We can also apply the fully dynamic min-cut algorithm of Thorup [41] which identifies cuts of size $n^{o(1)}$ in $n^{1/2+o(1)}$ worst-case time. For $c = n^{o(1)}$, we then maintain a c-certificate H in $O(m + n^{3/2+o(1)})$ total time. This includes telling which vertices are in the same c-edge-connected component. If we further want to answer queries about c-edge-connectivity between pairs of vertices, we can apply the fully-dynamic data structure of Jin and Sun [27] to the c-certificate H. By definition, the answers to these queries are the same in H and G, and the algorithm takes $n^{o(1)}$ time per update or query. Hence the total time for the updates remains $O(m + n^{3/2+o(1)})$, and we can tell if two vertices are c-edge-connected in $n^{o(1)}$ time.

1.3 Results

We will now give a more precise description of our reduction, including the log-factors hidden in the $\widetilde{O}(cn)$ bound. Let the *decremental c-certificate* problem be that of maintaining a c-certificate of G when G is subject to edge deletions. Recall that $T_c(n)$ denotes the amortized update time of a fully-dynamic c-edge-cut data structure. Thorup [39] showed the following.

▶ **Theorem 1** (Thorup [39])**.** *There exists a Las Vegas randomized algorithm for the decremental c-certificate problem with expected total update time $O(m \log(n^2/m) + n(c + \log n) \cdot T_c(n) \log^2 n)$. The maintained certificate undergoes $O(n \cdot (c + \log n))$ expected edge insertions and deletions throughout, assuming $\Theta(m)$ random bits are provided. These bounds similarly hold with high probability.*

In particular the total update time is $O(m)$ for very dense graphs with $\Omega(n^2)$ edges. Our main result, which we state below, shows that amortized constant update time can be obtained as long as the initial graph has $\Omega(n \operatorname{polylog}(n))$ edges.

▶ **Theorem 2.** *There exists a Monte Carlo randomized algorithm for the decremental c-certificate problem with total update time $O(m + n(c + \log n) \cdot T_c(n) \log^3 n + nc \log^7 n)$. The maintained certificate undergoes $O(nc \log^4 n)$ edge insertions and deletions throughout. The algorithm is correct with high probability. Within this time bound, the algorithm offers a final self-check after processing all updates.*

In fact, our algorithm is itself a reduction to $O(\log n)$ instances of the decremental c-certificate problem on a subgraph of G with $m' = O(m/\log^2 n)$ edges. To handle each of these instances, we use the state-of-the-art data structure (Theorem 1) which costs only $O(m' \log m') = O(m/\log n)$ (for non-sparse graphs), yielding a combined cost of $O(m)$. As a result, our improved reduction (Theorem 2) requires $\Theta(m/\log n)$ random bits to hold.

We can reduce the need for random bits dramatically paying a little extra cost per vertex. Our new randomized c-certificate that is the key to obtaining the new reduction requires only pairwise independent sampling to work. This is in sharp contrast with the certificate of Karger [30], used in the construction of Thorup's data structure (Theorem 1), which requires full independence, i.e., $\Theta(m)$ random bits. We show that we may instead plug our new certificate into Thorup's data structure at the cost of a single additional logarithmic factor in the running time. Since Karger's certificate constitutes the only use of randomness in Thorup's data structure, and full independence in our construction is required only for invoking Theorem 1, we obtain the below low-randomness version of our main result.

▶ **Theorem 3.** *There exists a Monte Carlo randomized algorithm for the decremental c-certificate problem with total update time $O(m + nc \cdot T_c(n) \log^4 n + nc \log^7 n)$. The maintained certificate undergoes $O(nc \log^4 n)$ edge insertions and deletions throughout. The algorithm is correct with high probability if $O(\operatorname{polylog} n)$ random bits are provided. Within this time bound, the algorithm offers the final self-check after processing all updates.*

By using Theorem 3 with best known fully dynamic algorithms for different values of c [23, 27, 41], we obtain:

▶ **Theorem 4.** *There exists Monte Carlo randomized decremental connectivity and decremental 2-edge-connectivity algorithms with $O(m + n \log^7 n)$ total update time and $O(1)$ query time.*

▶ **Theorem 5.** *Let $c = (\log n)^{o(1)}$. There exists a Monte Carlo randomized decremental c-edge-connectivity data structure which can answer queries to whether two vertices are in the same c-edge connected class in $O(n^{o(1)})$ time, and which has $O(m) + \tilde{O}(n^{3/2})$ total update time.*

▶ **Theorem 6.** *Let $c = O(n^{o(1)})$. There exists a Monte Carlo randomized decremental c-edge-connected components data structure with $O(m + n^{3/2+o(1)})$ total update time and $O(1)$ query time.*

While Theorems 5 and 6 are only optimal for graphs with $m = \Omega(n^{3/2+o(1)})$ edges, we do note that the improvement in runtime from $O(mT_c(n))$ to $O(m + nT_c(n) \operatorname{polylog} n)$ is in general more impressive when $T_c(n)$ is large. E.g., if $T_c(n) = \sqrt{n}$, for dense graphs with $\Omega(n^2)$ edges, the former bound is $O(m^{5/4})$ while the later is $O(m)$ which is a polynomial improvement.

All the above applications of our main result work using only $O(\operatorname{polylog} n)$ random bits. They moreover each have the self-check property as well. As discussed before, our new 2-edge-connectivity data structure implies an optimal $O(m)$-time unique perfect matching algorithm for $m = \Omega(n \operatorname{polylog} n)$.

1.3.1 Adaptive updates and unique perfect matching

All our time bounds are amortized. Amortized time bounds are particularly relevant for dynamic data structures used inside algorithms solving problems for static graphs. In such contexts, future updates often depend on answers to previous queries, and therefore we need algorithms that work with adaptive updates.

Our reduction works against adaptive updates as long as all the information it provides is uniquely defined from the input graph and the update sequence, hence not revealing any information about the random choices in our c-certificate H. We assume some linear orderings of the vertices and the edges, and define the representative (or ID) of a c-edge connected component to be the smallest vertex in it. The reduction will safely maintain the following public information about the c-edge-connected components of G: between deletions, each vertex stores a pointer to the representative of its c-edge connected component, so two vertices are in the same c-edge-connected component if and only if they have they point to the same representative. With the representative, we store the size of the c-edge connected component, and list its vertices in sorted order. Finally, we have a sorted list of all edges that go between c-edge-connected components. After each update, we can also reveal the representatives of the new c-edge-connected components, and the edges between these components. For the case of 2-edge-connectivity, the above means that we can maintain the bridges of a decremental graph and we can also maintain the connected components and their sizes without revealing what the current randomized certificate looks like. All this is needed for the unique perfect matching algorithm of Gabow, Kaplan, and Tarjan [15]. The algorithm is an extremely simple recursion based on the fact that a graph with a unique

Algorithm 1 Algorithm computing Thorup's certificate in the static setting.

Input : A graph $G = (V, E)$, where $n = |V|$, sampling probability P, parameter c
Returns: A set of $\tilde{O}(c \cdot n/P)$ edges giving a c-certificate of G

```
1 Function ThorupCertificate(V, E, P, c):
2     if |E| ≤ c · n then
3         return E
4     S ← subset of E, in which each edge is included independently with prob. P;
5     D ← edges of E connecting distinct c-edge-connected components of (V, S);
6     return D ∪ ThorupCertificate(V, S, P, c)
```

perfect matching has a bridge and all components have even sizes. The algorithm first asks for a bridge (u, v) of some component. If there is none, there is no unique matching. Otherwise we remove (u, v) and check the sizes of the components of u and v. If they are odd, (u, v) is in the unique matching, and we remove all other incident edges. Otherwise (u, v) is not in the unique matching. The important thing here is that the bridges do not tell us anything about our random 2-certificate of the 2-edge-connected components.

Thus we solve the static problem of deciding if a graph has a unique perfect matching in $O(m) + \widetilde{O}(n)$ time. If the self-verification reports a possible mistake, we simply rerun. Consequently we get a Las Vegas algorithm that terminates in $O(m) + \widetilde{O}(n)$ time with high probability.

Outline. Due to space constraints, in the remaining part of this extended abstract we give a rather extensive technical overview of our data structure. All the details and proofs can be found in the full version of this paper.

2 Technical overview

Our main technical contribution is a new construction of a sparse randomized c-certificate that witnesses the c-edge-connected components of G and can be maintained in constant time per edge deletion in G (assuming that the initial graph is not too sparse). In the static case, deterministic certificates of this kind have been known for decades [33]. However, they are not very robust in the decremental setting, where an adversary can constantly remove its edges forcing it to update frequently. Consequently, Thorup [39] used a randomized sample-based certificate to obtain his reduction. The general idea behind this approach is to ensure that the certificate is sparse and undergoes few updates. Ideally, the sparse certificate will only have to be updated whenever an edge from the certificate is deleted. Using a fully dynamic data structure on the certificate, we may obtain efficient algorithms provided that we don't spend too much time on maintaining the certificate. Thorup's reduction had an additive overhead $O(m \log(n^2/m))$ for maintaining the certificate, which we will reduce to the optimal $O(m)$. We shall, in fact, use Thorup's reduction as a subroutine, called on $O(\log n)$ decremental subproblems each starting with $O(m/\log^2 n)$ edges.

2.1 Thorup's construction [39]

Let us first briefly describe how Thorup's algorithm operates on certificates and highlight difficulties in improving his reduction to linear time. First of all, the c-certificate is constructed as follows (see Algorithm 1 for pseudocode). Initially, sample edges of G uniformly with

probability $P \leq 1/2$, thus obtaining a subgraph S. Then, compute the c-edge-connected components of S and form a certificate H out of two parts: (1) A *recursive* certificate of S, and (2) the subgraph D consisting of edges of G connecting distinct c-edge-connected components of S.

As proved by Karger [30], D has size $\widetilde{O}(cn/P)$ with high probability. Thorup [39] generalizes this by proving that D undergoes only $\widetilde{O}(cn/P)$ insertions throughout any sequence of edge deletions to S. Since D depends only on the c-edge-connected components of S, it is enough to have a c-certificate of S in order to define D. Hence, a c-certificate of S (which is a graph a size $O(mP)$, i.e., a constant factor smaller) is maintained under edge deletions recursively. The recursion stops when the size of the input graph is $O(cn)$. To maintain D at each recursive level, we first need to maintain the c-edge-connected components of the (recursive) certificate of S under edge deletions. The certificate of S can be (inductively) seen to have $\widetilde{O}(cn/P)$ edges and undergo $\widetilde{O}(cn/P)$ updates. As a result, for $P = 1/2$ maintaining its c-edge-connected components costs $\widetilde{O}(cn \cdot T_c(n))$ total time using the fully-dynamic c-edge-cut data structure. Since at each recursion level the certificate size decreases geometrically, the expected cost of all the dynamic c-edge-cut data structures is $\widetilde{O}(cn \cdot T_c(n))$. For $c = 1, 2$, $\widetilde{O}(cn \cdot T_c(n)) = \widetilde{O}(n)$.

The bottle-neck in Thorup's reduction. For non-sparse graphs, the bottleneck in Thorup's reduction is the additional cost of $O(m \log(n^2/m))$ which comes from the fact that, at each level of the recursion, when a c-edge-connected component in S splits into two components as a result of an edge deletion, we need to find edges of G between these two components in order to update D. This takes $O(m \log(n^2/m))$ total time throughout using a standard technique of iterating through the edges incident to the vertices in the smaller component every time a split happens [10]. The $O(\log(n^2/m))$ (instead of $O(\log(n))$) cost comes by noticing that a vertex can at most have q neighbors in a component of order q, and that after we go through the edges of a vertex i times it is in a component of order $\leq n/2^i$; hence it is only the first $O(\log(n/\deg(v)))$ times that all neighbors of v have to be considered, so, by applying Jensen's inequality, the total time spent on this becomes $O\left(\sum_{v \in V} \deg(v) \log(n/\deg(v))\right) = O(m \log(n^2/m))$.

It turns out very challenging to get rid of the $O(m \log(n^2/m))$ term associated with finding cuts when components split in Thorup's reduction. If we knew that all of these cuts were *small*, say of size at most δ, then we could apply a whole bag of tricks for efficiently finding them in a total time of $\tilde{O}(n\delta)$, e.g., using invertible Bloom lookup tables [19], or the XOR-trick [1, 2, 29]. Unfortunately, the bound of $\tilde{O}(cn/P)$ only gives an average bound on the number of edges between pairs of components, and in fact there can be pairs of components having as many as $\Omega(n^{1/3})$ edges between them, as we will later show. In order to resolve this, we have to introduce a new type of sample based c-edge certificate obtained by only removing cuts of size at most $\delta = O(c \operatorname{polylog} n)$ from G. In the following three subsections, we describe the ideas behind this new certificate, the technical challenges encountered in efficiently maintaining it, and why such a certificate is relevant for decremental connectivity algorithms.

2.2 Our c-certificate based on small cut samples

In this section we describe the construction of our c-certificate. For simplicity, we assume $c = 1$ for now.

The (simplified) algorithm for computing the certificate in the static setting is given as Algorithm 2. In order to obtain a conceptually simpler picture of the certificate, Algorithm 2 is described recursively where each recursive call takes as input a *minor* G' of G, namely

Algorithm 2 Algorithm computing our new certificate in the static setting.

Input : A graph $G = (V, E)$ where $n = |V|$ and $m = |E|$, sampling probability P, parameter δ

Returns : A set of $O(mP \log n + n\delta \log n)$ edges giving a 1-certificate of G

```
1  Function NewCertificate(V, E, P, δ):
2      if E = ∅ then
3          return ∅
4      D = ∅;
5      while G has a non-isolated vertex v of degree ≤ δ do
6          Remove from E all edges incident to v and add them to D
7      S ← subset of E, in which each edge is included independently with prob. P;
8      H ← (V, S);
9      G' ← graph obtained from G by contracting connected components of H;
10     return S ∪ D ∪ NewCertificate(V(G'), E(G'), P, δ)
```

the graph obtained by contracting the connected components of $H = (V, S)$, where S is a subset of edges of G (after pruning G of small cuts in lines 5-6) sampled with probability P. Adding an edge e of G' to the certificate, simply means that we add the corresponding edge of G. While Algorithm 2 gives a precise description of the static certificate at any given point, maintaining these minors in the dynamic setting is too costly. Because of that, in the dynamic algorithm, instead of using minors we work with a sequence of subgraphs of the initial graph that are easier to maintain dynamically.

Denote by ℓ the depth of the recursion in Algorithm 2. For $i = 1, \ldots, \ell$, let S_i be the union of samples S on the recursive levels $1, \ldots, i$ of Algorithm 2, so that S_ℓ contains all the edges sampled in the process. When an edge is deleted from G, it is removed from all the sampled subsets S in the recursion, and thus also from all the relevant subsets S_i. This way, after any sequence of deletions the certificate that we maintain only depends on the initial samples $S_1, S_2 \setminus S_1, \ldots, S_\ell \setminus S_{\ell-1}$ and the current graph G, not on the sequence of edge updates made to G so far. We may therefore describe the certificate statically.

The critical idea behind our certificate is to introduce a small-cut-parameter δ. Our certificate is obtained by iteratively removing certain cuts from G where each cut is allowed to be of size at most δ. We denote by $D \subset G$ the graph whose edge set consists of the edges removed in this process. The overall goal is to define this cut removal process in a way so that (1) each connected component of $G \setminus D$ is connected in S_l, and (2) it is easy to detect new small cuts under edge deletions issued to G. We then use $S_\ell \cup D$ as our connectivity certificate of G. Importantly, we want δ to be *as small as possible*, ideally $\delta = O(\text{polylog}(n))$. This is because $\tilde{O}(\delta n)$ will show up as an additive cost in our algorithm for maintaining the certificate.

We will describe shortly how this type of certificate can be used in the design of efficient decremental connectivity algorithms, but let us first demonstrate that the existence of such a cut removal process (satisfying both (1) and (2)) for a small δ is non-trivial.

First of all, we could simply remove *all* cuts from G of size at most δ leaving us with the $(\delta + 1)$-connected components. Karger's result [30] implies that with $\delta = O((c + \log n)/P)$ sufficiently large, these components will remain c-edge connected in S. However, in order to maintain the small cuts, we would need a decremental δ-edge connectivity algorithm. As $\delta > c$, this approach simply reduces our problem to a much harder one.

Suppose on the other hand that we attempted to use Thorup's sampling certificate [39] described above. To simplify the exposition, let's assume that $P = 1/2$. If D is the set of edges between connected components of S, $D \cup S$ is a certificate. Thorup's algorithm recurses on S to find a final certificate of G. At first sight it may seem like D can be constructed by iteratively removing cuts of size at most $\delta = O(\log n)$ between the connected components of S. After all, isn't it unlikely that a connected component of S has more than, say, $100 \log n$ unsampled outgoing edges when the sampling probability is $P = 1/2$? As alluring as this logic may be, it is flawed. Indeed, there exist graphs of maximum degree $O(\log n)$ such that after sampling with $P = 1/2$, some two connected components of the sampled subgraph, C_1 and C_2, will have $\Omega(n^{1/3})$ unsampled edges between them. At some point in the iterative process, we are thus forced to remove a cut of size $\Omega(n^{1/3})$ splitting C_1 and C_2, and we would have to choose δ of at least this size (but it is possible that other examples could show that δ would have to be even larger). Our algorithms spend total time $\tilde{O}(n\delta)$ on finding these cuts, and if $\delta = \Omega(n^{1/3})$, this is not good enough for a linear time algorithm for non-sparse graphs.

We remark that in this example, each vertex of G has degree $O(\log n)$ with high probability. Therefore, an alternative approach yielding cuts of size $O(\log n)$ would be to cut out one vertex at a time moving all incident edges to D. In particular this would cut the large sampled components C_1 and C_2 into singletons, one vertex at a time. We cannot proceed like this for general graphs which may have many vertices of large degree. Nevertheless, this simple idea will be critically used in our construction which we will now discuss.

Our actual certificate uses $\delta = \Theta(\frac{\log n}{P})$ and $P = 1/\operatorname{polylog} n$. To construct our certificate, we start by iteratively pruning G of the edges incident to vertices of degree less than δ, moving these edges to D. The graph left after the pruning $G_1 = G \setminus D$ satisfies that each vertex of positive degree has degree at least δ. Next, S_1 defines a sample of G_1, $H_1 = S_1 \cap G_1$. The expected degree of each vertex in H_1 which is not isolated in G_1 is at least $\delta \cdot P = \Theta(\log n)$, and thus we get that with constant probability a fraction of $3/4$ of the vertices with positive degree in the sampled subgraph H_1 have degree ≥ 4.

Using this property we show that H_1 can have at most $5n/6$ connected components. As a result, if we contract the connected components of H_1 in the pruned graph G_1, the resulting graph G_1' has at most $5n/6$ vertices. Finally, we construct a certificate for G_1' recursively using the samples $S_2 \setminus S_1, S_3 \setminus S_2, \ldots$, stopping when the contracted graph has no edges between the contracted vertices (here G played the role of G_0'). The constant factor decay in the number of components ensures that we are done after $\ell = O(\log n)$ steps with high probability. All edges of D are obtained as the removed edges of cuts of G of size less than δ, so D will have size $O(n\delta)$. Our certificate will simply be $S_\ell \cup D$ which we prove is in fact a certificate.

With this, we have thus completed the goal of obtaining a small cut sample certificate with δ as small as $O(\frac{\log n}{P})$. Abstractly, our certificate has a quite simple description: we alternate between sampling, removing small cuts around connected components in the sample, and finally contracting these components. However, in our implementation, we cannot afford to perform the contractions as described above explicitly, as updating them dynamically would be costly. As a result we end up solving a more challenging problem in the dynamic setting. Given a graph G and its subgraph H undergoing edge deletions, determine if any connected components of H is incident to at most δ edges of $G \setminus D$, i.e., induces a cut of at most δ edges. It turns out that since we are only concerned with cuts of size at most δ, we can in fact identify these cuts in total time $O(m) + \tilde{O}(\delta n)$. We will describe this in the following section.

A final property of our new randomized decremental certificate algorithm is that it requires only $O(\log^2 n)$ random bits to yield high-probability correctness bounds. This is in sharp contrast with Thorup's algorithm [39] which requires $\Omega(m)$ random bits. On a high level, the reason we can do with few random bits is that in each step of the construction of our certificate, we only need the bounds on the number of contracted components to hold with constant probability. Indeed, we will still only have $O(\log n)$ recursive levels with high probability. This means that for the probability bounds within a single recursive level, it suffices to apply Chebyshev's inequality. While the reduction of the number of required random bits is nice, the main point, however, is that with our new certificate we can get down to constant amortized update time per edge-deletion for decremental (2-edge)-connectivity for all but the sparsest graphs.

2.3 Maintaining our certificate

As edges are deleted from G, the recursive structure of the c-certificate H changes. Indeed, a deletion of an edge may cause the following changes in one of the recursive layers of H: (1) introduce a cut of size less than δ surrounding a c-edge-connected component or (2) break a c-edge-connected component in two. In the first case, the edges of the cut have to be moved to D, and deleted from the current and later layers of H, causing further cascading. When a c-edge-connected component (in a recursive layer) of H breaks in two, we need to determine whether either of the new components has less than δ outgoing edges in $G \setminus D$. If we use the standard technique of iterating over all the edges incident to vertices of the smaller component, this again incurs an $O(\log(n^2/m))$ cost per edge which is insufficient. However, as we only care about components with at most δ outgoing edges, it turns out that we can do better. We define the *boundary* $\partial_G(C)$ of a component C of some graph $H \subset G$ to be the set of edges of G with one endpoint in C, and another in $V \setminus C$. To overcome the $O(\log(n^2/m))$ cost per edge, we prove that we can maintain boundaries of size at most δ under splits using a Monte Carlo randomized algorithm in $O(m + n\delta \operatorname{polylog} n)$ total time. We achieve this by developing a fully dynamic data structure summarized as follows, that we believe may be of independent interest.

▶ **Theorem 7.** *Let $G = (V, E)$ be an initially empty graph subject to edge insertions and deletions and let s, $1 \leq s \leq n$, be an integral parameter. There exists a data structure that can process up to $O(\operatorname{poly} n)$ queries of the form "given some $S \subseteq V$, compute $\partial_G(S)$", where $\partial_G(S) = E(S, V \setminus S)$, so that with high probability each query is answered correctly in $O\left(|S|s + |E(S,V)| \cdot \frac{|\partial_G(S)|}{s} + \log n\right)$ time. The data structure is initialized in $O(ns)$ time and can be updated in constant time.*

We realize this result by deploying the so-called XOR-trick [29][5] for deciding if a boundary of some subset of vertices is non-empty, albeit in a somewhat unusual manner. We now briefly describe the method. Suppose each $e \in E$ is assigned a random bit-string x_e of length $\Theta(\log n)$, which fits in $O(1)$ machine words. Let $x_v = \bigoplus_{vw=e\in E} x_e$ denote the XOR of the respective bit-strings of edges incident to v. Then, one can prove that, given $S \subseteq V$, with high probability the XOR $\bigoplus_{u\in S} x_u$ is non-zero if and only if $\partial_G(S) \neq \emptyset$. The underlying idea is that if an edge e incident to $v \in S$ has its other endpoint also contained in S, its corresponding bit string x_e appears exactly twice in $\bigoplus_{u\in S} x_u$, and thus cancels out. So, emptiness of $\partial_G(S)$ can be tested in $O(|S|)$ time.

[5] See also [1, 2] for uses of the same idea in other contexts.

The XOR trick can also be used with no change to retrieve a non-empty boundary $\partial_G(S)$, but only when that boundary has precisely one element. In order to retrieve *some* element of $\partial_G(S)$, existing applications of the XOR-trick consider a polylogarithmic number of independent edge set samples, chosen such that one of the samples intersects $\partial_G(S)$ precisely on one edge (with high probability). This unavoidably introduces a polylogarithmic dependence in the cost per edge of the graph, which is prohibitive in our scenario.

The main idea behind Theorem 7 which allows us to deal with this problem is as follows. We partition the edge set E into $E_1, \dots, E_s$. Each $e \in E$ is assigned to one of these sets uniformly at random. We apply the XOR-trick for each of the edge-disjoint subgraphs (V, E_i) separately. This takes $O(s|S|)$ time and computes a set I of all i such that $\partial_G(S) \cap E_i \neq \emptyset$ (with high probability). Clearly, in order to find $\partial_G(S)$, we only need to look for this boundary's elements in $\left(\bigcup_{i \in I} E_i\right) \cap E_G(S, V)$. Note that the expected size of this set is $(|I|/s) \cdot |E_G(S, V)| \leq (|\partial_G(S)|/s) \cdot |E_G(S, V)|$. If we set s to be larger than the maximum size of a boundary that we would like to retrieve (in the algorithm we ensure that the ratio is polylogarithmic), we significantly reduce the set of candidate edges to consider and can search through them exhaustively. In total, as we show, only $O(|\partial_G(S)| + |E_G(S, V)| \cdot |\partial_G(S)|/s + \log n)$ edges are explored with high probability.

In our application, we end up using the data structure of Theorem 7 storing the (dynamic) graph $G \setminus D$, and handling small boundary (of size no more than $\delta = \operatorname{polylog} n$) queries for smaller sides $C \subseteq V$ of decomposing components of $\ell = O(\log n)$ dynamic subgraphs of $G \setminus D$. Throughout, the total size of the queried subsets C is $O(n \log^2 n)$. Consequently, the sum of $|E(C, V)|$ over these sets is $O(m \log^2 n)$. By setting $s = \delta \log^2 n$ in Theorem 7, we obtain that the required queries for δ-bounded boundaries $\partial_{G \setminus D}(C)$ can be processed in $O(n \operatorname{polylog} n + m)$ total time.

2.4 Combining our certificate with Thorup's algorithm

With the certificate as above, the overall idea for a decremental connectivity algorithm is to maintain a c-certificate of (each recursive layer of) the *decremental* graph $H = S \setminus D$ using the algorithm by Thorup [39]. By choosing $P = 1/\log^2 n$, S has $m' = O(m/\log^2 n)$ edges with high probability, so employing the algorithm of Theorem 1 on each recursive layer takes total time $O(m' \log^2 n + ncT_c(n) \operatorname{polylog} n) = O(m + ncT_c(n) \operatorname{polylog} n)$ with high probability. Let H^* be the c-certificate thus obtained for H. Using a fully dynamic c-edge-connectivity algorithm on $H^* \cup D$ (which undergoes $O(cn \operatorname{polylog} n)$ updates), we maintain a c-edge certificate of G. As $H^* \cup D$ undergoes $O(cn \operatorname{polylog} n)$ updates, running the fully dynamic algorithm takes total time $O(cnT_c(n) \operatorname{polylog} n)$.

We remark that for $c = 1, 2$ we could instead use a fully dynamic c-edge connectivity algorithm on H with polylogaritmic update and query time at the price of a smaller P (which would incur more log-factors in our final time bound). For $c > 2$, however, we only know that $T_c(n) = O(n^{1/2} \operatorname{poly}(c))$. Since running a fully dynamic algorithm on H takes total time $\Omega(mT_c(n)/\operatorname{polylog} n)$, this is insufficient to obtain linear time algorithms for dense graphs.

2.5 Final self-check

Let us finally describe the ideas behind the final self-checks claimed in Theorem 2 and 3 in a more general context. In particular, we show that if a randomized Monte Carlo dynamic algorithm satisfies some generic conditions then it can be augmented to detect, at the end of its execution, whether there is any chance that it answered any query incorrectly. That is, if the self-check passes then it is guaranteed that all queries were answered correctly

throughout the execution of the algorithm. Otherwise, it indicates that some queries might have been answered incorrectly. The self-check property is particularly useful in applications of dynamic algorithms as subroutines in algorithms solving static problems, that is, it enables static algorithms to exhibit Las Vegas guarantees instead of the Monte Carlo guarantees provided by the dynamic algorithm, as they can simply re-run the static algorithm with fresh randomness until the self-check passes.

The properties of a dynamic algorithm amenable to a self-check behavior are as follows:

- Once an error is made by the dynamic algorithm it should be detectable and any subsequent updates of the algorithm should not correct the error before it is detected.
- If the dynamic algorithm is stopped at any point in time, it should be able to still perform the self-check within the guaranteed running time of the algorithm.

In our algorithm, as long as the c-certificate maintained by our algorithm is correct, the c-edge-connectivity queries answered by our algorithm exhibit the same guarantees as the fully dynamic c-edge-connectivity algorithm running on the c-certificate H. Hence, we only need to detect potential errors in the process of maintaining the c-certificate H. Such errors only happen with probability $n^{-\Omega(1)}$.

By definition, a c-certificate $H \subseteq G$ of G is correct if for every "non-witness" edge (u, v) from $G \setminus H$, we have that u and v are c-edge-connected in H. We use $H = S_\ell \cup D$ where S_ℓ is decremental, and we impose the stronger requirement that if $(u, v) \in G \setminus H$, then u and v are c-edge-connected in S_ℓ. If this is not the case, we consider it an error.

Suppose we have an error with (u, v). Since S_ℓ is decremental, u and v cannot later become c-edge connected in S_ℓ. Thus, the error can only disappear if (u, v) is deleted from G or (u, v) is added to H. Therefore, all our self-checker needs to do is this: Whenever an edge from $G \setminus H$ is about to be deleted from G or about to be added to H, we first check that u and v are c-edge-connected in S_ℓ; otherwise we found an error. Also, if the algorithm is terminated before all edges are deleted, we perform that above check on all remaining edges. If any check finds an error, we flag the execution as invalid.

If an execution of our algorithm has not been flagged, we know that all queries have been answered correctly. Moreover, the execution is only flagged with probability $n^{-\Omega(1)}$.

As a final note, every vertex will maintain an ID of its c-edge-connected component in S_ℓ. Then u and v are the c-edge-connected in S_ℓ if and only if they have the same ID. This is checked in constant time, so these extra checks do not affect our overall asymptotic time bounds.

References

1 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 459–467. SIAM, 2012. doi:10.1137/1.9781611973099.40.

2 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In Michael Benedikt, Markus Krötzsch, and Maurizio Lenzerini, editors, *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 5–14. ACM, 2012. doi:10.1145/2213556.2213560.

3 S. Alstrup, T. Husfeldt, and T. Rauhe. Marked ancestor problems. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 534–543, 1998. doi:10.1109/SFCS.1998.743504.

4 Stephen Alstrup, Jens P. Secher, and Mikkel Thorup. Word encoding tree connectivity works. In David B. Shmoys, editor, *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA*, pages 498–499. ACM/SIAM, 2000. URL: http://dl.acm.org/citation.cfm?id=338219.338598.

5 Stephen Alstrup, Jens Peter Secher, and Maz Spork. Optimal on-line decremental connectivity in trees. *Information Processing Letters*, 64(4):161–164, 1997. doi:10.1016/S0020-0190(97)00170-1.

6 Joshua D. Batson, Daniel A. Spielman, Nikhil Srivastava, and Shang-Hua Teng. Spectral sparsification of graphs: theory and algorithms. *Commun. ACM*, 56(8):87–94, 2013. doi:10.1145/2492007.2492029.

7 Sayan Bhattacharya, Fabrizio Grandoni, Janardhan Kulkarni, Quanquan C. Liu, and Shay Solomon. Fully dynamic $(\Delta+1)$-coloring in $O(1)$ update time. *ACM Trans. Algorithms*, 18(2):10:1–10:25, 2022. doi:10.1145/3494539.

8 Julia Chuzhoy, Yu Gao, Jason Li, Danupon Nanongkai, Richard Peng, and Thatchaphol Saranurak. A deterministic algorithm for balanced cut with applications to dynamic connectivity, flows, and beyond. *CoRR*, abs/1910.08025, 2019. arXiv:1910.08025.

9 David Eppstein, Zvi Galil, Giuseppe F. Italiano, and Amnon Nissenzweig. Sparsification – A technique for speeding up dynamic graph algorithms. *J. ACM*, 44(5):669–696, 1997. doi:10.1145/265910.265914.

10 Shimon Even and Yossi Shiloach. An on-line edge-deletion problem. *J. ACM*, 28(1):1–4, 1981. doi:10.1145/322234.322235.

11 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005. doi:10.1016/j.tcs.2005.09.013.

12 Greg N. Frederickson. Data structures for on-line updating of minimum spanning trees, with applications. *SIAM J. Comput.*, 14(4):781–798, 1985. doi:10.1137/0214055.

13 M. Fredman and M. Saks. The cell probe complexity of dynamic data structures. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing (STOC)*, STOC '89, pages 345–354, New York, NY, USA, 1989. Association for Computing Machinery. doi:10.1145/73007.73040.

14 Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987. doi:10.1145/28869.28874.

15 Harold N. Gabow, Haim Kaplan, and Robert Endre Tarjan. Unique maximum matching algorithms. *J. Algorithms*, 40(2):159–183, 2001. doi:10.1006/jagm.2001.1167.

16 Harold N. Gabow and Robert Endre Tarjan. A linear-time algorithm for a special case of disjoint set union. *J. Comput. Syst. Sci.*, 30(2):209–221, 1985. doi:10.1016/0022-0000(85)90014-5.

17 Zvi Galil and Giuseppe F. Italiano. Maintaining the 3-edge-connected components of a graph on-line. *SIAM J. Comput.*, 22(1):11–28, 1993. doi:10.1137/0222002.

18 Dora Giammarresi and Giuseppe F. Italiano. Decremental 2- and 3-connectivity on planar graphs. *Algorithmica*, 16(3):263–287, 1996. doi:10.1007/BF01955676.

19 Michael T Goodrich and Michael Mitzenmacher. Invertible bloom lookup tables. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 792–799. IEEE, 2011.

20 Fabrizio Grandoni, Stefano Leonardi, Piotr Sankowski, Chris Schwiegelshohn, and Shay Solomon. $(1+\epsilon)$-approximate incremental matching in constant deterministic amortized time. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1886–1898. SIAM, 2019. doi:10.1137/1.9781611975482.114.

21 Monika Henzinger and Pan Peng. Constant-time dynamic $(\Delta+1)$-coloring. *ACM Trans. Algorithms*, 18(2):16:1–16:21, 2022. doi:10.1145/3501403.

22 Monika Rauch Henzinger and Valerie King. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *J. ACM*, 46(4):502–516, 1999. doi:10.1145/320211.320215.

23 Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *J. ACM*, 48(4):723–760, July 2001. doi:10.1145/502090.502095.

24 Jacob Holm and Eva Rotenberg. Good r-divisions imply optimal amortised decremental biconnectivity. *CoRR*, abs/1808.02568, 2018. arXiv:1808.02568.

25 Jacob Holm, Eva Rotenberg, and Mikkel Thorup. Dynamic bridge-finding in $\tilde{O}(\log^2 n)$ amortized time. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 35–52, 2018. doi:10.1137/1.9781611975031.3.

26 Shang-En Huang, Dawei Huang, Tsvi Kopelowitz, and Seth Pettie. Fully dynamic connectivity in $O(\log n(\log\log n)^2)$ amortized expected time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 510–520, 2017. doi:10.1137/1.9781611974782.32.

27 Wenyu Jin and Xiaorui Sun. Fully dynamic s-t edge connectivity in subpolynomial time (extended abstract). In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 861–872. IEEE, 2021. doi:10.1109/FOCS52979.2021.00088.

28 Bruce M. Kapron, Valerie King, and Ben Mountjoy. Dynamic graph connectivity in polylogarithmic worst case time. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1131–1142, 2013. doi:10.1137/1.9781611973105.81.

29 Bruce M. Kapron, Valerie King, and Ben Mountjoy. Dynamic graph connectivity in polylogarithmic worst case time. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '13, pages 1131–1142, USA, 2013. Society for Industrial and Applied Mathematics.

30 David R. Karger. Random sampling in cut, flow, and network design problems. *Math. Oper. Res.*, 24(2):383–413, 1999. doi:10.1287/moor.24.2.383.

31 Casper Kejlberg-Rasmussen, Tsvi Kopelowitz, Seth Pettie, and Mikkel Thorup. Faster worst case deterministic dynamic connectivity. In *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, pages 53:1–53:15, 2016. doi:10.4230/LIPIcs.ESA.2016.53.

32 Jakub Lacki and Piotr Sankowski. Optimal decremental connectivity in planar graphs. *Theory Comput. Syst.*, 61(4):1037–1053, 2017. doi:10.1007/s00224-016-9709-x.

33 Hiroshi Nagamochi and Toshihide Ibaraki. A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica*, 7(5&6):583–596, 1992. doi:10.1007/BF01758778.

34 Danupon Nanongkai, Thatchaphol Saranurak, and Christian Wulff-Nilsen. Dynamic minimum spanning forest with subpolynomial worst-case update time. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 950–961, 2017. doi:10.1109/FOCS.2017.92.

35 Mihai Pătraşcu and Erik D. Demaine. Lower bounds for dynamic connectivity. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pages 546–553, New York, NY, USA, 2004. Association for Computing Machinery. doi:10.1145/1007352.1007435.

36 Mihai Pătraşcu and Mikkel Thorup. Don't rush into a union: take time to find your roots. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 559–568, 2011. doi:10.1145/1993636.1993711.

37 Shay Solomon. Fully dynamic maximal matching in constant update time. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 325–334. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.43.

38 Robert Endre Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM*, 22(2):215–225, April 1975. doi:10.1145/321879.321884.

39 Mikkel Thorup. Decremental dynamic connectivity. *Journal of Algorithms*, 33(2):229–243, 1999. doi:10.1006/jagm.1999.1033.

40 Mikkel Thorup. Near-optimal fully-dynamic graph connectivity. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 343–350, 2000. doi:10.1145/335305.335345.

41 Mikkel Thorup. Fully-dynamic min-cut. *Comb.*, 27(1):91–127, 2007. doi:10.1007/s00493-007-0045-2.

42 Zhengyu Wang. An improved randomized data structure for dynamic graph connectivity. *CoRR*, abs/1510.04590, 2015. arXiv:1510.04590.

43 Christian Wulff-Nilsen. Faster deterministic fully-dynamic graph connectivity. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1757–1769. SIAM, 2013. doi:10.1137/1.9781611973105.126.

On Range Summary Queries

Peyman Afshani
Aarhus University, Denmark

Pingan Cheng
Aarhus University, Denmark

Aniket Basu Roy
Aarhus University, Denmark

Zhewei Wei
Renmin University of China, Beijing, China

Abstract

We study the query version of the approximate heavy hitter and quantile problems. In the former problem, the input is a parameter ε and a set P of n points in $\mathbb{R}^d$ where each point is assigned a color from a set C, and the goal is to build a structure such that given any geometric range γ, we can efficiently find a list of approximate heavy hitters in $\gamma \cap P$, i.e., colors that appear at least $\varepsilon|\gamma \cap P|$ times in $\gamma \cap P$, as well as their frequencies with an additive error of $\varepsilon|\gamma \cap P|$. In the latter problem, each point is assigned a weight from a totally ordered universe and the query must output a sequence S of $1 + 1/\varepsilon$ weights such that the i-th weight in S has approximate rank $i\varepsilon|\gamma \cap P|$, meaning, rank $i\varepsilon|\gamma \cap P|$ up to an additive error of $\varepsilon|\gamma \cap P|$. Previously, optimal results were only known in 1D [23] but a few sub-optimal methods were available in higher dimensions [4, 6].

We study the problems for two important classes of geometric ranges: 3D halfspace and 3D dominance queries. It is known that many other important queries can be reduced to these two, e.g., 1D interval stabbing or interval containment, 2D three-sided queries, 2D circular as well as 2D k-nearest neighbors queries. We consider the real RAM model of computation where integer registers of size w bits, $w = \Theta(\log n)$, are also available. For dominance queries, we show optimal solutions for both heavy hitter and quantile problems: using linear space, we can answer both queries in time $O(\log n + 1/\varepsilon)$. Note that as the output size is $\frac{1}{\varepsilon}$, after investing the initial $O(\log n)$ searching time, our structure takes on average $O(1)$ time to find a heavy hitter or a quantile! For more general halfspace heavy hitter queries, the same optimal query time can be achieved by increasing the space by an extra $\log_w \frac{1}{\varepsilon}$ (resp. $\log\log_w \frac{1}{\varepsilon}$) factor in 3D (resp. 2D). By spending extra $\log^{O(1)} \frac{1}{\varepsilon}$ factors in both time and space, we can also support quantile queries.

We remark that it is hopeless to achieve a similar query bound for dimensions 4 or higher unless significant advances are made in the data structure side of theory of geometric approximations.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Computational Geometry, Range Searching, Random Sampling, Geometric Approximation, Data Structures and Algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.7

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2305.03180`

Funding *Peyman Afshani, Pingan Cheng*: Supported by DFF (Det Frie Forskningsråd) of Danish Council for Independent Research under grant ID DFF−7014−00404.
Aniket Basu Roy: Supported by DFF (Det Frie Forskningsråd) of Danish Council for Independent Research under grant ID DFF−7014−00404 and the Independent Research Fund Denmark (DFF) under a Sapere Aude Research Leader grant No. 1051-00106B.
Zhewei Wei: Supported by National Natural Science Foundation of China (No. U2241212).

1 Introduction

Range searching is an old and fundamental area of computational geometry that deals with storing an input set $P \subset \mathbb{R}^d$ of n (potentially weighted) points in a data structure such that given a query range γ, one can answer certain questions about the subset of points inside γ. Range searching is often introduced within a general framework that allows a very diverse set of questions to be answered. For instance, if the points in P have been assigned integer or real weights, then one can count the points in γ (range counting), sum the total weights of the points in γ (weighted range counting), or find the maximum or minimum weight in γ (range max or min queries).

However, there are some important questions that cannot be answered within this general framework. Consider the following motivating example: our data includes the locations of houses in a city as well as their estimated values and given a query range γ, we are interested in the distribution of the house values within γ, for example, we might be interested to see if there's a large inequality in house values or not. Through classical results, we can find the most expensive and the least expensive houses (max and min queries), and the average value of the houses (by dividing the weighted sum of the values by the total number of houses in γ). Unfortunately, this information does not tell us much about the distribution of the house values within γ, e.g., one cannot compute the Gini index which is a widely-used measure of inequality of the distribution. Ideally, to know the exact distribution of values within γ, one must have all the values inside γ, which in the literature is known as a *range reporting* query which reports all the points inside the query range γ. However, this could be an expensive operation, e.g., it can take $\Omega(n)$ time if the query contains a constant fraction of the input points. A reasonable alternative is to ask for a "summary" query, one that can summarize the distribution. In fact, the streaming literature is rich with many important notions of summary that are used to concisely represent a large stream of data approximately but with high precision. Computing ε-quantiles can be considered as one of the most important concepts for a succinct approximation of a distribution and it also generalizes many of the familiar concepts, e.g., 0-quantile, 0.5-quantile, and 1-quantile that are also known as the minimum, the median, and the maximum of S. We now give a formal definition below.

Quantile summaries. Given a sequence of values $w_1 \leq \cdots \leq w_k$, a δ-quantile, for $0 \leq \delta \leq 1$, is the value with rank $\lfloor \delta k \rfloor$. By convention, 0-quantile and 1-quantiles are set to be the minimum and the maximum, i.e., w_1 and w_k respectively. An ε-quantile summary is then defined as the list of $1 + \varepsilon^{-1}$ values where the i-th value is the $i\varepsilon$-quantile, for $i = 0, \cdots, \varepsilon^{-1}$. As we will review shortly, computing exact quantiles is often too expensive so instead we focus on approximations. We define an approximate ε-quantile summary (AQS) to be a sequence of $1 + \varepsilon^{-1}$ values where the i-th value is between the $(i-1)$-quantile and the $(i+1)$-quantile[1], for $i = 0, \cdots, \varepsilon^{-1}$. An approximate quantile summary with a reasonably small choice of ε can give a very good approximation of the distribution. It also has the benefit that the query needs to output only $O(\varepsilon^{-1})$ values, regardless of the number of points inside the query range.

To obtain a relatively precise approximation of the distribution, ε needs to be chosen sufficiently small, and thus we consider it an additional parameter (and thus not a constant). This is also similar to the literature on streaming where the dependency on ε is important.

[1] For $a \leq 0$ (resp. $a \geq k$), we define the a-quantile to be the 0-quantile (resp. k-quantile).

1.1 Problem Definition, Previous Work, and Related Results

One of our main problems is the problem of answering approximate quantile summary (AQS) queries which is defined as follows.

▶ **Problem 1** (Approximate quantile summaries). *Consider an input set P of n points in $\mathbb{R}^d$ where each point $p \in P$ is assigned a weight w_p from a totally ordered universe. Given a value ε, we are asked to build a structure such that given a query range γ, it can return an AQS of $P \cap \gamma$ efficiently.*

It turns out that another type of "range summary queries" is extremely useful for building data structures for AQS queries.

Heavy hitter summaries. Consider a set P of k points where each point in P is assigned a color from the set $[n]$. Let f_i be the frequency of color i in P, i.e., the number of times color i appears among the points in P. A *heavy hitter summary (HHS)* with parameter ε, is the list of all the colors i with $f_i \geq \varepsilon k$ together with the value f_i. As before, working with exact HHS will result in very inefficient data structures and thus once again we turn to approximations. An *approximate heavy hitter summary (AHHS)* with parameter ε is a list, L, of colors such that every color i with $f_i \geq \varepsilon k$ is included in L and furthermore, every color $i \in L$ is also accompanied with an approximation, f'_i, of its frequency such that $f_i - \varepsilon k \leq f'_i \leq f_i + \varepsilon k$.

▶ **Problem 2** (Approximate heavy hitters summaries). *Consider an input set P of n points in $\mathbb{R}^d$ where each point in P is assigned a color from the set $[n]$. Given a parameter ε, we are asked to build a structure such that given a query γ, it can return an AHHS of the set $P \cap \gamma$.*

Observe that in both problems, the output size of a query is $O(1/\varepsilon)$ in the worst-case. Our main focus is to obtain data structures with the optimal worst-case query time of $O(\log n + \varepsilon^{-1})$. Note that it makes sense to define an *output-sensitive* variant where the query time is $O(\log n + k)$ where k is the output size. E.g., it could be the case for a AHHS query that the numbrer of heavy hitters is much fewer than ε^{-1}. This makes less sense for AQS queries, since unless the distribution of weights inside the query range γ is almost constant, an AQS will have $\Omega(\varepsilon^{-1})$ distinct values. As our main focus is on AQS, we only consider AHHS data structures with the worst-case query time of $O(\log n + \varepsilon^{-1})$.

A note about the notation. To reduce the clutter in the expressions of query time and space, we adopt the convention that $\log(\cdot)$ function is at least one, e.g., we define $\log_a b$ to be $\max\{1, \frac{\ln b}{\ln a}\}$ for any positive values a, b.

Previous Results

As discussed, classical range searching solutions focus on rather simple queries that can return sum, weighted sum, minimum, maximum, or the full list of points contained in a given query range. This is an extensively researched area with numerous results to cite and so we refer the reader to an excellent survey by Agarwal [5] that covers such classical results.

However, classical range searching data structures cannot give detailed statistical information about the set of points contained inside the query region, unless one opts to report the entire subset of points inside the query range, which could be very expensive if the set is large. Because of this, there have been a number of attempts to answer more informative queries. For example, "range median" queries have received quite a bit of attention [20, 10, 18]. Note that the median is the same as 0.5-quantile and thus these can be considered the

first attempts at answering quantile queries. However, optimal solution (linear space and logarithmic query time) to exact range median queries has only be found in 1D [10]. For higher dimensions, to the best of our knowledge, the only known technique is to reduce the problem to several range counting instances [10, 13], and it is a major open problem in the range searching field to find efficient data structures for exact range counting. Due to this barrier, the approximate version of the problem [9] has been studied.

Data summary queries have also received some amount of attention, especially in the context of geometric queries. Agarwal et al. [6] showed that the heavy hitters summary (as well as a few other data summaries) are "mergeable" and this gives a baseline solution for a lot of different queries in higher dimensions, although a straightforward application of their techniques gives sub-optimal dependency on ε. In particular, for $d = 2$ and for halfspace (or simplex) queries it yields a linear-space data structure with $O(\frac{\sqrt{n}}{\varepsilon})$ query time. For $d = 3$ the query time will be $O(n^{2/3}/\varepsilon)$. In general, in the naive implementation, the query time will be $O(f(n)/\varepsilon)$ where $f(n)$ is the query time of the corresponding "baseline" range searching query (see Table 1 for more information). A more efficient approach towards merging of summaries was taken by [17] where they study the problem in a communication complexity setting, however, it seems possible to adopt their approach to a data structure as well, in combination with standard application of partition trees; after building an optimal partition tree, for any node v in the tree, consider it as a player in the communication problem with the subset of points in the subtree of v as its input. At the query time, after identifying $O(n^{2/3})$ subsets that cover the query range, the goal would be to merge all the summaries involved. By plugging the results in [17] this can result in a linear-space data structure with query time of $\tilde{O}(n^{2/3} + n^{1/6}\varepsilon^{-3/2})$.

The issue of building optimal data structures for range summary queries was only tackled in 1D by Wei and Yi [24]. They built a data structure for answering a number of summary queries, including heavy hitters queries, and showed it is possible to obtain an optimal data structure with $O(n)$ space and $O(\log n + 1/\varepsilon)$ query time. Beyond this, only sub-optimal solutions are available. Recently, there have been efforts to tackle "range sampling queries" where the goal is to extract k random samples from the set $|P \cap \gamma|$ [3, 4, 16]. In fact, one of the main motivations to consider range sampling queries was to gain information about the distribution of the point set inside the query [3]. In particular, range sampling provides a general solution for obtaining a "data summary" and for example, it is possible to solve the heavy hitters query problem. However, it has a number of issues, in particular, it requires sampling at least $1/\varepsilon^2$ points from the set $|P \cap \gamma|$, and even then it will only provide a Monte Carlo type approximation which means to boost the probabilistic guarantee, even more points need to be sampled. For example, to get a high probability guarantee, $\Omega(\varepsilon^{-2}\log n)$ samples are required.

Type-2 Color Counting. These queries were introduced in 1995 by Gupta et al. [15] within the area of "colored range counting". In this problem, given a set of colored points, we want to report the frequencies of all the colors that appeared in a given query range. This is a well-studied problem, but mostly in the orthogonal setting, see e.g., [11].

AHHS queries can be viewed as approximate type-2 color counting queries but with an additive error. Consider a query with k points. If we allow error εk in type-2 counting, then we can ignore colors with frequencies fewer than εk but otherwise we have to report frequencies with error εk, which is equivalent to answering an AHHS query.

Other Related Problems. Karpinski and Nekrich [19] studied the problem of finding the most frequent colors in a given (orthogonal) query range. This problem has received further attention in the community [8, 7, 14]. But the problem changes fundamentally when we introduce approximations.

The Model of Computation. Our model of computation is the real RAM where we have access to real registers that can perform the standard operations on real numbers in constant time, but we also have access to $w = \Theta(\log n)$ bits long integer registers that can perform the standard operations on integers and extra nonstandard operations which can be implemented by table lookups since we only need binary operations on fewer than $\frac{1}{2}\log n$ bits. Note that our data structure works when the input coordinates are real numbers, however, at some point, we will make use of the capabilities of our model of computation to manipulate the bits inside its integer registers.

1.2 Our Contributions

Our main results and a comparison with the previously known results are shown in Table 1.

Overall, we obtain a series of new results for 3D AHHS and AQS query problems which improve the current results via mergeability and independent range sampling [6, 4] by up to a huge multiplicative $n^{\Omega(1)}$ factor in query time with almost the same linear-space usage. This improvement is quite nontrivial and requires an innovative combination of known techniques like the shallow cutting lemma, the partition theorem, ε-approximations, as well as some new ideas like bit-packing for nonorthogonal queries, solving AQS query problem using AHHS instances, rank-preserving geometric sampling and so on.

For dominance queries, we obtain the first optimal results. When $\varepsilon^{-1} = O(\log n)$ our halfspace AHHS results are also optimal. Note that for small values of ε, our halfspace AHHS results yield significant improvements in the query time over the previous approaches. Along the way, we also show improved results of the above problems for 2D as well as a slightly improved exact type-2 simplex color counting result.

2 Preliminaries

In this section, we introduce the main tools we will use in our results. For a comprehensive introduction to the tools we use, see the full version.

2.1 Shallow Cuttings and Approximate Range Counting

Given a set H of n hyperplanes in $\mathbb{R}^3$, the level of a point $q \in \mathbb{R}^3$ is the number of hyperplanes in H that pass below q. We call the locus of all points of level at most k the $(\leq k)$-level and the boundary of the locus is the k-level. A shallow cutting $\mathscr{C}$ for the $(\leq k)$-level of H (or a k-shallow cutting for short) is a collection of disjoint cells (tetrahedra) that together cover the $(\leq k)$-level of H with the property that every cell $C \in \mathscr{C}$ in the cutting intersects a set H_C, called the conflict list of C , of $O(k)$ hyperplanes in H. The shallow cutting lemma is the following.

▶ **Lemma 1.** *For any set of n hyperplanes in $\mathbb{R}^3$ and a parameter k, there exists an $O(k/n)$-shallow cutting of size $O(n/k)$ that covers the $(\leq k)$-level. The cells in the cutting are all vertical prisms unbounded from below (tetrahedra with a vertex at $(0, 0, -\infty)$).*

Table 1 Our main results compared with Mergeability-based [6] and Independent Range Sampling (IRS)-based [4] solution. The IRS-based solutions are randomized with success probability $1-\delta$ for a parameter $0<\delta<1$. F is the number of colors of the input. $w=\Theta(\log n)$ is the word size of the machine. † indicates optimal solutions.

Summary Query Types	Space	Query Time	Remark
Type-2 Simplex Color Counting	$O(n)$	$O\left(n^{1-\frac{1}{d}}+\frac{n^{1-\frac{1}{d}}F^{\frac{1}{d}}}{w^{\alpha}}\right)$	New
3D AHHS Halfspace	$O(n)$ $O(n)$ $O(n)$ $\boldsymbol{O(n\log_w \frac{1}{\varepsilon})}$	$O(\log n+\frac{1}{\varepsilon}n^{2/3})$ $\tilde{O}(n^{2/3}+\frac{1}{\varepsilon^{3/2}}n^{1/6})$ $O(\log n+\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ $\boldsymbol{O(\log n+\frac{1}{\varepsilon})}$	Mergeability-based [6] Monte Carlo [17] IRS-based [4] **New**
3D AHHS Dominance	$O(n)$ $O(n)$ $\boldsymbol{O(n)}$	$O(\log n+\frac{1}{\varepsilon}\log^3 n)$ $O(\log n+\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ $\boldsymbol{O(\log n+\frac{1}{\varepsilon})}$	Mergeability-based [6] IRS-based [4] **New**†
3D AQS Halfspace	$O(n)$ $O(n)$ $\boldsymbol{O(n\log^2\frac{1}{\varepsilon}\log_w\frac{1}{\varepsilon})}$	$O(\log n+\frac{1}{\varepsilon}n^{2/3}\log(\varepsilon n))$ $O(\log n+\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ $\boldsymbol{O(\log n+\frac{1}{\varepsilon}\log^2\frac{1}{\varepsilon})}$	Mergeability-based [6] IRS-based [4] **New**
3D AQS Dominance	$O(n)$ $O(n)$ $\boldsymbol{O(n)}$	$O(\log n+\frac{1}{\varepsilon}\log^3 n\log(\varepsilon n))$ $O(\log n+\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ $\boldsymbol{O(\log n+\frac{1}{\varepsilon})}$	Mergeability-based [6] IRS-based [4] **New**†

Furthermore, we can construct these cuttings for all k of form a^i simultaneously in $O(n\log n)$ time for any $a>1$. Given any point $q\in\mathbb{R}^3$, we can find the smallest level k that is above q as well the cell containing q in $O(\log n)$ time.

The above can also be applied to dominance ranges, which are defined as below. Given two points p and q in $\mathbb{R}^d$, p dominates q if and only if every coordinate of p is larger or equal to that of q. The subset of $\mathbb{R}^d$ dominated by p is known as a *dominance range*. When the query range in a range searching problem is a dominance range, we refer to it as a *dominance query*. As observed by Chan et al. [12], dominance queries can be simulated by a halfspace queries and thus Lemma 1 applies to them. See the full version for details.

We obtain the approximate version of the range counting result using shallow cuttings.

▶ **Theorem 2** (Approximate Range Counting [2]). *Let P be a set of n points in $\mathbb{R}^3$. One can build a data structure of size $O(n)$ for halfspace or dominance ranges such that given a query range γ, one can report $|\gamma\cap P|$ in $O(\log n)$ time with error $\alpha|\gamma\cap P|$ for any constant $\alpha>0$.*

2.2 ε-approximation

Another tool we will use is ε-approximation, which is a useful sampling technique:

▶ **Definition 3.** *Let (P,Γ) be a finite set system. Given any $0<\varepsilon<1$, a set $A\subseteq P$ is called an ε-approximation for (P,Γ) if for any $\gamma\in\Gamma$, $\left|\frac{|\gamma\cap A|}{|A|}-\frac{|\gamma\cap P|}{|P|}\right|\leq\varepsilon$.*

The set A above allows us to approximate the number of points of $\gamma\cap P$ with additive error $\varepsilon|P|$ by computing $|\gamma\cap A|$ exactly; essentially, ε-approximations reduce the approximate counting problem on the (big) set P to the exact counting problem on the (small) set A.

It has been shown that small-sized ε-approximations for set systems formed by points and halfspaces/dominance ranges exist:

▶ **Theorem 4** (ε-approximation [21, 22]). *There exist ε-approximations of size $O(\varepsilon^{-\frac{2d}{d+1}})$ and $O(\varepsilon^{-1}\log^{d+1/2}\varepsilon^{-1})$ for halfspace and dominance ranges respectively.*

3 Approximate Heavy Hitter Summary Queries

We solve approximate quantile summary (AQS) queries using improved results for approximate heavy hitter summary (AHHS) queries. We sketch the main ideas of our new AHHS solutions in this section and refer the readers to the full version for details. For the clarity of description, we use ε_0 to denote the target error for the AHHS queries. We will reserve ε as a general error parameter. We show the following.

▶ **Theorem 5.** *For $d = 3$, the approximate halfspace heavy hitter summary queries can be answered using $O(n\log_w(1/\varepsilon_0))$ space and with the optimal $O(\log n + 1/\varepsilon_0)$ query time.*

▶ **Theorem 6.** *For $d = 2$, the approximate halfspace heavy hitter summary queries can be answered using $O(n\log\log_w(1/\varepsilon_0))$ space and with the optimal $O(\log n + 1/\varepsilon_0)$ query time.*

▶ **Theorem 7.** *For $d = 2, 3$, the approximate dominance heavy hitter summary queries can be answered using the optimal $O(n)$ space and with the optimal $O(\log n + 1/\varepsilon_0)$ query time.*

3.1 Base Solution

The above results are built from a *base solution*, which solves the following problem:

▶ **Problem 3** (Coarse-Grained AHHS Queries). *Let P be a set of points in $\mathbb{R}^d$, each associated with a color. The problem is to store P in a structure such that given a query range q, one can estimate the frequencies of colors in $q \cap P$ with an additive error up to $\varepsilon|P|$ efficiently for some parameter $0 < \varepsilon < 1$.*

Note that here we allow more error (since the error is defined in the entire point set). To solve Problem 3, one crucial component we need is a better (exact) type-2 color counting structure for halfspaces. We combine several known techniques in a novel way with bit-packing to get the following theorem. See the full version for details.

▶ **Theorem 8.** *Given an integer parameter F, a set P of n points in $\mathbb{R}^d$ where each point is assigned a color from the set $[F]$, one can build a linear-sized data structure, such that given a query simplex q, it can output the number of times each color appears in $P \cap q$ in total time $\max\{O(n^{(d-1)/d}), O(n^{(d-1)/d}F^{1/d}/w^{\alpha})\}$, for some appropriate constant α and word size w.*

The main idea for getting a base solution is relatively straightforward. We group colors according to their frequencies where each group contains colors of roughly equal frequencies. However, we have to be careful about the execution and the analysis is a bit tricky. For example, if we place all the points in one copy of the data structure of Theorem 8, then we will get a sub-optimal result. However, by grouping the points correctly, and being stringent about the analysis, we can obtain the following.

▶ **Theorem 9.** *For $d \geq 3$, Problem 3 for simplex queries (the intersection of $d+1$ halfspaces) can be solved with $O(X)$ space for $X = \min\{|P|, \varepsilon^{-\frac{2d}{d+1}}\}$ and a query time of*

$$O\left(\frac{|P|^{1-\frac{2}{d-1}}}{w^{\alpha}\varepsilon^{\frac{2}{d-1}}}\right) + O\left(X^{\frac{d-1}{d}}\right)$$

where w is the word-size of the machine and α is some positive constant.

The main challenge is that we have two cases for the size of an ε-approximation on n points since it is bounded by $\min\left\{n, O(\varepsilon^{-\frac{2d}{d+1}})\right\}$ and also two cases for the query time of Theorem 8. However, the main idea is that since the total error budget is $\varepsilon|P|$, we can afford to pick a larger error parameter $\varepsilon_i = \frac{\varepsilon|P|}{|P_i|}$, where P_i is the set of points with color i. The details are presented in the full version.

3.2 Solving AHHS Queries

We first transform the problem into the dual space. So the point set P becomes a set H of hyperplanes and any query halfspace becomes a point q. We want to find approximate heavy hitters of hyperplanes of H below q. Here, we remark that obtaining a data structure with $O(n \log \frac{1}{\varepsilon_0})$ space is not too difficult: build a hierarchy of shallow cuttings covering level $2^i/\varepsilon_0$ for $i = 0, 1, \cdots, \log(\varepsilon_0 n)$ of the arrangement of H. For each shallow cutting cell Δ, we build the previous base structure for the conflict list $\mathcal{S}_\Delta$ for a parameter $\varepsilon = \varepsilon_0/c$ for a big enough constant c. Then, observe that for queries below level ε_0^{-1}, we can spend $O(\log n + \frac{1}{\varepsilon_0})$ time to find all the hyperplanes passing below the query and answer the AHHS queries explicitly and also for shallow cutting levels above level $\varepsilon_0^{-3/2}$, the total amount of space used by the base solution is $O(n)$. Thus, it turns out that the main difficulty lies in handling the levels between ε_0^{-1} and $\varepsilon_0^{-3/2}$.

To reduce the space to $O(\log_w \frac{1}{\varepsilon_0})$, recall that in the query time of the base structure, we have two terms $O(1/(\varepsilon_0 w^\alpha))$ and $O(X^{2/3})$. Observe that we can afford to set ε to be roughly ε_0/w^α and the first term will still be $O(\varepsilon_0^{-1})$ because we are at level below $\varepsilon_0^{-3/2}$, we have $X < \varepsilon_0^{-3/2}$ and so the second term will always be $O(\varepsilon_0^{-1})$! The effect of setting $\varepsilon = \varepsilon_0/w^\alpha$ is that now the base structure we built for a cell can output frequencies with a factor of w^α more precision, meaning it can be used for a factor of w^α many more levels. So we only need to build the base structure for shallow cuttings built for a factor of w^α! This gives us the $O(n \log_w \frac{1}{\varepsilon_0})$ space bound. Of course, here the output has size $O(\varepsilon^{-1}) = O(w^\alpha \varepsilon_0^{-1})$ and we cannot afford to examine all these colors. The final ingredient here is that we can maintain a list of $O(\varepsilon_0^{-1})$ candidate colors using shallow cuttings built for a factor of 2.

We remark that although the tools are standard, the combination of the tools and the analysis are quite nontrivial. Also when we have $\Theta(1/\varepsilon_0)$ heavy hitters, our query time is optimal. It is an interesting open problem if the query time can be made output sensitive.

4 Approximate Quantile Summary Queries

In this section, we solve Problem 1. We first show a general technique that uses our solution to AHHS queries solution to obtain an efficient solution for AQS queries. We show that for 3D halfspace and dominance ranges we can convert the solution for AHHS queries to a solution for AQS queries with an $O(\log^2 \frac{1}{\varepsilon})$ blow up in space and time. Then in Section 4.2, we present an optimal solution for dominance ranges based on a different idea.

First, we show how to solve AQS queries using the AHHS query solution. We describe the data structure for halfspaces, since as we have mentioned before, the same can be applied to dominance ranges in 3D as well. The high level idea of our structure is as follows: We first transform the problem into the dual space. This yields the problem instance where we have n weighted hyperplanes and given a query point q, we would like to extract an approximate quantile summary for the hyperplanes that pass below q. To do this, we build hierarchical shallow cuttings. For each cell in each cutting, we collect the hyperplanes in its conflict list and then divide them into $O(\frac{1}{\varepsilon_0})$ groups according to the increasing order of their weights.

Given a query point in the dual space, we first find the cutting and the cell containing it, and then find an approximated rank of each group, within the subset below the query. This is done by generating an AHHS problem instance and applying Theorem 5. We construct the instance in a way such that the rank approximated will only have error small enough such that we can afford to scan through the groups and pick an arbitrary hyperplane in corresponding groups to form an approximate ε_0-quantile summary.

4.1 The Data Structure and the Query Algorithm

We dualize the set P of n input points which gives us a set $H = \overline{P}$ of n hyperplanes. We then build a hierarchy of shallow cuttings where the i-th shallow cutting, $\mathscr{C}_i$, is a k_i-shallow cutting where $k_i = \frac{2^i}{\varepsilon_0}$, for $i = 0, 1, 2, \cdots, \log(\varepsilon_0 n)$. Consider a cell Δ in the i-th shallow cutting and its conflict list $\mathcal{S}_\Delta$. Let $\epsilon = \frac{\varepsilon_0}{c}$ for a big enough constant c. We partition $\mathcal{S}_\Delta$ into $t = \frac{1}{\epsilon}$ groups $G_1, G_2, \cdots, G_t$ sorted by weight, meaning, the weight of any hyperplane in G_j is no larger than that of any hyperplane in G_{j+1} for $j = 1, 2, \cdots, t-1$.

For each group G_j, we store the smallest weight among the hyperplanes it contains, as its representative. To make the description shorter, we make the simplifying assumption that t is a power of 2 (if not, we can add some dummy groups). We arrange the groups G_j as the leaves of a balanced binary tree $\mathcal{T}$ and let $V(\mathcal{T})$ be the set of vertices of $\mathcal{T}$. Next, we build the following set A_Δ of colored hyperplanes, associated with Δ: Let $\varepsilon' = \frac{\epsilon}{\log^2 t}$. For every vertex $v \in V(\mathcal{T})$, let G_v to be the set of all the hyperplanes contained in the subtree of v; we add an ε'-approximation, E_v, of G_v to A_Δ with color v. Using Theorem 5, we store the points dual to hyperplanes in A_Δ in a data structure Ψ_Δ for AHHS queries with error parameter ε'. This completes the description of our data structure.

The query algorithm. A given query q is answered as follows. Let us quickly go over the standard parts: We consider the query in the dual space and thus q is considered to be a point. Let k be the number of hyperplanes passing below q. Observe that by Theorem 2, we can find a $(1+\alpha)$ factor approximation, k^*, of k in $O(\log n)$ time for any constant α, using a data structure that consumes linear space. This allows us to find the first k_i-shallow cutting $\mathscr{C}_i$ with $k_{i-1} < k \leq k_i$. The cell $\Delta \in \mathscr{C}_i$ containing q can also be found in $O(\log n)$ time using a standard point location data structure (e.g., see [1]).

The interesting part of the query is how to handle the query after finding the cell Δ. Let H_q be the subset of H that lies below q. Recall that $\mathcal{S}_\Delta$ is the subset of H that intersects Δ. The important property of Δ is that $H_q \subset \mathcal{S}_\Delta$ and also $|\mathcal{S}_\Delta| = O(|H_q|) = O(k)$.

We query the data structure Ψ_Δ built for Δ to obtain a list of colors and their approximate counts where the additive error in the approximation is at most $\varepsilon'|A_\Delta|$. To continue with the description of the query algorithm, let us use the notation g_j to denote the subset of G_j that lies below q, and let $g = \cup_{j=1}^{t} g_j$ and thus $|g| = k$.

Note that while the query algorithm does not have direct access to g, or k, we claim that using the output of the data structure Ψ_Δ, we can calculate the approximate rank of the elements of g_i within g up to an additive error of $\varepsilon_0 k$. Again, we can use tree $\mathcal{T}$ to visualize this process. Recall that in Ψ_Δ, every vertex $v \in V(\mathcal{T})$ represents a unique color in the data structure Ψ_Δ and the data structure returns an AHHS summary with error parameter ε'. This allows us to estimate the number of elements of E_v that pass below q with error $\varepsilon'|A_\Delta|$ and since E_v is an ε'-approximation of G_v, this allows us to estimate the number of elements of G_v that pass below q with error at most $2\varepsilon'|A_\Delta|$. Consider the leaf node that represents $g_j \subset G_j$ and the path π that connects it to the root of $\mathcal{T}$. The approximate rank, r_j, of g_j is calculated as follows. Consider a subtree with root u that hang to the left of the path π (as

shown in Figure 1). If color u does not appear in the output of the AHHS query, then we can conclude that at most $2\varepsilon'|A_\Delta|$ of its hyperplanes pass below q and in this case we do nothing. If it does appear in the output of the AHHS query, then we know the number of hyperplanes in its subtree that pass below q up to an additive error of $2\varepsilon'|A_\Delta|$ and in this case, we add this estimate to r_j. In both cases, we are off by an additive error of $2\varepsilon'|A_\Delta|$. We repeat this for every subtree that hangs to the left of π. The number of such subtree is at most $\log t$ and thus the total error is at most $2\varepsilon'|A_\Delta|\log t$. Now observe that

$$2\varepsilon'|A_\Delta|\log t = 2\frac{\epsilon}{\log^2 t}\cdot \log t|\mathcal{S}_\Delta|\cdot\log t = O(\varepsilon k) = O\left(\frac{\varepsilon_0 k}{c}\right) \leq \varepsilon_0 k$$

which follows by setting c large enough and observing the fact that $|A_\Delta| \leq \log t|\mathcal{S}_\Delta|$ since every hyperplane in $\mathcal{S}_\Delta$ is duplicated $\log t$ times.

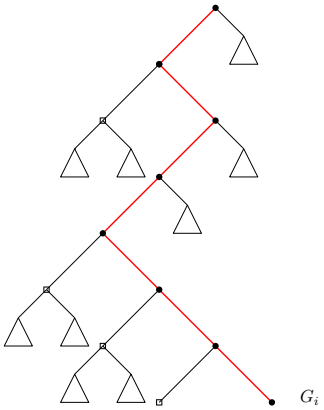


Figure 1 Compute the Approximate Rank of a Group: The approximated rank of G_i is calculated as the sum of all the approximate counts of square nodes.

We are now almost done. We just proved that in each g_i, we know the rank of its elements within g up to an additive error of $\varepsilon_0 k$. This means that picking one element from each G_i gives us a super-set of an AQS; in the last stage of the query algorithm we simply prune the unnecessary elements as follows: We scan all the leave in $\mathcal{T}$ from left to right, i.e., consider the group G_j for $j = 1$ to t and compute the quantile summary in a straightforward fashion. To be specific, we initialize a variable $j' = 0$ and then consider G_j, for $j = 1$ to t. The first time r_j exceeds a quantile boundary, i.e., $r_j \geq j'\varepsilon_0 k^*$, we add the hyperplane with the lowest weight in G_j to the approximate ε_0-quantile summary, and then increment j'.

Analysis

Based on the previous paragraph, the correctness is established. Thus, it remains to analyze the space and query complexities. We start with the former.

Space Usage. Consider the structure Ψ_Δ built for cell Δ from a k_i-shallow cutting $\mathscr{C}_i$. Observe that $\sum_{v\in V(\mathcal{T})}|G_v| = |\mathcal{S}_\Delta|\log t$ since in the sum every hyperplane will be counted $\log t$ times. E_v is an ε'-approximation of G_v and thus

$$|E_v| \leq \min\left\{\varepsilon'^{-3/2}, G_v\right\} \tag{1}$$

which implies

$$|A_\Delta| = \sum_{v \in V(\mathcal{T})} |E_v| \leq \min\left\{\varepsilon'^{-3/2} 2t, |\mathcal{S}_\Delta| \log t\right\} \tag{2}$$

where the first part follows as there are at most $2t$ vertices in $\mathcal{T}$ and the second part follows from (1). We build an instance of Theorem 5 on the set A_Δ which by Theorem 5 uses $O(|A_\Delta| \log_w \frac{1}{\varepsilon'})$ space. Assuming Δ belongs to a k_i-shallow cutting $\mathscr{C}_i$, we have $|\mathcal{S}_\Delta| = O(k_i)$ and there are $O(n/k_i)$ cells in $\mathscr{C}_i$. Observe that

$$\sum_{\Delta \in \mathscr{C}_i} |A_\Delta| = \sum_{\Delta \in \mathscr{C}_i} \min\left\{\varepsilon'^{-3/2} 2t, |\mathcal{S}_\Delta| \log t\right\} = \sum_{\Delta \in \mathscr{C}_i} O\left(\min\left\{\varepsilon'^{-3/2} t, k_i \log t\right\}\right) = O\left(\min\left\{\frac{n}{k_i}\varepsilon_0^{-3}, n \log \frac{1}{\varepsilon_0}\right\}\right). \tag{3}$$

Thus, the total space used for $\mathscr{C}_i$ is

$$O\left(\min\left\{\frac{n}{k_i}\varepsilon_0^{-3} \log_w \frac{1}{\varepsilon_0}, n \log_w \frac{1}{\varepsilon_0} \log \frac{1}{\varepsilon_0}\right\}\right).$$

Finally, observe that there can be at most $O(\log \frac{1}{\varepsilon_0})$ levels where the second term dominates; to be specific, at least when k_i exceeds ε_0^{-4}, the first term dominates and the total space used by those levels is $O(n)$ as k_i's form a geometric series. So the total space usage of our structure is $O(n \log^2 \frac{1}{\varepsilon_0} \log_w \frac{1}{\varepsilon_0})$.

Query Time. By Lemma 1, we can find the desired cutting cell in time $O(\log n)$. Next, we query the data structure Ψ_Δ which by Theorem 5 uses $O(\log n + \varepsilon'^{-1}) = O(\log n + \frac{\epsilon}{\log^2 t}) = O(\log n + \frac{1}{\varepsilon_0} \log^2 \frac{1}{\varepsilon_0})$ query time. Scanning the groups and pruning the output of the data structure Ψ_Δ takes asymptotically smaller time and thus it can be absorbed in the above expression. Therefore, we obtain the following result.

▶ **Theorem 10.** *Given an input consisting of an error parameter ε_0, and a set P of n points in $\mathbb{R}^3$ where each point $p \in P$ is associated with a weight w_p from a totally ordered universe, one can build a data structure that uses $O(n \log^2 \frac{1}{\varepsilon_0} \log_w \frac{1}{\varepsilon_0})$ space such that given any query halfspace h, it can answer an AQS query with parameter ε_0 in time $O(\log n + \frac{1}{\varepsilon_0} \log^2 \frac{1}{\varepsilon_0})$.*

For the case of 2D, we can just replace Ψ_Δ with the structure in Theorem 6, and we immediately get the following:

▶ **Theorem 11.** *Given an input consisting of an error parameter ε_0, and a set P of n points in $\mathbb{R}^2$ where each point $p \in P$ is associated with a weight w_p from a totally ordered universe, one can build a data structure that uses $O(n \log^2 \frac{1}{\varepsilon_0} \log \log_w \frac{1}{\varepsilon_0})$ space such that given any query halfspace h, it can answer an AQS query with parameter ε_0 in time $O(\log n + \frac{1}{\varepsilon_0} \log^2 \frac{1}{\varepsilon_0})$.*

4.2 Dominance Approximate Quantile Summary Queries

Now we turn our attention to dominance ranges. We will show a structure similar to that for halfspace queries. The main difference is that we now use exact type-2 color counting as an auxiliary structure to estimate the rank of each group. This saves us roughly $\log^2 \frac{1}{\varepsilon_0}$ factors for both space and query time and so we can answer quantile queries in the optimal $O(\log n + \frac{1}{\varepsilon_0})$ time. To reduce the space to linear, we need more ideas. We first present a suboptimal but simpler structure to demonstrate our main idea. Then we modify this structure to get the desired optimal structure. We use shallow cuttings in the primal space.

4.2.1 A Suboptimal $O(n \log\log \frac{1}{\varepsilon_0})$ Space Solution

We first describe a data structure that solves the dominance AQS problem with $O(n \log\log \frac{1}{\varepsilon_0})$ space and the optimal $O(\log n + \frac{1}{\varepsilon_0})$ query time.

Rank-Preserving Approximation for Weighted Points. Let S be a weighted point set where every point has been assigned a weight from a totally ordered universe. Let $r_S(p)$ be the rank of a point p in the set S. Consider a geometric set system $(P, \mathscr{D})$, where P is a set of weighted points in $\mathbb{R}^3$ and $\mathscr{D}$ is a family of subsets of P induced by 3D dominance ranges. We mention a way to construct a sample A for P and a parameter ϵ such that

$$\left| \frac{r_{P\cap\mathcal{D}}(p)}{|P|} - \frac{r_{A\cap\mathcal{D}}(p)}{|A|} \right| \leq \epsilon \tag{4}$$

for any point $p \in P$ and any range $\mathcal{D} \in \mathscr{D}$. First note that taking an ϵ-approximation for P does **not** work since it does not take the weights of P into consideration. Our simple but important observation is that we can lift the points P into 4D by adding their corresponding weights as the fourth coordinate. Let us call this new point set P' and let $(P', \mathscr{D}')$ be the set system in 4D induced by 4D dominance ranges. Consider an ϵ-approximation A' for P' and let A be the projection of A' into the first three dimensions (i.e., by removing the weights again). A will be our sample for P and to distinguish it from an unweighted approximation, we call it *rank-preserving ϵ-approximation*. Indeed, for any point $p \in P$ with weight w_p and any $\mathcal{D} \in \mathscr{D}$, $r_{P\cap\mathcal{D}}(p)$ (resp. $r_{A\cap\mathcal{D}}(p)$) is equal to the number of points in P' (resp. A') contained in 4D dominance range $\mathcal{D} \times (-\infty, w_p)$. By the definition of ϵ-approximation, property (4) holds.

We now turn our attention to the AQS for 3D dominance queries.

The Data Structure and The Query Algorithm. Similar to the structure we presented for halfspace queries, we build $\frac{2^i}{\varepsilon_0}$-shallow cuttings for $i = 0, 1, \cdots, \log(\varepsilon_0 n)$. Let $\kappa = O(1)$ be the constant such that $O(\frac{1}{\varepsilon_0}\log^\kappa \frac{1}{\varepsilon_0})$ is the size of the ε_0-approximation for dominance ranges in 4D. Consider one k-shallow cutting $\mathscr{C}$. We consider two cases:

- If $k \leq \frac{1}{\varepsilon_0}\log^\kappa \frac{1}{\varepsilon_0}$, for each cell Δ in the cutting $\mathscr{C}$, we collect the points in its conflict list $\mathcal{S}_\Delta$ and divide them into $t = \frac{1}{\epsilon}$ groups $G_1, G_2, \cdots, G_t$ according to their weights (meaning, the weights in G_i are no larger than weights in group G_{i+1}) where $\epsilon = \frac{\varepsilon_0}{c}$ for a big enough constant c as we did for halfspace queries.
- For $k > \frac{1}{\varepsilon_0}\log^\kappa \frac{1}{\varepsilon_0}$, we take a rank-preserving ϵ-approximation of $\mathcal{S}_\Delta$ first, and then divide the approximation into $t = \frac{1}{\epsilon}$ groups, just like the above case. Again, for each group, we store the smallest weight among the points it contains.

We build the following structure for each cell Δ.

Let N be the number of points in all the t groups we generated for a cell Δ. We collect groups $G_{i\cdot\alpha+1}, G_{i\cdot\alpha+2}, \cdots, G_{(i+1)\cdot\alpha}$ into a cluster $\mathcal{C}_i$ for each $i = 0, 1, \cdots, t/\alpha - 1$ where $\alpha = (\log\log \frac{1}{\varepsilon_0})^3$. For each group j in cluster $\mathcal{C}_i$ for $j = 1, 2, \cdots, \alpha$, we color the points in the group with color j. Then we build the following type-2 color counting structure Ψ_i for $\mathcal{C}_i$. Let N_i be the total number of points in $\mathcal{C}_i$:

- First, we store three predecessor search data structures, one for each coordinate. This allows us to map the input coordinates as well as the query coordinates to rank space.
- Next, we build a grid of size $\sqrt[3]{N_i} \times \sqrt[3]{N_i} \times \sqrt[3]{N_i}$ such that each slice contains $\sqrt[3]{N_i^2}$ points. For each grid point, we store the points it dominates in a frequency vector using the compact representation.

- Finally, we recurse on each grid slab (i.e., three recursions, one for each dimension). The recursion stops when the number of points in the subproblem becomes smaller than $N_* = N_i^{\eta}$ for some small enough constant η.
- For these "leaf" subproblems, note that the total number of different answers to queries is bounded by $O(N_i^{3\eta})$. We build a lookup table which records the corresponding frequency vectors for these answers. Note that since at every step we do a rank space reduction, the look up can be simply done in $O(1)$ time, after reducing the coordinates of the query to rank space.

The query algorithm. Given a query q, we first locate the grid cell C containing q and this gives us three ranks. Using the ranks for x and y, we obtain an entry and using the rank of z, we find the corresponding word and the corresponding frequency vector stored in the lower corner of C. We get three more frequency vectors by recursing to three subproblems. We merge the three frequency vectors to generate the final answer. This completes the description of the structure we build for each family $\mathcal{C}_i$.

To answer a query q, we first find the first shallow cutting level above q and the corresponding cutting cell Δ. We then query the data structure described above to get the count the number of points dominated by q in each of the t groups. Then by maintaining a running counter, we scan through the t groups from left to right to construct the approximate ε_0-quantile summary.

Space Usage. For the space usage, note that there are N_i grid points in each recursive level and the recursive depth is $O(1)$. There are α colors and the frequency of a color is no more than N_i. So the total number of words needed to store frequency vectors is $O(N_i \frac{\alpha \log N_i}{w})$. When the problem size is below N_i^{η}, for each subproblem, we store a lookup table using $O(N_i^{3\eta} \frac{\alpha \log N_i}{w})$ words. So the total number of words used for the bottom level is $O(\frac{N_i}{N_i^{\eta}}) \cdot O(N_i^{3\eta} \frac{\alpha \log N_i}{w}) = O(N_i \frac{N_i^{2\eta} \alpha \log N_i}{w})$. Note that by our construction and $\varepsilon_0 \geq \frac{1}{n}$, $N = O(\frac{1}{\varepsilon_0} \log^{\kappa} \frac{1}{\varepsilon_0})$, $\alpha = (\log\log \frac{1}{\varepsilon_0})^3 \leq (\log\log n)^3$ and $N_i = O(\alpha \frac{N}{1/\varepsilon_0}) = O(\alpha \log^{\kappa} \frac{1}{\varepsilon_0}) = O(\alpha \log^{\kappa} n)$. Since by assumption, $w = \Omega(\log n)$, by picking η in $N_* = N_i^{\eta}$ to be a small enough constant, the space usage for frequency vectors satisfy

$$f(N_i) = \begin{cases} 3\sqrt[3]{N_i} f(\sqrt[3]{N_i^2}) + O(\frac{N_i}{w^{1-o(1)}}), \text{for } N_i \geq N_* \\ O(\frac{N_i}{w^{1-\beta}}), \text{otherwise} \end{cases},$$

for some constant $0 < \beta < 1$, which solves to $O(\frac{N_i (\log N_i)^3}{w^{1-\beta}}) = O(\frac{N_i}{w^{1-\tau}})$ for some constant $0 < \tau < 1$. Since the recursive depth is $O(1)$, the space usage for all the predecessor searching structures is $O(N_i)$. Therefore the space usage of Ψ_i is $O(N)$. So the total space for each shallow cutting cell Δ is bounded by $\frac{N}{N_i} \cdot O(N_i) = O(N)$.

For $k_i \geq \frac{1}{\varepsilon_0} \log^{\kappa} \frac{1}{\varepsilon_0}$, $N = O(\frac{1}{\varepsilon_0} \log^{\kappa} \frac{1}{\varepsilon_0})$. So the total space usage for them is bounded by

$$\sum_{i=\kappa \log\log \frac{1}{\varepsilon_0}}^{\varepsilon_0 n} O\left(\frac{n}{k_i}\right) \cdot O(N) = \sum_{i=\kappa \log\log \frac{1}{\varepsilon_0}}^{\varepsilon_0 n} O\left(\frac{n\varepsilon_0}{2^i}\right) \cdot O\left(\frac{1}{\varepsilon_0} \log^{\kappa} \frac{1}{\varepsilon_0}\right) = O(n).$$

For $k_i < \frac{1}{\varepsilon_0} \log^{\kappa} \frac{1}{\varepsilon_0}$, $N = k_i$ and so we have space bound

$$\sum_{i=0}^{\kappa \log\log \frac{1}{\varepsilon_0}} O\left(\frac{n}{k_i}\right) \cdot O(N) = \sum_{i=0}^{\kappa \log\log \frac{1}{\varepsilon_0}} O\left(\frac{n}{k_i}\right) \cdot O(k_i) = O\left(n \log\log \frac{1}{\varepsilon_0}\right).$$

This completes our space bound proof.

Query Time. For the query time, we first spend $O(\log n)$ time to find an appropriate shallow cutting level and the corresponding cell by the property of shallow cuttings. Then we query Ψ_i for $i = 0, 1, \cdots, t/\alpha - 1$ to estimate the count for each group in the cell. For each Ψ_i, note that each predecessor searching takes $O(\log N_i)$ time. Also each frequency vector can fit in one word and so we can merge two frequency vectors in time $O(1)$. This gives us the following recurrence relation for the query time

$$g(N_i) = \begin{cases} 3g(\sqrt[3]{N_i^2}) + O(\log N_i), \text{for } N_i \geq N_* \\ O(\log N_i), \text{otherwise} \end{cases},$$

which solves to $O((\log N_i)^3) = O((\log\log \frac{1}{\varepsilon_0})^3) = O(\alpha)$. Since we need to query t/α such data structures to get the count for all groups, the total query time for count estimation is $O(t) = O(1/\varepsilon_0)$. Then we scan through the groups and report the approximate quantiles which takes again $O(1/\varepsilon_0)$ time. So the total query time is $O(\log n + \frac{1}{\varepsilon_0})$.

Correctness. Given a query q, let k be the actual number of points dominated by q. By the property of shallow cuttings, we find a cell Δ containing q in the shallow cutting level k_i above it such that $k \leq k_i \leq 2k$. When $k_i < \frac{1}{\varepsilon_0}\log^\kappa \frac{1}{\varepsilon_0}$, after we estimate the count in each group, since the estimation is exact and each group has size $\frac{|\mathcal{S}_\Delta|}{t} = \frac{\varepsilon_0|\mathcal{S}_\Delta|}{c}$, each quantile we output will have error at most $\frac{\varepsilon_0|\mathcal{S}_\Delta|}{c}$. For $k_i \geq \frac{1}{\varepsilon_0}\log^\kappa \frac{1}{\varepsilon_0}$, we introduce error $\epsilon|\mathcal{S}_\Delta|$ in the ϵ-approximation, but since we use exact counting for each group, the total error will not increase as we add up ranks of groups. So the total error is at most $\frac{2\varepsilon_0|\mathcal{S}_\Delta|}{c}$. In both cases, the total error is at most $\varepsilon_0 k$ for a big enough c.

4.2.2 An Optimal Solution for 3D Dominance AQS

In this section, we modify the data structure in the previous section to reduce the space usage to linear. It can be seen from the space analysis that the bottleneck is shallow cuttings with $k_i \leq \frac{1}{\varepsilon_0}\log^\kappa \frac{1}{\varepsilon_0}$. For the structures built for these levels, the predecessor searching structures take linear space at each level which leads to a super linear space usage in total. To address this issue, we do a rank space reduction for points in the cells of these levels before constructing Ψ_i's so that we can use the integer register to spend sublinear space for the predecessor searching structures.

Rank Space Reduction Structure. We consider the cells in the $\frac{1}{\varepsilon_0}\log^\kappa \frac{1}{\varepsilon_0}$-shallow cutting. Let $A = \log^{\kappa+1} \frac{1}{\varepsilon_0}$. For the points in the conflict list $\mathcal{S}_\Delta$ of a shallow cutting cell $\Delta \in \mathscr{C}$, we build a grid of size $A \times A \times A$ such that each slice of the grid contains $O(1/(\varepsilon_0 \log \frac{1}{\varepsilon_0}))$ points. The coordinate of each grid point consists of the ranks of its three coordinates in the corresponding dimensions. For each of the $O(\frac{A}{\varepsilon_0 \log(1/\varepsilon_0)})$ points in $\mathcal{S}_\Delta$, we round it down to the closest grid point dominated by it. This reduces the coordinates of the points down to $O(\log\log \frac{1}{\varepsilon_0})$ bits and now we can apply the sub-optimal solution from the previous subsection which leads to an $O(n)$ space solution. To be more specific, we build the hierarchical shallow cuttings for $k_i \leq \frac{1}{\varepsilon_0}\log^\kappa \frac{1}{\varepsilon_0}$ locally for hyperplanes in $\mathcal{S}_\Delta$ and apply the previous solution with a value $\varepsilon' = \varepsilon/c$ for a large enough constant c.

Query Algorithm and the query time. The query algorithm is similar to that for the previous suboptimal solution. The only difference is that when the query q is in a shallow cutting level smaller than $\frac{1}{\varepsilon_0}\log^\kappa \frac{1}{\varepsilon_0}$, we use the rank space reduction structure to reduce q

to the rank space. Let q' be the grid point obtained after reducing q to rank space. Observe that the set of points dominated by q can be written as the union of the points dominated by q' and the subset of points dominated by q in three grid slabs of A that contain q. We get an ε'-quantile for the former set using the data structure implemented on the grid points. The crucial observation is that there are $O(\varepsilon_0^{-1}/\log\frac{1}{\varepsilon_0})$ points in the slabs containing q and thus we can afford to build an approximate ε'-quantile summary of these points in $O(\frac{1}{\varepsilon_0})$ time. We can then merge these two quantiles and return the answer as the result. By setting c in the definition of ε' small enough, we make sure that the result is a valid ε_0 quantile summary. This also yields a query time (after locating the correct cell Δ in the shallow cutting) of $O(\frac{1}{\varepsilon_0})$.

Correctness. Since we build shallow cutting $k_i \leq \frac{1}{\varepsilon_0}\log^{\kappa}\frac{1}{\varepsilon_0}$ inside each cell in $\frac{1}{\varepsilon_0}\log^{\kappa}\frac{1}{\varepsilon_0}$-shallow cutting, the transformed coordinates are consistent. As we described above, this introduces error to the counts Ψ_i's outputs, but since we correct the error explicitly afterwards, the counts we get are still exact. The remaining is the same as the suboptimal solution and so our structure finds ε_0-quantile properly.

Space Usage. For the rank space reduction structure, we need to store a predecessor searching structure for the query, which takes space linear in the number of slices which is $O(A)$. We build this structure for each cell in the A/ε_0-shallow cutting level and there are $O(n\varepsilon_0/A)$ cells in total, and so the space usage is $O(n\varepsilon_0)$. Building shallow cuttings inside each cell will only increase the space by a constant factor by the property of shallow cuttings.

For each Ψ_i, by our analysis in the suboptimal solution, the frequency vectors will take $O(\frac{N}{w^{1-\tau}})$ space. Now since the coordinates of the points and queries are integers of size at most A, it takes $O(\log A) = O(\log\log\frac{1}{\varepsilon_0})$ bits to encode a coordinate. Since the word size is $w = \Omega(\log n)$, we need only $O(\frac{N_i \log A}{w})$ space to build the predecessor searching structures for Ψ_i. In total, we spend $O(\frac{N}{w^{1-o(1)}})$ space for each shallow cutting level less than A/ε_0. So, the total space usage is $O(n)$. We conclusion this section by the following theorem.

▶ **Theorem 12.** *Given an input consisting of a parameter $\varepsilon_0 > 0$, and a set P of n points in $\mathbb{R}^3$ where each point $p \in P$ is associated with a weight w_p from a totally ordered universe, one can build a data structure that uses the optimal $O(n)$ space such that given any dominance query γ, the data structure can answer an AQS query with parameter ε_0 in the optimal query time of $O(\log n + \frac{1}{\varepsilon_0})$.*

5 Open Problems

Our results bring many interesting open problems. First, for type-2 color counting problems, we showed a linear-sized structure for simplex queries. It is not clear if the query time can be reduced with more space. It is an intriguing open problem to figure out the correct space-time tradeoff for the problem. Note that our query time in Theorem 8 depends on the number of colors in total. It is unclear if the query time can be made output-sensitive. This seems difficult and unfortunately there seems to be no suitable lower bound techniques to settle the problem. Furthermore, since improving exact simplex range counting results is already very challenging, it makes sense to consider the approximate version of the problem with multiplicative errors.

Second, for heavy-hitter queries, there are two open problems. In our solution, the space usage is optimal with up to some extra polylogarithmic factor (in $\frac{1}{\varepsilon}$). An interesting challenging open problem is if the space usage can be made linear. On the other hand, our

query time is not output-sensitive. Technically speaking, there can be less than $1/\varepsilon$ heavy hitters, and in this case, it would be interesting to see if $O(\log n + k)$ query time can be obtained for k output heavy hitters with (close to) linear space[2].

Third, for AQS queries, our data structure for halfspace ranges is suboptimal. The main reason is that we need a type-2 range counting solution as a subroutine. For halfspace ranges, our exact type-2 solution is too costly, and so we have to switch to an approximate version. This introduces some error and as a result, we need to use a smaller error parameter, which leads to extra polylogarithmic factors in both time and space. In comparison, we obtain an optimal solution for dominance AQS queries through exact type-2 counting. Currently, it seems quite challenging to improve the exact type-2 result for halfspace queries and some different ideas probably are needed to improve our results.

Finally, it is also interesting to investigate approximate quantile summaries, or heavy hitter summaries (or other data summaries or data sketches used in the streaming literature) for a broader category of geometric ranges. In this paper, our focus has been on very fast data structures, preferably those with optimal $O(\log n + \frac{1}{\varepsilon})$ query time, but we know such data structures do not exist for many important geometric ranges. For example, with linear space, simplex queries require $O(n^{(d-1)/d})$ time and there are some matching lower bounds. Nonetheless, it is an interesting open question whether approximate quantile or heavy hitter summary can be built for simplex queries in time $O(n^{(d-1)/d} + \frac{1}{\varepsilon})$ using linear or near-linear space; as we review in the introduction, the general approaches result in sub-optimal query times of $O(n^{(d-1)/d} \cdot \frac{1}{\varepsilon})$ or $O(n^{(d-1)/d} + \frac{1}{\varepsilon^2})$.

References

1 Peyman Afshani and Timothy M. Chan. Optimal halfspace range reporting in three dimensions. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 180–186, 2009.

2 Peyman Afshani, Chris Hamilton, and Norbert Zeh. A general approach for cache-oblivious range reporting and approximate range counting. *Computational Geometry: Theory and Applications*, 43:700–712, 2010. preliminary version at SoCG'09.

3 Peyman Afshani and Jeff M. Phillips. Independent Range Sampling, Revisited Again. In *Symposium on Computational Geometry (SoCG)*, pages 4:1–4:13, 2019.

4 Peyman Afshani and Zhewei Wei. Independent Range Sampling, Revisited. In *Proceedings of European Symposium on Algorithms (ESA)*, volume 87, pages 3:1–3:14, 2017.

5 Pankaj K. Agarwal. Range searching. In J. E. Goodman, J. O'Rourke, and C. Toth, editors, *Handbook of Discrete and Computational Geometry*. CRC Press, Inc., 2016.

6 Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff M. Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 23–34, 2012.

7 Djamal Belazzougui, Travis Gagie, J. Ian Munro, Gonzalo Navarro, and Yakov Nekrich. Range majorities and minorities in arrays. *Algorithmica*, 83(6):1707–1733, 2021.

8 Djamal Belazzougui, Travis Gagie, and Gonzalo Navarro. Better space bounds for parameterized range majority and minority. In *Algorithms and data structures*, volume 8037 of *Lecture Notes in Comput. Sci.*, pages 121–132. Springer, Heidelberg, 2013.

9 Prosenjit Bose, Evangelos Kranakis, Pat Morin, and Yihui Tang. Approximate range mode and range median queries. In *Proceedings of Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, 2005.

[2] We thank an anonymous referee for suggesting the "output-sensitive" version.

10 Gerth Stølting Brodal, Beat Gfeller, Allan Grønlund Jørgensen, and Peter Sanders. Towards optimal range medians. *Theoretical Computer Science*, 412(24):2588–2601, 2011.

11 Timothy M. Chan, Qizheng He, and Yakov Nekrich. Further Results on Colored Range Searching. In *Symposium on Computational Geometry (SoCG)*, volume 164, pages 28:1–28:15, 2020.

12 Timothy M. Chan, Kasper Green Larsen, and Mihai Pătraşcu. Orthogonal range searching on the RAM, revisited. In *Symposium on Computational Geometry (SoCG)*, pages 1–10, 2011.

13 Timothy M. Chan and Gelin Zhou. Multidimensional range selection. In *Algorithms and computation*, volume 9472 of *Lecture Notes in Comput. Sci.*, pages 83–92. Springer, Heidelberg, 2015.

14 Stephane Durocher, Meng He, J. Ian Munro, Patrick K. Nicholson, and Matthew Skala. Range majority in constant time and linear space. *Inform. and Comput.*, 222:169–179, 2013.

15 P. Gupta, R. Janardan, and M. Smid. Further results on generalized intersection searching problems: Counting, reporting, and dynamization. *Journal of Algorithms*, 19(2):282–317, 1995.

16 Xiaocheng Hu, Miao Qiao, and Yufei Tao. Independent range sampling. In Richard Hull and Martin Grohe, editors, *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, pages 246–255. ACM, 2014. doi:10.1145/2594538.2594545.

17 Zengfeng Huang and Ke Yi. The communication complexity of distributed epsilon-approximations. *SIAM Journal of Computing*, 46(4):1370–1394, 2017.

18 Allan Grønlund Jørgensen and Kasper Green Larsen. Range selection and median: Tight cell probe lower bounds and adaptive data structures. In *Proc. 22ndProceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 805–813, 2011.

19 Marek Karpinski and Yakov Nekrich. Searching for frequent colors in rectangles. In *Proceedings of the 20th Annual Canadian Conference on Computational Geometry, Montréal, Canada, August 13-15, 2008*, 2008.

20 Danny Krizanc, Pat Morin, and Michiel Smid. Range mode and range median queries on lists and trees. *Nordic J. Comput.*, 12(1):1–17, 2005.

21 Jiří Matoušek. *Geometric Discrepancy: An Illustrated Guide*. Algorithms and Combinatorics. Springer Berlin Heidelberg, 2009.

22 Jeff M. Phillips. Coresets and sketches. *CoRR*, abs/1601.00617, 2016. arXiv:1601.00617.

23 Zhewei Wei and Ke Yi. Beyond simple aggregates: Indexing for summary queries. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 117–128. ACM, 2011.

24 Ke Yi, Lu Wang, and Zhewei Wei. Indexing for summary queries: Theory and practice. *ACM Transactions on Database Systems (TODS)*, 39(1), January 2014.

Stable Matching: Choosing Which Proposals to Make

Ishan Agarwal
New York University, NY, USA

Richard Cole
New York University, NY, USA

Abstract

To guarantee all agents are matched in general, the classic Deferred Acceptance algorithm needs complete preference lists. In practice, preference lists are short, yet stable matching still works well. This raises two questions:

- Why does it work well?
- Which proposals should agents include in their preference lists?

We study these questions in a model, introduced by Lee [17], with preferences based on correlated cardinal utilities: these utilities are based on common public ratings of each agent together with individual private adjustments. Lee showed that for suitable utility functions, in large markets, with high probability, for most agents, all stable matchings yield similar valued utilities. By means of a new analysis, we strengthen Lee's result, showing that in large markets, with high probability, for *all* but the agents with the lowest public ratings, all stable matchings yield similar valued utilities. We can then deduce that for *all* but the agents with the lowest public ratings, each agent has an easily identified length $O(\log n)$ preference list that includes all of its stable matches, addressing the second question above. We note that this identification uses an initial communication phase.

We extend these results to settings where the two sides have unequal numbers of agents, to many-to-one settings, e.g. employers and workers, and we also show the existence of an ϵ-Bayes-Nash equilibrium in which every agent makes relatively few proposals. These results all rely on a new technique for sidestepping the conditioning between the tentative matching events that occur over the course of a run of the Deferred Acceptance algorithm. We complement these theoretical results with an experimental study.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Mathematics of computing → Matchings and factors; Theory of computation → Random network models

Keywords and phrases Stable matching, randomized analysis

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.8

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2204.04162`

Funding This work was supported in part by NSF Grant CCF-1909538.

Acknowledgements We thank the reviewers for their helpful feedback.

1 Introduction

Consider a doctor applying for residency positions. Where should she apply? To the very top programs for her specialty? Or to those where she believes she has a reasonable chance of success (if these differ)? And if the latter, how does she identify them? We study these questions in the context of Gale and Shapley's deferred acceptance (DA) algorithm [5]. It is well-known that in DA the optimal strategy for the proposing side is to list their choices in order of preference. However, this does not address which choices to list.

The DA algorithm is widely used to compute matchings in real-world applications: the National Residency Matching Program (NRMP), which matches future residents to hospital programs [25]; university admissions programs which match students to programs, e.g. in Chile [24], school choice programs, e.g. for placement in New York City's high schools [1], the Israeli psychology Masters match [9], and no doubt many others (e.g. [7]).

Recall that each agent provides the mechanism a list of its possible matches in preference order, including the possibility of "no match" as one of its preferences. These mechanisms promise that the output will be a stable matching with respect to the submitted preference lists. In practice, preference lists are relatively short. This may be directly imposed by the mechanism or could be a reflection of the costs – for example, in time or money – of determining these preferences. Note that a short preference list is implicitly stating that the next preference after the listed ones is "no match".

Thus it is important to understand the impact of short preference lists. Roth and Peranson observed that the NRMP data showed that preference lists were short compared to the number of programs and that these preferences yielded a single stable partner for most participants; we note that this single stable partner could be the "no match" choice, and in fact this is the outcome for a constant fraction of the participants. They also confirmed this theoretically for the simplest model of uncorrelated random preferences; namely that with the preference lists truncated to the top O(1) preferences, almost all agents have a unique stable partner. Subsequently, in [10] the same result was obtained in the more general popularity model which allows for correlations among different agents' preferences; in their model, the first side – men – can have arbitrary preferences; on the second side – women – preferences are selected by weighted random choices, the weights representing the "popularity" of the different choices. These results were further extended by Kojima and Parthak in [15].

The popularity model does not capture behavior in settings where bounds on the number of proposals lead to proposals being made to plausible partners, i.e. partners with whom one has a realistic chance of matching. One way to capture such settings is by way of tiers [2], also known as block correlation [4]. Here agents on each side are partitioned into tiers, with all agents in a higher tier preferred to agents in a lower tier, and with uniformly random preferences within a tier. Tiers on the two sides may have different sizes. If we assign tiers successive intervals of ranks equal to their size, then, in any stable matching, the only matches will be between agents in tiers whose rank intervals overlap.

A more nuanced way of achieving these types of preferences bases agent preferences on cardinal utilities; for each side, these utilities are functions of an underlying common assessment of the other side, together with idiosyncratic individual adjustments for the agents on the other side. These include the separable utilities defined by Ashlagi, Braverman, Kanoria and Shi in [2], and another class of utilities introduced by Lee in [17]. This last model will be the focus of our study.

To make this more concrete, we review a simple special case of Lee's model, the *linear separable model*. Suppose that there are n men and n women seeking to match with each other. Each man m has a public rating r_m, a uniform random draw from $[0, 1]$. These ratings can be viewed as the women's joint common assessment of the men. In addition, each woman w has an individual adjustment, which we call a score, $s_w(m)$ for man m, again a uniform random draw from $[0, 1]$. All the draws are independent. Woman w's utility for man m is given by $\frac{1}{2}[r_m + s_w(m)]$; her full preference list has the men in decreasing utility order. The men's utilities are defined similarly.

Lee stated that rather than being assumed, short preference lists should arise from the model; this appears to have been a motivation for the model he introduced. A natural first step would be to show that for some or all stable matchings, the utility of each agent can

be well-predicted, for this would then allow the agents to limit themselves to the proposals achieving such a utility. Lee proved an approximate version of this statement, namely that with high probability (w.h.p., for short) most agents obtain utility within a small ϵ of an easily-computed individual benchmark. However, this does not imply that agents can restrict their proposals to a reduced utility range. (See the paragraph preceding Definition 5 for the specification of the benchmarks.)

Our work seeks to resolve this issue. We obtain the following results. Note that in these results, when we refer to the bottommost agents, we mean when ordered by decreasing public rating. Also, we let the term loss mean the difference between an agent's benchmark utility and their achieved utility.

1. We show that in the linearly separable model, for any constant $c > 0$, with probability $1 - 1/n^c$, in every stable matching, apart from a sub-constant σ fraction of the bottommost agents, *all* the other agents obtain utility equal to an easily-computed individual benchmark $\pm\epsilon$, where ϵ is also sub-constant.
   We show that both $\sigma, \epsilon = \widetilde{\Theta}(n^{-1/3})$.[1] As we will see, this implies, w.h.p., that for all the agents other than the bottommost σ fraction, each agent has $\Theta(\ln n)$ possible edges (proposals) that could be in any stable matching, namely the proposals that provide both agents utility within ϵ of their benchmark. Furthermore, we show our bound is tight: with fairly high probability, there is no matching, let alone stable matching, providing every agent a partner if the values of ϵ and σ are reduced by a suitable constant factor.
   An interesting consequence of this lower bound on the agents' utilities is that the agents can readily identify a moderate sized subset of the edge set to which they can safely restrict their applications. More precisely, any woman w outside the bottommost σ fraction, knowing only her own public rating, the public ratings of the men, and her own private score for each man, can determine a preference list of length $\widetilde{\Theta}(n^{1/3})$ which, w.h.p, will yield the same result as her true full-length list. Our analysis also shows that if w obtained the men's private scores for these proposals, then w.h.p. she could safely limit herself to a length $O(\ln n)$ preference list.
2. The above bounds apply not only to the linearly separable model, but to a significantly more general bounded derivative model (in which derivatives of the utility functions are bounded).
3. The result also immediately extends to settings with unequal numbers of men and women. Essentially, our analysis shows that the loss for an agent is small if there is a σ fraction of agents of lower rank on the opposite side. Thus even on the longer side, w.h.p., the topmost $n(1-\sigma)$ agents all obtain utility close to their benchmark, where n is the size of the shorter side. This limits the "stark effect of competition" [3] – namely that the agents on the longer side are significantly worse off – to a lower portion of the agents on the longer side.
4. The result extends to the many-to-one setting, in which agents on one side seek multiple matches. Our results are given w.r.t. a parameter d, the number of matches that each agent on the "many" side desires. For simplicity, we assume this parameter is the same for all these agents. In fact, we analyze a more general many-to-many setting.
5. A weaker result with arbitrarily small $\sigma, \epsilon = \Theta(1)$ holds when there is no restriction on the derivatives of the utility functions, which we call the general values model. Again, we show this bound cannot be improved in general. This setting is essentially the general

[1] The $\widetilde{\Theta}(\cdot)$ notation means up to a poly-logarithmic term; here $\sigma, \epsilon = \Theta((n/\ln n)^{-1/3})$.

setting considered by Lee [17]. He had shown there was a σ fraction of agents who might suffer larger losses; our bound identifies this σ fraction of agents as the bottommost agents.

6. In the bounded derivative model, with slightly stronger constraints on the derivatives, we also show the existence of an ϵ-Bayes-Nash equilibrium in which no agent proposes more than $O(\ln^2 n)$ times and all but the bottommost $O((\ln n/n)^{1/3})$ fraction of the agents make only the $O(\ln n)$ proposals identified in (1) above. Here $\epsilon = \Theta(\ln n/n^{1/3})$.

These results all follow from a lemma showing that, w.h.p., each non-bottommost agent has at most a small loss. In turn, the proof of this lemma relies on a new technique which sidesteps the conditioning inherent to runs of DA in these settings.

Experimental results

Much prior work has been concerned with preference lists that have a constant bound on their length. For moderate values of n, say $n \in [10^3, 10^6]$, $\ln n$ is quite small, so our $\Theta(\ln n)$ bound may or may not be sufficiently small in practice for this range of n. What matters are the actual constants hidden by the Θ notation, which our analysis does not fully determine. To help resolve this, we conducted a variety of simulation experiments.

We have also considered how to select the agents to include in the preference lists, when seeking to maintain a constant bound on their lengths, namely a bound that, for the values of n we considered, was smaller than the $\Theta(\ln n)$ bound determined by the above simulations; again, our investigation was experimental.

Other Related work

The random preference model was introduced by Knuth [12] (for a version in English see [13]), and subsequently extensively analyzed [20, 14, 21, 18, 23, 22, 16]. In this model, each agent's preferences are an independent uniform random permutation of the agents on the other side. An important observation was that when running the DA algorithm, the proposing side obtained a match of rank $\Theta(\ln n)$ on the average, while on the other side the matches had rank $\Theta(n/\ln n)$.

A recent and unexpected observation in [3] was the "stark effect of competition": that in the random preferences model the short side, whether it was the proposing side or not, was the one to enjoy the $\Theta(\ln n)$ rank matches. Subsequent work showed that this effect disappeared with short preference lists in a natural modification of the random preferences model [11]. Our work suggests yet another explanation for why this effect may not be present: it does not require that short preference lists be imposed as an external constraint, but rather that the preference model generates few edges that might ever be in a stable matching.

The number of edges present in any stable matching has also been examined for a variety of settings. When preference lists are uniform the expected number of stable pairs is $\Theta(n \ln n)$ [21]; when they are arbitrary on one side and uniform on the other side, the expected number is $O(n \ln n)$ [14]. This result continues to hold when preference lists are arbitrary on the men's side and are generated from general popularities on the women's side [6]. Our analysis shows that in the linear separable model (and more generally in the bounded derivative setting) the expected number of stable pairs is also $O(n \ln n)$.

Another important issue is the amount of communication needed to identify who to place on one's preference lists when they have bounded length. In general, the cost is $\Omega(n)$ per agent (in an n agent market) [8], but in the already-mentioned separable model of Ashlagi et al. [2] this improves to $\widetilde{O}(\sqrt{n})$ given some additional constraints, and further improves to

$O(\ln^4 n)$ in a tiered separable market [2]. We note that for the bounded derivatives setting, with high probability, the communication cost will be $O(n^{1/3}\ln^{2/3} n)$ for all agents except the bottommost $\Theta(n^{2/3}\ln^{1/3} n)$, for whom the cost can reach $O(n^{2/3}\ln^{1/3} n)$.

Another approach to selecting which universities to apply to was considered by Shorrer who devised a dynamic program to compute the optimal choices for students assuming universities had a common ranking of students [26].

Roadmap

In Section 2 we review some standard material. In Section 3 we state our main result in two parts: Theorem 6, which bounds the losses in the setting of the linear model, and Theorem 8, which shows it suffices to limit preference lists to a small set of edges. We prove these theorems in Sections 4 and 5, respectively. We also present some numerical simulations for the linear separable model in Section 6 We conclude with a brief discussion of open problems in Section 7.

In the appendices of the full version of the paper, we formally state and prove all the other results alluded to in the introduction and we also present further numerical simulations for the linear separable model. For the reader's convenience, in the text that follows, we provide pointers to these appendices, as appropriate. We note that Appendix A provides a complete summary of the content in these appendices.

2 Preliminaries

2.1 Stable Matching and the Deferred Acceptance (DA) Algorithm

Let M be a set of n men and W a set of n women. Each man m has an ordered list of women that represents his preferences, i.e. if a woman w comes before a woman w' in m's list, then m would prefer matching with w rather than w'. The position of a woman w in this list is called m's ranking of w. Similarly each woman w has a ranking of her preferred men[2]. The stable matching task is to pair (match) the men and women in such a way that no two people prefer each other to their assigned partners. More formally:

▶ **Definition 1** (Matching). *A matching is a pairing of the agents in M with the agents in W. It comprises a bijective function μ from M to W, and its inverse $\nu = \mu^{-1}$, which is a bijective function from W to M.*

▶ **Definition 2** (Blocking pair). *A matching μ has a blocking pair (m, w) if and only if:*

1. *m and w are not matched: $\mu(m) \neq w$.*
2. *m prefers w to his current match $\mu(m)$.*
3. *w prefers m to her current match $\nu(w)$.*

▶ **Definition 3** (Stable matching). *A matching μ is stable if it has no blocking pair.*

Gale and Shapley [5] proposed the seminal deferred acceptance (DA) algorithm for the stable matching problem. We present the woman-proposing DA algorithm (Algorithm 1); the man-proposing DA is symmetric. The following facts about the DA algorithm are well known. We state them here without proof and we shall use them freely in our analysis.

[2] Throughout this paper, we assume that each man m (woman w) ranks all the possible women (men), i.e. m's (w's) preference list is complete.

Algorithm 1 Woman Proposing Deferred Acceptance (DA) Algorithm.

Initially, all the men and women are unmatched.
while *some woman w with a non-empty preference list is unmatched* **do**
  let m be the first man on her preference list;
  if *m is currently unmatched* **then**
    tentatively match w to m.
  end
  if *m is currently matched to w', and m prefers w to w'* **then**
    make w' unmatched and tentatively match w to m.
  else
    remove m from w's preference list.
  end
end

▶ **Observation 4.**

1. *DA terminates and outputs a stable matching.*
2. *The stable matching generated by DA is independent of the order in which the unmatched agents on the proposing side are processed.*
3. *Woman-proposing DA is woman-optimal, i.e. each woman is matched with the best partner she could be matched with in any stable matching.*
4. *Woman-proposing DA is man-pessimal, i.e. each man is matched with the worst partner he could be matched with in any stable matching.*

2.2 Useful notation and definitions

There are n men and n women. In all of our models, each man m has a utility $U_{m,w}$ for the woman w, and each woman w has a utility $V_{m,w}$ for the man m. These utilities are defined as

$$U_{m,w} = U(r_w, s_m(w)), \text{ and}$$
$$V_{m,w} = V(r_m, s_w(m)),$$

where r_m and r_w are common public ratings, $s_m(w)$ and $s_w(m)$ are private scores specific to the pair (m, w), and $U(\cdot,\cdot)$ and $V(\cdot,\cdot)$ are continuous and strictly increasing functions from $\mathbb{R}_+^2$ to $\mathbb{R}_+$. The ratings are independent uniform draws from $[0, 1]$ as are the scores.

In the *Linear Separable Model*, each man m assigns each woman w a utility of $U_{m,w} = \lambda \cdot r_w + (1-\lambda) \cdot s_m(w)$, where $0 < \lambda < 1$ is a constant. The women's utilities for the men are defined analogously as $V_{m,w} = \lambda \cdot r_m + (1-\lambda) \cdot s_w(m)$. All our experiments are for this model.

We let $\{m_1, m_2, \ldots, m_n\}$ be the men in descending order of their public ratings and $\{w_1, w_2, \ldots, w_n\}$ be a similar ordering of the women. We say that m_i has public rank i, or rank i for short, and similarly for w_i. We also say that m_i and w_i are *aligned*. In addition, we often want to identify the men or women in an interval of public ratings. Accordingly, we define $M(r, r')$ to be the set of men with public ratings in the range (r, r'), and $M[r, r']$ to be the set with public ratings in the range $[r, r']$; we also use the notation $M(r, r']$ and $M[r, r')$ to identify the men with ratings in the corresponding semi-open intervals. We use an analogous notation, with W replacing M, to refer to the corresponding sets of women.

We will be comparing the achieved utilities in stable matchings to the following benchmarks: the rank i man has as benchmark $U(r_{w_i}, 1)$, the utility he would obtain from the combination of the rank i woman's public rating and the highest possible private score; and similarly for the women. Based on this we define the loss an agent faces as follows.

▶ **Definition 5** (Loss). *Suppose man m and woman w both have rank i. The* loss *m sustains from a match of utility u is defined to be $U(r_w, 1) - u$. The loss for women is defined analogously.*

In our analysis we will consider a complete bipartite graph whose two sets of vertices correspond to the men and women, respectively. For each man m and woman w, we view the possible matched pair (m, w) as an edge in this graph. Thus, throughout this work, we will often refer to edges being proposed, as well as edges satisfying various conditions.

3 Upper Bound in The Linear Separable Model

To illustrate our proof technique for deriving upper bounds, we begin by stating and proving our upper bound result for the special case of the linear separable model with $\lambda = \frac{1}{2}$.

▶ **Theorem 6.** *In the linear separable model with $\lambda = 1/2$, when there are n men and n women, for any given constant $c > 0$, for large enough n, with probability at least $1 - n^{-c}$, in every stable matching, for every i, with $r_{w_i} \geq \overline{\sigma} \triangleq 3\overline{L}/2$, agent m_i suffers a loss of at most $\overline{L}$, where $\overline{L} = (16(c+2)\ln n/n)^{1/3}$, and similarly for the agents w_i.*

In words, w.h.p., all but the bottommost agents (those whose aligned agents have public rating less than $\overline{\sigma}$) suffer a loss of no more than $\overline{L}$. This is a special case of our basic upper bound for the bounded utilities model (Theorem 12).

One of our goals is to be able to limit the number of proposals the proposing side needs to make. We identify the edges that could be in some stable matching, calling them acceptable edges. Our definition is stated generally so that it covers all our results; accordingly we replace the terms $\overline{L}$ and $\overline{\sigma}$ in Theorem 6 with parameters L and σ.

▶ **Definition 7** (Acceptable edges). *Let $0 < \sigma < 1$ and $0 < L < 1$ be two parameters. An edge (m_i, w_j) is* (L, σ)-man-acceptable *either if it provides m_i utility at least $U(r_{w_i}, 1) - L$, or if $m_i \in M[0, \sigma)$. The definition of (L, σ)-woman-acceptable is symmetric. Finally, (m_i, w_j) is* (L, σ)-acceptable *if it is both (L, σ)-man and (L, σ)-woman-acceptable.*

To prove our various results, we choose L and σ so that w.h.p. the edges in every stable matching are (L, σ)-*acceptable*. We call this high probability event $\mathcal{E}$. We will show that if $\mathcal{E}$ occurs, then running DA on the set of acceptable edges, or any superset of the acceptable edges obtained via loss thresholds, produces the same stable matching as running DA on the full set of edges.

▶ **Theorem 8.** *If $\mathcal{E}$ occurs, then running woman-proposing DA with the edge set restricted to the acceptable edges or to any superset of the acceptable edges obtained via loss thresholds (including the full edge set) result in the same stable matching.*

The implication is that w.h.p. a woman can safely restrict her proposals to her acceptable edges, or to any overestimate of this set of edges obtained by her setting an upper bound on the loss she is willing to accept. There is a small probability – at most n^{-c} – that this may result in a less good outcome, which can happen only if $\mathcal{E}$ does not occur. Note that Theorem 8 applies to every utility model we consider. Then, w.h.p., every stable matching gives each woman w, whose aligned agent m has public rating $r_m \geq \overline{\sigma} = \Omega((\ln n/n)^{1/3})$, a partner with public rating in the range $[r_m - 2\overline{L}, r_m + \frac{5}{2}\overline{L}]$ (see Theorem 25 in Appendix F.1). The bound $r_m - 2\overline{L}$ is a consequence of the bound on the woman's loss; the bound $r_m + \frac{5}{2}\overline{L}$ is a consequence of the bound on the men's losses. An analogous statement applies to the men.

This means that if we are running woman-proposing DA, each of these women might as well limit her proposals to her woman-acceptable edges, which is at most the men with public ratings in the range $r_m \pm \Theta(\overline{L})$ for whom she has private scores of at least $1 - \Theta(\overline{L})$. In expectation, this yields $\Theta(n^{1/3}(\ln n)^{2/3})$ men to whom it might be worth proposing. It also implies that a woman can have a gain of at most $\Theta(\overline{L})$ compared to her target utility.

If, in addition, each man can inexpensively signal the women who are man-acceptable to him, then the women can further limit their proposals to just those men providing them with a signal; this reduces the expected number of proposals these women can usefully make to just $\Theta(\ln n)$.

4 Sketch of the Proof of Theorem 6

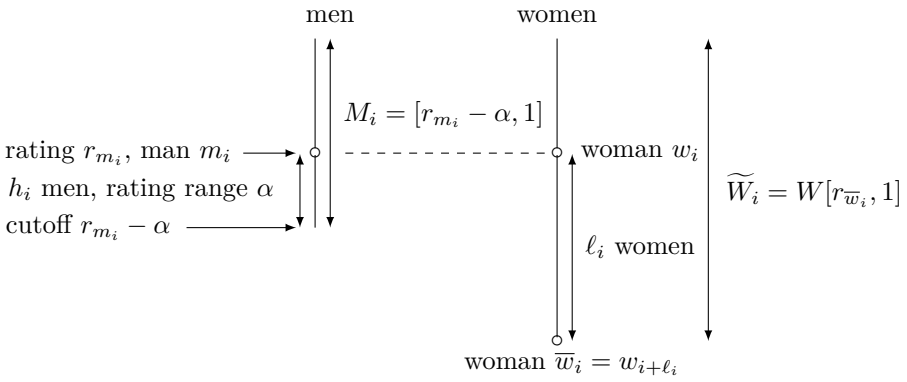


Figure 1 Illustrating Lemma 11.

We begin by outlining the main ideas used in our analysis. Our goal is to show that when we run woman proposing DA, w.h.p. each man receives a proposal that gives him a loss of at most L (except possibly for men among the bottommost $\Theta(nL)$). As the outcome is the man-pessimal stable matching, this means that w.h.p., in all stable matchings, these men have a loss of at most L. By symmetry, the same bound holds for the women.

Next, we provide some intuition for the proof of this result. See Fig. 1. Our analysis uses 3 parameters $\alpha, \beta, \gamma = \Theta(L)$. Let m_i be a non-bottommost man. We consider the set of men with public rank at least $r_{m_i} - \alpha$: $M_i = M[r_{m_i} - \alpha, 1]$. We consider a similar, slightly larger set of women: $\widetilde{W}_i = W[r_{w_i} - 3\alpha, 1]$. Now we look at the best proposals by the women in $\widetilde{W}_i$, i.e. the ones they make first. Specifically, we consider the proposals that give these women utility at least $V(r_{m_i} - \alpha, 1)$, proposals that are therefore guaranteed to be to the men in M_i. Let $\left|M_i\right| = i + h_i$ and $\left|W_i\right| = i + \ell_i$. In expectation, $\ell_i - h_i = 2\alpha n$. Necessarily, at least $\ell_i - h_i + 1$ women in M_i cannot match with men in $M_i \setminus \{m_i\}$. But, as we will see, these women all have probability at least β of having a proposal to m_i which gives them utility at least $V(r_{m_i} - \alpha, 1)$. These are proposals these women must make before they make any proposals to men with public rating less than $r_{m_i} - \alpha$. Furthermore, for each of these proposals, m_i has probability at least γ of having a loss of L or less. Thus, in expectation, m_i receives at least $2\alpha\beta\gamma n$ proposals which give him a loss of L or less.

We actually want a high-probability bound. So we choose α, β, γ so that $\alpha\beta\gamma n \geq c \log n$ for a suitable constant $c > 0$, and then apply a series of Chernoff bounds. There is one difficulty. The Chernoff bounds requires the various proposals to be independent. Unfortunately, in

general, this does not appear to be the case. However, we are able to show that the failure probability for our setting is at most the failure probability in an artificial setting in which the events are independent, which yields the desired bound.

We now embark on the actual proof.

We formalize the men's rating cutoff with the notion of DA stopping at public rating r.

▶ **Definition 9** (DA stops). *The women* stop at public rating r *if, in each woman's preference list, all the edges with utility less than $V(r, 1)$ are removed. The women* stop at man m *if, in each woman's preference list, all the edges following her edge to m are removed. The women* double cut at man m and public rating r*, if they each stop at m or r, whichever comes first. Men stopping and double cutting are defined similarly. Finally, an edge is said to* survive *the cutoff if it is not removed by the stopping.*

To obtain our bounds for man m_i, we will have the women double cut at rating $r_{m_i} - \alpha$ and at man m_i, where $\alpha > 0$ is a parameter we will specify later.

Our upper bounds in all of the utility models depend on a parameterized key lemma (Lemma 11) stated shortly. This lemma concerns the losses the men face in the woman-proposing DA; a symmetric result applies to the women. The individual theorems follow by setting the parameters appropriately. Our key lemma uses three parameters: $\alpha, \beta, \gamma > 0$. To avoid rounding issues, we will choose α so that αn is an integer. The other parameters need to satisfy the following constraints.

$$\text{for } r \geq \alpha: \qquad V(r - \alpha, 1) \leq V(r, 1 - \beta) \tag{1}$$

$$\text{for } r \geq 3\alpha: \qquad U(r, 1) - U(r - 3\alpha, 1 - \gamma) \leq L \tag{2}$$

Equation (1) relates the range of private values that will yield a woman an edge to m_i that survives the cut at $r_{m_i} - \alpha$, or equivalently the probability of having such an edge. Observation 10 below, shows that Equation (2) identifies the range of m_i's private values for proposals from $\widetilde{W}_i$ that yield him a loss of at most L (for we will ensure the women in $\widetilde{W}_i$ have public rating at least $r_{w_i} - 3\alpha$).

▶ **Observation 10.** *Consider the proposal from woman w to the rank i man m_i. Suppose the rank i woman w_i has rating $r_{w_i} \geq 3\alpha$. If w has public rating $r \geq r_{w_i} - 3\alpha$ and m_i's private score for w is at least $1-\gamma$, then m_i's utility for w is at least $U(r_{w_i} - 3\alpha, 1-\gamma) \geq U(r_{w_i}, 1) - L$.*

In the linear separable model with $\lambda = \frac{1}{2}$, we set $\alpha = \beta = \gamma$ and $L = 2\alpha$.

The next lemma determines the probability that man m_i receives a proposal causing him a loss of at most L. The lemma calculates this probability in terms of the parameters we just defined. Note that the result does not depend on the utility functions $U(\cdot, \cdot)$ and $V(\cdot, \cdot)$ being linear. In fact, the same lemma applies to much more general utility models which we also study (see Appendix C) and it is the crucial tool we use in all our upper bound proofs.

In what follows, to avoid heavy-handed notation, by $r_{m_i} - \alpha$ we will mean $\max\{0, r_{m_i} - \alpha\}$.

In order to state our next result crisply, we define the following Event $\mathcal{E}_i$. It concerns a run of woman-proposing DA with double cut at the rank i man m_i and at public rating $r_{m_i} - \alpha$. Let $h_i = \left|M[r_{m_i} - \alpha, r_{m_i})\right|$, $\ell_i = \left|W[r_{w_i} - 3\alpha, r_{w_i})\right|$, and $\overline{w}_i$ be the woman with rank $i + \ell_i$. See Figure 1 for an illustration of these definitions. Event $\mathcal{E}_i$ occurs if $r_{w_i} \geq 3\alpha$ and between them the $i + \ell_i$ women in $W[r_{w_i} - 3\alpha, 1]$ make at least one proposal to m_i that causes him a loss of at most L.

Finally we define Event $\mathcal{E}$: it happens if $\mathcal{E}_i$ occurs for all i such that $r_{w_i} \geq 3\alpha$.

▶ **Lemma 11.** *Let $\alpha > 0$ and $L > 0$ be given, and suppose that β and γ satisfy* (1) *and* (2)*, respectively. Then, Event $\mathcal{E}$ occurs with probability at least $1 - p_f$, where the failure probability*

$$p_f = n \cdot \exp(-\alpha(n-1)/12) + n \cdot \exp(-\alpha(n-1)/24) + n \exp(-\alpha\beta n/8) + n \cdot \exp(-\alpha\beta\gamma n/2).$$

The following simple claim notes that the men's loss when running the full DA is no larger than when running double-cut DA.

▷ Claim 12. Suppose a woman-proposing double-cut DA at man m_i and rating $r_{m_i} - \alpha$ is run, and suppose m_i incurs a loss of L. Then in the full run of woman-proposing DA, m_i will incur a loss of at most L.

Proof. Recall that when running the women-proposing DA the order in which unmatched women are processed does not affect the outcome. Also note that as the run proceeds, whenever a man's match is updated, the man obtains an improved utility. Thus, in the run with the full edge set we can first use the edges used in the double-cut DA and then proceed with the remaining edges. Therefore if in the double-cut DA m_i has a loss of L, in the full run m_i will also have a loss of at most L. ◁

To illustrate how this lemma is applied, we now prove Theorem 6. Note that $\overline{L}$ is the value of L used in this theorem. Our other results use other values of L.

Proof of Theorem 6. By Lemma 11, in the double-cut DA, for all i with $r_{w_i} \geq 3\alpha$, m_i obtains a match giving him loss at most $\overline{L}$, with probability at least $1 - n \cdot \exp(-\alpha(n-1)/12) - n \cdot \exp(-\alpha n/24) - n \exp\big(-\alpha^2 n/8\big) - n \cdot \exp\big(-\alpha^3 n/2\big)$.

By Claim 12, m_i will incur a loss of at most $\overline{L}$ in the full run of woman-proposing DA with at least as large a probability. But this is the man-pessimal match. Consequently, in every stable match, m_i has a loss of at most $\overline{L}$. By symmetry, the same bound applies to each woman w_i such that $r_{m_i} \geq 3\alpha$.

We choose $\overline{L} = [16(c+2)\ln n/n]^{1/3}$. Recalling that $\alpha = \overline{L}/2$, we see that for large enough n the probability bound, over all the men and women, is at most $1 - n^{-c}$. The bounds $r_{w_i} \geq 3\alpha$ and $r_{m_i} \geq 3\alpha$ imply we can set $\overline{\sigma} = 3\alpha = \frac{3}{2}\overline{L}$. ◀

Proof of Lemma 11. We run the double-cut DA in two phases, defined as follows. Recall that $h_i = \big|M[r_{m_i} - \alpha, r_{m_i})\big|$ and $\ell_i = \big|W[r_{w_i} - 3\alpha, r_{w_i})\big|$. Note that women with rank at most $i + \ell_i$ have public rating at least $r_{w_i} - 3\alpha$.

Phase 1. Every unmatched woman with rank at most $i + \ell_i$ keeps proposing until her next proposal is to m_i, or she runs out of proposals.

Phase 2. Each unmatched women makes her next proposal, if any, which will be a proposal to m_i.

Our analysis is based on the following four claims. The first two are simply observations that w.h.p. the number of agents with public ratings in a given interval is close to the expected number. We defer the proofs to the appendix.

A critical issue in this analysis is to make sure the conditioning induced by the successive steps of the analysis does not affect the independence needed for subsequent steps. To achieve this, we use the Principle of Deferred Decisions, only instantiating random values as they are used. Since each successive bound uses a different collection of random variables this does not present a problem.

▷ Claim 13. Let $\mathcal{B}_1$ be the event that for some i, $h_i \geq \frac{3}{2}\alpha(n-1)$. $\mathcal{B}_1$ occurs with probability at most $n \cdot \exp(-\alpha(n-1)/12)$. The only randomness used in the proof are the choices of the men's public ratings. The same bound applies to the women.

Proof (Sketch). As $\mathrm{E}[h_i] = \alpha(n-1)$, w.h.p., $h_i < \frac{3}{2}\alpha(n-1)$. This claim uses a Chernoff bound with the randomness coming from the public ratings of the men. ◁

▷ Claim 14. Let $\mathcal{B}_2$ be the event that for some i, $\ell_i \leq \frac{5}{2}\alpha(n-1)$. $\mathcal{B}_2$ occurs with probability at most $n \cdot \exp(-\alpha(n-1)/24)$. The only randomness used in the proof are the choices of the women's public ratings. The same bound applies to the men.

Proof. This is very similar to the proof of Claim 13. ◁

▷ Claim 15. Let $\mathcal{B}_3$ be the event that between them, the women with rank at most $i + \ell_i$ make fewer than $\frac{1}{2}\alpha\beta n$ Step 2 proposals to m_i. If events $\mathcal{B}_1$ and $\mathcal{B}_2$ do not occur, then $\mathcal{B}_3$ occurs with probability at most $\exp(-\alpha\beta n/8)$. The only randomness used in the proof are the choices of the women's private scores.

This bound uses the private scores of the women and employs a novel argument given below to sidestep the conditioning among these proposals.

▷ Claim 16. If none of the events $\mathcal{B}_1$, $\mathcal{B}_2$, or $\mathcal{B}_3$ occur, then at least one of the Step 2 proposals to m_i will cause him a loss of at most L with probability at least $1-(1-\gamma)^{\alpha\beta n/2} \geq 1 - \exp(-\alpha\beta\gamma n/2)$. The only randomness used in the proof are the choices of the men's private scores.

Proof. Note that each Phase 2 proposal is from a woman w with rank at most $i + \ell_i$. As already observed, her public rating is at least $r_{w_i} - 3\alpha$. Recall that man m_i's utility for w equals $U(r_w, s_{m_i}(w)) \geq U(r_{w_i} - 3\alpha, s_{m_i}(w))$. To achieve utility at least $U(r_{w_i}, 1) - L \leq U(r_{w_i} - 3\alpha, 1-\gamma)$ (using (2)) it suffices to have $s_{m_i}(w) \geq 1-\gamma$, which happens with probability γ. Consequently, utility at least $U(r_{w_i}, 1) - L$ is achieved with probability at least γ.

For each Phase 2 proposal these probabilities are independent as they reflect m_i's private scores for each of these proposals. Therefore the probability that there is no proposal providing m_i a loss of at most L is at most

$$\big(1-\gamma\big)^{\alpha\beta n/2} \leq \exp(\alpha\beta\gamma n/2).$$ ◁

Concluding the proof of Lemma 11: The overall failure probability summed over all n choices of i is

$$n \cdot \exp(-\alpha(n-1)/12) + n \cdot \exp(-\alpha(n-1)/24) + n\exp(-\alpha\beta n/8) + n \cdot \exp(-\alpha\beta\gamma n/2).$$ ◀

Proof of Claim 15. First, we simplify the action space by viewing the decisions as being made on a discrete utility space, as specified in the next claim, proved in the appendix.

▷ Claim 17. For any $\delta > 0$, there is a discrete utility space in which for each woman the probability of selecting m_i is only increased, and the probability of having any differences in the sequence of actions in the original continuous setting and the discrete setting is at most δ.

We represent the possible computations of the double-cut DA in this discrete setting using a tree T. Each woman will be going through her possible utility values in decreasing order, with the possible actions of the various women being interleaved in the order given by the DA processing. Each node u corresponds to a woman w processing her next utility value. The possible choices at this utility are each represented by an edge descending from u. These choices are:

i. Proposing to some man (among those men w has not yet proposed to); or

ii. "no action". This corresponds to w making no proposal achieving the current utility.

We observe the following important structural feature of tree T. Let S be the subtree descending from the edge corresponding to woman w proposing to m_i; in S there are no further actions of w, i.e. no nodes at which w makes a choice, because the double cut DA cuts at the proposal to m_i.

The assumption that $\mathcal{B}_1$ and $\mathcal{B}_2$ do not occur means that for all i, $h_i < \frac{3}{2}\alpha(n-1)$ and $\ell_i > \frac{5}{2}\alpha(n-1)$, and therefore $\ell_i - h_i > \alpha(n-1)$.

At each leaf of T, up to $i + h_i - 1$ women will have been matched with someone other than m_i. The other women either finished with a proposal to m_i or both failed to match and did not propose to m_i. Let w be a woman in the latter category. Then, on the path to this leaf, w will have traversed edges corresponding to a choice at each discrete utility in the range $[V(r_{m_i} - \alpha, 1), V(1,1)]$.

We now create an extended tree, T_x, by adding a subtree at each leaf; this subtree will correspond to pretending there were no matches; the effect is that each women will take an action at all their remaining utility values in the range $[V(r_{m_i} - \alpha, 1), V(1,1)]$, except that in the sub-subtrees descending from edges that correspond to some woman w selecting m_i, w has no further actions. For each leaf in the unextended tree, the probability of the path to that leaf is left unchanged. The probabilities of the paths in the extended tree are then calculated by multiplying the path probability in the unextended tree with the probabilities of each woman's choices in the extended portion of the tree.

Next, we create an artificial mechanism $\mathcal{M}$ that acts on tree T_x. The mechanism $\mathcal{M}$ is allowed to put $i + h_i - 1$ "blocks" on each path; blocks can be placed at internal nodes. A block names a woman w and corresponds to her matching (but we no longer think of the matches as corresponding to the outcome of the edge selection; they have no meaning beyond making all subsequent choices by this woman be the "no action" choice).

DA can be seen as choosing to place up to $i + h_i - 1$ blocks at each of the nodes corresponding to a leaf of T. $\mathcal{M}$ will place its blocks so as to minimize the probability p of paths with at least $\frac{1}{2}\alpha\beta n$ women choosing edges to m_i. Clearly p is a lower bound on the probability that the double-cut DA makes at least $\frac{1}{2}\alpha\beta n$ proposals in Step 2. Given a choice of blocks we call the resulting probability of having fewer than $\frac{1}{2}\alpha\beta n$ women choosing edges to m_i the *blocking probability.*

▷ Claim 18. The probability that $\mathcal{M}$ makes at least $\frac{1}{2}\alpha\beta n$ proposals to m_i is at least $1 - \exp(-\alpha\beta n/8)$.

▶ **Corollary 19.** *The probability that the double-cut DA makes at least $\frac{1}{2}\alpha\beta n$ proposals to m_i is at least $1 - \exp(-\alpha\beta n/8)$.*

Proof. For any fixed δ, by Claim 18, the probability that $\mathcal{M}$ makes at least $\frac{1}{2}\alpha\beta n$ proposals to m_i is at least $1 - \exp(-\alpha\beta n/8)$. By construction, the probability is only larger for the double-cut DA in the discrete space.

Therefore, by Claim 12, the probability that the double-cut DA makes at least $\frac{1}{2}\alpha\beta n$ proposals to m_i in the actual continuous space is at least $1-\exp(-\alpha\beta n/8)-\delta$, and this holds for any $\delta > 0$, however small. Consequently, this probability is at least $1-\exp(-\alpha\beta n/8)$. ◀

Proof of Claim 18. We will show that the most effective blocking strategy is to block as many women as possible before they have made any choices. This leaves at least $(i+\ell_i)-(i-1+h_i) \geq 1+\alpha(n-1) \geq \alpha n$ women unmatched. Then, as we argue next, each of these remaining at least αn women w has independent probability at least β that their proposal to m_i is cutoff-surviving. To be cutoff-surviving, it suffices that $V(r_{m_i}, s_w(m_i)) \geq V(r_{m_i}-\alpha, 1)$. But we know by (1) that $V(r_{m_i}-\alpha, 1) \leq V(r_{m_i}, 1-\beta)$, and therefore it suffices that $s_w(m_i) \geq 1-\beta$, which occurs with probability β.

Consequently, in expectation, there are at least $\alpha\beta n$ proposals to m_i, and therefore, by a Chernoff bound, at least $\frac{1}{2}\alpha\beta n$ proposals with probability at least $\exp(-\alpha\beta n/8)$.

We consider the actual blocking choices made by $\mathcal{M}$ and modify them bottom-up in a way that only reduces the probability of there being $\frac{1}{2}\alpha\beta n$ or more proposals to m_i.

Clearly, $\mathcal{M}$ can choose to block the same maximum number of women on every path as it never hurts to block more women (we allow the blocking of women who have already proposed to m_i even though it does not affect the number of proposals to m_i).

Consider a deepest block at some node u in the tree, and suppose b women are blocked at u. Let v be a sibling of u. As this is a deepest block, there will be no blocks at proper descendants of u, and furthermore as there are the same number of blocks on every path, v will also have b blocked women.

Observe that if there is no blocking in a subtree, then the probability that a woman makes a proposal to m_i is independent of the outcomes for the other women. Therefore the correct blocking decision at node u is to block the b women with the highest probabilities of otherwise making a proposal to m_i, which we call their *proposing probabilities*; the same is true at each of its siblings v.

Let x be u's parent. Suppose the action at node x concerns woman $\widetilde{w}_x$. Note that the proposing probability for any woman $w \neq \widetilde{w}_x$ is the same at u and v because the remaining sequence of actions for woman w is the same at nodes u and v, and as they are independent of the actions of the other women, they yield the same probability of selecting m_i at some point.

We need to consider a number of cases.

Case 1. w is blocked at every child of x.

Then we could equally well block w at node x.

Case 2. At least one woman other than $\widetilde{w}_x$ is blocked at some child of x.

Each such blocked woman w has the same proposing probability at each child of x. Therefore by choosing to block the women with the highest proposing probabilities, we can ensure that at each node either $\widetilde{w}_x$ plus the same $b-1$ other women are blocked, or these $b-1$ woman plus the same additional woman $w' \neq \widetilde{w}_x$ are blocked. In any event, the blocking of the first $b-1$ women can be moved to x.

Case 2.1. $\widetilde{w}_x$ is not blocked at any child of x.

Then the remaining identical blocked woman at each child of x can be moved to x.

Case 2.2. $\widetilde{w}_x$ is blocked at some child of x but not at all the children of x.

Notice that we can avoid blocking $\widetilde{w}_x$ at the child u of x corresponding to selecting m_i, as the proposing probability for $\widetilde{w}_x$ after it has selected m_i is 0, so blocking any other women would be at least as good. Suppose that $w \neq \widetilde{w}_x$ is blocked at node u.

Let v be another child of x at which $\widetilde{w}_x$ is blocked. Necessarily, $p_{v,\widetilde{w}_x}$, the proposing probability for $\widetilde{w}_x$ at node v, is at least the proposing probability $p_{v,w}$ for w at node v (for

otherwise w would be blocked at node v); also, $p_{v,w}$ equals the proposing probability for w at every child of x including u; in addition, $p_{v,\widetilde{w}_x}$ equals the proposing probability for $\widetilde{w}_x$ at every child of x other than u. It follows that w is blocked at u and $\widetilde{w}_x$ can be blocked at every other child of x. But then blocking $\widetilde{w}_x$ at x only reduces the proposing probability.

Thus in every case one should move the bottommost blocking decisions at a collection of sibling nodes to a single blocking decision at their parent. ◁

◁

5 Making Fewer Proposals

We identify a sufficient set of edges that contains all stable matchings, and on which the DA algorithm produces the same outcome as when it runs on the full edge set.

▶ **Definition 20** (Viable edges). *An edge (m, w) is* man-viable *if, according to m's preferences, w is at least as good as the woman he is matched to in the man-pessimal stable match. Woman-viable is defined symmetrically. An edge is* viable *if it is both man and woman-viable. E_v is the set of all viable edges.*

▶ **Lemma 21.** *Running woman-proposing DA with the edge set restricted to E_v and with any superset obtained via loss thresholds, including the full edge set, results in the same stable matching.*

Proof. Suppose a new stable matching, S, now exists in the restricted edge set: it could not be present when using the full edge set, therefore there must be a blocking edge (m, w) in the full edge set. But neither m nor w would have removed this edge when forming their restricted edge set since for both of them it is better than an edge they did not remove (the edge they are matched with in S).

It follows that w.h.p. the set of stable matchings is the same when using E_v (or any super set of it generated by truncation with larger loss thresholds) and the whole set. Thus woman-proposing DA run on the restricted edge set will yield the same stable matching as on the full edge set. ◀

Proof of Theorem 8. If $\mathcal{E}$ occurs, the set of acceptable edges contains all the viable edges. Furthermore, the acceptable edges are defined by means of loss thresholds. The result now follows from Lemma 21. ◀

For some of the very bottommost agents, almost all edges may be acceptable. However, in the bounded derivatives model, with slightly stronger constraints on the derivatives, we also show (see Appendix H) the existence of an ϵ-Bayes-Nash equilibrium in which all but a bottom $\Theta((\ln n/n)^{1/3})$ fraction of agents use only $\Theta(\ln n)$ edges, and all agents propose using at most $\Theta(\ln^2 n)$ edges, with $\epsilon = O(\ln n/n^{1/3})$.

6 Numerical Simulations

We present several simulation results which are complementary to our theoretical results. Throughout this section, we focus on the linear separable model.

6.1 NRMP Data

We used NRMP data to motivate some of our choices of parameters for our simulations. The NRMP provides extensive summary data [19]. We begin by discussing this data.

Over time, the number of positions and applicants has been growing. We mention some numbers for 2021. There were over 38,000 positions available and a little over 42,000 applicants. The main match using the DA algorithm (modified to allow for couples, who comprise a little over 5% of the applicants) filled about 95% of the available positions. The NRMP also ran an aftermarket, called SOAP, after which about 0.5% of the positions remained unfilled.

The positions cover many different specialities. These specialities vary hugely in the number of positions available, with the top 11, all of size at least 1,000, accounting for 75% of the positions. In addition, about 75% of the doctors apply to only one speciality. We think that as a first approximation, w.r.t. the model we are using, it is reasonable to view each speciality as a separate market. Accordingly, we have focused our simulations on markets with 1,000–2,000 positions (though the largest speciality in the NRMP data had over 9,000 positions).

On average, doctors listed 12.5 programs in their preference lists, hospital programs listed 88 doctors, and the average program size was 6.5 (all numbers are approximate). While there is no detailed breakdown of the first two numbers, it is clear they vary considerably over the individual doctors and hospitals. For our many-to-one simulations we chose to use a fixed size for the hospital programs. Our simulations cause the other two numbers to vary over the individual doctors and programs because the public ratings and private scores are chosen by a random process.

6.2 Numbers of Available Edges

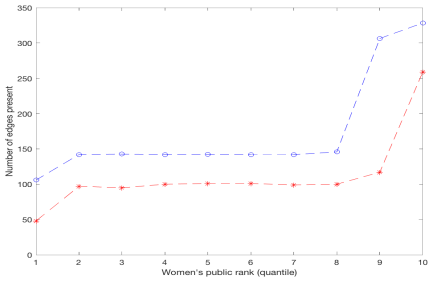
The first question we want to answer is how long do the preference lists need to be in order to have a high probability of including all acceptable edges, for all but the bottommost agents?

We chose bottommost to mean the bottom 20% of the agents, based on where the needed length of the preference lists started to increase in our experiments for $n = 1{,}000$–$2{,}000$.

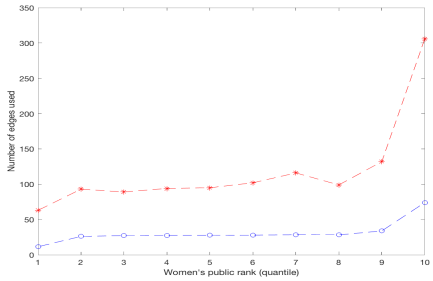
We ran experiments with $\lambda = 0.5, 0.67, 0.8$, corresponding to the public rating having respectively equal, twice, and four times the weight of the private scores in their contribution to the utility. We report the results for $\lambda = 0.8$. The edge sets were larger for smaller values of λ, but the results were qualitatively the same. We generated 100 random markets and determined the smallest value of L that ensured all agents were matched in all 100 markets. $L = 0.12$ sufficed. In Figure 2, we show results by decile of women's rank (top 10%, second 10%, etc.), specifically the average length of the preference list and the average number of edges proposed by a woman in woman-proposing DA, over these 100 randomly generated markets. We also show the max and min values over the 100 runs; these can be quite far from the average value. Note that the min values in Figure 2(a) are close to the max values in Figure 2(b), which suggests that being on the proposing side does not significantly reduce the value of L that the women could use compared to the value the men use. We also show data for a typical single run in Figure 3.

We repeated the simulation for the many-to-one setting. In Figure 4, we show the results for 2000 workers and 250 companies, each with 8 positions. Now, on average, a typical worker (i.e. among the top 80%) has an average preference list length of 55 and makes 7 proposals.

The one-to-one results show that for non-bottommost agents, the preference lists have length 150 on the average, while women make 30 proposals on the average (these numbers

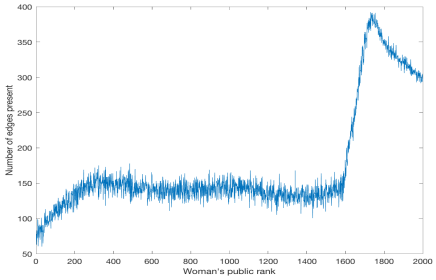


(a) Number of edges in the acceptable edge set, per woman, by decile; average in blue with circles, minimum in red with stars. ($n = 2{,}000$, $\lambda = 0.8$, $L = 0.12$.)

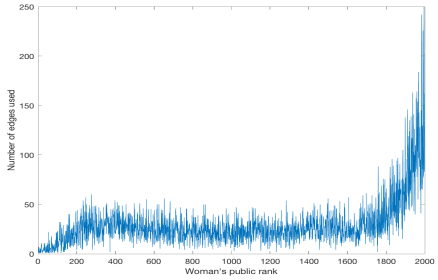


(b) Number of edges in the acceptable edge set proposed during the run of DA, per women, by decile; average in blue with circles, maximum in red with stars.

Figure 2 One-to-one case: summary statistics.



(a) Number of edges in the acceptable edge set for each woman.



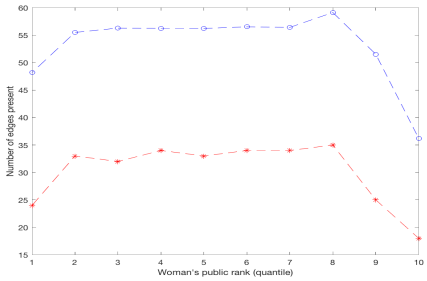
(b) Number of edges in the acceptable edge set proposed by each woman.

Figure 3 One-to-one case: a typical run.

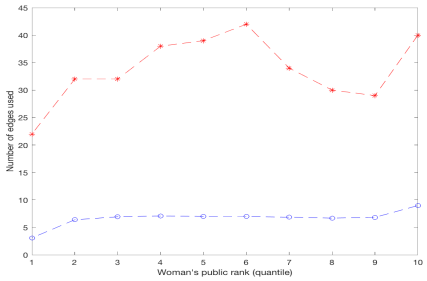
are slightly approximate). What is going on? We believe that the most common matches provide a small loss or gain ($\Theta(n^{-1/3})$ in our theoretical bounds) as opposed to the maximum loss possible ($\Theta(n^{-1/3} \ln^{1/3} n)$ in our theoretical bounds), as is indicated by our distribution bound on the losses (see item 4 in Appendix E.1). The question then is where do these edges occur in the preference list, and the answer is about one fifth of the way through (for one first has the edges providing a gain, which only go to higher up agents on the opposite side, and then one has the edges providing a loss, and these go both up and down). However, a few of the women will need to go through most of their list, as indicated by the fact that the max and min lines (for example in Figure 4) roughly coincide.

This effect can also be seen in the many-to-one experiment but it is even more stark on the worker's side. The reason is that the number of companies with whom a worker w might match which are above w, based on their public ratings alone, is $\Theta(L_c n_c)$, while the number below w is $\Theta(L_w n_c)$, a noticeably larger number. (See Appendix F.1 for a proof of these bounds.) The net effect is that there are few edges that provide w a gain, and so the low-loss edges, which are the typical matches, are reached even sooner in this setting.

Now we turn to why the number of edges in the available edge set per woman changes at the ends of the range. There are two factors at work. The first factor is due to an increasing loss bound as we move toward the bottommost women, which increases the sizes of their available edge sets. The second factor is due to public ratings. For a woman w the range of men's public ratings for its acceptable edges is $[r_m - \Theta(\overline{L}), r_m + \Theta(\overline{L})]$, where m is aligned with w. But at the ends a portion of this range will be cut off, reducing the number of



(a) Many to One Setting: Number of edges in the acceptable edge set per worker, by decile; average in blue with circles, minimum in red with stars. (n_w = 2,000, $d = 8$, $\lambda = 0.8$, $L_c = 0.14$, $L_w = 0.24$.)



(b) Number of edges in the acceptable edge set proposed during the run of DA, per worker, by decile; average in blue with circles, maximum in red with stars.

Figure 4 Many to One Setting.

acceptable edges, with the effect more pronounced for low public ratings. Because $\lambda = 0.8$, initially, as we move to lower ranked women, the gain due to increasing the loss bound dominates the loss due to a reduced public rating range, but eventually this reverses. Both effects can be clearly seen in Figure 3(a), for example.

6.3 Unique Stable Partners

Another interesting aspect of our simulations is that they showed that most agents have a unique stable partner. This is similar to the situation in the popularity model when there are short preference lists, but here this result appears to hold with full length preference lists. In Figure 5, we show the outcome on a typical run and averaged over 100 runs, for $n = 2{,}000$ in the one-to-one setting. We report the results for the men, but as the setting is symmetric they will be similar for the women. On the average, among the top 90% of agents by rank, 0.5% (10 of 1,800) had more than one stable partner, and among the remainder another 2% had multiple stable partners (40 of 200).

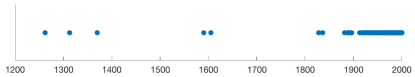
Also, as suggested by the single run illustrated in Figure 5(a), the pair around public rank 1,600 and the triple between 1,200 and 1,400 have multiple stable partners which they can swap (or exchange via a small cycle of swaps) to switch between different stable matchings. This pattern is typical for the very few men with multiple stable partners outside the bottommost region.

6.4 Constant Number of Proposals

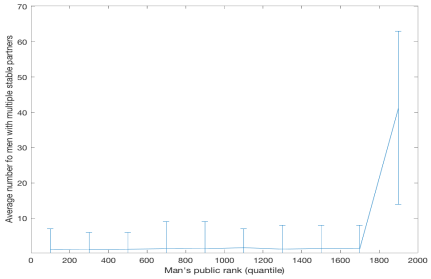
Our many-to-one experiments suggest that the length of the preference lists needed by our model are larger than those observed in the NRMP data. In addition, even though there is a simple rule for identifying these edges, in practice the communication that would be needed to identify these edges may well be excessive. In light of this it is interesting to investigate what can be done when the agents have shorter preference lists.

We simulated a strategy where the workers' preference lists contain only a constant number of edges. We construct an *Interview Edge Set* which contains the edges (w, c) satisfying the following conditions:

1. Let r_w and r_c be the public ratings of w and c respectively. Then $|r_w - r_c| \leq p$.



(a) Public ranks of men with multiple stable partners in a typical run.



(b) Average numbers of men with multiple stable partners, by decile.

Figure 5 Unique stable partners, one-to-one setting.

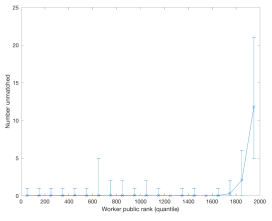
2. The private score w has for c as well as the private score of c for w are both greater than q.

We choose the parameters p and q so as to have 15 edges per agent on average. Many combinations of p and q would work. We chose a pair that caused relatively few mismatches. We then ran worker proposing DA on the Interview Edge Set.

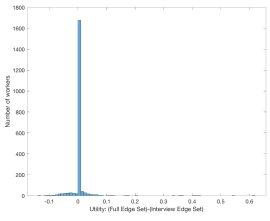
One way of identifying these edges is with the following communication protocol: the workers signal the companies which meet their criteria (the workers' criteria); the companies then reply to those workers who meet their criteria. In practice this would be a lot of communication on the workers's side, and therefore it may be that an unbalanced protocol where the workers use a larger q_w as their private score cutoff and the companies a correspondingly smaller q_c is more plausible. Clearly this will affect the losses each side incurs when there is a match, but we think it will have no effect on the non-match probability, and as non-matches are the main source of losses, we believe our simulation is indicative. We ran the above experiment with $p = 0.19$ and $q = 0.60$, with the company capacity being 8. Figure 6(a) shows the locations of unmatched workers in a typical run of this experiment while 6(b) shows the average numbers of unmatched workers per quantile (of public ratings) over 100 runs. We observe that the number of unmatched workers is very low (about 1.5% of the workers) and most of these are at the bottom of the public rating range.

Figure 6(c) compares the utility obtained by the workers in the match obtained by running worker-proposing DA on the Interview Edge Set to the utility they obtain in the worker-optimal stable match. We observe that only a small number of workers have a significantly worse outcome when restricted to the Interview Edge Set.

(a) Public ranks of unmatched workers in a typical run.



(b) Average numbers of unmatched workers by public rating decile.



(c) Distribution of workers' utilities with worker-proposing DA: (full edge set result) – (Interview edge set result).

Figure 6 Constant number of proposals.

7 Discussion and Open Problems

Our work shows that in the bounded derivatives model, apart from a sub-constant fraction of the agents, each of the other agents has $O(\ln n)$ easily identified edges on their preference list which cover all their stable matches w.h.p.

As described in Section 6, our experiments for the one-to-one setting yield a need for what appear to be impractically large preference lists. While the results in the many-to-one setting are more promising, even here the preference lists appear to be on the large side. Also, while our rule for identifying the edges to include is simple, in practice it may well require too much communication to identify these edges. At the same time, our outcome is better than what is achieved in practice: we obtain a complete match with high probability, whereas in the NRMP setting a small but significant percentage of positions are left unfilled. Our conclusion is that it remains important to understand how to effectively select smaller sets of edges.

In the popularity model, it is reasonable for each agent to simply select their favorite partners. But in the current setting, which we consider to be more realistic, it would be an ineffective strategy, as it would result in most agents remaining unmatched. Consequently, we believe the main open issue is to characterize what happens when the number of edges k that an agent can list is smaller than the size of the allowable edge set. We conjecture that following a simple protocol for selecting edges to list, such as the one we use in our experiments (see Section 6.4), will lead to an ϵ-Bayes-Nash equilibrium, where ϵ is a decreasing function of k. Strictly speaking, as the identification of allowable edges requires communication, we need to consider the possibility of strategic communication, and so one would need to define a notion of ϵ-equilibrium akin to a Subgame Perfect equilibrium. We conjecture that even with this, it would still be an ϵ-equilibrium.

Finally, it would be interesting to resolve whether the experimentally observed near uniqueness of the stable matching for non-bottom agents is a property of the linear separable model. We conjecture that in fact it also holds in the bounded derivatives model.

References

1. Atila Abdulkadiroğlu, Parag A. Pathak, and Alvin E. Roth. The new york city high school match. *American Economic Review*, 95(2):364–367, May 2005. doi:10.1257/000282805774670167.
2. Itai Ashlagi, Mark Braverman, Yash Kanoria, and Peng Shi. Clearing matching markets efficiently: Informative signals and match recommendations. *Management Science*, 66(5):2163–2193, 2019. doi:10.1287/mnsc.2018.3265.
3. Itai Ashlagi, Yash Kanoria, and Jacob D. Leshno. Unbalanced random matching markets: The stark effect of competition. *Journal of Political Economy*, 125(1), 2017. doi:10.1086/689869.
4. Peter Coles, Alexey Kushnir, and Muriel Niederle. Preference signaling in matching markets. *American Economic Journal: Microeconomics*, 5(2):99–134, May 2013. doi:10.1257/mic.5.2.99.
5. D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962. URL: http://www.jstor.org/stable/2312726.
6. Hugo Gimbert, Claire Mathieu, and Simon Mauras. Incentives in popularity-based random matching markets, 2019. arXiv:1904.03890v1.
7. Yannai A. Gonczarowski, Noam Nisan, Lior Kovalio, and Assaf Romm. Matching for the israeli "Mechinot" gap-year programs: Handling rich diversity requirements. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, EC '19, page 321, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3328526.3329620.

8 Yannai A. Gonczarowski, Noam Nisan, Rafail Ostrovsky, and Will Rosenbaum. A stable marriage requires communication. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 1003–1017, USA, 2015. Society for Industrial and Applied Mathematics.

9 Avinatan Hassidim, Assaf Romm, and Ran I. Shorrer. Redesigning the israeli psychology master's match. *American Economic Review*, 107(5):205–09, May 2017. doi:10.1257/aer.p20171048.

10 Nicole Immorlica and Mohammad Mahdian. Incentives in large random two-sided markets. *ACM Trans. Econ. Comput.*, 3(3), June 2015. doi:10.1145/2656202.

11 Yash Kanoria, Seungki Min, and Pengyu Qian. In which matching markets does the short side enjoy an advantage? In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1374–1386. SIAM, 2021.

12 Donald E. Knuth. *Mariages stables et leurs relations avec d'autres problèmes combinatoires : introduction à l'analyse mathémathique des algorithmes*. Les Presses de l'Université de Montréal, 1976.

13 Donald E. Knuth. *Stable Marriage and Its Relation to Other Combinatorial Problems: An Introduction to the Mathematical Analysis of Algorithms*, volume 10. CRM Proceedings & Lecture Notes, 1996.

14 Donald E Knuth, Rajeev Motwani, and Boris Pittel. Stable husbands. *Random Structures & Algorithms*, 1(1):1–14, 1990.

15 Fuhito Kojima and Parag A. Pathak. Incentives and stability in large two-sided matching markets. *American Economic Review*, 99(3):608–27, June 2009. doi:10.1257/aer.99.3.608.

16 Ron Kupfer. The influence of one strategic agent on the core of stable matchings. In *WINE*, 2020.

17 SangMok Lee. Incentive compatibility of large centralized matching markets. *The Review of Economic Studies*, 84(1):444–463, 2016.

18 Stephan Mertens. Random stable matchings. *Journal of Statistical Mechanics: Theory and Experiment*, 2005, October 2005. doi:10.1088/1742-5468/2005/10/P10008.

19 nrmp.org. Results and data, 2021 main residency match, May 2021. URL: https://www.nrmp.org/match-data-analytics/residency-data-reports/.

20 Boris Pittel. The average number of stable matchings. *SIAM Journal on Discrete Mathematics*, 2(4):530–549, 1989.

21 Boris Pittel. On Likely Solutions of a Stable Marriage Problem. *The Annals of Applied Probability*, 2(2):358–401, 1992. doi:10.1214/aoap/1177005708.

22 Boris Pittel. On likely solutions of the stable matching problem with unequal numbers of men and women. *Mathematics of Operations Research*, 44(1):122–146, 2019.

23 Boris Pittel, Larry Shepp, and Eugene Veklerov. On the number of fixed pairs in a random instance of the stable marriage problem. *SIAM Journal on Discrete Mathematics*, 21(4):947–958, 2008.

24 Ignacio Rios, Tomás Larroucau, Giorgiogiulio Parra, and Roberto Cominetti. Improving the chilean college admissions system. *Operations Research*, 69(4):1186–1205, 2021. doi:10.1287/opre.2021.2116.

25 Alvin E. Roth and Elliott Peranson. The redesign of the matching market for american physicians: Some engineering aspects of economic design. *American Economic Review*, 89(4):748–780, September 1999. doi:10.1257/aer.89.4.748.

26 Ran I. Shorrer. Simultaneous search: Beyond independent successes. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, EC '19, pages 347–348, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3328526.3329599.

Expander Decomposition with Fewer Inter-Cluster Edges Using a Spectral Cut Player

Daniel Agassy
Tel Aviv University, Israel

Dani Dorfman
Tel Aviv University, Israel

Haim Kaplan
Tel Aviv University, Israel

Abstract

A (ϕ, ϵ)-expander decomposition of a graph G (with n vertices and m edges) is a partition of V into clusters $V_1, \ldots, V_k$ with conductance $\Phi(G[V_i]) \geq \phi$, such that there are at most ϵm inter-cluster edges. Such a decomposition plays a crucial role in many graph algorithms. We give a randomized $\tilde{O}(m/\phi)$ time algorithm for computing a $(\phi, \phi \log^2 n)$-expander decomposition. This improves upon the $(\phi, \phi \log^3 n)$-expander decomposition also obtained in $\tilde{O}(m/\phi)$ time by [Saranurak and Wang, SODA 2019] (SW) and brings the number of inter-cluster edges within logarithmic factor of optimal.

One crucial component of SW's algorithm is a non-stop version of the cut-matching game of [Khandekar, Rao, Vazirani, JACM 2009] (KRV): The cut player does not stop when it gets from the matching player an unbalanced sparse cut, but continues to play on a trimmed part of the large side. The crux of our improvement is the design of a non-stop version of the cleverer cut player of [Orecchia, Schulman, Vazirani, Vishnoi, STOC 2008] (OSVV). The cut player of OSSV uses a more sophisticated random walk, a subtle potential function, and spectral arguments. Designing and analysing a non-stop version of this game was an explicit open question asked by SW.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases Exapander Decomposition, Cut-Matching Game

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.9

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2205.10301` [1]

Funding Israel science foundation (ISF) grant 1595/19, Israel science foundation (ISF) grant 2854/20, and the Blavatnik research foundation.

1 Introduction

The *conductance* of a cut $(S, V \setminus S)$ is $\Phi_G(S, V \setminus S) = \frac{|E(S, V \setminus S)|}{\min(\mathbf{vol}(S), \mathbf{vol}(V \setminus S))}$, where $\mathbf{vol}(S)$ is the sum of the degrees of the vertices of S. The conductance of a graph G is the smallest conductance of a cut in G.

A (ϕ, ϵ)-*expander decomposition* of a graph G is a partition of the vertices of G into clusters $V_1, \ldots, V_k$ with conductance $\Phi(G[V_i]) \geq \phi$ such that there are at most ϵm inter-cluster edges, where $\phi, \epsilon \geq 0$. We consider the problem of computing in almost linear time ($\tilde{O}(m)$ time) a (ϕ, ϵ)-expander decomposition for a given graph G and $\phi > 0$, while minimizing ϵ as a function of ϕ. It is known that a (ϕ, ϵ)-expander decomposition, with $\epsilon = O(\phi \log n)$, always exists and that $\epsilon = \Theta(\phi \log n)$ is optimal [23, 2].

Expander decomposition algorithms have been used in many cutting edge results, such as directed/undirected Laplacian solvers [27, 11], graph sparsification [9, 10], distributed algorithms [6], and maximum flow algorithms [15]. Expander decomposition was also used [10]

(in the deterministic case) in order to break the $O(\sqrt{n})$ dynamic connectivity bound and achieve an improved running time of $O(n^{o(1)})$ per operation. It was also used in the recent breakthrough result by Chen et al. [8], who showed algorithms for maximum flow and minimum cost flow in almost linear time.

Given an $f(n)$-approximation algorithm for the problem of finding a minimum conductance cut, one can get a $(\phi, O(f(n) \cdot \phi \log n))$-expander decomposition algorithm by recursively computing approximate cuts (and thus splitting V) until all components are certified as expanders. In particular, using an exact minimum conductance cut algorithm ensures the existence of an expander decomposition with $\epsilon = O(\phi \log n)$ as mentioned above. Using the polynomial algorithms of [20, 4] which provide the best approximation ratios of $O\left(\sqrt{\phi}\right)$ and $O\left(\sqrt{\log n}\right)$, respectively, for conductance, gives polynomial time expander decomposition algorithms with $\epsilon = O\left(\phi^{3/2} \log n\right)$ and $\epsilon = O\left(\phi \log^{\frac{3}{2}} n\right)$. However, these decomposition algorithms might lead to a linear recursion depth, and therefore have superlinear time complexity.

To get a near linear time algorithm using this recursive approach, one must be able to efficiently compute low conductance cuts with additional guarantees. We get such cuts using the cut-matching framework of [16] (abbreviated as KRV). In order to present our results in the appropriate context we now give a brief background on the cut-matching framework.

Cut-matching. Edge-expansion is a connectivity measure related to conductance. The *edge-expansion* of a cut $(S, V \setminus S)$ is $h_G(S, V \setminus S) = \frac{|E(S, V \setminus S)|}{\min(|S|, |V \setminus S|)}$ and the *edge-expansion* of a graph G is the smallest edge-expansion of a cut in G.

The cut-matching game is a technique that reduces the approximation task for sparsest cut (in terms of edge-expansion) to a polylogarithmic number of maximum flow problems. The resulting approximation algorithm for sparsest cut is remarkably simple and robust.

The cut-matching game is played between a *cut player* and a *matching player*, as follows. We start with an empty graph G_0 on n vertices. At round t, the cut player chooses a bisection $(S_t, \overline{S_t})$ of the vertices (we assume n is even). In response, the matching player presents a perfect matching M_t between the vertices of S_t and $\overline{S_t}$ and the game graph is updated to $G_t = G_{t-1} \cup M_t$. Note that this graph may contain parallel edges. The game ends when G_t is a sufficiently good edge-expander. The goal of this game is to devise a strategy for the cut player that maximizes the ratio $r(n) := \phi/T$, where T is the number of rounds and $\phi = h(G_T)$ is the edge-expansion of G_T. KRV showed that one can translate a cut strategy of quality $r(n)$ into a sparsest cut algorithm of approximation ratio $1/r(n)$ by applying a binary search on a sparsity parameter ϕ until we certify that $h(G) \geq \phi$ and $h(G) = O(\phi/r(n))$.

KRV devised a randomized cut-player strategy that finds the bisection using a stochastic matrix that corresponds to a random walk on all previously discovered matchings. Their walk traverses the previous matchings in order and with probability half takes a step according to each matching. They showed that the matrix corresponding to this random walk can actually be embedded (as a flow matrix) into G_t with constant congestion. They terminate when the random walk matrix is close to uniform (i.e. having constant edge-expansion), resulting in G_T for $T = O\left(\log^2 n\right)$, having constant edge-expansion.

Orecchia et al. [21] (abbreviated as OSVV) took the same approach but devised a more sophisticated random walk and used Cheeger's inequality [7] in order to show that G_T, for $T = O\left(\log^2 n\right)$, has $\Omega(\log n)$ edge-expansion. That is, they got a ratio of $r(n) = \Omega\left(\frac{1}{\log n}\right)$.

Equipped with this background we now get back to expander decomposition, and focus on the $\tilde{O}(m/\phi)$ time algorithm by Saranurak and Wang [23] (abbreviated as SW). Their algorithm is randomized, follows the recursive scheme described above, and computes a

$(\phi, \phi \log^3 n)$-expander decomposition in $O\left(\frac{m \log^4 n}{\phi}\right)$ time. Its number of inter-cluster edges is off by a factor of $O\left(\log^2 n\right)$ from optimal and off by a factor of $O\left(\log^{\frac{3}{2}} n\right)$ from the aforementioned best achievable polynomial time construction.

One core component of this algorithm is a variation of the cut-matching game (inspired by Räcke et al. [22]). In this variation, the game graph $G_t = (V_t, E_t)$ may lose vertices (*i.e.*, $V_{t+1} \subseteq V_t$) throughout the game and the objective of the cut player is to make V_T a *near expander* in G_T (see Definition 9). The result of each round does not consist of a perfect matching in V_t, but rather a subset to remove from V_t and a matching of the remaining vertices. The game ends either with a balanced cut of low conductance, or with an unbalanced cut of low conductance, such that the larger side is a *near expander*. This allows SW to avoid recurring on the large side of the cut. Indeed, if the cut is balanced, they run recursively on both sides, and if it is unbalanced, they use the fact that the large side is a *near expander* and "trim" it by finding a large subset of this side which is an expander. Then, they run recursively on the smaller side combined with the "trimmed" vertices. SW's analysis of the new cut-matching game is based on the ideas and the potential function of KRV while carefully taking into account of the shrinkage of the game graph.

An open question, raised by SW, was whether one can adapt the technique of the cut-matching strategy of OSVV to improve their decomposition. A major obstacle is how to perform an OSVV-like spectral analysis when we lose vertices throughout the process and need to bound the near-expansion of the final piece. This is challenging as the analysis of OSVV is already somewhat more complicated than that of KRV: It uses a different lazy random walk and a subtle potential to measure progress towards near expansion. Moreover Cheeger's inequality is suitable to show high expansion and the object we are targeting is a near expander.

Our contribution. In this paper we answer this question of SW affirmatively. We present and analyze an expander decomposition algorithm with a new cut-player inspired by OSVV. This improves the result of SW and gives a randomized $\tilde{O}(m/\phi)$ time algorithm for computing an $(\phi, \phi \log^2 n)$-expander decomposition (Theorem 18). This brings the number of inter-cluster edges to be off only by $O(\log n)$ factor from the best possible.

To achieve this we overcome two main technical challenges: (1) We generalize the lazy random walk of the cut player of OSVV and the subtle potential tracking its progress, to the setting in which the vertex set shrinks (by ripping off of it small cuts as in SW). (2) We show that when the generalized potential is small the remaining part of the game graph is a near expander. This required a generalization of Cheeger's inequality appropriate for our purpose (see Lemma 33).

Our techniques may be applied in similar contexts. One concrete such context is the construction of tree-cut sparsifiers. Specifically, one could try to use our technique to improve the $O\left(\log^4 n\right)$-approximate tree-cut sparsifier construction of [22] by a factor of $\log n$. (Note that [22] in fact construct a tree-flow sparsifier, which is a stronger notion.)

The cut-matching framework [16] is formalized for edge-expansion rather than conductance. Consequently, SW and others whose primary objective is conductance had to transform the graph into a *subdivision-graph* in order to use this framework. The subdivision graph is obtained by adding a new vertex (called a *split-node*) in the middle of each edge e, splitting e into a path of length two. Consequently, the analysis has to translate cuts of low expansion in the modified graph (the *subdivision graph*) to cuts of low conductance in the original graph. This transformation complicates the algorithms and their analysis.

To avoid this transformation we revisit the seminal results of KRV and OSVV and redo them directly for conductance. This is not trivial and requires subtle changes to the cut players, and the matching players, and the potentials measuring progress towards a graph with small conductance. In particular the matching player does not produce a matching anymore but rather what we call a d_G-matching, which is a graph with the same degrees as G.

Our new cut-matching algorithm is then described using this natural reformulation of the cut-matching framework directly for conductance, removing the complications that would have followed from using the split graph.

We believe that our clean presentations of the cut-matching framework for conductance would prove useful for other applications of cut-matching that require optimization for conductance rather than expansion.

Further related work. Computing the expansion and the conductance of a graph G is NP-hard [18, 25], and there is a long line of research on approximating these connectivity measures. The best known polynomial algorithms for approximating the minimum conductance cut have either $O\left(\sqrt{\log n}\right)$ [4, 24] or $O\left(\sqrt{\Phi(G)}\right)$ approximation ratios [20]. Approximation algorithms for expansion and conductance play a crucial role in algorithms for expander decomposition [23, 5, 10], expander hierarchies [12, 14], and tree flow sparsifiers [22].

In his thesis, Orecchia [19] elaborates on the two cut-matching strategies described in OSVV, one based on a lazy random walk, called C_{NAT}, and a more sophisticated one based on the *heat-kernel* random walk, called C_{EXP}. Orecchia proves (Theorem 4.1.5 of [19]) that using C_{NAT} or C_{EXP}, after $T = \Theta(\log^2 n)$ iterations, the graph G_T has expansion $\Omega(\log n)$ (and thereby conductance $\Omega\left(\frac{1}{\log n}\right)$, since it is regular with degrees $\Theta(\log^2 n)$). Orecchia also bounds the second largest eigenvalue of the normalized Laplacian of G_T. However, Orecchia does not show how to use cut-matching to get approximation algorithms for the conductance of G.

In a recent paper [3] Ameranis *et al.* use a generalized notion of expansion, also mentioned in [19], where we normalize the number of edges crossing the cut by a general measure (μ) of the smaller side of the cut. They define a corresponding generalized version of the cut-matching game, and show how to use a cut strategy for this game to get an approximation algorithm for two generalized cut problems. They claim that one can construct a cut strategy for this measure using ideas from [19].[1]

Both SW and our result can be implemented in $\tilde{O}(m)$ time using the recent result of [17], by replacing Bounded-Distance-Flow (Lemma 21) and the "Trimming Step" of [23] with the algorithm of [17, Section 8]. This $\tilde{O}(m)$ hides many log factors and requires more complicated machinery.

The structure of this paper is as follows. Section 2 contains additional definitions. In order to provide the appropriate context for our work, Section 3 gives an overview of the cut-matching games in [16] and [21] and highlights the differences between them. In the full version of this paper, we give a complete and self-contained description of these approximation algorithms directly **for conductance**. A reader knowledgeable in the Cut-Matching game can skip directly to Section 4. In Section 4 we present our new non-stop spectral cut player and expander decomposition algorithm. Section 5 contains the analysis of our algorithm. Due to the space constraints some of the proofs are omitted, and are available in the full version of this paper [1].

[1] The details of such a cut player do not appear in [3] or [19].

To be consistent with common terminology we refer to a graph with conductance at least ϕ as a ϕ-expander (rather than ϕ-conductor.) No confusion should arise since in the rest of this paper we focus on conductance and do not use the notion of edge-expansion anymore. In this paper we only focus on unweighted graphs, although our algorithm can be adapted to the case of integral, polynomially bounded weights.

2 Preliminaries

We denote the transpose of a vector or a matrix x by x'. That is, if v is a column vector then v' is the corresponding row vector. For a vector $v \in \mathbb{R}^n_{\geq 0}$, define $\sqrt{v}$ to be vector whose coordinates are the square roots of those of v. Given $A \in \mathbb{R}^{n\times n}$, we denote by $A(i,j)$ the element at the i'th row and j'th column of A. We denote by $A(i,), A(,i)$ the i'th row and column of A, respectively. We define both $A(i,)$ and $A(,i)$ as column vectors. We use the abbreviation $A(i) := A(i,)$ only with respect to the rows of A. Given a vector $v \in \mathbb{R}^n$, we denote its i'th element by $v(i)$. For disjoint $A, B \subseteq V$, we denote by $E_G(A,B)$ the set of edges connecting A and B. We sometimes omit the subscript when the graph is clear from the context. If $A = V \setminus B$, then we call (A,B) a *cut*.

▶ **Fact 1.** *Let $X, Y \in \mathbb{R}^{n\times n}, m \in \mathbb{N}$, then $\mathrm{Tr}(XY) = \mathrm{Tr}(YX)$.*

▶ **Fact 2.** *Let $X, Y \in \mathbb{R}^{n\times n}$ be symmetric matrices and let $k \in \mathbb{N}$. Then*

$$\mathrm{Tr}\left((XYX)^{2^k}\right) \leq \mathrm{Tr}\left(X^{2^k}Y^{2^k}X^{2^k}\right).$$

▶ **Definition 3** (d_G,$\mathbf{vol}_G(S)$)**.** *Given a graph G, the vector $d_G \in \mathbb{R}^n$ is defined as $d_G(v) = \deg_G(v)$. To simplify the notation, we denote $d := d_G$ whenever the graph G is clear from the context. For $S \subseteq V$, we denote by $\mathbf{vol}_G(S) := \sum_{v\in S} d_G(v)$ the* volume *of S.*

▶ **Definition 4** ($G\{A\}$)**.** *Let $G = (V,E)$ be a graph, and let $A \subseteq V$ be a set of vertices. We define the graph $G\{A\} = (V', E')$ as the graph induced by A with self-loops added to preserve the degrees: $V' = A, E' = \{\{u,v\} \in E : u,v \in A\} \cup \{\{u,u\} : u \in A, v \in V \setminus A, \{u,v\} \in E\}$.*

▶ **Definition 5** (d-Matching)**.** *Given a vector $d \in \mathbb{N}^n$ and a collection of pairs $M = \{(u_i,v_i)\}_{i=1}^m$. We say that M is a* d-matching *if the graph defined by M (i.e., the graph whose edges are M) satisfies $d_M(v) = d(v)$, for every v.*

▶ **Definition 6** (d_G-stochastic)**.** *A matrix $F \in \mathbb{R}^{n\times n}$ is* d_G-stochastic *with respect to a graph G if the following two conditions hold: (1) $F \cdot \mathbb{1}_n = d_G$ and (2) $\mathbb{1}'_n \cdot F = d'_G$.*

▶ **Definition 7** (Laplacian, Normalized Laplacian)**.** *Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix and let $d = A \cdot \mathbb{1}_n$, $D = \mathbf{diag}(d)$. The* Laplacian of A *is defined as $\mathcal{L}(A) = D - A$. The* normalized-Laplacian of A *is defined as $\mathcal{N}(A) = D^{-\frac{1}{2}}\mathcal{L}(A)D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$. The (normalized) Laplacian of an undirected graph is defined analogously using its adjacency matrix.*

▶ **Definition 8** (Conductance)**.** *Let $G = (V,E)$ and $S \subset V$, $S \neq \emptyset$. The* conductance *of the cut $(S, V \setminus S)$, denoted by $\Phi_G(S, V \setminus S)$, is*

$$\Phi_G(S, V \setminus S) = \frac{|E(S, V \setminus S)|}{\min(\mathbf{vol}(S), \mathbf{vol}(V \setminus S))}.$$

The conductance of G is defined to be $\Phi(G) = \min_{S\subseteq V} \Phi_G(S, V \setminus S)$.

▶ **Definition 9** (Expander, Near-Expander). *Let $G = (V, E)$. We say that G is a ϕ-*expander *if $\Phi(G) \geq \phi$. Let $A \subseteq V$. We say that A is a* near ϕ-expander *in G if*

$$\min_{S \subseteq A} \frac{|E(S, V \setminus S)|}{\min(\mathbf{vol}(S), \mathbf{vol}(A \setminus S))} \geq \phi.$$

That is, a near expander is allowed to use cut edges that go outside of A. Note that the above definition applies to both directed and undirected graphs.

▶ **Definition 10** (Embedding). *Let $G = (V, E)$ be an undirected graph. Let $F \in \mathbb{R}_{\geq 0}^{V \times V}$ be a matrix (not necessarily symmetric). We say that F is* embeddable in G with congestion c, *if there exists a multi-commodity flow f in G, with $|V|$ commodities, one for each vertex (vertex v is the source of its commodity), such that, simultaneously for each $(u, v) \in V \times V$, f routes $F(u, v)$ units of u's commodity from u to v, and the total flow on each edge is at most c.* [2]

If F is the weighted adjacency matrix of a graph H on the same vertex set V, we say that H is embeddable in G with congestion c if F is embeddable in G with congestion c.

▶ **Lemma 11.** *Let G, H be two graphs on the same vertex set V. Let $A \subseteq V$. Let $\alpha > 0$ be a constant such that for each $v \in V$, $d_G(v) = \alpha \cdot d_H(v)$. Assume that H is embeddable in G with congestion c, and that A is a near ϕ-expander in H. Then, A is a near $\frac{\phi}{c\alpha}$-expander in G.*

▶ **Corollary 12.** *Let G, H be two graphs on the same vertex set V. Let $\alpha > 0$ be a constant such that for each $v \in V$, $d_G(v) = \alpha \cdot d_H(v)$. Assume that H is embeddable in G with congestion c, and that H is a ϕ-expander. Then, G is a $\frac{\phi}{c\alpha}$-expander.*

Proof. This follows from Lemma 11 by choosing $A = V$. ◀

3 Approximating conductance via cut-matching

In preparation for our expander decomposition algorithm we give a high level overview of the conductance approximation algorithms of [16] and [21]. [16] and [21] described their results for edge-expansion rather than conductance. In the full version of this paper, we give a complete description and analysis of these algorithms for conductance. This translation from edge-expansion to conductance is not trivial as both the cut player, the matching player, and the analysis have to be carefully modified to take the degrees into account. Here we give a high level overview of the key components of these algorithms and the differences between them so one can better absorb our main algorithm in Section 4.2.

The cut-matching game of [16] (in the conductance setting) works as follows.

The Cut-Matching game for conductance, with parameters T and a degree vector d:

- The game is played on a series of graphs G_i. Initially, $G_0 = \emptyset$.
- In iteration t, the cut player produces two multisets of size m, $L_t, R_t \subseteq V$, such that each $v \in V$ appears in $L_t \cup R_t$ exactly $d(v)$ times.
- The matching player responds with a d-matching M_t that only matches vertices in L_t to vertices in R_t.
- We set $G_{t+1} = G_t \cup M_t$.
- The game ends at iteration T, and the *quality* of the game is $r := \Phi(G_T)$. Note that the volume of G_t increases from one iteration to the next.

[2] This definition requires to route $F(u, v) = F(v, u)$ both from u to v and from v to u if F is symmetric.

Given a strategy for the cut player of quality r, one can create a $\frac{1}{r}$ approximation algorithm for the conductance of a given graph G. To this end, the matching player has to provide matchings that can be embedded in G.

The difference between the results of [16] and [21] is mainly in the cut player. They both run the game for $T = \Theta(\log^2 n)$ iterations but [16]'s cut player achieves quality of $r = \Omega\left(\frac{1}{\log^2 n}\right)$ whereas [21]'s achieves quality of $r = \Omega\left(\frac{1}{\log n}\right)$. Notice that the cut player produces the stated expansion result in G_T regardless of the matchings given by the matching player.

3.1 KRV's Cut-Matching Game for Conductance

The cut player implicitly maintains a d_G-stochastic flow matrix (*i.e.*, representing flow demands) $F_t \in \mathbb{R}^{n\times n}$, and the graph G_t which is the union of the matchings that it obtained so far from the matching player (t is the index of the round). The flow F_t and the graph G_t have two crucial properties. First, we can embed F_t in G_t with $O(1)$ congestion (See Definition 10). Second, after $T = \Theta(\log^2 n)$ rounds, with high probability, F_T will have constant conductance.[3] Since the degrees in G_T are factor of $O(\log^2 n)$ larger than the degrees in F_T (when we think of F_T as a weighted graph) then it follows by Corollary 12 that G_T is $\Omega(1/\log^2 n)$ expander. Note that the cut player is unrelated to the input graph G in which we would like to approximate the conductance. Its goal is to produce the expander G_T.

At the beginning, $F_0 = D = \mathbf{diag}(d)$, and G_0 is the empty graph on $V = [n]$. The cut player updates F_t as follows. It draws a random unit vector $r \in \mathbb{R}^n$ orthogonal to $\sqrt{d}$ and computes the projections $u_i = \frac{1}{d(i)}\langle D^{-\frac{1}{2}} F_t(i), r\rangle$.[4] The cut player computes these projections in $O(m\log^2 n)$ time since the vector of all projections is $u := D^{-1}F_t D^{-\frac{1}{2}} \cdot r$ and F_t is defined (see below) as a multiplication of $\Theta(\log^2 n)$ sparse matrices, each having $O(m)$ non-zero entries. The cut player sorts the projections as $u_{i_1} \le \ldots \le u_{i_n}$. Consider the sequence $Q = (u_{i_1}, u_{i_1}, \ldots, u_{i_1}, u_{i_2}, u_{i_2}, \ldots, u_{i_2}, \ldots, u_{i_n}, \ldots, u_{i_n})$, where each u_{i_j} appears $d(i_j)$ times. Then, $|Q| = 2m$. Take $L_t \subseteq Q$ to be the multi-set containing the first m elements, and $R_t = Q \setminus L_t$ to be the multi-set containing the last m elements. Define $\eta \in \mathbb{R}$ such that $L_t \subseteq \{i_k : u_{i_k} \le \eta\}$ and $R_t \subseteq \{i_k : u_{i_k} \ge \eta\}$. Note that a vertex can appear both in L_t and in R_t, if $u_{i_j} = \eta$. For a vertex $v \in V$, denote by m_v the number of times v appears in L_t, and by $\bar{m}_v$ the number of times v appears in R_t. That is, except for (maybe) one vertex, for any $v \in V$, either $m_v = 0$ and $\bar{m}_v = d(v)$ or $m_v = d(v)$ and $\bar{m}_v = 0$.

The cut player hands out the partition L_t, R_t to the matching player who sends back a d_G-matching M_t (we think of M_t as an $n \times n$ matrix with at most m non-zero entries that encodes the matching) between L_t and R_t. The cut player updates its flow matrix using M_t and sets $F_{t+1}(v) = \frac{1}{2}F_t(v) + \sum_{(v,u)\in M_t} \frac{1}{2d(u)} F_t(u)$ (in matrix form $F_{t+1} = \frac{1}{2}\left(I + M_t \cdot D^{-1}\right) F_t$).[5] This update keeps F_t a d_G-stochastic matrix for all t. The cut player also defines the graph G_{t+1} as $G_{t+1} = G_t \cup M_t$. This completes the description of the cut player of [16] adapted for conductance.

[3] We think about F_t as a weighted graph on $V = [n]$. The definitions of conductance, expander and near-expander for weighted graphs are the same as Definitions 8-9 where $|E(S, V \setminus S)|$ is the sum of the weights of the edges crossing the cut.

[4] Recall that $F_t(i)$ is a column vector.

[5] Note that it is possible that some $u \in V$ appears in the sum $\sum_{(v,u)\in M_t} \frac{1}{2d(u)} F_t(u)$ multiple times, if v is matched to u multiple times in M_t.

The matching player constructs an auxiliary flow problem on $G' := G \cup \{s, t\}$, where s is a new vertex which would be the source and t is a new vertex which would be the sink. We add an arc (s, v) for each $v \in L_t$ of capacity m_v and we add an arc (v, t) of capacity $\bar{m}_v$ for each $v \in R_t$. The capacity of each edge $e \in G$ is set to be $c = \Theta\left(\frac{1}{\phi \log^2 n}\right)$, where c is an integer. The matching player computes a maximum flow g from s to t in this network.

If the value of g is less than m, then the matching player uses the minimum cut in G' separating the source from the sink to find a cut in G of conductance $O(\phi \log^2 n)$. Otherwise, it decomposes g to a set of paths, each carrying exactly one unit of flow from a vertex $u \in L_t$ to a vertex $v \in R_t$.[6] Then it defines the d_G-matching M_t as $M_t = ((v_j, u_j))_{j=1}^{m}$, where v_j and u_j are the endpoints of path j. We view M_t as a symmetric $n \times n$ matrix, such that $M_t(v, u)$ is the number of paths between v and u. The matching player connects the game to the input graph G. Indeed, by solving the maximum flow problems in G it guarantees that the expander G_T is embeddable in G with congestion $O(cT) = O(1/\phi)$. Since the degrees of G_T are a factor of $O(\log^2 n)$ larger than the degrees of G and G_T is $\Omega(1/\log^2 n)$ expander, we get that G is a $\Omega(\phi)$-expander (see Corollary 12). The following theorem summarizes the properties of this algorithm.

▶ **Theorem 13** ([16]'s cut-matching game for conductance). *Given a graph G and a parameter $\phi > 0$, there exists a randomized algorithm, whose running time is dominated by computing a polylogarithmic number of maximum flow problems, that either*

1. *Certifies that $\Phi(G) = \Omega(\phi)$ with high probability; or*
2. *Finds a cut $(S, V \setminus S)$ in G whose conductance is $\Phi_G(S, V \setminus S) = O(\phi \log^2 n)$.*

If the matching player finds a sparse cut in any iteration then we terminate with Case (2). On the other hand, if the game continues for $T = O(\log^2 n)$ rounds then since the cut player can embed F_T in G_T and the matching player can embed G_T in G, and since F_t is an expander, then we get Case (1).

The running time of the cut player is $O(m \log^4 n)$. The matching player solves $O(\log^2 n)$ maximum flow problems. By using the most recent maximum flow algorithm of [8], we get the matching player to run in $O\left(m^{1+o(1)}\right)$ time. Alternatively, we can adapt the cut-matching game, and use a version of the Bounded-Distance-Flow algorithm (which was called *Unit-Flow* in [23]; see Lemma 21), to get a running time of $\tilde{O}(\frac{m}{\phi})$ for the matching player. We can also get $\tilde{O}(m)$ running time using the recent result [17].

The key part of the analysis is to show that F_T is indeed an $\Omega(1)$-expander for any choice of d_G-matchings of the matching player. To this end, we keep track of the progress of the cut player using the potential function

$$\psi(t) = \sum_{i \in V} \sum_{j \in V} \frac{1}{d(i) \cdot d(j)} \left(F_t(i,j) - \frac{d(i)d(j)}{2m} \right)^2 = \left\| D^{-\frac{1}{2}} F_t D^{-\frac{1}{2}} - \frac{1}{2m} \sqrt{d} \sqrt{d'} \right\|_F^2$$

where the matrix norm which we use here is the Frobenius norm (sum of the squares of the entries). This potential represents the distance between the normalized flow matrix $\bar{F}_t = D^{-\frac{1}{2}} F_t D^{-\frac{1}{2}}$ and the (normalized) uniform random walk distribution $d_G d'_G / 2m$. Let $P = I - \frac{1}{2m} \sqrt{d} \sqrt{d'}$ be the projection matrix on the orthogonal complement of the span of the vector $\sqrt{d}$, then we can also write this potential as

[6] Note that there can be multiple flow paths between a pair of vertices $u \in L_t$ and $v \in R_t$. Furthermore, if $u \in L_t \cap R_t$ then it is possible that a path starts and ends at u.

$$\psi(t) = \left\|\bar{F}_t P\right\|_F^2 = \mathrm{Tr}\left((\bar{F}_t P)(\bar{F}_t P)'\right) = \mathrm{Tr}(\bar{F}_t P^2 \bar{F}_t') = \mathrm{Tr}(P\bar{F}_t'\bar{F}_t).$$

The first equality holds since F_t is d-stochastic and the last equality is due to Fact 1 (and that $P^2 = P$ as a projection matrix).

The crux of the proof is to show that after T rounds this potential is smaller than $1/(16m^2)$ which implies that for every pair of vertices u and v, $F_T(u,v) \geq d(v)d(u)/(4m)$. From this we get a lower bound of $1/4$ on the conductance of every cut.

3.2 OSVV's Cut-Matching Game for Conductance

The cut player of [21] also maintains (implicitly) a flow matrix F_t and the union G_t of the d_G-matchings it got from the matching player. Let $P = I - \frac{1}{2m}\sqrt{d}\sqrt{d'}$ be the projection to the subspace orthogonal to $\sqrt{d}$ as before (hence $P^2 = P$). Let $\delta = \Theta(\log n)$ be a power of 2. Here the matrix $W_t = (PD^{-\frac{1}{2}}F_tD^{-\frac{1}{2}}P)^\delta$ takes the role of $D^{-\frac{1}{2}}F_tD^{-\frac{1}{2}}$ from the cut player of Section 3.1.

In round t the cut player computes the projections $u_i = \frac{1}{\sqrt{d(i)}}\langle W_t(i), r\rangle$, and defines L_t and R_t based on these projections as in the previous section.[7] Then it gets a d_G-matching M_t between L_t and R_t from the matching player. It defines $N_t = \frac{\delta-1}{\delta}D + \frac{1}{\delta}M_t$ and updates the flow to be $F_{t+1} = N_t \cdot D^{-1}F_tD^{-1}N_t$. If we think of F_t as a random walk then $D^{-1}N_t$ is a lazy step that we add before and after the walk F_t to get F_{t+1}. It holds that F_{t+1} is d_G-stochastic and moreover that for all rounds t, F_t is embeddable in G_t with congestion $\frac{4}{\delta} = O(1/\log n)$. Note that here we embed F_t in G_t with smaller congestion than in Section 3.1. We can still prove, however, that F_T for $T = O(\log^2 n)$ is a $\Omega(1)$ expander and therefore, G_T is a $\Omega(1/\log n)$ expander.

The matching player solves the same flow problem as in Section 3.1 but with an integer capacity value of $c = \Theta(\frac{1}{\phi \log n})$ on the edges of G. If the value of maximum flow is less than m then it finds a cut of conductance $O(\phi \log n)$, and otherwise it returns the matching that it derives from a decomposition of the flow into paths. The matching player guarantees that the expander G_T is embeddable in G with congestion $O(cT) = O(\log n/\phi)$. Since the degrees of G_T are larger by a factor of $O(\log^2 n)$ than the degrees of G and G_T is $\Omega(1/\log n)$-expander, we get that G is a $\Omega(\phi)$-expander (see Lemma 11). The following theorem summarizes the properties of this algorithm.

▶ **Theorem 14** ([21]'s cut-matching game for conductance). *Given a graph G and a parameter $\phi > 0$, there exists a randomized algorithm, whose running time is dominated by computing a polylogarithmic number of maximum flow problems, that either*

1. *Certifies that $\Phi(G) = \Omega(\phi)$ with high probability; or*
2. *Finds a cut $(S, V \setminus S)$ in G whose conductance is $\Phi_G(S, V \setminus S) = O(\phi \log n)$.*

The running time of the cut player is dominated by computing the projections in $O(m \log^3 n)$ time per iteration for a total of $O(m \log^5 n)$ time. The matching player solves $O(\log^2 n)$ maximum flow problems. Again, we can modify the algorithm so that its running time is $\tilde{O}(\frac{m}{\phi})$ or $\tilde{O}(m)$, similarly to the previous subsection.

[7] Computing these projections takes $O(m \log^3 n)$ time since F_t is a multiplication of $\Theta(\log^2 n)$ sparse matrices, each with $O(m)$ non-zero entries. Therefore W_t is a multiplication of $\Theta(\log^3 n)$ matrices, each of which is either P or a sparse matrix.

As in Section 3.1, the key part of the analysis is to show that F_T is indeed an $\Omega(1)$-expander for any choice of d_G-matchings of the matching player. Here we keep track of the progress of the cut player using the potential function

$$\psi(t) = \left\| (D^{-\frac{1}{2}} F_t D^{-\frac{1}{2}})^\delta - \frac{1}{2m}\sqrt{d}\sqrt{d'} \right\|_F^2 .$$

Recall that $W_t = (PD^{-\frac{1}{2}} F_t D^{-\frac{1}{2}} P)^\delta$, so we can rewrite the potential function as

$$\psi(t) = \left\| (D^{-\frac{1}{2}} F_t D^{-\frac{1}{2}})^\delta P \right\|_F^2 = \mathrm{Tr}(P(D^{-\frac{1}{2}} F_t D^{-\frac{1}{2}})^{2\delta} P) \overset{(4)}{=} \mathrm{Tr}((PD^{-\frac{1}{2}} F_t D^{-\frac{1}{2}} P)^{2\delta}) = \mathrm{Tr}(W_t^2) ,$$

where equality (4) follows since F_t is d-stochastic and the fact that $P^2 = P$. A careful argument shows that after $T = O(\log^2 n)$ iterations, $\psi(T) \leq 1/n$. From this we deduce that the second smallest eigenvalue of the normalized Laplacian of F_T is at least $1/2$ and then by Cheeger's inequality [7] we get that $\Phi(F_T) = \Omega(1)$.

4 Expander decomposition via spectral Cut-Matching

To put our main result in context we first show how SW [23] modified the cut-matching game of KRV [16] for their expander decomposition algorithm.

4.1 SW's Cut-Matching for expander decomposition

SW [23] take a recursive approach to find an expander decomposition. One can use the cut-matching game to find a sparse cut, but if the cut is unbalanced, we want to avoid recursing on the large side.

In order to refrain from recursing on the large side of the cut, SW changed the cut-matching game as follows. The cut player now maintains a partition of V into a small set R and a large set $A = V \setminus R$, where initially $R = \emptyset$ and $A = V$. In each iteration the cut and the matching player interact as follows.

- The cut player computes two disjoint sets $A^l, A^r \subseteq A$ such that $|A^l| \leq n/8$ and $|A^r| \geq n/2$.
- The matching player returns a partition $(S, A \setminus S)$ of A, which may be empty ($S = \emptyset$), and a matching of $A^l \setminus S$ to a subset of $A^r \setminus S$.

The cut player computes the sets A^l and A^r by projecting the rows of a *flow-matrix* F that it maintains (as in KRV [16]) onto a random unit vector r, and applying a result by [22] to generate the sets A^l and A^r from the values of the projections. For the matching player, SW use a flow-based algorithm which simultaneously gives a cut $(S, A \setminus S)$ of conductance $O(\phi \log^2 n)$ of $G[A]$, and a matching of the vertices left in $A^l \setminus S$ to vertices of $A^r \setminus S$ (S may be empty when $G[A]$ has conductance $\geq \phi$). If the matching player found a sparse cut $(S, A \setminus S)$ then the cut player updates the partition (R, A) of V by moving S from A to R.

The game terminates either when the volume of R gets larger than $\Omega(m/\log^2 n)$ or after $O(\log^2 n)$ rounds. In the latter case, SW proved that the remaining set A (which is large) is a near ϕ-expander in G (see Definition 9).

To prove that after $T = \Theta(\log^2 n)$ iterations, the remaining set A is a near ϕ-expander, SW essentially followed the footsteps of KRV and used a similar potential. The argument is more complicated since they have to take the shrinkage of A into account. SW did not use a version of KRV suitable to conductance as we give in the full version. Therefore, they had to modify the graph by adding a split node for each edge, essentially reducing conductance to edge-expansion, a reduction that made their algorithm and analysis somewhat more complicated. The following theorem summarized the properties of the cut-matching game of [23].

▶ **Theorem 15** (Theorem 2.2 of [23]). *Given a graph $G = (V, E)$ of m edges and a parameter $0 < \phi < 1/\log^2 n$,[8] there exists a randomized algorithm, called "the cut-matching step", which takes $O((m \log n)/\phi)$ time and terminates in one of the following three cases:*

1. *We certify that G has conductance $\Phi(G) = \Omega(\phi)$ with high probability.*
2. *We find a cut (R, A) of G of conductance $\Phi_G(R, A) = O(\phi \log^2 n)$, and $\mathbf{vol}(R), \mathbf{vol}(A)$ are both $\Omega(\frac{m}{\log^2 n})$, i.e., we find a relatively balanced low conductance cut.*
3. *We find a cut (R, A) of G with $\Phi_G(R, A) \leq c_0 \phi \log^2 n$ for some constant c_0, and $\mathbf{vol}(R) \leq \frac{m}{10c_0 \log^2 n}$, and with high probability A is a near ϕ-expander in G.*

SW derived an expander decomposition algorithm from this modified cut-matching game by recursing on both sides of the cut only if Case (2) occurs. In Case (3) they find a large subset $B \subseteq A$ which is an expander (in what they called the *trimming step*), add $A \setminus B$ to R and recur only on R. The main result of [23] is as follows.

▶ **Theorem 16** (Theorem 1.2 of [23]). *Given a graph $G = (V, E)$ of m edges and a parameter ϕ, there is a randomized algorithm that with high probability finds a partitioning of V into clusters $V_1, \ldots, V_k$ such that $\forall i : \Phi_{G\{V_i\}} = \Omega(\phi)$ and there are at most $O(\phi m \log^3 n)$ inter cluster edges.[9] The running time of the algorithm is $O(m \log^4 n/\phi)$.*

4.2 Our contribution: Spectral cut player for expander decomposition

SW [23] left open the question if one can improve their expander decomposition algorithm using tools similar to the ones that allowed OSVV [21] to improve the conductance approximation algorithm of KRV [16]. We give a positive answer to this question. Specifically we improve the cut-matching game of SW and derive the following improved version of Theorem 15.

▶ **Theorem 17.** *Given a graph $G = (V, E)$ of m edges and a parameter $0 < \phi < \frac{1}{\log n}$,[10] there exists a randomized algorithm which takes $O\left(m \log^5 n + \frac{m \log^2 n}{\phi}\right)$ time and must end in one of the following three cases:*

1. *We certify that G has conductance $\Phi(G) = \Omega(\phi)$ with high probability.*
2. *We find a cut (R, A) in G of conductance $\Phi_G(R, A) = O(\phi \log n)$, and $\mathbf{vol}(R), \mathbf{vol}(A)$ are both $\Omega(\frac{m}{\log n})$, i.e, we find a relatively balanced low conductance cut.*
3. *We find a cut (R, A) with $\Phi_G(R, A) \leq c_0 \phi \log n$ for some constant c_0, and $\mathbf{vol}(R) \leq \frac{m}{10c_0 \log n}$, and with high probability A is a near $\Omega(\phi)$-expander in G.*

The proof of Theorem 17 is given in Section 5. Theorem 17 implies the following theorem

▶ **Theorem 18.** *Given a graph $G = (V, E)$ of m edges and a parameter ϕ, there is a randomized algorithm that with high probability finds a partition of V into clusters $V_1, ..., V_k$ such that $\forall i : \Phi_{G\{V_i\}} = \Omega(\phi)$ and $\sum_i |E(V_i, V \setminus V_i)| = O(\phi m \log^2 n)$. The running time of the algorithm is $O(m \log^7 n + \frac{m \log^4 n}{\phi})$.*[11]

To get Theorem 17 we use the following cut player and matching player.

[8] The theorem is trivial if $\phi \geq \frac{1}{\log^2 n}$, because any cut $(A, V \setminus A)$ has conductance $\Phi_G(A, V \setminus A) \leq 1$. We can therefore assume that $\phi < \frac{1}{\log^2 n}$.

[9] $G\{V_i\}$ is defined in Definition 4.

[10] The theorem is trivial if $\phi \geq \frac{1}{\log n}$, because any cut $(A, V \setminus A)$ has conductance $\Phi_G(A, V \setminus A) \leq 1$. We can therefore assume that $\phi < \frac{1}{\log n}$.

[11] Note that if $\phi \leq \frac{1}{\log^3 n}$, then the running time matches the running time of [23] in Theorem 16. In case that $\phi \geq \frac{1}{\log^3 n}$, we get a slightly worse running time of $O(m \log^7 n)$ instead of $O(\frac{m \log^4 n}{\phi})$.

4.3 Cut player

Like in Section 3, we consider a d-stochastic flow matrix $F_t \in \mathbb{R}^{n\times n}$, and a series of graphs G_t. F_0 is initialized as $F_0 = D := \mathbf{diag}(d)$, and G_0 is initialized as the empty graph on $V = [n]$. Here the cut player also maintains a low conductance cut $A_t \subseteq V, R_t = V \setminus A_t$, such that after $T = \Theta(\log^2 n)$ rounds, with high probability, A_T is a near expander in G_T. At the beginning, $A_0 = V$, $R_0 = \emptyset$,

Since the new cut-matching game consists of iteratively shrinking the domain $A_t \subseteq V$, we start by generalizing our matrices from Section 3 to this context of shrinking domain.

▶ **Definition 19** ($I_t, d_t, D_t, P_t, \mathbf{vol}_t$). *We define the following variables*[12]

1. $I_t \in \mathbb{R}^{n\times n}$ *is the diagonal* $0/1$ *matrix that have* 1*'s on the diagonal entries corresponding to* A_t*.*
2. $d_t = I_t \cdot d \in \mathbb{R}^n$*, i.e the projection of* d *onto* A_t*.*
3. $D_t = I_t \cdot D = \mathbf{diag}(d_t) \in \mathbb{R}^{n\times n}$*.*
4. $\mathbf{vol}_t = \mathbf{vol}_G(A_t)$*.*
5. $P_t = I_t - \frac{1}{\mathbf{vol}_t}\sqrt{d_t}\sqrt{d_t'} \in \mathbb{R}^{n\times n}$*.*

We define the matrix $W_t = (P_t D^{-\frac{1}{2}} F_t D^{-\frac{1}{2}} P_t)^\delta$, where $\delta = \Theta(\log n)$ is set in Lemma 33, that plays a crucial role in this section. This definition is similar to the definition of W_t in Section 3.2, but with P_t instead of P. This makes us "focus" only on the remaining vertices A_t, as any row/column of W_t corresponding to a vertex $v \in R_t$ is zero. The matrix W_t is used in this section to define the projections that our algorithm uses to update F_t. It is also used in Section 5.3 to define the potential that measures how far is the remaining part of the graph from a near expander. In particular, we show in Lemma 33 and Corollary 34 that if W_T^2 has small eigenvalues (which will be the case when the potential is small) then A_T is near-expander in G_T.

Let $r \in \mathbb{R}^n$ be a random unit vector. Consider the projections $u_i = \frac{1}{\sqrt{d(i)}}\langle W_t(i), r\rangle$, for $i \in A_t$. Note that because $P_t\sqrt{d_t} = 0$, and W_t is symmetric:

$$\sum_{i\in A_t} d(i)u_i = \sum_{i\in A_t}\sqrt{d(i)}\,\langle W_t(i), r\rangle = \left\langle \sum_{i\in A_t}\sqrt{d(i)}W_t(i), r\right\rangle = \left\langle W_t\sqrt{d_t}, r\right\rangle = 0$$

We use the following lemma to partition (some of) the remaining vertices into two multisets A_t^l and A_t^r.[13] The lemma follows by applying Lemma 3.3 in [22] on the multiset of the u_i's, where each u_i appears with multiplicity of $d(i)$.

▶ **Lemma 20** (Lemma 3.3 in [22]). *Given* $u_i \in \mathbb{R}$ *for all* $i \in A_t$*, such that* $\sum_{i\in A_t} d(i)u_i = 0$*, we can find in time* $O(|A_t|\cdot\log(|A_t|))$ *a multiset of source nodes* $A_t^l \subseteq A_t$*, a multiset of target nodes* $A_t^r \subseteq A_t$*, and a separation value* η *such that each* $i \in A_t$ *appears in* $A_t^l \cup A_t^r$ *at most* $d(i)$ *times, and additionally:*

1. η *separates the sets* A_t^l, A_t^r*, i.e., either* $\max_{i\in A_t^l} u_i \le \eta \le \min_{j\in A_t^r} u_j$*, or* $\min_{i\in A_t^l} u_i \ge \eta \ge \max_{j\in A_t^r} u_j$*,*
2. $|A_t^r| \ge \frac{\mathbf{vol}_t}{2}$*,* $|A_t^l| \le \frac{\mathbf{vol}_t}{8}$*,*
3. $\forall i \in A_t^l : (u_i - \eta)^2 \ge \frac{1}{9}u_i^2$*,*
4. $\sum_{i\in A_t^l} m_i u_i^2 \ge \frac{1}{80}\sum_{i\in A_t} d(i)u_i^2$*, where* m_i *is the number of times* i *appears in* A_t^l*.*

[12] These variables are the analogs of $I, d, D, \mathbf{vol}(G)$ and P (respectively) from Section 3.2 in $G[A_t]$.
[13] Note that this does not produce a bisection of V.

Note that a vertex could appear both in A_t^l and in A_t^r, if $u_{i_j} = \eta$. The cut player sends A_t^l, A_t^r and A_t to the matching player.

In turn, the matching player (see Subsection 4.4) returns a cut $(S_t, A_t \setminus S_t)$ and a matching M_t of $A_t^l \setminus S_t$ to $A_t^r \setminus S_t$ (each vertex of A_t^l is matched to a vertex of A_t^r). We add self-loops to M_t to preserve the degrees (that is, M_t is d-stochastic). Define $N_t = \frac{\delta-1}{\delta} D + \frac{1}{\delta} M_t$. The cut player then updates F_t similarly to Section 3.2: $F_{t+1} = N_t \cdot D^{-1} F_t D^{-1} N_t$. Like in the previous sections, we also define the graph G_{t+1} as $G_{t+1} = G_t \cup M_t$.[14]. We define $A_{t+1} = A_t \setminus S_t$.

4.4 Matching player

The matching player receives A_t^l and A_t^r and the current A_t. For a vertex $v \in V$, denote by m_v the number times v appears in A_t^l, and by $\bar{m}_v$ the number of times v appears in A_t^r. The matching player solves the flow problem on $G[A_t]$, specified by Lemma 21 below. This lemma is similar to Lemma B.6 in [23] and is proved using the *Bounded-Distance-Flow* algorithm (called *Unit-Flow* by [13, 23]). The details are provided in the full version of this paper [1]. Note that we can get running time of $\tilde{O}(m)$ mentioned in the introduction by replacing this subroutine is with a fair-cut computation as shown in [17, Section 8].

▶ **Lemma 21.** *Let $G = (V, E)$ be a graph with n vertices and m edges, let $A^l, A^r \subseteq V$ be multisets such that $|A^r| \geq \frac{1}{2}m, |A^l| \leq \frac{1}{8}m$, and let $0 < \phi < \frac{1}{\log n}$ be a parameter. For a vertex $v \in V$, denote by m_v the number times v appears in A^l, and by $\bar{m}_v$ the number of times v appears in A^r. Assume that $m_v + \bar{m}_v \leq d(v)$. We define the flow problem $\Pi(G)$, as the problem in which a source s is connected to each vertex $v \in A^l$ with an edge of capacity m_v and each vertex $v \in A^r$ is connected to a sink t with an edge of capacity $\bar{m}_v$. Every edge of G has the same capacity $c = \Theta\left(\frac{1}{\phi \log n}\right)$, which is an integer. A feasible flow for $\Pi(G)$ is a maximum flow that saturates all the edges outgoing from s. Then, in time $O(\frac{m}{\phi})$, we can find either*

1. *A feasible flow f for $\Pi(G)$; or*
2. *A cut S where $\Phi_G(S, V \setminus S) \leq \frac{7}{c} = O(\phi \log n)$, $\mathbf{vol}(V \setminus S) \geq \frac{1}{3}m$ and a feasible flow for the problem $\Pi(G - S)$, where we only consider the sub-graph $G[V \setminus S \cup \{s, t\}]$ (that is, vertices $v \in A^l \setminus S$ are sources of m_v units, and vertices $v \in A^r \setminus S$ are sinks of $\bar{m}_v$ units).*

▶ Remark 22. It is possible that $A^l \subseteq S$, in which case the feasible flow for $\Pi(G - S)$ is trivial (the total source mass is 0).

Let S_t be the cut returned by the lemma. If the lemma terminates with the first case, we denote $S_t = \emptyset$. Since c is an integer, we can decompose the returned flow into a set of paths (using *e.g.* dynamic trees [26]), each carrying exactly one unit of flow from a vertex $u \in A_t^l \setminus S_t$ to a vertex $v \in A_t^r \setminus S_t$. Note that multiple paths can route flow between the same pair of vertices. If $u \in A_t^l \cap A_t^r$ then it is possible that a path starts and ends at u. Each $u \in A_t^l \setminus S_t$ is the endpoint of exactly $m_u \leq d(u)$ paths, and each $v \in A_t^r \setminus S_t$ is the endpoint of *at most* $\bar{m}_v \leq d(v)$ paths. Define the "matching"[15] $\tilde{M}_t$ as $\tilde{M}_t = ((u_i, v_i))_{i=1}^{|A_t^l \setminus S_t|}$, where u_i and v_i are the endpoints of path i. We can view $\tilde{M}_t$ as a symmetric $n \times n$ matrix, such that $\tilde{M}_t(u, v)$ is the number of paths from u to v. We turn $\tilde{M}_t$ into a d-stochastic matrix by increasing its diagonal entries by $d - \tilde{M}_t \mathbb{1}_n$. Formally, we set $M_t := \tilde{M}_t + \mathbf{diag}(d - \tilde{M}_t \mathbb{1}_n)$.

[14] G_{t+1} may have self-loops.

[15] Note that this is **not** a matching or a d-matching, but rather a graph that connects vertices of A_t^ℓ to vertices of A_t^r, whose degrees are bounded by d.

Notice that $d - \tilde{M}_t \mathbb{1}_n$ has only non-negative entries, so M_t also has non-negative entries. Intuitively, we can think of M_t as the response of the matching player to the subsets A_t^l and A_t^r given by the cut player.

5 Analysis

This section is organized as follows. Subsection 5.1 presents in detail the algorithm for Theorem 17. Subsection 5.2 shows that F_t is embeddable in G_t with congestion $\frac{4}{\delta}$ and that G_t is embeddable in G with congestion $c \cdot t$. Subsection 5.3 shows that if we reach round T, then with high probability, A_T is a near $\Omega(\phi)$-expander in G. Finally, in Subsection 5.4 we prove Theorem 17.

5.1 The Algorithm

Similarly to Section 3.2, let $\delta = \Theta(\log n)$ be a power of 2, let $T = \Theta(\log^2 n)$ and $c = \Theta(\frac{1}{\phi \log n})$. We choose c to be an integer. The algorithm follows along the same lines as the algorithm of SW in Section 4.1. The only modifications are the usage of our new cut player and that the algorithm stops if $\mathbf{vol}(R_t) > \frac{m \cdot c \cdot \phi}{70} = \Omega(\frac{m}{\log n})$. In each round t, we implicitly update F_t (see Section 4.3). Like SW, in order to keep the running time near linear, we use the flow routine *Bounded-Distance-Flow* [13, 23] which is mentioned in Subsection 4.4. This routine may also return a cut $S_t \subseteq A_t$ with $\Phi_{G[A]}(S_t, A_t \setminus S_t) \leq \frac{1}{c}$, in which case we "move" S_t to R_{t+1}. After T rounds, F_T certifies that the remaining part of A_T is a near ϕ-expander.

5.2 F_t is embeddable in G

To begin the analysis of the algorithm, we first define a blocked matrix. This notion will be useful when our matrices "operate" only on vertices of A_t.

▶ **Definition 23.** *Let $A \subseteq V$. A matrix $B \in \mathbb{R}^{n \times n}$ is* A-blocked *if $B(i,j) = 0$ for all $i \neq j$ such that $(i,j) \notin A \times A$.*

▶ **Lemma 24.** *The following holds for all t:*
1. *M_t, N_t, F_t and W_t are symmetric.*
2. *M_t, N_t and F_t are d-stochastic.*
3. *M_t and N_t are A_{t+1}-blocked.*

▶ **Lemma 25.** *For all rounds t, F_t is embeddable in G_t with congestion $\frac{4}{\delta}$.*

▶ **Lemma 26.** *For all rounds t, G_t is embeddable in G with congestion ct.*

5.3 A_T is a near expander in F_T

In this section we prove that after $T = \Theta(\log^2 n)$ rounds, with high probability, A_T is a near $\Omega(1)$-expander in F_T, which will imply that it is a near $\Omega(\phi)$-expander in G.

The section is organized as follows. Lemma 27 contains matrix identities and Lemma 28 specifies a spectral property that our proof requires. We then define a potential function and lower bound the decrease in potential in Lemmas 29-32. Finally, in Lemma 33 and Corollary 34 we use the lower bound on the potential at round T, to show that with high probability A_T is a near $\Omega(1)$-expander in F_T and a near $\Omega(\phi)$-expander in G.

▶ **Lemma 27.** *The following relations hold for all t:*
1. *For any A_t-blocked d-stochastic matrix $B \in \mathbb{R}^{n \times n}$ we have $I_t D^{-\frac{1}{2}} B D^{-\frac{1}{2}} = D^{-\frac{1}{2}} B D^{-\frac{1}{2}} I_t$ and $P_t \cdot D^{-\frac{1}{2}} B D^{-\frac{1}{2}} = D^{-\frac{1}{2}} B D^{-\frac{1}{2}} \cdot P_t$.*

2. $I_t P_t = P_t$, $I_t^2 = I_t$ *and* $P_t^2 = P_t$.
3. $P_t P_{t+1} = P_{t+1} P_t = P_{t+1}$.
4. $P_t = D^{-\frac{1}{2}} \mathcal{L}(\frac{1}{\mathbf{vol}_t} d_t d_t') D^{-\frac{1}{2}}$ *(recall the Laplacian defined in Definition 7).*
5. *for any* $v \in \mathbb{R}^n$*, it holds that* $v' \mathcal{L} \left(\frac{1}{\mathbf{vol}_t} d_t d_t' \right) v = \left\| D_t^{\frac{1}{2}} v \right\|_2^2 - \frac{1}{\mathbf{vol}_t} \langle v, d_t \rangle^2$.
6. *For any* $B \in \mathbb{R}^{n \times n}$, $\mathrm{Tr}(I_t B B') = \sum_{i \in A_t} \|B(i)\|_2^2$.

We define the potential $\psi(t) = \mathrm{Tr}[W_t^2] = \sum_{i \in A_t} \|W_t(i)\|_2^2$, where W_t was defined as $W_t = (P_t D^{-\frac{1}{2}} F_t D^{-\frac{1}{2}} P_t)^\delta$. This is the same potential from Section 3.2 with the new definition of W_t. Intuitively, by projecting using P_t instead of P, the potential only "cares" about the vertices of A_t. As show in Lemma 33, having small potential will certify that A_T is a near expander in F_t.

Before we bound the decrease in potential, we recall Definition 7 of a normalized Laplacian $\mathcal{N}(A) = D^{-\frac{1}{2}} \mathcal{L}(A) D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, where A is a symmetric d-stochastic matrix.

▶ **Lemma 28.** *For any matrix* $A \in \mathbb{R}^{n \times n}$, $\mathrm{Tr}(A'(I - (D^{-\frac{1}{2}} N_t D^{-\frac{1}{2}})^{4\delta}) A) \geq \frac{1}{3} \mathrm{Tr}(A' \mathcal{N}(M_t) A)$.

The following lemma bounds the decrease in potential. The bound takes into account both the contribution of the matched vertices and the removal of S_t from A_t.

▶ **Lemma 29.** *For each round* t*,*

$$\psi(t) - \psi(t+1) \geq \frac{1}{3} \sum_{\{i,k\} \in M_t} \left\| \left(\frac{W_t(i)}{\sqrt{d(i)}} - \frac{W_t(k)}{\sqrt{d(k)}} \right) \right\|_2^2 + \sum_{j \in S_t} d(j) \left\| \frac{W_t(j)}{\sqrt{d(j)}} \right\|_2^2$$

Proof. To simplify the notation, we denote $\bar{N}_t := D^{-\frac{1}{2}} N_t D^{-\frac{1}{2}}$ and $\bar{F}_t := D^{-\frac{1}{2}} F_t D^{-\frac{1}{2}}$. We rewrite the potential in the next iteration as follows:

$$\begin{aligned}
\psi(t+1) &= \mathrm{Tr}(W_{t+1}^2) = \mathrm{Tr}\left(\left(P_{t+1} D^{-\frac{1}{2}} F_{t+1} D^{-\frac{1}{2}} P_{t+1} \right)^{2\delta} \right) \\
&= \mathrm{Tr}\left(\left(P_{t+1} D^{-\frac{1}{2}} (N_t D^{-1} F_t D^{-1} N_t) D^{-\frac{1}{2}} P_{t+1} \right)^{2\delta} \right) \\
&= \mathrm{Tr}\left(\left(P_{t+1} D^{-\frac{1}{2}} (N_t D^{-\frac{1}{2}} D^{-\frac{1}{2}} F_t D^{-\frac{1}{2}} D^{-\frac{1}{2}} N_t) D^{-\frac{1}{2}} P_{t+1} \right)^{2\delta} \right) \\
&= \mathrm{Tr}\left(\left(P_{t+1} \bar{N}_t \bar{F}_t \bar{N}_t P_{t+1} \right)^{2\delta} \right) \overset{(6)}{=} \mathrm{Tr}\left(\left(\bar{N}_t P_{t+1} \bar{F}_t P_{t+1} \bar{N}_t \right)^{2\delta} \right) \\
&\overset{(7)}{=} \mathrm{Tr}\left(\left(\bar{N}_t P_{t+1} P_t \bar{F}_t P_t P_{t+1} \bar{N}_t \right)^{2\delta} \right) = \mathrm{Tr}\left(\left(\bar{N}_t P_{t+1} (P_t \bar{F}_t P_t) P_{t+1} \bar{N}_t \right)^{2\delta} \right) ,
\end{aligned}$$

where equality (6) follows from Lemma 27 (1) for N_t (which is A_{t+1}-blocked d-stochastic by Lemma 24), and equality (7) follows from Lemma 27 (3).

By Properties (1) and (2) of Lemma 27 it holds that $\bar{N}_{t+1} P_{t+1} = P_{t+1} \bar{N}_{t+1} = P_{t+1} \bar{N}_{t+1} P_{t+1}$. Therefore, the potential can be written in terms of symmetric matrices:

$$\begin{aligned}
\psi(t+1) &= \mathrm{Tr}\left(\left((P_{t+1} \bar{N}_t P_{t+1})(P_t \bar{F}_t P_t)(P_{t+1} \bar{N}_t P_{t+1}) \right)^{2\delta} \right) \\
&\leq \mathrm{Tr}((P_{t+1} \bar{N}_t P_{t+1})^{2\delta} (P_t \bar{F}_t P_t)^{2\delta} (P_{t+1} \bar{N}_t P_{t+1})^{2\delta}) \\
&\overset{(2)}{=} \mathrm{Tr}((P_{t+1} \bar{N}_t P_{t+1})^{4\delta} (P_t \bar{F}_t P_t)^{2\delta}) = \mathrm{Tr}((\bar{N}_t P_{t+1})^{4\delta} W_t^2) \\
&\overset{(4)}{=} \mathrm{Tr}(\bar{N}_t^{4\delta} P_{t+1} W_t^2) \overset{(5)}{=} \mathrm{Tr}(\bar{N}_t^{2\delta} P_{t+1} \bar{N}_t^{2\delta} W_t^2) \overset{(6)}{=} \mathrm{Tr}(W_t \bar{N}_t^{2\delta} P_{t+1} \bar{N}_t^{2\delta} W_t) \\
&\overset{(7)}{=} \mathrm{Tr}(W_t \bar{N}_t^{2\delta} D^{-\frac{1}{2}} \mathcal{L}\left(\frac{1}{\mathbf{vol}_{t+1}} d_{t+1} d_{t+1}' \right) D^{-\frac{1}{2}} \bar{N}_t^{2\delta} W_t) \\
&= \mathrm{Tr}\left(\left(D^{-\frac{1}{2}} \cdot \bar{N}_t^{2\delta} W_t \right)' \cdot \mathcal{L}\left(\frac{1}{\mathbf{vol}_{t+1}} d_{t+1} d_{t+1}' \right) \cdot \left(D^{-\frac{1}{2}} \cdot \bar{N}_t^{2\delta} W_t \right) \right) ,
\end{aligned}$$

where the inequality follows from Fact 2, equality (2) follows from Fact 1. Equalities (4) and (5) follow from Properties (1) and (2) of Lemma 27 (and from the fact that N_t is A_{t+1}-blocked d-stochastic, by Lemma 24). Equality (6) again uses Fact 1, and equality (7) follows from Lemma 27 (4).

Let $Z_t = D^{-\frac{1}{2}} \cdot \bar{N}_t^{2\delta} W_t$. By applying Lemma 27 (5) we get

$$\begin{aligned}
\psi(t+1) &\leq \operatorname{Tr}\left(Z_t' \mathcal{L}\left(\frac{1}{\mathbf{vol}_{t+1}} d_{t+1} d_{t+1}'\right) Z_t\right) = \sum_{i=1}^{n} (Z_t(,i))' \mathcal{L}\left(\frac{1}{\mathbf{vol}_{t+1}} d_{t+1} d_{t+1}'\right) Z_t(,i) \\
&\overset{(2)}{=} \sum_{i=1}^{n} \left(\left\| D_{t+1}^{\frac{1}{2}} Z_t(,i) \right\|_2^2 - \frac{1}{\mathbf{vol}_{t+1}} \langle Z_t(,i), d_{t+1} \rangle^2\right) \leq \sum_{i=1}^{n} \left\| D_{t+1}^{\frac{1}{2}} Z_t(,i) \right\|_2^2 \\
&= \sum_{i=1}^{n} \sum_{j \in A_{t+1}} \left(\sqrt{d(j)} Z_t(j,i)\right)^2 = \sum_{j \in A_{t+1}} \left\| \left(D_{t+1}^{\frac{1}{2}} Z_t\right)(j) \right\|_2^2 \overset{(5)}{=} \sum_{j \in A_{t+1}} \left\| \left(\bar{N}_t^{2\delta} W_t\right)(j) \right\|_2^2 \\
&= \sum_{j \in A_t} \left\| \left(\bar{N}_t^{2\delta} W_t\right)(j) \right\|_2^2 - \sum_{j \in S_t} \left\| \left(\bar{N}_t^{2\delta} W_t\right)(j) \right\|_2^2, \qquad (1)
\end{aligned}$$

where equality (2) holds by Property (5) of Lemma 27 and equality (5) holds since we only sum rows in A_{t+1}. Since $\bar{N}_t$ is diagonal outside A_{t+1} (by the definition of M_t), we have that $\left(\bar{N}_t^{2\delta} W_t\right)(j) = W_t(j)$, for every $j \in S_t$. Thus,

$$\sum_{j \in S_t} \left\| \left(\bar{N}_t^{2\delta} W_t\right)(j) \right\|_2^2 = \sum_{j \in S_t} \|W_t(j)\|_2^2 . \qquad (2)$$

By Lemma 27 (6), we get

$$\begin{aligned}
\sum_{j \in A_t} \left\| \left(\bar{N}_t^{2\delta} W_t\right)(j) \right\|_2^2 &= \operatorname{Tr}(I_t \cdot \bar{N}_t^{2\delta} \cdot W_t^2 \cdot \bar{N}_t^{2\delta}) = \operatorname{Tr}(\bar{N}_t^{2\delta} \cdot I_t \cdot W_t^2 \cdot \bar{N}_t^{2\delta}) \\
&= \operatorname{Tr}(\bar{N}_t^{2\delta} \cdot W_t^2 \cdot \bar{N}_t^{2\delta}) = \operatorname{Tr}(\bar{N}_t^{4\delta} W_t^2) \qquad (3)
\end{aligned}$$

where second equality holds since N_t is A_{t+1}-blocked d-stochastic (by Lemma 24), so in particular it is A_t-blocked d-stochastic, and we can use Lemma 27 (1). The third equality holds because $I_t W_t = I_t (P_t \bar{F}_t P_t)^\delta$ and $I_t P_t = P_t$ (by Lemma 27 (2)), and the last equality follows from Fact 1. Plugging Equations (2) and (3) into (1) we get the following bound on the decrease in potential:

$$\begin{aligned}
\psi(t) - \psi(t+1) &\geq \operatorname{Tr}((I - \bar{N}_t^{4\delta}) W_t^2) + \sum_{j \in S_t} \|W_t(j)\|_2^2 \\
&= \operatorname{Tr}(W_t (I - \bar{N}_t^{4\delta}) W_t) + \sum_{j \in S_t} \|W_t(j)\|_2^2 \geq \frac{1}{3} \operatorname{Tr}(W_t \mathcal{N}(M_t) W_t) + \sum_{j \in S_t} \|W_t(j)\|_2^2 \\
&= \frac{1}{3} \operatorname{Tr}((D^{-\frac{1}{2}} W_t)' \mathcal{L}(M_t)(D^{-\frac{1}{2}} W_t)) + \sum_{j \in S_t} d(j) \left\| \frac{W_t(j)}{\sqrt{d(j)}} \right\|_2^2 \\
&= \frac{1}{3} \sum_{\{i,k\} \in M_t} \left\| \frac{W_t(i)}{\sqrt{d(i)}} - \frac{W_t(k)}{\sqrt{d(k)}} \right\|_2^2 + \sum_{j \in S_t} d(j) \left\| \frac{W_t(j)}{\sqrt{d(j)}} \right\|_2^2
\end{aligned}$$

where the second inequality follows Lemma 28, and the last equality follows from by Laplacian matrix properties. ◀

The following lemma states that the potential is expected to drop by a factor of $1 - \Omega(1/\log n)$.

▶ **Lemma 30.** *For each round t,*

$$\mathbb{E}\left[\frac{1}{3}\sum_{\{i,k\}\in M_t}\left\|\frac{W_t(i)}{\sqrt{d(i)}}-\frac{W_t(k)}{\sqrt{d(k)}}\right\|_2^2+\sum_{j\in S_t}d(j)\left\|\frac{W_t(j)}{\sqrt{d(j)}}\right\|_2^2\right]\geq\frac{1}{3000\alpha\log n}\psi(t)-\frac{3}{n^{\alpha/16}}$$

for every $\alpha > 48$, where the expectation is over the unit vector $r \in \mathbb{R}^n$.

The following two corollaries follow by Lemmas 29 and 30.

▶ **Corollary 31.** *For each round t, $\mathbb{E}[\psi(t+1)] \leq \left(1-\frac{1}{3000\alpha\log n}\right)\psi(t)+\frac{3}{n^{\alpha/16}}$, where the expectation is over the unit vector $r \in \mathbb{R}^n$.*

▶ **Corollary 32** (Total Decrease in Potential)**.** *With high probability over the choices of r, $\psi(T) \leq \frac{1}{n}$.*

The following lemma uses the low potential to derive the near-expansion of A_T in F_T.

▶ **Lemma 33** (Variation of Cheeger's inequality)**.** *Let $H = (V, \bar{E})$ be a graph on n vertices, such that F_T is its weighted adjacency matrix. Assume that $\psi(T) \leq \frac{1}{n}$. Then, A_T is a near $\frac{1}{5}$-expander in H.*

Proof. Recall that F_T is symmetric and d-stochastic. Let $k = \mathbf{vol}(A_T)$. Let $S \subseteq A_T$ be a cut, and denote $d_S \in \mathbb{R}^n$ to be the vector where $d_S(u) = \begin{cases} d(u) & \text{if } u \in S, \\ 0 & \text{otherwise.} \end{cases}$ Additionally, denote $\ell = \mathbf{vol}(S) \leq \frac{1}{2}k$. Note that $\left\|\sqrt{d_S}\right\|_2^2 = \ell$.

Denote by $\bar{\lambda} \geq 0$ the largest singular value of $X_T := P_T D^{-\frac{1}{2}} F_T D^{-\frac{1}{2}} P_T$ (square root of the largest eigenvalue of $(P_T D^{-\frac{1}{2}} F_T D^{-\frac{1}{2}} P_T)^2$). Because $\mathrm{Tr}(X_T^{2\delta}) = \psi(T) \leq \frac{1}{n}$, we have in particular that the largest eigenvalue of $X_T^{2\delta}$ is at most $\frac{1}{n}$, so we have $\bar{\lambda} \leq \frac{1}{n^{\frac{1}{\delta}}}$. We choose $\delta = \Theta(\log n)$ such that $\frac{1}{n^{\frac{1}{\delta}}} \leq \frac{1}{20}$, so $\bar{\lambda} \leq \frac{1}{20}$.

In order to prove near-expansion we need to lower bound $|E_{F_T}(S, V \setminus S)|$. We do so by upper bounding $|E_{F_T}(S, S)| = \mathbb{1}_S' F_T \mathbb{1}_S$. Note that $\mathbb{1}_S' F_T \mathbb{1}_S = \mathbb{1}_S'(I_T F_T I_T)\mathbb{1}_S$. Observe the following relation between X_T and $I_T F_T I_T$:

$$\begin{aligned}
D^{\frac{1}{2}} X_T D^{\frac{1}{2}} &= D^{\frac{1}{2}}(P_T D^{-\frac{1}{2}} F_T D^{-\frac{1}{2}} P_T) D^{\frac{1}{2}} \\
&= D^{\frac{1}{2}}(I_T - \frac{1}{k}\sqrt{d_T}\sqrt{d_T'}) D^{-\frac{1}{2}} F_T D^{-\frac{1}{2}} (I_T - \frac{1}{k}\sqrt{d_T}\sqrt{d_T'}) D^{\frac{1}{2}} \\
&= (I_T - \frac{1}{k} d_T \mathbb{1}_T') F_T (I_T - \frac{1}{k}\mathbb{1}_T d_T') \\
&= I_T F_T I_T - \frac{1}{k} d_T \mathbb{1}_T' F_T I_T - \frac{1}{k} I_T F_T \mathbb{1}_T d_T' + \frac{1}{k^2} d_T \mathbb{1}_T' F_T \mathbb{1}_T d_T'.
\end{aligned}$$

Rearranging the terms, we get

$$I_T F_T I_T = D^{\frac{1}{2}} X_T D^{\frac{1}{2}} + \frac{1}{k} d_T \mathbb{1}_T' F_T I_T + \frac{1}{k} I_T F_T \mathbb{1}_T d_T' - \frac{1}{k^2} d_T \mathbb{1}_T' F_T \mathbb{1}_T d_T' \ .$$

Therefore

$$\begin{aligned}
|E_{F_T}(S, S)| &= \mathbb{1}_S' F_T \mathbb{1}_S \\
&= \mathbb{1}_S' \left(D^{\frac{1}{2}} X_T D^{\frac{1}{2}} + \frac{1}{k} d_T \mathbb{1}_T' F_T I_T + \frac{1}{k} I_T F_T \mathbb{1}_T d_T' - \frac{1}{k^2} d_T \mathbb{1}_T' F_T \mathbb{1}_T d_T' \right) \mathbb{1}_S.
\end{aligned}$$

We analyze the summands separately. The first summand can be bounded using $\bar{\lambda}$, the largest singular value of X_T:

$$\mathbb{1}'_S D^{\frac{1}{2}} X_T D^{\frac{1}{2}} \mathbb{1}_S = \sqrt{d'_S} X \sqrt{d_S} = \left\langle \sqrt{d_S}, X\sqrt{d_S} \right\rangle \leq \left\| \sqrt{d_S} \right\|_2 \left\| X_T \sqrt{d_S} \right\|_2 \leq \left\| \sqrt{d_S} \right\|_2^2 \bar{\lambda} \leq \frac{\ell}{20},$$

where the first inequality is the Cauchy-Schwartz inequality. Observe that the second and third summands are equal:

$$\frac{1}{k}\mathbb{1}'_S d_T \mathbb{1}'_T F_T I_T \mathbb{1}_S = \frac{\ell}{k}\mathbb{1}'_T F_T \mathbb{1}_S = \frac{\ell}{k}\mathbb{1}'_S F_T \mathbb{1}_T = \frac{1}{k}\mathbb{1}'_S I_T F_T \mathbb{1}_T d'_T \mathbb{1}_S,$$

where the second equality follows by transposing and since F_T is symmetric. We now bound the sum of the second, third and fourth summands:

$$\mathbb{1}'_S \left(\frac{1}{k} d_T \mathbb{1}'_T F_T I_T + \frac{1}{k} I_T F_T \mathbb{1}_T d'_T - \frac{1}{k^2} d_T \mathbb{1}'_T F_T \mathbb{1}_T d'_T \right) \mathbb{1}_S = \frac{2\ell}{k}\mathbb{1}'_T F_T \mathbb{1}_S - \frac{\ell^2}{k^2}\mathbb{1}'_T F_T \mathbb{1}_T$$
$$\leq \left(\frac{2\ell}{k} - \frac{\ell^2}{k^2} \right) \mathbb{1}'_T F_T \mathbb{1}_S \leq \left(\frac{2\ell}{k} - \frac{\ell^2}{k^2} \right) \mathbb{1}' F_T \mathbb{1}_S = \left(\frac{2\ell}{k} - \frac{\ell^2}{k^2} \right) d' \mathbb{1}_S = \frac{\ell}{k}\left(2 - \frac{\ell}{k}\right)\ell,$$

where the first inequality follows since $S \subseteq A_t$. Note that $\frac{\ell}{k} \in [0, \frac{1}{2}]$. The last inequality is true because for $\frac{\ell}{k}$ in this range, $\left(\frac{2\ell}{k} - \frac{\ell^2}{k^2} \right) \geq 0$. Moreover, because $\frac{\ell}{k} \in \left[0, \frac{1}{2}\right]$, we have $\frac{\ell}{k}\left(2 - \frac{\ell}{k}\right) \leq \frac{3}{4}$. Therefore, $|E_{F_T}(S, S)| \leq \frac{1}{20}\ell + \frac{3}{4}\ell = \frac{4}{5}\ell$, and

$$|E(S, V \setminus S)| = \sum_{u \in S} \sum_{v \in V \setminus S} F_T(u, v) = \sum_{u \in S} \sum_{v \in V} F_T(u, v) - \sum_{u \in S} \sum_{v \in S} F_T(u, v)$$
$$= \sum_{u \in S} d(u) - \sum_{u \in S} \sum_{v \in S} F_T(u, v) \geq \ell - \frac{4}{5}\ell = \frac{\ell}{5}\ .$$

So $\Phi_G(S, V \setminus S) = \frac{|E(S, V \setminus S)|}{\mathbf{vol}(S)} \geq \frac{1}{5}$, and this is true for all cuts $S \subseteq A$ with $\frac{\mathbf{vol}(S)}{\mathbf{vol}(A)} \leq \frac{1}{2}$. ◀

▶ **Corollary 34.** *If we reach round T, then with high probability, A_T is a near $\Omega(\phi)$-expander in G.*

Proof. Assume we reach round T. By Corollary 32 and Lemma 33, with high probability, A_T is a near $\Omega(1)$-expander in F_T. By Lemma 25, F_T is embeddable in G_T with congestion $O(\frac{1}{\delta})$. Note that G_T is a union of T d_G-matchings $\{M_t\}_{t=1}^T$, each having $d_{M_t} = d_G = d_{F_T}$. Therefore, $d_{G_T} = T \cdot d_{F_T}$. So by Lemma 11, A_T is a near $\Omega(\frac{\delta}{T})$-expander in G_T. By Lemma 26, G_T is embeddable in G with congestion cT. Together with the fact that $d_G = \frac{1}{T} \cdot d_{G_T}$, we get by Lemma 11 again, that A is a near $\Omega(\frac{\delta}{cT})$-expander in G. Recall that $c = O\left(\frac{1}{\phi \log n}\right)$, $\delta = \Theta(\log n)$, and $T = O(\log^2 n)$. Therefore, A is an near $\Omega(\phi)$-expander in G. ◀

5.4 Proof of Theorem 17

We are now ready to prove Theorem 17.

Proof of Theorem 17. Recall that S_t denotes the cut returned by Lemma 21 at iteration t, so that $A_{t+1} = A_t \setminus S_t$.

Observe first that in any round t, we have $\Phi_G(A_t, R_t) \leq \frac{7}{c} = O(\phi \log n)$. This is because $R_t = \bigcup_{0 \leq t' < t} S_{t'}$ and by Lemma 21, for each t', $\Phi_{G[A_{t'}]}(S_{t'}, V \setminus S_{t'}) \leq \frac{7}{c} = O(\phi \log n)$.

Assume the algorithm terminates because $\mathbf{vol}(R_t) > \frac{m \cdot c \cdot \phi}{70} = \Omega(\frac{m}{\log n})$. We also have, by Lemma 21, that $\mathbf{vol}(A_t) = \Omega(m) = \Omega(\frac{m}{\log n})$. Then (A_t, R_t) is a balanced cut where $\Phi_G(A_t, R_t) = O(\phi \log n)$. We end in Case (2) of Theorem 17.

Otherwise, the algorithm reached round T and we apply Corollary 34. If $R = \emptyset$, then we obtain the first case of Theorem 17 because the whole vertex set V is, with high probability, a near $\Omega(\phi)$-expander, which means that G is an $\Omega(\phi)$-expander. Otherwise, we write $c = \frac{c_1}{\phi \log n}$ for some constant c_1, and let $c_0 := \frac{7}{c_1}$. We have $\Phi_G(A_T, R_T) \leq \frac{7}{c} = \frac{7}{c_1}\phi \log n = c_0 \phi \log n$. Additionally, $\mathbf{vol}(R_T) \leq \frac{m \cdot c \cdot \phi}{70} = \frac{m \cdot c_1}{70 \log n} = \frac{m}{10 c_0 \log n}$, and, with high probability, A_T is a near $\Omega(\phi)$-expander in G, which means we obtain the third case of Theorem 17.

To bound the running time, note that the algorithm performs at most $T = \Theta(\log^2 n)$ iterations and each iteration's running time is dominated by computing $W_t \cdot r$ in $O(t \cdot \delta \cdot m)$ and by running the matching player (Lemma 21) in $O(\frac{m}{\phi})$. ◀

References

1. Daniel Agassy, Dani Dorfman, and Haim Kaplan. Expander decomposition with fewer inter-cluster edges using a spectral cut player. *arXiv preprint*, 2022. `arXiv:2205.10301`.
2. Vedat Levi Alev, Nima Anari, Lap Chi Lau, and Shayan Oveis Gharan. Graph clustering using effective resistance. *arXiv preprint*, 2017. `arXiv:1711.06530`.
3. Konstantinos Ameranis, Lorenzo Orecchia, Kunal Talwar, and Charalampos Tsourakakis. Practical almost-linear-time approximation algorithms for hybrid and overlapping graph clustering. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, pages 17071–17093, 2022.
4. Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 56(2):1–37, 2009.
5. Yi-Jun Chang and Thatchaphol Saranurak. Improved distributed expander decomposition and nearly optimal triangle enumeration. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODS)*, pages 66–73, 2019.
6. Yi-Jun Chang and Thatchaphol Saranurak. Deterministic distributed expander decomposition and routing with applications in distributed derandomization. In *Proceedings of the 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 377–388, 2020.
7. Jeff Cheeger. A lower bound for the smallest eigenvalue of the laplacian. *Problems in analysis*, 625(195-199):110, 1970.
8. Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. *arXiv preprint*, 2022. `arXiv:2203.00671`.
9. Timothy Chu, Yu Gao, Richard Peng, Sushant Sachdeva, Saurabh Sawlani, and Junxing Wang. Graph sparsification, spectral sketches, and faster resistance computation via short cycle decompositions. In *Proceedings of the 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 18–85, 2020.
10. Julia Chuzhoy, Yu Gao, Jason Li, Danupon Nanongkai, Richard Peng, and Thatchaphol Saranurak. A deterministic algorithm for balanced cut with applications to dynamic connectivity, flows, and beyond. In *Proceedings of the 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1158–1167, 2020.
11. Michael B. Cohen, Jonathan Kelner, John Peebles, Richard Peng, Anup B. Rao, Aaron Sidford, and Adrian Vladu. Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs. In *Proceedings of the 49th Annual Symposium on Theory of Computing (STOC)*, pages 410–419, 2017.
12. Gramoz Goranci, Harald Räcke, Thatchaphol Saranurak, and Zihan Tan. The expander hierarchy and its applications to dynamic graph algorithms. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2212–2228. SIAM, 2021.
13. Monika Henzinger, Satish Rao, and Di Wang. Local flow partitioning for faster edge connectivity. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1919–1938, 2017.

14 Yiding Hua, Rasmus Kyng, Maximilian Probst Gutenberg, and Zihang Wu. Maintaining expander decompositions via sparse cuts. *arXiv preprint*, 2022. `arXiv:2204.02519`.

15 Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 217–226, 2014.

16 Rohit Khandekar, Satish Rao, and Umesh Vazirani. Graph partitioning using single commodity flows. *Journal of the ACM*, 56(4):1–15, 2009.

17 Jason Li, Danupon Nanongkai, Debmalya Panigrahi, and Thatchaphol Saranurak. Near-linear time approximations for cut problems via fair cuts. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 240–275, 2023.

18 David W. Matula and Farhad Shahrokhi. Sparsest cuts and bottlenecks in graphs. *Discrete Applied Mathematics*, 27(1-2):113–123, 1990.

19 Lorenzo Orecchia. *Fast approximation algorithms for graph partitioning using spectral and semidefinite-programming techniques*. PhD thesis, Berkeley, 2011.

20 Lorenzo Orecchia, Sushant Sachdeva, and Nisheeth K. Vishnoi. Approximating the exponential, the lanczos method and an $\tilde{O}(m)$-time spectral algorithm for balanced separator. In *Proceedings of the 44th Annual Symposium on Theory of Computing (STOC)*, pages 1141–1160, 2012.

21 Lorenzo Orecchia, Leonard J. Schulman, Umesh V. Vazirani, and Nisheeth K. Vishnoi. On partitioning graphs via single commodity flows. In *Proceedings of the 40th Annual Symposium on Theory of Computing (STOC)*, pages 461–470, 2008.

22 Harald Räcke, Chintan Shah, and Hanjo Täubig. Computing cut-based hierarchical decompositions in almost linear time. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 227–238, 2014.

23 Thatchaphol Saranurak and Di Wang. Expander decomposition and pruning: Faster, stronger, and simpler. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2616–2635, 2019.

24 Jonah Sherman. Breaking the multicommodity flow barrier for $O(\sqrt{\log n})$-approximations to sparsest cut. In *Proceedings of the 50th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 363–372. IEEE, 2009.

25 Jiří Šíma and Satu Elisa Schaeffer. On the NP-completeness of some graph cluster measures. In *International Conference on Current Trends in Theory and Practice of Computer Science (SOFTSEM)*, pages 530–537. Springer, 2006.

26 Daniel D. Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26(3):362–391, 1983.

27 Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th Annual Symposium on Theory of Computing (FOCS)*, pages 81–90, 2004.

Locality in Online, Dynamic, Sequential, and Distributed Graph Algorithms

Amirreza Akbari
Aalto University, Espoo, Finland

Navid Eslami
Aalto University, Espoo, Finland
Sharif University of Technology, Tehran, Iran

Henrik Lievonen
Aalto University, Espoo, Finland

Darya Melnyk
Aalto University, Espoo, Finland
TU Berlin, Germany

Joona Särkijärvi
Aalto University, Espoo, Finland

Jukka Suomela
Aalto University, Espoo, Finland

Abstract

In this work, we give a unifying view of locality in four settings: distributed algorithms, sequential greedy algorithms, dynamic algorithms, and online algorithms. We introduce a new model of computing, called the online-LOCAL model: the adversary presents the nodes of the input graph one by one, in the same way as in classical online algorithms, but for each node we get to see its radius-T neighborhood before choosing the output. Instead of *looking ahead* in time, we have the power of *looking around* in space.

We compare the online-LOCAL model with three other models: the LOCAL model of distributed computing, where each node produces its output based on its radius-T neighborhood, the SLOCAL model, which is the sequential counterpart of LOCAL, and the dynamic-LOCAL model, where changes in the dynamic input graph only influence the radius-T neighborhood of the point of change.

The SLOCAL and dynamic-LOCAL models are sandwiched between the LOCAL and online-LOCAL models. In general, all four models are distinct, but we study in particular *locally checkable labeling problems* (LCLs), which is a family of graph problems extensively studied in the context of distributed graph algorithms. We prove that for LCL problems in paths, cycles, and rooted trees, all four models are roughly equivalent: the locality of any LCL problem falls in the same broad class – $O(\log^* n)$, $\Theta(\log n)$, or $n^{\Theta(1)}$ – in all four models. In particular, this result enables one to generalize prior lower-bound results from the LOCAL model to all four models, and it also allows one to simulate e.g. dynamic-LOCAL algorithms efficiently in the LOCAL model.

We also show that this equivalence does not hold in two-dimensional grids or general bipartite graphs. We provide an online-LOCAL algorithm with locality $O(\log n)$ for the 3-coloring problem in bipartite graphs – this is a problem with locality $\Omega(n^{1/2})$ in the LOCAL model and $\Omega(n^{1/10})$ in the SLOCAL model.

2012 ACM Subject Classification Theory of computation → Online algorithms; Computing methodologies → Distributed algorithms; Theory of computation → Dynamic graph algorithms

Keywords and phrases Online computation, spatial advice, distributed algorithms, computational complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.10

Category Track A: Algorithms, Complexity and Games

50th International Colloquium on Automata, Languages, and Programming (ICALP 2023).
Editors: Kousha Etessami, Uriel Feige, and Gabriele Puppis; Article No. 10; pp. 10:1–10:20

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Related Version *Full Version*: https://arxiv.org/abs/2109.06593

Funding This work was supported in part by the Academy of Finland, Grant 333837.

Acknowledgements We would like to thank Alkida Balliu, Sameep Dahal, Chetan Gupta, Fabian Kuhn, Dennis Olivetti, Jan Studený, and Jara Uitto for useful discussions. We would also like to thank the anonymous reviewers for the very helpful feedback they have provided for previous versions of this work.

1 Introduction

In *online graph algorithms*, the adversary reveals the input graph one node at a time: In step i, the adversary presents some node v_i. The algorithm gets to see the subgraph induced by the nodes $v_1, \ldots, v_i$, and the algorithm has to respond by labeling node v_i. For example, in *online graph coloring*, the algorithm has to pick a color for node v_i in such a way that the end result is a proper coloring of the input graph.

In this work, we consider a more general setting, which we call the online-LOCAL model: in step i, the algorithm gets to see the subgraph induced by all nodes that are within distance T from $v_1, \ldots, v_i$. That is, the algorithm can look T hops further in the input graph around the nodes presented by the adversary. For $T = 0$, this corresponds to the usual online model. For $T = n$, on the other hand, any graph problem (in connected graphs) is solvable in this setting. The key question is what value of T is sufficient for a given graph problem. Put otherwise, what is the *locality* of a given online problem?

It turns out that this question is very closely connected to questions studied in the context of *distributed* graph algorithms, and we can identify problem classes in which the online setting coincides with the distributed setting. However, we also see surprising differences, the prime example being the problem of 3-coloring bipartite graphs, which is a fundamentally *global* problem in the distributed setting, while we show that we can do much better in the online setting.

1.1 Contribution 1: landscape of models

In Section 2, we define the online-LOCAL model, and we also recall the definitions of three models familiar from the fields of distributed and dynamic graph algorithms:

- The LOCAL model [37, 44]: the nodes are processed *simultaneously* in parallel; each node looks at its radius-T neighborhood and picks its own output.
- The SLOCAL model [26]: the nodes are processed *sequentially* in an adversarial order; each node in its turn looks at its radius-T neighborhood and picks its own output (note that here the output of a node may depend on the outputs of other nodes that were previously processed).
- The dynamic-LOCAL model: the adversary *constructs* the graph by adding nodes and edges one by one; after each modification, the algorithm can only update the solution within the radius-T neighborhood of the point of change. While this is not one of the standard models, there is a number of papers [3, 9, 11, 21, 28, 34, 43] that implicitly make use of this model. We also occasionally consider the dynamic-LOCAL$^{\pm}$ model, in which we can have both additions and deletions.

In Section 3, we show that we can sandwich SLOCAL and both versions of dynamic-LOCAL between LOCAL and online-LOCAL, as shown in Figure 1. In particular, this implies that if we can prove that LOCAL and online-LOCAL are equally expressive for some family of graph problems, we immediately get the same result also for SLOCAL and dynamic-LOCAL. This is indeed what we achieve in our next contribution.

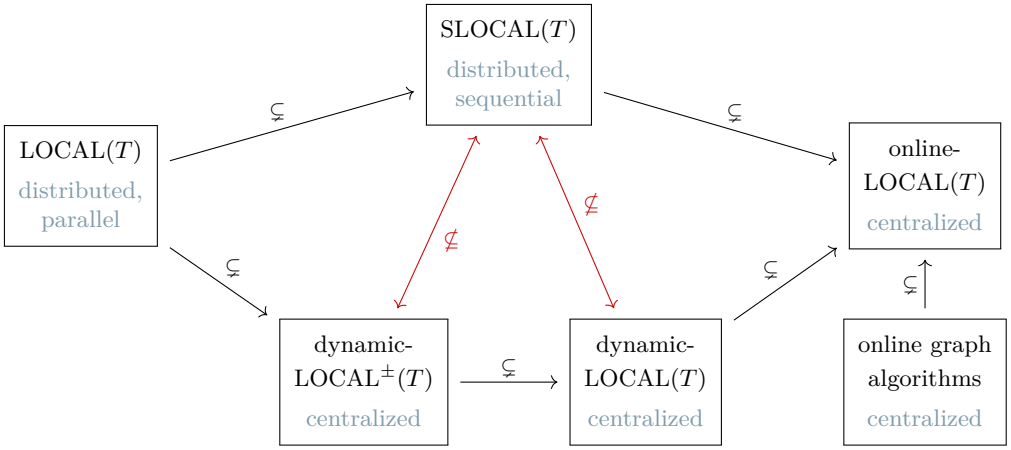


Figure 1 The landscape of models – see Section 2 for the definitions. Each box represents the set of problems solvable with locality $O(T)$ in the given model of computation (except for online graph algorithms, which do not have a notion of locality). For example, any problem with locality $O(T)$ in the LOCAL model can also be solved with locality $O(T)$ in both the SLOCAL and the online-LOCAL models. On the other hand, the SLOCAL and the dynamic-LOCAL models are incomparable, as there exist problems that are solvable with locality $O(T)$ in one of the models but that require $\omega(T)$ locality in the other model.

1.2 Contribution 2: collapse for LCLs in rooted regular trees

A lot of focus in the study of distributed graph algorithms and the LOCAL model has been on understanding *locally checkable labeling problems* (in brief, LCLs) [4,5,7,8,13,15,17,18,42]. These are problems where feasible solutions are defined with local constraints – a solution is feasible if it looks good in all constant-radius neighborhoods (see Definition 3). Coloring graphs of maximum degree Δ with k colors (for some constants Δ and k) is an example of an LCL problem.

In Section 5, we study LCL problems in *paths*, *cycles*, and *rooted regular trees*, and we show that all four models are approximately equally strong in these settings – see Table 1. For example, we show that if the locality of an LCL problem in rooted trees is $n^{\Theta(1)}$ in the LOCAL model, it is also $n^{\Theta(1)}$ in the dynamic-LOCAL, SLOCAL, and online-LOCAL models.

Table 1 In all four models, LCL problems have got the same locality classes in paths, cycles, and rooted trees. Here $n^{\Theta(1)}$ refers to locality $\Theta(n^\alpha)$ for some constant $\alpha > 0$. See Section 5 for more details.

	LOCAL		SLOCAL		dynamic-LOCAL		online-LOCAL
LCLs in paths and cycles	$O(\log^* n)$	$\Leftrightarrow$	$O(1)$	$\Leftrightarrow$	$O(1)$	$\Leftrightarrow$	$O(1)$
	$\Theta(n)$	$\Leftrightarrow$	$\Theta(n)$	$\Leftrightarrow$	$\Theta(n)$	$\Leftrightarrow$	$\Theta(n)$
LCLs in rooted regular trees	$O(\log^* n)$	$\Leftrightarrow$	$O(1)$	$\Leftrightarrow$	$O(1)$	$\Leftrightarrow$	$O(1)$
	$\Theta(\log n)$	$\Leftrightarrow$	$\Theta(\log n)$	$\Leftrightarrow$	$\Theta(\log n)$	$\Leftrightarrow$	$\Theta(\log n)$
	$n^{\Theta(1)}$	$\Leftrightarrow$	$n^{\Theta(1)}$	$\Leftrightarrow$	$n^{\Theta(1)}$	$\Leftrightarrow$	$n^{\Theta(1)}$

By previous work, we know that LCL complexities in paths, cycles, and rooted regular trees are *decidable* in the LOCAL model [4, 7, 18]. Our equivalence result allows us to extend this decidability to the SLOCAL, dynamic-LOCAL, and online-LOCAL models. For example, there is an algorithm that gets as input the description of an LCL problem in rooted trees and produces as output in which of the classes of Table 1 it is, for any of the four models.

1.3 Contribution 3: 3-coloring bipartite graphs in online-LOCAL

Given the equivalence results for LCLs in paths, cycles, and rooted regular trees, it would be tempting to conjecture that the models are approximately equal for LCLs in any graph class. In Section 4, we show that this is not the case: we provide an exponential separation between the SLOCAL and online-LOCAL models for the problem of 3-coloring bipartite graphs. By prior work it is known that in the LOCAL model, the locality of 3-coloring is $\Omega(n^{1/2})$ in two-dimensional grids [13], which are a special case of bipartite graphs; using this result we can derive a lower bound of $\Omega(n^{1/10})$ also for the SLOCAL model (see the full version). In Section 4, we prove the following:

▶ **Theorem 1.** *There is an online-LOCAL algorithm that finds a* 3*-coloring in bipartite graphs with locality* $O(\log n)$*.*

That is, in bipartite graphs, there is an LCL problem that requires locality $n^{\Omega(1)}$ in the LOCAL and SLOCAL models and is solvable with locality $O(\log n)$ in the online-LOCAL model.

The algorithm that we present for coloring bipartite graphs is also interesting from the perspective of competitive analysis of online algorithms. With locality $O(\log n)$, the online-LOCAL algorithm can compute a 3-coloring. Since bipartite graphs are 2-colorable, this gives us a 1.5-competitive online-LOCAL algorithm. On the other hand, it has been shown that any online algorithm for coloring bipartite graphs is at least $\Omega(\log n)$-competitive [10], with a matching algorithm presented in [38]. This result shows how much the competitive ratio of an algorithm can be improved by increasing the view of each node.

1.4 Contribution 4: locality of online coloring

As a corollary of our work, together with results on distributed graph coloring from prior work [13, 19, 37], we now have a near-complete understanding of the locality of graph coloring in paths, cycles, rooted trees, and grids in both distributed and online settings. Table 2 summarizes our key results. For the proofs of the localities in the online-LOCAL model, see Sections 4 and 5.

1.5 Motivation

Before we discuss the key technical ideas, we briefly explain the practical motivation for the study of online-LOCAL and dynamic-LOCAL models. As a running example, consider the challenge of providing public services (e.g. local schools) in a rapidly growing city. The future is unknown, depending on future political decisions, yet the residents need services every day.

The offline solution would result in a city-wide redesign of e.g. the entire school network every time the city plan is revised; this is not only costly but also disruptive. On the other hand, a strict online solution without any consideration of the future would commit to a solution that is far from optimal. The models that we study in this work capture the essence of two natural strategies for coping with such a situation:

Table 2 The locality of the vertex coloring problem in distributed vs. online settings, for two graph families: rooted trees and paths (with n nodes) and 2-dimensional grids (with $\sqrt{n} \times \sqrt{n}$ nodes). Note that most results for the online-LOCAL model follow from the equivalence results discussed in Section 1.2. See Sections 4 and 5 and the full version for more details.

	colors	competitive ratio	LOCAL	SLOCAL	online-LOCAL	references
Rooted trees and paths	2	1	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	trivial
	3	1.5	$\Theta(\log^* n)$	$O(1)$	$O(1)$	[19, 37]
	4	2	$\Theta(\log^* n)$	$O(1)$	$O(1)$	[19, 37]
	...					
Grids	2	1	$\Theta(n^{1/2})$	$\Theta(n^{1/2})$	$\Theta(n^{1/2})$	trivial
	3	1.5	$\Theta(n^{1/2})$	$\boldsymbol{\Omega(n^{1/10})}$	$\boldsymbol{O(\log n)}$	Section 4, [13]
	4	2	$\Theta(\log^* n)$	$O(1)$	$O(1)$	[13]
	5	2.5	$\Theta(\log^* n)$	$O(1)$	0	[13]
	...					

- Redesign the public service network only in the local neighborhoods in which there are new developments. This corresponds to the dynamic-LOCAL model, and the locality parameter T captures the redesign cost and the disruption it causes.
- Wait until new developments in a neighborhood are completed before providing permanent public services in the area. This corresponds to the online-LOCAL model, and the locality parameter T captures the inconvenience for the residents (the width of the "buffer zone" without permanent public services around areas in which the city plan is not yet finalized).

These two models make it possible to formally explore trade-offs between the quality of the solution in the long term vs. the inconvenience of those living close to the areas where the city is changing. In these kinds of scenarios, the key challenge is not related to the computational cost of finding an optimal solution (which is traditionally considered in the context of dynamic graph algorithms) but to the quality of the solution (which is typically the focus in online algorithms). The key constraint is not the availability of information on the current state of the world (which is traditionally considered in distributed graph algorithms), but the cost of changing the solution.

1.6 Techniques and key ideas

For the equivalence in paths and cycles (Section 5.1), we first make use of *pumping-style arguments* that were introduced by Chang and Pettie [17] in the context of distributed algorithms. We show that such ideas can be used to also analyze locality in the context of online algorithms: we start by showing that we can "speed up" (or "further localize") online-LOCAL algorithms with a *sublinear* locality to online-LOCAL algorithms with a *constant* locality in paths and cycles. Then, once we have reached constant locality in the online-LOCAL model, we show how to turn it into a LOCAL-model algorithm with locality $O(\log^* n)$. In this part, the key insight is that *we cannot directly simulate* online-LOCAL in LOCAL. Instead, we can use an online-LOCAL algorithm with a constant locality to find a *canonical labeling* for each possible input-labeled fragment, and use this information to design a LOCAL-model algorithm. The main trick is that we first present only disconnected path fragments to an online-LOCAL algorithm, and force it to commit to some output labeling in each fragment without knowing how the fragments are connected to each other.

In the case of rooted regular trees (Section 5.2), we face the same fundamental challenge: we cannot directly simulate black-box online-LOCAL algorithms in the LOCAL model. Instead, we need to look at the combinatorial properties of a given LCL problem Π. We proceed in two steps: (1) Assume that the locality of Π is $n^{\Theta(1)}$ in the LOCAL model; we need to show that the locality is $n^{\Theta(1)}$ also in the online-LOCAL model. Using the result of [7], high LOCAL-model locality implies that the structure of Π has to have certain "inflexibilities", and we use this property to present a strategy that the adversary can use to force any online-LOCAL algorithm with locality $n^{o(1)}$ to fail. (2) Assume that we have an online-LOCAL algorithm A for Π with locality $o(\log n)$; we need to show that the locality is $O(\log^* n)$ in the LOCAL model. Here we design a family of inputs and a strategy of the adversary that forces algorithm A to construct a "certificate" (in the sense of [7]) that shows that Π is efficiently solvable in the LOCAL model.

For 3-coloring bipartite graphs in online-LOCAL (Section 4), we make use of the following ideas. We maintain a collection of graph fragments such that each of the fragments has got a boundary that is properly 2-colored. Each such fragment has got one of two possible parities (let us call them here "odd" and "even") with respect to the underlying bipartition. We do not know the global parity of a given graph fragment until we have seen almost the entire graph. Nevertheless, it is possible to merge two fragments and maintain the invariant: if two fragments A and B have parities that are not compatible with each other, we can surround either A or B with a *barrier* that uses the third color, and thus change parities. Now we can merge A and B into one fragment that has got a properly 2-colored boundary. The key observation here is that we can make a *choice* between surrounding A vs. B, and if we always pick the one with the smallest number of nested barriers, we never need to use more than a logarithmic number of nested barriers. It turns out that this is enough to ensure that seeing up to distance $O(\log n)$ suffices to color any node chosen by the adversary.

1.7 Open questions

Our work gives rise to a number of open questions. First, we can take a more fine-grained view of the results in Tables 1 and 2:

1. Is there any problem in rooted trees with locality $\Theta(n^\alpha)$ in the online-LOCAL model and locality $\Theta(n^\beta)$ in the LOCAL model, for some $\alpha < \beta$?
2. Is it possible to find a 3-coloring in 2-dimensional grids in the dynamic-LOCAL model with locality $O(\log n)$?
3. Is it possible to find a 3-coloring in bipartite graphs in the online-LOCAL model with locality $o(\log n)$?

Perhaps even more interesting is what happens if we consider *unrooted* trees instead of rooted trees. In unrooted trees we can separate *randomized* and *deterministic* versions of the LOCAL model [16], and SLOCAL is strong enough to derandomize randomized LOCAL-model algorithms [25]; hence the key question is:

4. Does randomized-LOCAL $\approx$ SLOCAL $\approx$ dynamic-LOCAL $\approx$ online-LOCAL hold for LCL problems in unrooted trees?

Finally, our work shows a trade-off between the competitive ratio and the locality of coloring: With locality $O(\log n)$, one can achieve $O(1)$-coloring of a bipartite graph, and to achieve locality 0, one needs to use $\Omega(\log n)$ colors. This raises the following question:

5. What trade-offs exist between the locality and number of colors needed to color a (bipartite) graph in the online-LOCAL model?

2 Definitions and related work

Throughout this work, graphs are simple, undirected, and finite, unless otherwise stated. We write $G = (V, E)$ for a graph G with the set of nodes V and the set of edges E, and we use n to denote the number of nodes in the graph. For a node v and a natural number T, we use $B(v, T)$ to denote the set of all nodes in the radius-T neighborhood of node v. For a set of nodes U, we write $G[U]$ for the subgraph of G induced by U. By radius-T neighborhood of v we refer to the induced subgraph $G[B(v, T)]$, together with possible input and output labelings.

We use the following notation for graph problems. We write $\mathcal{G}$ for the family of graphs, Σ for the set of input labels, and Γ for the set of output labels. For a graph $G = (V, E)$, we write $I\colon V \to \Sigma$ for the input labeling and $L\colon V \to \Gamma$ for the output labeling. We consider here node labelings, but edge labelings can be defined in an analogous manner. A *graph problem* Π associates with each possible *input* (G, I) a set of feasible *solutions* L; this assignment must be invariant under graph isomorphism.

Locality. In what follows, we define five models of computing: LOCAL, SLOCAL, two versions of dynamic-LOCAL, and online-LOCAL. In all of these models, an algorithm is characterized by a *locality* T (a.k.a. locality radius, local horizon, time complexity, or round complexity, depending on the context). In general, T can be a function of n. We assume that the algorithm knows the value of n.

In each of these models $\mathcal{M}$, we say that algorithm $\mathcal{A}$ solves problem Π if, for each possible input (G, I) and for each possible adversarial choice, the labeling L produced by $\mathcal{A}$ is a feasible solution. We say that problem Π has locality T in model $\mathcal{M}$ if T is the pointwise smallest function such that there exists an $\mathcal{M}$-model algorithm $\mathcal{A}$ that solves Π with locality at most T.

LOCAL model. In the LOCAL model of distributed computing [37, 44], the adversary labels the nodes with unique identifiers from $\{1, 2, \ldots, \mathrm{poly}(n)\}$. In a LOCAL model algorithm, each node in parallel chooses its local output based on its radius-T neighborhood (the output may depend on the graph structure, input labels, and the unique identifiers).

Naor and Stockmeyer [42] initiated the study of the locality of LCL problems (see Definition 3) in the LOCAL model. Today, LCL problems are well classified with respect to their locality for the special cases of paths [4, 5, 13, 18, 42], grids [13], directed and undirected trees [5, 7, 8, 15, 17] as well as general graphs [13, 42], with only a few unknown gaps [7].

SLOCAL model. In the SLOCAL model [26], we have got adversarial unique identifiers similar to the LOCAL model, but the nodes are processed sequentially with respect to an adversarial input sequence $\sigma = v_1, v_2, v_3, \ldots, v_n$. Each node v is equipped with an unbounded local memory; initially, all local memories are empty. When a node v is processed, it can query the local memories of the nodes in its radius-T neighborhood, and based on this information, it has to decide what is its own final output and what to store in its own local memory.

The SLOCAL model has been used as a tool to e.g. better understand the role of randomness in the LOCAL model [25, 26]. It is also well-known that SLOCAL is strictly stronger than LOCAL. For example, it is trivial to find a maximal independent set greedily in the SLOCAL model, while this is a nontrivial problem in the general case in the LOCAL model [6, 36]. There are many LCL problems with LOCAL-locality $\Theta(\log^* n)$ [19, 37], and

all of them have SLOCAL-locality $O(1)$. There are also LCL problems (e.g. the so-called *sinkless orientation* problem), where the locality in the (deterministic) LOCAL model is $\Theta(\log n)$, while the locality in the (deterministic) SLOCAL model is $\Theta(\log \log n)$ [16, 25].

Dynamic-LOCAL model. To our knowledge, there is no standard definition or name for what we call dynamic-LOCAL here; however, the idea has appeared implicitly in a wide range of work. For example, many efficient dynamic algorithms for graph problems, such as vertex or edge coloring, maximal independent set, and maximal matching, also satisfy the property that the solution is only modified in the (immediate) local neighborhood of a point of change [3, 9, 11, 21, 28, 34, 43], and hence all of them fall in the class dynamic-LOCAL.

We use the following definition for dynamic-LOCAL: Computation starts with an empty graph G_0. In step i, the adversary constructs a supergraph G_i of G_{i-1} such that G_i and G_{i-1} differ in only one edge or one node; let C_i denote the set of nodes v in G_i with $G_i[B(v,T)] \neq G_{i-1}[B(v,T)]$, i.e., nodes that are within distance at most T from the point of change. In each step, the algorithm has to produce a feasible labeling L_i for problem Π in graph G_i, and the labeling can only be modified in the local neighborhood of a point of change, i.e., $L_i(v) = L_{i-1}(v)$ for all $v \notin C_i$.

Note that we defined the dynamic-LOCAL model for the *incremental* case, where nodes and edges are only added. If we do not require that G_i is a supergraph of G_{i-1}, we arrive at what we call the dynamic-LOCAL$^{\pm}$ model with both additions and deletions.

Online graph algorithms. In online graph algorithms, nodes are processed sequentially with respect to an adversarial input sequence $\sigma = v_1, v_2, \ldots, v_n$. Let $\sigma_i = v_1, v_2, \ldots, v_i$ denote the first i nodes of the sequence, and let $G_i = G[\{v_1, v_2, \ldots, v_i\}]$ be the subgraph induced by these nodes. When the adversary presents a node v_i, the algorithm has to label v_i based on σ_i and G_i.

Online algorithms on graphs have been studied for many problems such as matching [35] and independent set [30], but closest to our work is the extensive literature on online graph coloring [1, 10, 29, 31, 33, 38, 47]. There is also prior work that has considered various ways to strengthen the notion of online algorithms; the performance of online algorithms can be improved by letting the algorithm know the input graph [20, 32], by giving it an advice string [12, 14, 22] with knowledge about the request sequence, or allowing the algorithm to delay decisions [23]. The online-LOCAL model can be interpreted as online graph algorithms with spatial advice, and it can also be interpreted as a model where the online algorithm can delay its decision for node v until it has seen the whole neighborhood around v (this interpretation is equivalent to the definition we give next).

Online-LOCAL model. We define the online-LOCAL model as follows. The nodes are processed sequentially with respect to an adversarial input sequence $\sigma = v_1, v_2, \ldots, v_n$. Let $\sigma_i = v_1, v_2, \ldots, v_i$ denote the first i nodes of the sequence, and let $G_i = G\big[\bigcup_{j=1}^{i} B(v_j, T)\big]$ be the subgraph induced by the radius-T neighborhoods of these nodes. When the adversary presents a node v_i, the algorithm has to label v_i based on σ_i and G_i.

Observe that any online graph algorithm is an online-LOCAL algorithm with locality 0. Further note that in the online-LOCAL model, unique identifiers would not give any additional information. This is because the nodes can always be numbered with respect to the point in time when the algorithm first sees them in some G_i.

Yet another way to interpret the online-LOCAL model is that it is an extension of the SLOCAL model, where the algorithm is equipped with unbounded *global memory* where it can store arbitrary information on what has been revealed so far. When they introduced

the SLOCAL model, Ghaffari, Kuhn, and Maus [26] mentioned the possibility of such an extension but pointed out that it would make the model "too powerful", as just one bit of global memory would already make it possible to solve e.g. leader election (and this observation already shows that the online-LOCAL model is indeed strictly stronger than the SLOCAL model). In our work, we show that even though online-LOCAL can trivially solve e.g. leader election thanks to the global memory, it is not that easy to exploit this extra power in the context of LCL problems. Indeed, online-LOCAL turns out to be as weak as SLOCAL when we look at LCL problems in paths, cycles, and rooted trees.

Local computation algorithms. We do not discuss local computation algorithms (LCAs) [2, 24, 39, 40, 46, 46] in this work in more detail, but we briefly point out a direct connection between the online-LOCAL model and LCAs. It is known that for a broad family of graph problems (that includes LCLs), we can w.l.o.g. assume that whenever the adversary queries a node v, the LCA makes probes to learn a *connected* subgraph around node v [27]. For such problems, an online-LOCAL algorithm with locality T is at least as strong as an LCA that makes T probes per query: an LCA can learn some *subgraph* of the radius-T neighborhood of v and, depending on the size of the state space, remember some part of that, while in the online-LOCAL model the algorithm can learn the entire radius-T neighborhood of v and remember all of that. We leave a more detailed exploration of the distinction between distance (how far to see) and volume (how much to see), in the spirit of e.g. [41, 45], for future work.

3 Landscape of models

As an introduction to the models, we first check that all relations in Figure 1 indeed hold. In each case, we are interested in *asymptotic* equivalence: for example, when we claim that $A \subsetneq B$, the interpretation is that locality T in model A implies locality $O(T)$ in model B, but the converse is not true. Note that the relation between the online-LOCAL problems and the online graph algorithms has already been discussed in Sections 1 and 2.

Inclusions. Let us first argue that the subset relations in Figure 1 hold. These cases are trivial:

- Any LOCAL algorithm can be simulated in the SLOCAL model, and any SLOCAL algorithm can be simulated in the online-LOCAL model (this is easiest to see if one interprets online-LOCAL as an extension of SLOCAL with the global memory).
- Any dynamic-LOCAL$^{\pm}$ algorithm can be directly used in the dynamic-LOCAL model (an algorithm that supports both additions and deletions can handle additions).

These are a bit more interesting cases:

- To simulate a LOCAL algorithm A in the dynamic-LOCAL$^{\pm}$ model, we can simply recompute the entire output with A after each change. If the locality of A is T, then the output of A only changes within distance T from a point of change.
- To simulate a dynamic-LOCAL algorithm A in the online-LOCAL model, we proceed as follows: When the adversary reveals a node v, we feed v along with the new nodes in its radius-$O(T)$ neighborhood to A edge by edge. Now there will not be any further changes within distance T from v, and hence A will not change the label $L(v)$ of v anymore. Hence the online-LOCAL algorithm can also label v with $L(v)$.

Table 3 Problems that we use to separate the models, and the bounds that we show for their locality.

Problem	LOCAL	SLOCAL	dynamic-LOCAL$^{\pm}$	dynamic-LOCAL	online-LOCAL
3-coloring paths	$\Omega(\log^* n)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
weak reconstruction	$\Omega(n)$	$\Omega(n)$	$O(1)$	$O(1)$	$O(1)$
cycle detection	$\Omega(n)$	$\Omega(n)$	$\Omega(n)$	$O(1)$	$O(1)$
component-wise leader election	$\Omega(n)$	$\Omega(n)$	$\Omega(n)$	$\Omega(n)$	$O(1)$
nested orientation	$\omega(1)$	$O(1)$	$\omega(1)$	$\omega(1)$	$O(1)$

Separations. To prove the separations of Figure 1, we make use of the classic distributed graph problem of 3-*coloring paths*, as well as the following problems that are constructed to highlight the differences between the models:

- *Weak reconstruction:* in each connected component C there has to be at least one node v such that its label $L(v)$ is an encoding of a graph isomorphic to C.
- *Cycle detection:* for each cycle there has to be at least one node that outputs "yes", and each node that outputs "yes" has to be part of at least one cycle.
- *Component-wise leader election:* in each connected component exactly one node has to be marked as the leader.
- *Nested orientation:* find an acyclic orientation of the edges and label each node recursively with its own identifier, the identifiers of its neighbors, and the labels of its in-neighbors (see the full version for the precise definition).

We can prove the bounds shown in Table 3 for the locality of these problems in the five models; see the full version for the details. Now each separation in Figure 1 follows from one of the rows of Table 3.

4 3-coloring bipartite graphs

In this section, we present our Contribution 3: we design an algorithm for 3-coloring bipartite graphs in the online-LOCAL model and show that this gives us an exponential separation between the SLOCAL and online-LOCAL models. This section also serves as an introduction into the algorithmic techniques that work in online-LOCAL. Equipped with this understanding, in Section 5, we start to develop more technical tools that we need for our Contribution 2.

By prior work [13], it is known that the locality of 3-coloring in $\sqrt{n} \times \sqrt{n}$ grids is at least $\Omega(\sqrt{n})$ in the LOCAL model. The aforementioned paper considers the case of *toroidal* grid graphs, but the same argument can be applied for non-toroidal grids (in essence, if you could color locally anywhere in the middle of a non-toroidal grid, you could also apply the same algorithm to color a toroidal grid). We can easily extend this result to show a polynomial lower bound for 3-coloring grids in the SLOCAL model:

▶ **Theorem 2.** *There is no SLOCAL algorithm that finds a 3-coloring in 2-dimensional grids with locality $o(n^{1/10})$.*

To prove the result, we show that we can simulate SLOCAL algorithms sufficiently efficiently in the LOCAL model. We use the standard technique of first precomputing a distance-$o(n^{1/10})$ coloring, and then using the colors as a schedule for applying the SLOCAL algorithm. Such a simulation can be done efficiently and would lead to a LOCAL algorithm running in $o(\sqrt{n})$ time, which is a contradiction. The full proof of the lower bound is presented

in the full version of the paper. As grids are bipartite graphs, the problem of 3-coloring in grids already gives an exponential separation between the SLOCAL and online-LOCAL models. For the special case of grids, we discuss the known locality bounds for the coloring problem in the full version. A summary of these results can be found in Table 2.

In Section 4.1, we introduce the 3-coloring algorithm in the online-LOCAL, and we use the special case of grids in order to visualize it. Besides providing a natural separation between the SLOCAL and online-LOCAL models, the 3-coloring problem also shows the advantage of allowing online algorithms to look around: while the best online coloring algorithm on bipartite graphs is $\Theta(\log n)$-competitive, our algorithm in the online-LOCAL model achieves a competitive ratio of 1.5.

Note that an optimal solution would be to color a bipartite graph with 2 colors. In all models that we consider here, we know it is not possible to solve 2-coloring with locality $o(n)$, the worst case being a path with n nodes. We show that allowing an online-LOCAL algorithm to use only one extra color makes it possible to find a valid coloring with locality $O(\log n)$:

▶ **Theorem 1.** *There is an online-LOCAL algorithm that finds a 3-coloring in bipartite graphs with locality $O(\log n)$.*

4.1 Algorithm for 3-coloring bipartite graphs in online-LOCAL

Algorithm overview. The high-level idea of our online-LOCAL algorithm is to color the presented nodes of the graph with 2 colors until the algorithm sees two areas where the 2-colorings are not compatible. In essence, when the adversary presents a node far from any other node the algorithm has seen, the algorithm blindly start constructing a 2-coloring. When the adversary presents nodes in the neighborhood of already colored nodes, the algorithm simply expands the 2-colored component – we call such a component a *group*. The algorithm keeps expanding such properly 2-colored groups until, eventually, two groups with incompatible 2-colorings meet (i.e., groups that have different *parities*). Then, the algorithm uses the third color in order to create a *barrier* around one of the groups, effectively flipping its parity. Our algorithm thereby makes use of the knowledge of previously queried neighborhoods that are given by the online-LOCAL model: the algorithm is *committing* to colors for nodes in the revealed subgraphs before they are queried.

Algorithm in detail. At the beginning, no nodes are revealed to the algorithm, and we therefore say that all nodes are *unseen*. We refer to connected components of the subgraph G_i of G as *groups*. With each of these groups, we associate a *border count*, which is a natural number that is initially 0. The algorithm uses colors 0 and 1 for the 2-coloring, reserving color 2 as the barrier color. Each time the adversary points at a node v_i, the algorithm gets to see the radius-T neighborhood $B(v_i, T)$ of this node. Now consider different types of nodes in $B(v_i, T+1)$. There are three different cases that the algorithm needs to address (we visualize them in Figures 2–4 using grids as an example):

1. *All nodes in $B(v_i, T+1)$ are unseen.* In this case, the nodes in $B(v_i, T)$ form a new connected component, i.e. a new *group*. This group has a *border count* of 0. The algorithm colors v_i with 0, thus fixing the parity for this group (see Figure 2).
2. *The algorithm has already seen some nodes in $B(v_i, T+1)$, but all of them belong to the same group.* In this case, the adversary has shown an area next to an existing group. If v_i was already committed to a color, the algorithm uses that color. Otherwise, the algorithm colors v_i according to the 2-coloring of the group. All nodes in $B(v_i, T)$ are now considered to be in this group (see Figure 3).

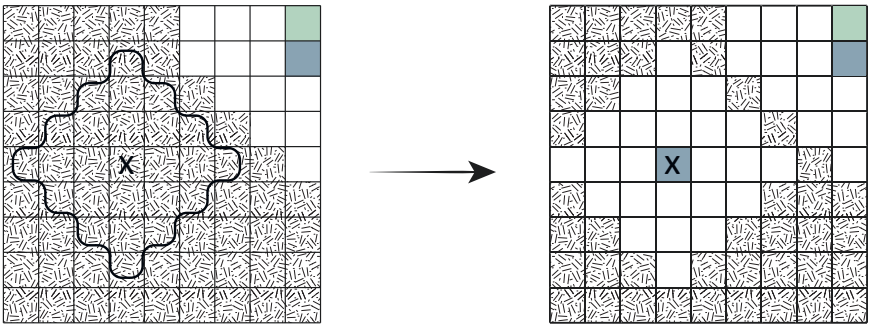


Figure 2 3-coloring algorithm, case 1/3. The adversary queries node x. Here node x is in the middle of an unseen region (shaded). The algorithm creates a new group (white) and fixes the color of node x arbitrarily.

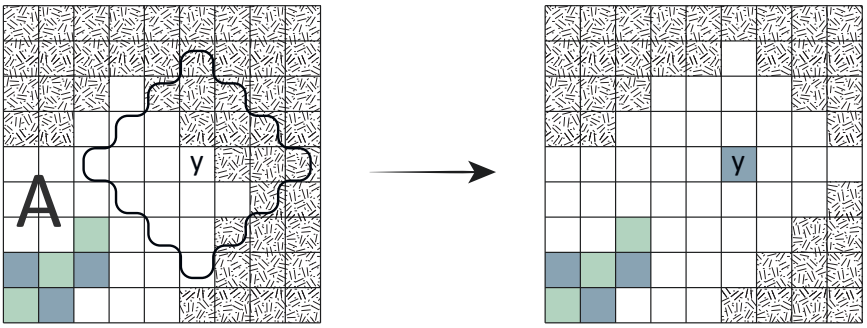


Figure 3 3-coloring algorithm, case 2/3. The adversary queries node y. Some nodes in the local neighborhood of y are already part of a group (white), and hence y joins this group. The algorithm fixes the color of node y so that it is consistent with the coloring of the group.

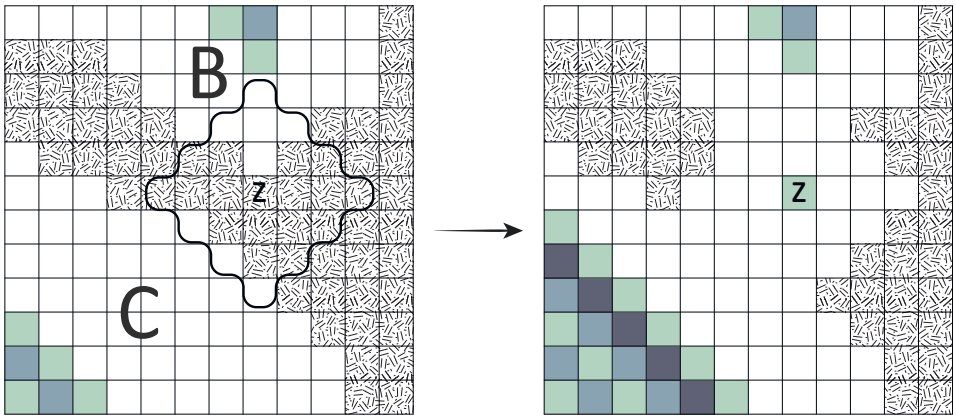


Figure 4 3-coloring algorithm, case 3/3. The adversary queries node z. Some nodes in the local neighborhood of z belong to two different groups, B and C. The algorithm merges the groups. As they have incompatible parities, the algorithm adds a new border around one of the groups, in this case C, as both groups have the same number of borders around them and the algorithm can choose arbitrarily. Nodes in the local neighborhood of z join the group, and z is colored in a way compatible with the coloring of the newly created group.

Algorithm 1 join_groups(A, B).

Input: Groups A, B
Output: Group X
if *A and B have different parities* **then**
  Let S be the group with the smaller border count. If they are equal, $S = A$;
  For all nodes of color 0 in S, commit all uncolored neighbors to color 1;
  For all nodes of color 1 in S, commit all uncolored neighbors to color 2;
  For all nodes of color 2 in S, commit all uncolored neighbors to color 0;
  Increase border count of S by 1.
end
Set all nodes in groups A, B to be in group X;
Set the border count for X to be the maximum of border counts for A, B and S;
return X

3. *There are nodes in $B(v_i, T+1)$ that belong to different groups.* In this case, the algorithm has to join groups. Here, we only define the join of two groups A and B; if there are more groups, this join can be applied iteratively.
   If A and B have different parities (i.e., the 2-colorings at their boundaries are not compatible), the algorithm takes the group with the smaller border count and uses a layer of nodes of color 2 to create a barrier that changes its parity, and then it increases the group's border count; see Algorithm 1 for the details. Then, the algorithm joins the groups, that are now compatible, and sets the border count of the newly created group to the maximum of the border counts of A and B.
   By merging all groups in the local neighborhood of v_i, the algorithm eventually ends up in a situation where v_i only sees nodes in a single group, and we are in a scenario similar to case 2 above: nodes in the local neighborhood of v_i also join the newly created group, and if v_i has not already committed to a color, the algorithm colors it according to the 2-coloring of this group (see Figure 4).

4.2 Analysis of the 3-coloring Algorithm in online-LOCAL

In order to show the correctness of the coloring algorithm, we first prove that this process creates a valid 3-coloring provided that all our commitments remain within the visible area, that is, inside subgraph G_i. Next, we show that by choosing $T(n) = O(\log n)$, all our commitments indeed remain inside the visible area. Together, these parts prove Theorem 1.

Validity of the 3-coloring. We first prove that our algorithm always continues a valid 3-coloring, as long as it does not need to make commitments to unseen nodes. We consider all three cases of the algorithm individually.

1. *All nodes in $B(v_i, T+1)$ are unseen.* In this case, the algorithm colors v_i with 0. As all neighboring nodes were unseen, they have not been committed to any color, and thus this case causes no errors.
2. *The algorithm has already seen some nodes in $B(v_i, T+1)$, but all of them belong to the same group.* In this case, the algorithm would either use the committed color or the parity of the group. As previously committed colors do not cause errors, and the group has consistent parity, this case cannot cause any errors.

3. *There are nodes in $B(v_i, T+1)$ that belong to different groups.* In this case, we want to join groups without breaking the coloring. If the two groups have the same parity, clearly, no errors can be caused by continuing the 2-coloring. The interesting case is when the two groups have different parities. Then, we need to show that the new commitments made by Algorithm 1 do not create any errors.

   Let S be the group with the smaller border count. By examining Algorithm 1, we can see that all colored nodes that have uncolored neighbors are either of color 0 or color 1: only in line 4, nodes can be colored with color 2, and all of those nodes' neighbors are then colored in line 5. Thus, in order for an error to occur, there either needs to be two nodes of colors 0 and 1 that have uncolored neighbors and different parities in S, or the algorithm commits to a color of a node that it has not yet seen. This could cause an error, as two groups could commit a single node to two different colors.

   As for the first case, we assume that all nodes in S that have uncolored neighbors also have consistent parity. This trivially holds for a group that has border count 0, as all colored nodes in it have the same parity. From the assumption, it follows that all nodes colored with 1 in line 3 have the same parity, so they cannot create an error. After this, all colored nodes with uncolored neighbors in the group have the same parity, and are colored with 1. Thus all nodes colored with 2 in line 4 also have the same parity, as do the nodes colored with 0 in line 5. As these are the only lines where nodes are colored, this procedure cannot create any errors. It also ensures that, after the procedure, the only colored nodes in the group that have uncolored neighbors are the nodes colored in line 5, which have the same parity. Therefore, our assumption holds for all groups. Those nodes also have a parity different from the nodes in S that had uncolored neighbors before this procedure, so in essence, we have flipped the parity of group S to match the parity of the other group.

   As for the second case, this can be avoided by choosing a large enough T, so that all commitments remain within the visible area of G_i. Next, we discuss how to choose such a T.

Locality of the 3-coloring algorithm. In this part, we prove that by choosing locality $T(n) = 3\lceil \log_2 n \rceil = O(\log n)$, no nodes outside the visible area of G_i need to be committed.

We first make the observation that a group with border count b contains at least 2^b nodes; this is a simple induction:

$b = 0$: A newly created group contains at least 1 node.

$b > 0$: Consider the cases in which Algorithm 1 returns a group X with border count b. One possibility is that A or B already had border count b, and hence by assumption it already contained at least 2^b nodes. The only other possibility is that both A and B had border count exactly $b-1$, they had different parities, one of the border counts was increased, and hence X has now got a border count of b. But, in this case, both A and B contained at least 2^{b-1} nodes each.

Hence the border count is bounded by $b \leq \log_2 n$ in a graph with n nodes.

We next consider the maximum distance between a node that the adversary has queried and a node with a committed color. Note that the only place where the algorithm commits a color to a node that the adversary has not queried yet is when building a border around a group. There are three steps (lines 3–5) where the algorithm commits to the color of a neighbor of a committed node, and thus effectively extends the distance by at most one in each step. Therefore, if the border count is b, in the worst case, the algorithm commits a color for a node that is within distance $3b$ from a node that was queried by the adversary.

As we have $b \leq \log_2 n$, a locality of $3\lceil \log_2 n \rceil \geq 3b$ suffices to ensure that all the commitments of the algorithm are safely within the visible region. This concludes the proof of Theorem 1.

5 LCL problems in paths, cycles, rooted regular trees

We just showed that the online-LOCAL model is much more powerful than LOCAL and SLOCAL for an LCL on bipartite graphs and grids. In this section, we discuss what happens when we restrict our attention to LCL problems in paths, cycles, and trees. We start by defining LCL problems more formally.

We say that Π is a *locally verifiable problem* with verification radius r if the following holds: there is a collection of labeled local neighborhoods $\mathcal{T}$ such that L is a feasible solution for input (G, I) if and only if for all nodes v, the radius-r neighborhood of v in (G, I, L) is in $\mathcal{T}$. Informally, a solution is feasible if it looks good in all radius-r neighborhoods.

▶ **Definition 3** (Locally checkable labeling [42]). *A locally verifiable problem Π is a* locally checkable labeling (LCL) *problem if the set of the input labels Σ is finite, the set of the output labels Γ is finite, and there is a natural number Δ such that maximum degree of any graph $G \in \mathcal{G}$ is at most Δ.*

Note that in LCL problems, $\mathcal{T}$ is also finite since there are only finitely many possible non-isomorphic labeled local neighborhoods.

It turns out that in the case of paths, cycles, and rooted regular trees, the LOCAL, SLOCAL, dynamic-LOCAL, and online-LOCAL models are all approximately equally expressive for LCL problems. In particular, all classification and decidability results related to LCLs in paths, cycles, and rooted regular trees in the LOCAL model [4, 7, 18] directly apply also in the online-LOCAL model, the SLOCAL model, and both versions of the dynamic-LOCAL model.

We show first that the LOCAL and online-LOCAL models are equivalent in the case of paths and cycles, even when the LCL problems can have inputs. We then continue to prove that the models are equivalent also in the more general case of LCL problems rooted regular trees, but in this case we do not consider the possibility of having input labels.

Formally, we prove the following theorem for cycles and paths:

▶ **Theorem 4.** *Let Π be an LCL problem in paths or cycles (possibly with inputs). If the locality of Π is T in the online-LOCAL model, then its locality is $O(T + \log^* n)$ in the LOCAL model.*

For the case of rooted trees, we prove the following two theorems:

▶ **Theorem 5.** *Let Π be an LCL problem in rooted regular trees (without inputs). Problem Π has locality $n^{\Omega(1)}$ in the LOCAL model if and only if it has locality $n^{\Omega(1)}$ in the online-LOCAL model.*

▶ **Theorem 6.** *Let Π be an LCL problem in rooted regular trees (without inputs). Problem Π has locality $\Omega(\log n)$ in the LOCAL model if and only if it has locality $\Omega(\log n)$ in the online-LOCAL model.*

These two theorems show that all LCL problems in rooted regular trees belong to one of the known complexity classes $O(\log^* n)$, $\Theta(\log n)$ and $n^{\Omega(1)}$ in all of the models we study. In what follows, we introduce the high-level ideas of the proofs of these theorems. For full proofs, we refer the reader to the full version.

5.1 Cycles and paths

We prove Theorem 4 by first showing that any LCL problem in cycles and paths has either locality $O(1)$ or $\Omega(n)$ in the online-LOCAL model. Next, we show that if a problem is solvable with locality $O(1)$ in the online-LOCAL model, then it is also solvable in locality $O(\log^* n)$ in the LOCAL model. These steps are described by the following two lemmas:

▶ **Lemma 7.** *Let Π be an LCL problem in paths or cycles (possibly with inputs), and let $\mathcal{A}$ be an online-LOCAL algorithm solving Π with locality $o(n)$. Then, there exists an online-LOCAL algorithm $\mathcal{A}'$ solving Π with locality $O(1)$.*

The high-level idea of the proof of Lemma 7 is to construct a large virtual graph P' such that when the original algorithm runs on the virtual graph P', the labeling produced by the algorithm is locally compatible with the labeling in the original graph P. We ensure this by applying a pumping-lemma-style argument on the LCL problem. The proof uses similar ideas as the ones presented by Chang and Pettie [17].

▶ **Lemma 8.** *Let Π be an LCL problem in paths or cycles (possibly with inputs), and let $\mathcal{A}$ be an online-LOCAL algorithm solving Π with locality $O(1)$. Then, there exists a LOCAL algorithm $\mathcal{A}'$ solving Π with locality $O(\log^* n)$.*

The high-level idea of the proof of Lemma 8 is to use the constant locality online-LOCAL algorithm to construct a canonical output labeling for each possible neighborhood of input labels. The fast LOCAL algorithm can then use these canonical labelings in disjoint neighborhoods of the real graph, and the construction of the canonical labelings ensures that the labeling also extends to the path segments between these neighborhoods.

The full proofs of these lemmas can be found in the full version of this paper. In order to prove Theorem 4, it is sufficient to combine these lemmas with the fact that the possible localities on paths and cycles in the LOCAL model are $O(1)$, $\Theta(\log^* n)$ and $\Theta(n)$ [18].

5.2 Rooted regular trees

We prove the equivalence of the LOCAL and the online-LOCAL models for LCL problems in rooted regular trees in two parts. We start out with Theorem 5 and show that if an LCL problem requires locality $n^{\Omega(1)}$ in the LOCAL model, then for every locality-$n^{o(1)}$ online-LOCAL algorithm we can construct an input instance which the algorithm must fail to solve. To prove Theorem 6, we show that a locality-$o(\log n)$ online-LOCAL algorithm for solving an LCL problem implies that there exists a locality-$O(\log^* n)$ LOCAL algorithm for solving that same problem. In the following, we outline the proofs of both theorems; the full proofs can be found in the full version of the paper. Before considering the full proof, we advise the reader to look at the example in the full version of this paper, where we show that the 2.5-coloring problem requires locality $\Omega(\sqrt{n})$ in the online-LOCAL model.

Proof outline of Theorem 5. Our proof is based on the fact that any LCL problem requiring locality $n^{\Omega(1)}$ in the LOCAL model has a specific structure. In particular, the problem can be decomposed into a sequence of *path-inflexible* labels and the corresponding sequence of more and more restricted problems [7]. Informally, a label is path-inflexible if two nodes having that label can exist only at specific distances apart from each other. For example, when 2-coloring a graph, two nodes having label 1 can exist only at even distances from each other. The problems in the path-inflexible decomposition are formed by removing the path-inflexible labels from the previous problem in the sequence until an empty problem is reached.

This decomposition of the problem into restricted problems with path-inflexible labels allows us to construct an input graph for any locality-$n^{o(1)}$ online-LOCAL algorithm. In particular, we force the algorithm to commit labels in disjoint fragments of the graph. Any label that the algorithm uses must be a path-inflexible label in some problem of the sequence of restricted problems. By combining two fragments containing labels that are path-inflexible in the same problem, we can ensure that the algorithm cannot solve that problem in the resulting graph. Hence the algorithm must use a label from a problem earlier in the sequence. At some point, the algorithm must use labels that are path-inflexible in the original problem. At that point, we can combine two fragments having path-inflexible labels in the original problem in such a way that no valid labeling for the original problem exists, and hence the algorithm must fail to solve the problem on the resulting graph.

Proof outline of Theorem 6. Here, we show that a locality-$o(\log n)$ online-LOCAL algorithm solving an LCL problem implies that there exists a *certificate for* $O(\log^* n)$ *solvability* for that problem. It is known that the existence of such a certificate for a problem implies that there exists a locality-$O(\log^* n)$ LOCAL algorithm for solving the problem [7].

Informally, the certificate for $O(\log^* n)$ solvability for LCL problem Π with label set Γ and arity δ consists of a subset $\Gamma_{\mathcal{T}} = \{\gamma_1, \ldots, \gamma_t\}$ of labels Γ, and two sequences of correctly labeled complete δ-ary trees $\mathcal{T}^1$ and $\mathcal{T}^2$. The leaves of each tree in the sequence $\mathcal{T}^1$ (resp. $\mathcal{T}^2$) are labeled in the same way using only labels from set $\Gamma_{\mathcal{T}}$. For every label of set $\Gamma_{\mathcal{T}}$, there exists a tree in both of the sequences having a root labeled with that label.

We can use the online-LOCAL algorithm to construct such a certificate. We do this by constructing exponentially many deep complete δ-ary trees and using the algorithm to label nodes in the middle of those trees. We then glue these trees together in various ways. When the trees are glued together, we use the online-LOCAL algorithm to label the rest of the nodes to form one tree of the sequence. We repeat this procedure until all trees of both sequences have been constructed.

References

1 Susanne Albers and Sebastian Schraink. Tight bounds for online coloring of basic graph classes. *Algorithmica*, 83(1):337–360, 2021. doi:10.1007/s00453-020-00759-7.

2 Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. In *Proc. 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2012)*, pages 1132–1139. SIAM, 2012. doi:10.1137/1.9781611973099.89.

3 Sepehr Assadi, Krzysztof Onak, Baruch Schieber, and Shay Solomon. Fully dynamic maximal independent set with sublinear update time. In *Proc. 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2018)*, pages 815–826, 2018. doi:10.1145/3188745.3188922.

4 Alkida Balliu, Sebastian Brandt, Yi-Jun Chang, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. The distributed complexity of locally checkable problems on paths is decidable. In *Proc. 38th ACM Symposium on Principles of Distributed Computing (PODC 2019)*, pages 262–271. ACM Press, 2019. doi:10.1145/3293611.3331606.

5 Alkida Balliu, Sebastian Brandt, Yuval Efron, Juho Hirvonen, Yannic Maus, Dennis Olivetti, and Jukka Suomela. Classification of distributed binary labeling problems. In *Proc. 34th International Symposium on Distributed Computing (DISC 2020)*, pages 17:1–17:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.DISC.2020.17.

6 Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. Lower bounds for maximal matchings and maximal independent sets. *Journal of the ACM*, 68(5), 2021. doi:10.1145/3461458.

7 Alkida Balliu, Sebastian Brandt, Dennis Olivetti, Jan Studený, Jukka Suomela, and Aleksandr Tereshchenko. Locally checkable problems in rooted trees. In *Proc. 40th ACM Symposium on Principles of Distributed Computing (PODC 2021)*, pages 263–272. ACM Press, 2021. doi:10.1145/3465084.3467934.

8 Alkida Balliu, Juho Hirvonen, Dennis Olivetti, and Jukka Suomela. Hardness of minimal symmetry breaking in distributed computing. In *Proc. 38th ACM Symposium on Principles of Distributed Computing (PODC 2019)*, pages 369–378. ACM Press, 2019. doi:10.1145/3293611.3331605.

9 Leonid Barenboim and Tzalik Maimon. Fully dynamic graph algorithms inspired by distributed computing: Deterministic maximal matching and edge coloring in sublinear update-time. *ACM Journal of Experimental Algorithmics*, 24, 2019.

10 Dwight R. Bean. Effective coloration. *The Journal of Symbolic Logic*, 41(2):469–480, 1976. doi:10.2307/2272247.

11 Sayan Bhattacharya, Deeparnab Chakrabarty, Monika Henzinger, and Danupon Nanongkai. Dynamic algorithms for graph coloring. In *Proc. 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2018)*, pages 1–20. SIAM, 2018. doi:10.1137/1.9781611975031.1.

12 Maria Paola Bianchi, Hans-Joachim Böckenhauer, Juraj Hromkovič, and Lucia Keller. Online coloring of bipartite graphs with and without advice. In *Computing and Combinatorics*, pages 519–530, 2012. doi:10.1007/978-3-642-32241-9_44.

13 Sebastian Brandt, Juho Hirvonen, Janne H. Korhonen, Tuomo Lempiäinen, Patric R. J. Östergård, Christopher Purcell, Joel Rybicki, Jukka Suomela, and Przemysław Uznański. LCL problems on grids. In *Proc. 36th ACM Symposium on Principles of Distributed Computing (PODC 2017)*, pages 101–110. ACM Press, 2017. doi:10.1145/3087801.3087833.

14 Elisabet Burjons, Juraj Hromkovič, Xavier Muñoz, and Walter Unger. Online graph coloring with advice and randomized adversary. In *Proc. 42nd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2016)*, pages 229–240. Springer, 2016. doi:10.1007/978-3-662-49192-8_19.

15 Yi-Jun Chang. The complexity landscape of distributed locally checkable problems on trees. In *Proc. 34th International Symposium on Distributed Computing (DISC 2020)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.DISC.2020.18.

16 Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. An Exponential Separation between Randomized and Deterministic Complexity in the LOCAL Model. In *Proc. 57th IEEE Symposium on Foundations of Computer Science (FOCS 2016)*, pages 615–624. IEEE, 2016. doi:10.1109/FOCS.2016.72.

17 Yi-Jun Chang and Seth Pettie. A Time Hierarchy Theorem for the LOCAL Model. *SIAM Journal on Computing*, 48(1):33–69, 2019. doi:10.1137/17M1157957.

18 Yi-Jun Chang, Jan Studený, and Jukka Suomela. Distributed graph problems through an automata-theoretic lens. In *Proc. 28th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2021)*, pages 31–49. Springer, 2021. doi:10.1007/978-3-030-79527-6_3.

19 Richard Cole and Uzi Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Information and Control*, 70(1):32–53, 1986. doi:10.1016/S0019-9958(86)80023-7.

20 Stefan Dobrev, Rastislav Královič, and Richard Královič. Independent set with advice: The impact of graph knowledge. In *Proc. 10th Workshop on Approximation and Online Algorithms (WAOA 2012)*. Springer, 2013. doi:10.1007/978-3-642-38016-7_2.

21 Yuhao Du and Hengjie Zhang. Improved algorithms for fully dynamic maximal independent set, 2018. arXiv:1804.08908.

22 Yuval Emek, Pierre Fraigniaud, Amos Korman, and Adi Rosén. Online computation with advice. In *Proc. 36th edition of the International Colloquium on Automata, Languages and Programming (ICALP 2009)*, pages 427–438. Springer, 2009. doi:10.1007/978-3-642-02927-1_36.

23 Yuval Emek, Shay Kutten, and Roger Wattenhofer. Online matching: Haste makes waste! In *Proc. 48th Annual ACM Symposium on Theory of Computing (STOC 2016)*, pages 333–344, 2016. doi:10.1145/2897518.2897557.

24 Guy Even, Moti Medina, and Dana Ron. Deterministic stateless centralized local algorithms for bounded degree graphs. In *Proc. 22nd European Symposium on Algorithms (ESA 2014)*, pages 394–405. Springer, 2014. doi:10.1007/978-3-662-44777-2_33.

25 Mohsen Ghaffari, David G. Harris, and Fabian Kuhn. On derandomizing local distributed algorithms. In *Proc. 59th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2018)*, pages 662–673. IEEE, 2018. doi:10.1109/FOCS.2018.00069.

26 Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. On the complexity of local distributed graph problems. In *Proc. 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2017)*, pages 784–797. ACM Press, 2017. doi:10.1145/3055399.3055471.

27 Mika Göös, Juho Hirvonen, Reut Levi, Moti Medina, and Jukka Suomela. Non-local probes do not help with many graph problems. In *Proc. 30th International Symposium on Distributed Computing (DISC 2016)*. Springer, 2016. doi:10.1007/978-3-662-53426-7_15.

28 Manoj Gupta and Shahbaz Khan. Simple dynamic algorithms for maximal independent set and other problems, 2018. arXiv:1804.01823.

29 András Gyárfás and Jenő Lehel. On-line and first fit colorings of graphs. *Journal of Graph Theory*, 12(2):217–227, 1988. doi:h10.1002/jgt.3190120212.

30 Magnús M. Halldórsson, Kazuo Iwama, Shuichi Miyazaki, and Shiro Taketomi. Online independent sets. *Theoretical Computer Science*, 289(2):953–962, 2002. doi:10.1016/S0304-3975(01)00411-X.

31 Magnús M. Halldórsson. Parallel and on-line graph coloring. *Journal of Algorithms*, 23(2):265–280, 1997. doi:10.1006/jagm.1996.0836.

32 Magnús M. Halldórsson. Online coloring known graphs. *Electronic Journal of Combinatorics*, 7, 2000. doi:10.37236/1485.

33 Magnús M. Halldórsson and Mario Szegedy. Lower bounds for on-line graph coloring. *Theoretical Computer Science*, 130(1):163–174, 1994. doi:10.1016/0304-3975(94)90157-0.

34 Zoran Ivković and Errol L. Lloyd. Fully dynamic maintenance of vertex cover. In *Proc. 19th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 1993)*. Springer, 1994.

35 Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proc. 22nd Annual ACM Symposium on Theory of Computing (STOC 1990)*, pages 352–358, 1990. doi:10.1145/100216.100262.

36 Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: Lower and upper bounds. *Journal of the ACM*, 63(2):1–44, 2016. doi:10.1145/2742012.

37 Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992. doi:10.1137/0221015.

38 László Lovász, Michael Saks, and W.T. Trotter. An on-line graph coloring algorithm with sublinear performance ratio. *Discrete Mathematics*, 75(1):319–325, 1989. doi:10.1016/0012-365X(89)90096-4.

39 Yishay Mansour, Aviad Rubinstein, Shai Vardi, and Ning Xie. Converting online algorithms to local computation algorithms. In *Proc. 39th International Colloquium on Automata, Languages and Programming (ICALP 2012)*, pages 653–664. Springer, 2012. doi:10.1007/978-3-642-31594-7_55.

40 Yishay Mansour and Shai Vardi. A local computation approximation scheme to maximum matching. In *Proc. 16th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2013) and 17th International Workshop on Randomization and Computation (RANDOM 2013)*, pages 260–273. Springer, 2013. doi:10.1007/978-3-642-40328-6_19.

41 Darya Melnyk, Jukka Suomela, and Neven Villani. Mending partial solutions with few changes. In *Proc. 25th International Conference on Principles of Distributed Systems (OPODIS 2022)*, Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

42 Moni Naor and Larry Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995. doi:10.1137/S0097539793254571.

43 Ofer Neiman and Shay Solomon. Simple deterministic algorithms for fully dynamic maximal matching. *ACM Transactions on Algorithms*, 12(1), 2015. doi:10.1145/2700206.

44 David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, 2000. doi:10.1137/1.9780898719772.

45 Will Rosenbaum and Jukka Suomela. Seeing far vs. seeing wide: volume complexity of local graph problems. In *Proc. 39th ACM Symposium on Principles of Distributed Computing (PODC 2020)*, pages 89–98. ACM Press, 2020. doi:10.1145/3382734.3405721.

46 Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. In *Proc. 2nd Symposium on Innovations in Computer Science (ICS 2011)*, pages 223–238, 2011.

47 Sundar Vishwanathan. Randomized online graph coloring. *Journal of Algorithms*, 13(4):657–669, 1992. doi:10.1016/0196-6774(92)90061-G.

An Efficient Algorithm for All-Pairs Bounded Edge Connectivity

Shyan Akmal
MIT EECS and CSAIL, Cambridge, MA, USA

Ce Jin
MIT EECS and CSAIL, Cambridge, MA, USA

Abstract

Our work concerns algorithms for a variant of Maximum Flow in unweighted graphs. In the All-Pairs Connectivity (APC) problem, we are given a graph G on n vertices and m edges, and are tasked with computing the maximum number of edge-disjoint paths from s to t (equivalently, the size of a minimum (s,t)-cut) in G, for all pairs of vertices (s,t). Over undirected graphs, it is known that APC can be solved in essentially optimal $n^{2+o(1)}$ time. In contrast, the true time complexity of APC over directed graphs remains open: this problem can be solved in $\tilde{O}(m^\omega)$ time, where $\omega \in [2, 2.373)$ is the exponent of matrix multiplication, but no matching conditional lower bound is known.

Following [Abboud et al., ICALP 2019], we study a bounded version of APC called the k-Bounded All Pairs Connectivity (k-APC) problem. In this variant of APC, we are given an integer k in addition to the graph G, and are now tasked with reporting the size of a minimum (s,t)-cut only for pairs (s,t) of vertices with min-cut value less than k (if the minimum (s,t)-cut has size at least k, we can just report it is "large" instead of computing the exact value).

Our main result is an $\tilde{O}((kn)^\omega)$ time algorithm solving k-APC in directed graphs. This is the first algorithm which solves k-APC faster than simply solving the more general APC problem exactly, for all $k \geq 3$. This runtime is $\tilde{O}(n^\omega)$ for all $k \leq \text{poly}(\log n)$, which essentially matches the optimal runtime for the $k = 1$ case of k-APC, under popular conjectures from fine-grained complexity. Previously, this runtime was only achieved for general directed graphs when $k \leq 2$ [Georgiadis et al., ICALP 2017]. Our result employs the same algebraic framework used in previous work, introduced by [Cheung, Lau, and Leung, FOCS 2011]. A direct implementation of this framework involves inverting a large random matrix. Our new algorithm is based off the insight that for solving k-APC, it suffices to invert a low-rank random matrix instead of a generic random matrix.

We also obtain a new algorithm for a variant of k-APC, the k-Bounded All-Pairs Vertex Connectivity (k-APVC) problem, where for every pair of vertices (s,t), we are now tasked with reporting the maximum number of internally vertex-disjoint (rather than edge-disjoint) paths from s to t if this number is less than k, and otherwise reporting that this number is at least k.

Our second result is an $\tilde{O}(k^2 n^\omega)$ time algorithm solving k-APVC in directed graphs. Previous work showed how to solve an easier version of the k-APVC problem (where answers only need to be returned for pairs of vertices (s,t) which are not edges in the graph) in $\tilde{O}((kn)^\omega)$ time [Abboud et al, ICALP 2019]. In comparison, our algorithm solves the full k-APVC problem, and is faster if $\omega > 2$.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases maximum flow, all-pairs, connectivity, matrix rank

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.11

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2305.02132 [3]

Funding *Shyan Akmal*: Supported in part by NSF grants CCF-2129139 and CCF-2127597.
Ce Jin: Partially supported by NSF Grant CCF-2129139 and a Siebel Scholarship.

Acknowledgements The first author thanks Virginia Vassilevska Williams for insightful discussions on algorithms for computing matrix rank.

1 Introduction

Computing maximum flows is a classic problem which has been extensively studied in graph theory and computer science. In unweighted graphs, this task specializes to computing connectivities, an interesting computational problem in its own right. Given a graph G on n vertices and m edges, for any vertices s and t in G, the *connectivity* $\lambda(s,t)$ from s to t is defined to be the maximum number of edge-disjoint paths[1] from s to t. Since maximum flow can be computed in almost-linear time, we can compute $\lambda(s,t)$ for any given vertices s and t in $m^{1+o(1)}$ time [5].

What if instead of merely returning the value of a single connectivity, our goal is to compute all connectivities in the graph? This brings us to the All-Pairs Connectivity (APC) problem: in this problem, we are given a graph G as above, and are tasked with computing $\lambda(s,t)$ for all pairs of vertices (s,t) in G. In undirected graphs, APC can be solved in $n^{2+o(1)}$ time [2], so that this "all-pairs" problem is essentially no harder than outputting a single connectivity in dense graphs.

In directed graphs, APC appears to be much harder, with various conditional lower bounds (discussed in Section 1.2) suggesting it is unlikely this problem can be solved in quadratic time. Naively computing the connectivity separately for each pair yields an $n^2m^{1+o(1)}$ time algorithm for this problem. Using the flow vector framework (discussed in Section 3), it is possible to solve APC in directed graphs in $\tilde{O}(m^\omega)$ time[2] [7], where ω is the exponent of matrix multiplication. Known algorithms imply that $\omega < 2.37286$ [4], so the $\tilde{O}(m^\omega)$ time algorithm is faster than the naive algorithm whenever the input graph is not too dense.

Our work focuses on a bounded version of the APC problem, which we formally state as the k-Bounded All-Pairs Connectivity (k-APC) problem: in this problem, we are given a *directed* graph G as above, and are tasked with computing $\min(k, \lambda(s,t))$ for all pairs of vertices (s,t) in G. Intuitively, this is a relaxation of the APC problem, where our goal is to compute the exact values of $\lambda(s,t)$ only for pairs (s,t) with small connectivity. For all other pairs, it suffices to report that the connectivity is large, where k is our threshold for distinguishing between small and large connectivity values.

When $k = 1$, the k-APC problem is equivalent to computing the transitive closure of the input graph (in this problem, for each pair of vertices (s,t), we are tasked with determining if G contains a path from s to t), which can be done in $\tilde{O}(n^\omega)$ time [8]. Similarly, for the special case of $k = 2$, it is known that k-APC can be solved in $\tilde{O}(n^\omega)$ time, by a divide-and-conquer algorithm employing a cleverly tailored matrix product [10]. As we discuss in Section 1.2, there is evidence that these runtimes for k-APC when $k \leq 2$ are essentially optimal.

Already for $k = 3$ however, it is open whether k-APC can be solved faster than computing the *exact values* of $\lambda(s,t)$ for all pairs (s,t) of vertices! Roughly speaking, this is because the known $\tilde{O}(m^\omega)$ time algorithm for APC involves encoding the connectivity information in the inverse of an $m \times m$ matrix, and inverting an $m \times m$ matrix takes $O(m^\omega)$ time in general. This encoding step appears to be necessary for k-APC as well. For $k = 2$, clever combinatorial observations about the structure of strongly connected graphs allow one to skip this computation, but for $k \geq 3$ it is not clear at all from previous work how to avoid this bottleneck. Moreover, it is consistent with existing hardness results that k-APC could be solved in $O(n^\omega)$ time for any constant k.

---

[1] By Menger's theorem, $\lambda(s,t)$ is also equal to the minimum number of edges that must be deleted from the graph G to produce a graph with no s to t path.

[2] Given a function f, we write $\tilde{O}(f)$ to denote $f \cdot \text{poly}(\log f)$.

▶ **Open Problem 1.** Can k-APC be solved in faster than $\tilde{O}(m^\omega)$ time for $k = 3$?

Due to this lack of knowledge about the complexity of k-APC, researchers have also studied easier versions of this problem. Given vertices s and t in the graph G, we define the *vertex connectivity* $\nu(s,t)$ from s to t to be the maximum number of internally vertex-disjoint paths from s to t. We can consider vertex connectivity analogues of the APC and k-APC problems. In the All-Pairs Vertex Connectivity (APVC) problem, we are given a graph G on n vertices and m edges, and are tasked with computing the value of $\nu(s,t)$ for all pairs of vertices (s,t) in G. In the k-Bounded All-Pairs Vertex Connectivity (k-APVC) problem, we are given the same input G as above, but are now tasked with only computing $\min(k, \nu(s,t))$ for all pairs of vertices (s,t) in G.

The k-APVC problem does not face the $O(m^\omega)$ barrier which existing algorithmic techniques for k-APC seem to encounter, intuitively because it is possible to encode all the vertex-connectivity information of a graph in the inverse of an $n \times n$ matrix instead of an $m \times m$ matrix. As a consequence, [1] was able to present an $\tilde{O}((kn)^\omega)$ time algorithm which computes $\min(k, \nu(s,t))$ for all pairs of vertices (s,t) such that (s,t) is not an edge. Given this result, it is natural to ask whether the more general k-APVC and k-APC problems can also be solved in this same running time.

▶ **Open Problem 2.** Can k-APVC be solved in $\tilde{O}((kn)^\omega)$ time?

▶ **Open Problem 3.** Can k-APC be solved in $\tilde{O}((kn)^\omega)$ time?

1.1 Our Contribution

We resolve all three open problems raised in the previous section.

First, we present a faster algorithm for k-APC, whose time complexity matches the runtime given by previous work for solving an easier version of k-APVC.

▶ **Theorem 4.** *For any positive integer k, k-APC can be solved in $\tilde{O}((kn)^\omega)$ time.*

This is the first algorithm which solves k-APC faster than simply solving APC exactly using the $\tilde{O}(m^\omega)$ time algorithm of [7], for all constant $k \geq 3$.

Second, we present an algorithm for k-APVC, which is faster than the $\tilde{O}((kn)^\omega)$ time algorithm from [1] (which only solves a restricted version of k-APVC) if $\omega > 2$.

▶ **Theorem 5.** *For any positive integer k, k-APVC can be solved in $\tilde{O}(k^2 n^\omega)$ time.*

1.2 Comparison to Previous Results

Conditional Lower Bounds

The field of fine-grained complexity contains many popular conjectures (which hypothesize lower bounds on the complexity of certain computational tasks) which are used as the basis of conditional hardness results for problems in computer science. In this section, we review known hardness results for APC and its variants. The definitions of the problems and conjectures used in this section can be found in...

Assuming that Boolean Matrix Multiplication (BMM) requires $n^{\omega - o(1)}$ time, it is known that k-APC and k-APVC require $n^{\omega - o(1)}$ time to solve, even for $k = 1$ [8]. In particular, this hypothesis implies our algorithms for k-APC and k-APVC are optimal for constant k.

Assuming the Strong Exponential Time Hypothesis (SETH), previous work shows that APC requires $(mn)^{1-o(1)}$ time [12, Theorem 1.8], APVC requires $m^{3/2-o(1)}$ time [14, Theorem 1.7], and k-APC requires $\left(kn^2\right)^{1-o(1)}$ time [12, Theorem 4.3].

Let $\omega(1,2,1)$ be the smallest real number[3] such that we can compute the product of an $n \times n^2$ matrix and $n^2 \times n$ matrix in $n^{\omega(1,2,1)+o(1)}$ time. Assuming the 4-Clique Conjecture, the k-APVC problem over directed graphs (and thus the k-APC problem as well) requires $\left(k^2 n^{\omega(1,2,1)-2}\right)^{1-o(1)}$ time [1]. This conjecture also implies that solving APVC even in undirected graphs requires $n^{\omega(1,2,1)-o(1)}$ time [11].

Algorithms for Related Problems

As mentioned previously, no nontrivial algorithms for k-APC over general directed graphs were known for $k \geq 3$, prior to our work. However, faster algorithms were already known for k-APC over directed acyclic graphs (DAGs). In particular, [1] presented two algorithms to solve k-APC in DAGs, running in $2^{O(k^2)}mn$ time and $(k \log n)^{4^k+o(k)} n^\omega$ time respectively.

In comparison, our algorithm from Theorem 4 solves k-APC in *general* directed graphs, is faster than the former algorithm whenever $m \geq n^{\omega-1}$ or $k \geq \omega(\sqrt{\log n})$ (for example), is always faster than the latter algorithm, and is significantly simpler from a technical perspective than these earlier arguments. However, these algorithms for k-APC on DAGs also return cuts witnessing the connectivity values, while our algorithm does not.

In the special case of undirected graphs, APVC can be solved in $m^{2+o(1)}$ time [14, Theorem 1.8], which is faster than the aforementioned $\tilde{O}(m^\omega)$ time algorithm if $\omega > 2$. Over undirected graphs, k-APVC can be solved in $k^3 m^{1+o(1)} + n^2 \operatorname{poly}(\log n)$ time. In comparison, our algorithm from Theorem 5 can handle k-APVC in both undirected *and* directed graphs, and is faster for large enough values of k in dense graphs.

In directed planar graphs with maximum degree d, [7, Theorem 1.5] proves that APC can be solved in $O\left(d^{\omega-2} n^{\omega/2+1}\right)$ time.

In [15], the authors consider a symmetric variant of k-APC. Here, the input is a directed graph G on n vertices and m edges, and the goal is to compute for all pairs of vertices (s,t), the value of $\min(k, \lambda(s,t), \lambda(t,s))$. This easier problem can be solved in $O(kmn)$ time [15].

1.3 Organization

The rest of this paper is devoted to proving Theorems 4 and 5. In Section 2 we introduce notation, some useful definitions, and results on matrix computation which will be useful in proving correctness of our algorithms. In Section 3 we provide an intuitive overview of our algorithms for k-APC and k-APVC. In Section 4 we describe a framework of "flow vectors" for capturing connectivity values, and in Section 5 use this framework to prove Theorem 4. In Section 6 we present helpful results about vertex-connectivity, and in Section 7 use these results to prove Theorem 5.

2 Preliminaries

Graph Assumptions

Throughout, we let G denote a directed graph on n vertices and m edges. Without loss of generality, we assume that the underlying undirected graph of G is connected, i.e., G is weakly connected (since, if not, we could simply run our algorithms separately on each weakly connected component of G), so we have $m \geq n-1$. We assume G has no self-loops, since these do not affect the connectivity or vertex-connectivity values between distinct vertices.

[3] Known fast matrix multiplication algorithms imply that $\omega(1,2,1) < 3.25669$ [9, Table 2].

In Sections 4 and 5 we focus on the k-APC problem, and so allow G to have parallel edges between vertices (i.e., G can be a multigraph). We assume however, without loss of generality, that for any distinct vertices s and t, there are at most k edges from s to t (since if there were more than k parallel edges from s to t, we could delete some and bring the count of parallel edges down to k without changing the value of $\min(k, \lambda(s,t))$). In Sections 6 and 7 we focus on the k-APVC problem, and so assume that G is a simple graph with no parallel edges, since parallel edges from u to v cannot affect the value of a vertex connectivity $\nu(s,t)$, unless $u = s$ and $v = t$, in which case the value of $\nu(s,t)$ is simply increased by the number of additional parallel edges from s to t.

Graph Terminology and Notation

Given an edge e from u to v in G, we write $e = (u, v)$. We call u the tail of e and v the head of e. Vertices which are tails of edges entering a vertex v are called *in-neighbors* of v. Similarly, vertices which are heads of edges exiting v are called *out-neighbors* of v. Given a vertex u in G, we let $E_{\text{in}}(u)$ denote the set of edges entering u, and $E_{\text{out}}(u)$ denote the set of edges exiting u. Similarly, $V_{\text{in}}(u)$ denotes the set of in-neighbors of u, and $V_{\text{out}}(u)$ denotes the set of out-neighbors of u. Furthermore, we define $V_{\text{in}}[u] = V_{\text{in}}(u) \cup \{u\}$ and $V_{\text{out}}[u] = V_{\text{out}}(u) \cup \{u\}$. Finally, let $\deg_{\text{in}}(u) = |E_{\text{in}}(u)|$ and $\deg_{\text{out}}(u) = |E_{\text{out}}(u)|$ denote the indegree and outdegree of u respectively.

Given vertices s and t, an (s,t)-cut is a set C of edges, such that deleting the edges in C produces a graph with no s to t path. By Menger's theorem, the size of a minimum (s,t)-cut is equal to the connectivity $\lambda(s,t)$ from s to t. Similarly, an (s,t)-vertex cut is a set of C' of vertices with $s, t \notin C'$, such that deleting C' produces a graph with no s to t path. Clearly, a vertex cut exists if and only if (s,t) is not an edge. When (s,t) is not an edge, Menger's theorem implies that the size of a minimum (s,t)-vertex cut is equal to the vertex connectivity $\nu(s,t)$ from s to t.

Matrix Notation

Let A be a matrix. For indices i and j, we let $A[i,j]$ denote the (i,j) entry of A. More generally, if S is a set of row indices and T a set of column indices, we let $A[S,T]$ denote the submatrix of A restricted to rows from S and columns from T. Similarly, $A[S,*]$ denotes A restricted to rows from S, and $A[*,T]$ denotes A restricted to columns from T. We let $A^\top$ denote the transpose of A. If A is a square matrix, then we let $\text{adj}(A)$ denote the adjugate of A. If A is invertible, we let A^{-1} denote its inverse. If a theorem, lemma, or proposition statement refers to A^{-1}, it is generally asserting that A^{-1} exists (or if A is a random matrix, asserting that A^{-1} exists with some probability) as part of the statement. We let I denote the identity matrix (the dimensions of this matrix will always be clear from context). Given a vector $\vec{v}$, for any index i we let $\vec{v}[i]$ denote the i^{th} entry in $\vec{v}$. We let $\vec{0}$ denote the zero vector (the dimensions of this vector will always be clear from context). Given a positive integer k, we let $[k] = \{1, \ldots, k\}$ denote the set of the first k positive integers.

Matrix and Polynomial Computation

Given a prime p, we let $\mathbb{F}_p$ denote the finite field on p elements. Arithmetic operations over elements of $\mathbb{F}_p$ can be performed in $\tilde{O}(\log p)$ time.

We now recall some well-known results about computation with matrices and polynomials, which will be useful for our algorithms.

▶ **Proposition 6.** *Let A be an $a \times b$ matrix, and B be a $b \times a$ matrix. If $(I - BA)$ is invertible, then the matrix $(I - AB)$ is also invertible, with inverse*

$$(I - AB)^{-1} = I + A(I - BA)^{-1}B.$$

Proof. It suffices to verify that the product of $(I - AB)$ with the right hand side of the above equation yields the identity matrix. Indeed, we have

$$\begin{aligned}
&(I - AB)\left(I + A(I - BA)^{-1}B\right) \\
&= I + A(I - BA)^{-1}B - AB - ABA(I - BA)^{-1}B \\
&= I + A(I - BA)^{-1}B - AB - A\big(I - (I - BA)\big)(I - BA)^{-1}B \\
&= I + A(I - BA)^{-1}B - AB - A(I - BA)^{-1}B + AB,
\end{aligned}$$

which simplifies to I, as desired. ◀

▶ **Proposition 7.** *Let A be an $a \times a$ matrix over $\mathbb{F}_p$. We can compute the inverse A^{-1} (if it exists) in $O(a^\omega)$ field operations.*

▶ **Proposition 8** ([6, Theorem 1.1])**.** *Let A be an $a \times b$ matrix over $\mathbb{F}_p$. Then for any positive integer k, we can compute $\min(k, \operatorname{rank} A)$ in $O(ab + k^\omega)$ field operations.*

▶ **Proposition 9** (Schwartz-Zippel Lemma [13, Theorem 7.2])**.** *Let $f \in \mathbb{F}_p[x_1, \ldots, x_r]$ be a degree d, nonzero polynomial. Let $\vec{a}$ be a uniform random point in $\mathbb{F}_p^r$. Then $f(\vec{a})$ is nonzero with probability at least $1 - d/p$.*

3 Proof Overview

3.1 Flow Vector Encodings

Previous algorithms for APC [7] and its variants work in two steps:

Step 1: Encode
In this step, we prepare a matrix M which implicitly encodes the connectivity information of the input graph.

Step 2: Decode
In this step, we iterate over all pairs (s, t) of vertices in the graph, and for each pair run a small computation on a submatrix of M to compute the desired connectivity value.

The construction in the **encode** step is based off the framework of *flow vectors*, introduced in [7] as a generalization of classical techniques from network-coding. We give a high-level overview of how this method has been previously applied in the APC problem.[4]

Given the input graph G, we fix a source vertex s. Let $d = \deg_{\text{out}}(s)$, and let $\mathbb{F}$ be some ground field.[5] Our end goal is to assign to each edge e in the graph a special vector $\vec{e} \in \mathbb{F}^d$ which we call a *flow vector*.

First, for each edge $e \in E_{\text{out}}(s)$, we introduce a d-dimensional vector $\vec{v_e}$. These vectors intuitively correspond to some starting flow that is pumping out of s. It is important that these vectors are linearly independent (and previous applications have always picked these vectors to be distinct d-dimensional unit vectors). We then push this flow through the rest of the graph, by having each edge get assigned a vector which is a random linear combination

[4] For the APVC problem we employ a different, but analogous, framework described in Section 3.3.
[5] In our applications, we will pick $\mathbb{F}$ to be a finite field of size $\text{poly}(m)$.

of the flow vectors assigned to the edges entering its tail. That is, given an edge $e = (u, v)$ with $u \neq s$, the final flow vector $\vec{e}$ will be a random linear combination of the flow vectors for the edges entering u. If instead the edge $e = (s, v)$ is in $E_{\text{out}}(s)$, the final flow vector $\vec{e}$ will be a random linear combination of the flow vectors for the edges entering s, added to the initial flow $\vec{v}_e$.

The point of this random linear combination is to (with high probability) preserve linear independence. In this setup, for any vertex v and integer ℓ, if some subset of ℓ flow vectors assigned to edges in $E_{\text{in}}(v)$ is independent, then we expect that every subset of at most ℓ flow vectors assigned to edges in $E_{\text{out}}(v)$ is also independent. This sort of behavior turns out to generalize to preserving linear independence of flow vectors across cuts, which implies that (with high probability) for any vertex t, the rank of the flow vectors assigned to edges in $E_{\text{in}}(t)$ equals $\lambda(s, t)$.

Intuitively, this is because the flow vectors assigned to edges in $E_{\text{in}}(t)$ will be a linear combination of the $\lambda(s, t)$ flow vectors assigned to edges in a minimum (s, t)-cut, and the flow vectors assigned to edges in this cut should be independent.

Collecting all the flow vectors as column vectors in a matrix allows us to produce a single matrix M_s, such that computing the rank of $M_s[*, E_{\text{in}}(t)]$ yields the desired connectivity value $\lambda(s, t)$ (computing these ranks constitutes the **decode** step mentioned previously). Previous work [7, 1] set the initial pumped $\vec{v}_e$ to be distinct unit vectors. It turns out that for this choice of starting vectors, it is possible to construct a single matrix M (independent of a fixed choice of s), such that rank queries to submatrices of M correspond to the answers we wish to output in the APC problem and its variants.

In Section 3.2 we describe how we employ the flow vector framework to prove Theorem 4. Then in Section 3.3, we describe how we modify these methods to prove Theorem 5.

3.2 All-Pairs Connectivity

Our starting point is the $\tilde{O}(m^\omega)$ time algorithm for APC presented in [7], which uses the flow vector encoding scheme outlined in Section 3.1.

Let K be an $m \times m$ matrix, whose rows and columns are indexed by edges in the input graph. For each pair (e, f) of edges, if the head of e coincides with the tail of f, we set $K[e, f]$ to be a uniform random field element in $\mathbb{F}$. Otherwise, $K[e, f] = 0$. These field elements correspond precisely to the coefficients used in the random linear combinations of the flow vector framework. Define the matrix

$$M = (I - K)^{-1}. \tag{1}$$

Then [7] proves that with high probability, for any pair (s, t) of vertices, we have

$$\operatorname{rank} M[E_{\text{out}}(s), E_{\text{in}}(t)] = \lambda(s, t). \tag{2}$$

With this setup, the algorithm for APC is simple: first compute M (the **encode** step), and then for each pair of vertices (s, t), return the value of $\operatorname{rank} M[E_{\text{out}}(s), E_{\text{in}}(t)]$ as the connectivity from s to t (the **decode** step).

By Equation (1), we can complete the **encode** step in $\tilde{O}(m^\omega)$ time, simply by inverting an $m \times m$ matrix with entries from $\mathbb{F}$. It turns out we can also complete the **decode** step in the same time bound. So this gives an $\tilde{O}(m^\omega)$ time algorithm for APC.

Suppose now we want to solve the k-APC problem. A simple trick (observed in the proof of [1, Theorem 5.2] for example) in this setting can allow us to speed up the runtime of the **decode** step. However, it is not at all obvious how to speed up the **encode** step. To

implement the flow vector scheme of Section 3.1 as written, it seems almost inherent that one needs to invert an $m \times m$ matrix. Indeed, an inability to overcome this bottleneck is stated explicitly as part of the motivation in [1] for focusing on the k-APVC problem instead.

Our Improvement

The main idea behind our new algorithm for k-APC is to work with a low-rank version of the matrix K used in Equation (1) for the **encode** step.

Specifically, we construct certain random sparse matrices L and R with dimensions $m \times kn$ and $kn \times m$ respectively. We then set $K = LR$, and argue that with high probability, the matrix M defined in Equation (1) for this choice of K satisfies

$$\operatorname{rank} M[E_{\text{out}}(s), E_{\text{in}}(t)] = \min(k, \lambda(s,t)). \tag{3}$$

This equation is just a k-bounded version of Equation (2). By Proposition 6, we have

$$M = (I - K)^{-1} = (I - LR)^{-1} = I + L(I - RL)^{-1}R.$$

Note that $(I - RL)$ is a $kn \times kn$ matrix. So, to compute M (and thus complete the **encode** step) we no longer need to invert an $m \times m$ matrix! Instead we just need to invert a matrix of size $kn \times kn$. This is essentially where the $\tilde{O}\left((kn)^\omega\right)$ runtime in Theorem 4 comes from.

Conceptually, this argument corresponds to assigning flow vectors through the graph by replacing random linear combinations with random "low-rank combinations." That is, for an edge $e \in E_{\text{out}}(u)$ exiting a vertex u, we define the flow vector at e to be

$$\vec{e} = \sum_{i=1}^{k} \left(\sum_{f \in E_{\text{in}}(u)} L_i[f,u]\vec{f} \right) \cdot R_i[u,e],$$

where the inner summation is over all edges f entering u, $\vec{f}$ denotes the flow vector assigned to edge f, and the $L_i[f,u]$ and $R_i[u,e]$ terms correspond to random field elements uniquely determined by the index i and some (edge, vertex) pair.

Here, unlike in the method described in Section 3.1, the coefficient in front of $\vec{f}$ in its contribution to $\vec{e}$ is not uniquely determined by the pair of edges f and e. Rather, if edge f enters node u, then it has the same set of "weights" $L_i[f,u]$ it contributes to every flow vector exiting u. However, since we use k distinct weights, this restricted rule for propagating flow vectors still suffices to compute $\min(k, \lambda(s,t))$.

A good way to think about the effect of this alternate approach is that now for any vertex v and any integer $\ell \leq k$, if some subset of ℓ flow vectors assigned to edges in $E_{\text{in}}(v)$ is independent, then we expect that every subset of at most ℓ flow vectors assigned to edges in $E_{\text{out}}(v)$ is also independent. In the previous framework, this result held even for $\ell > k$. By relaxing the method used to determine flow vectors, we achieve a weaker condition, but this is still enough to solve k-APC.

This modification makes the **encode** step more complicated (it now consists of two parts: one where we invert a matrix, and one where we multiply that inverse with other matrices), but speeds it up overall. To speed up the **decode** step, we use a variant of an observation from the proof of [1, Theorem 5.2] to argue that we can assume every vertex in our graph has indegree and outdegree k. By Proposition 8 and Equation (3), this means we can compute $\min(k, \lambda(s,t))$ for all pairs (s,t) of vertices in $\tilde{O}(k^\omega n^2)$ time. So the bottleneck in our algorithm comes from the **encode** step, which yields the $\tilde{O}\left((kn)^\omega\right)$ runtime.

3.3 All-Pairs Vertex Connectivity

Our starting point is the $\tilde{O}((kn)^\omega)$ time algorithm in [1], which computes $\min(k, \nu(s,t))$ for all pairs of vertices (s,t) which are not edges. That algorithm is based off a variant of the flow vector encoding scheme outlined Section 3.1. Rather than assign vectors to edges, we instead assign flow vectors to vertices (intuitively this is fine because we are working with vertex connectivities in the k-APVC problem). The rest of the construction is similar: we imagine pumping some initial vectors to s and its out-neighbors, and then we propagate the flow through the graph so that at the end, for any vertex v, the flow vector assigned to v is a random linear combination of flow vectors assigned to in-neighbors of v.[6]

Let K be an $n \times n$ matrix, whose rows and columns are indexed by vertices in the input graph. For each pair (u,v) of vertices, if there is an edge from u to v, we set $K[u,v]$ to be a uniform random element in $\mathbb{F}$. Otherwise, $K[u,v] = 0$. These entries correspond precisely to coefficients used in the random linear combinations of the flow vector framework.

Now define the matrix

$$M = (I - K)^{-1}. \tag{4}$$

Then we argue that for any pair (s,t) of vertices, we have

$$\operatorname{rank} M[V_{\text{out}}[s], V_{\text{in}}[t]] = \begin{cases} \nu(s,t) + 1 & \text{if } (s,t) \text{ is an edge} \\ \nu(s,t) & \text{otherwise.} \end{cases} \tag{5}$$

Previously, [1, Proof of Lemma 5.1] sketched a different argument, which shows that $\operatorname{rank} M[V_{\text{out}}(s), V_{\text{in}}(t)] = \nu(s,t)$ when (s,t) is not an edge.

We use Equation (5) to solve k-APVC. For the **encode** step, we compute M. By Equation (4), we can do this by inverting an $n \times n$ matrix, which takes $\tilde{O}(n^\omega)$ time. For the **decode** step, by Equation (5) and Proposition 8, we can compute $\min(k, \nu(s,t))$ for all pairs (s,t) of vertices in asymptotically

$$\sum_{s,t} (\deg_{\text{out}}(s) \deg_{\text{in}}(t) + k^\omega) = m^2 + k^\omega n^2$$

time, where the sum is over all vertices s and t in the graph. The runtime bound we get here for the **decode** step is far too high – naively computing the ranks of submatrices is too slow if the graph has many high-degree vertices.

To avoid this slowdown, [1] employs a simple trick to reduce degrees in the graph: we can add layers of k new nodes to block off the ingoing and outgoing edges from each vertex in the original graph. That is, for each vertex s in G, we add a set S of k new nodes, replace the edges in $E_{\text{out}}(s)$ with edges from s to all the nodes in S, and add edges from every node in S to every vertex originally in $V_{\text{out}}(s)$. Similarly, for each vertex t in G, we add a set T of k new nodes, replace the edges in $E_{\text{in}}(t)$ with edges from all the nodes in T to t, and add edges from every vertex originally in $V_{\text{in}}(t)$ to every node in T.

It is easy to check that this transformation preserves the value of $\min(k, \nu(s,t))$ for all pairs (s,t) of vertices in the original graph where (s,t) is not an edge. Moreover, all vertices in the original graph have indegree and outdegree exactly k in the new graph. Consequently, the **decode** step can now be implemented to run in $\tilde{O}(k^\omega n^2)$ time. Unfortunately, this

[6] Actually, this behavior only holds for vertices $v \notin V_{\text{out}}[s]$. The rule for flow vectors assigned to vertices in $V_{\text{out}}[s]$ is a little more complicated, and depends on the values of the initial pumped vectors.

construction increases the number of vertices in the graph from n to $(2k+1)n$. As a consequence, in the **encode** step, the matrix K we work with is no longer $n \times n$, but instead is of size $(2k+1)n \times (2k+1)n$. Now inverting $I - K$ to compute M requires $\tilde{O}((kn)^\omega)$ time, which is why [1] obtains this runtime for their algorithm.

Our Improvement

Intuitively, the modification used by [1] to reduce degrees in the graph feels very inefficient. This transformation makes the graph larger in order to "lose information" about connectivity values greater than k. Rather than modify the graph in this way, can we modify the flow vector scheme itself to speed up the **decode** step? Our algorithm does this, essentially modifying the matrix of flow vectors to simulate the effect of the previously described transformation, without ever explicitly adding new nodes to the graph.

Instead of working directly with the matrix M from Equation (4), for each pair (s,t) of vertices we define a $(k+1) \times (k+1)$ matrix

$$M_{s,t} = B_s \left(M[V_{\text{out}}[s], V_{\text{in}}[t]]\right) C_t$$

which is obtained from multiplying a submatrix of M on the left and right by small random matrices B_s and C_t, with $k+1$ rows and columns respectively. Since B_s has $k+1$ rows and C_t has $k+1$ columns, we can argue that with high probability, Equation (5) implies that

$$\operatorname{rank} M_{s,t} = \begin{cases} \min(k+1, \nu(s,t)+1) & \text{if } (s,t) \text{ is an edge} \\ \min(k+1, \nu(s,t)) & \text{otherwise.} \end{cases}$$

So we can compute $\min(k, \nu(s,t))$ from the value of $\operatorname{rank} M_{s,t}$. This idea is similar to the preconditioning method used in algorithms for computing matrix rank efficiently (see [6] and the references therein). Conceptually, we can view this approach as a modification of the flow vector framework. Let $d = \deg_{\text{out}}(s)$. As noted in Section 3.1, previous work

1. starts by pumping out distinct d-dimensional unit vectors to nodes in $V_{\text{out}}(s)$, and then
2. computes the rank of all flow vectors of vertices in $V_{\text{in}}(t)$.

In our work, we instead

1. start by pumping out $(d+1)$ random $(k+1)$-dimensional vectors to nodes in $V_{\text{out}}[s]$, and then
2. compute the rank of $(k+1)$ random linear combinations of flow vectors for vertices in $V_{\text{in}}[t]$.

This alternate approach suffices for solving the k-APVC problem, while avoiding the slow $\tilde{O}((kn)^\omega)$ **encode** step of previous work.

So, in the **decode** step of our algorithm, we compute $\min(k, \nu(s,t))$ for each pair (s,t) of vertices by computing the rank of the $(k+1) \times (k+1)$ matrix $M_{s,t}$, in $\tilde{O}(k^\omega n^2)$ time overall.

Our **encode** step is more complicated than previous work, because not only do we need to compute the inverse $(I-K)^{-1}$, we also have to construct the $M_{s,t}$ matrices. Naively computing each $M_{s,t}$ matrix separately is too slow, so we end up using an indirect approach to compute all entries of the $M_{s,t}$ matrices *simultaneously*, with just $O(k^2)$ multiplications of $n \times n$ matrices. This takes $\tilde{O}(k^2 n^\omega)$ time, which is the bottleneck for our algorithm.

4 Flow Vector Encoding

The arguments in this section are similar to the arguments from [7, Section 2], but involve more complicated proofs because we work with low-rank random matrices as opposed to generic random matrices.

Fix a source vertex s in the input graph G. Let $d = \deg_{\text{out}}(s)$ denote the number of edges leaving s. Let $e_1, \ldots, e_d \in E_{\text{out}}(s)$ be the outgoing edges from s.

Take a prime $p = \Theta(m^5)$. Let $\vec{u}_1, \ldots, \vec{u}_d$ be distinct unit vectors in $\mathbb{F}_p^d$.

Eventually, we will assign each edge e in G a vector $\vec{e} \in \mathbb{F}_p^d$, which we call a *flow vector*. These flow vectors will be determined by a certain system of vector equations. To describe these equations, we first introduce some symbolic matrices.

For each index $i \in [k]$, we define an $m \times n$ matrix X_i, whose rows are indexed by edges of G and columns are indexed by vertices of G, such that for each edge $e = (u, v)$, entry $X_i[e, v] = x_{i,ev}$ is an indeterminate. All entries in X_i not of this type are zero. Similarly, we define $n \times m$ matrices Y_i, with rows indexed by vertices of G and columns indexed by edges of G, such that for every edge $f = (u, v)$, the entry $Y_i[u, f] = y_{i,uf}$ is an indeterminate. All entries in Y_i not of this type are zero. Let X be the $m \times kn$ matrix formed by horizontally concatenating the X_i matrices. Similarly, let Y be the $kn \times m$ matrix formed by vertically concatenating the Y_i matrices. Then we define the matrix

$$Z = XY = X_1Y_1 + \cdots + X_kY_k. \tag{6}$$

By construction, Z is an $m \times m$ matrix, with rows and columns indexed by edges of G, such that for any edges $e = (u, v)$ and $f = (v, w)$, we have

$$Z[e, f] = \sum_{i=1}^{k} x_{i,ev} y_{i,vf} \tag{7}$$

and all other entries of Z are set to zero.

Consider the following procedure. We assign independent, uniform random values from $\mathbb{F}_p$ to each variable $x_{i,ev}$ and $y_{i,uf}$. Let L_i, L, R_i, R, and K be the matrices over $\mathbb{F}_p$ resulting from this assignment to X_i, X, Y_i, Y, and Z respectively. In particular, we have

$$K = LR. \tag{8}$$

Now, to each edge e, we assign a flow vector $\vec{e} \in \mathbb{F}_p^d$, satisfying the following equalities:

1. Recall that $e_1, \ldots, e_d$ are all the edges exiting s, and $\vec{u}_1, \ldots, \vec{u}_d$ are distinct unit vectors in $\mathbb{F}_p^d$. For each edge $e_i \in E_{\text{out}}(s)$, we require its flow vector satisfy
$$\vec{e}_i = \left(\sum_{f \in E_{\text{in}}(s)} \vec{f} \cdot K[f, e_i] \right) + \vec{u}_i. \tag{9}$$
2. For each edge $e = (u, v)$ with $u \neq s$, we require its flow vector satisfy
$$\vec{e} = \sum_{f \in E_{\text{in}}(u)} \vec{f} \cdot K[f, e]. \tag{10}$$

A priori it is not obvious that flow vectors satisfying the above two conditions exist, but we show below that they do (with high probability). Let H_s be the $d \times m$ matrix whose columns are indexed by edges in G, such that the column associated with e_i is $\vec{u}_i$ for each index i, and the rest of the columns are zero vectors. Let F be the $d \times m$ matrix, with columns indexed by edges in G, whose columns $F[*, e] = \vec{e}$ are flow vectors for the corresponding edges. Then Equations (9) and (10) are encapsulated by the simple matrix equation

$$F = FK + H_s. \tag{11}$$

The following lemma shows we can solve for F in the above equation, with high probability.

▶ **Lemma 10.** *We have* $\det(I - K) \neq 0$*, with probability at least* $1 - 1/m^3$*.*

Proof. Since the input graph has no self-loops, by Equation (7) and the discussion immediately following it, we know that the diagonal entries of the $m \times m$ matrix Z are zero. By Equation (7), each entry of Z is a polynomial of degree at most two, with constant term set to zero. Hence, $\det(I - Z)$ is a polynomial over $\mathbb{F}_p$ with degree at most $2m$, and constant term equal to 1. In particular, this polynomial is nonzero. Then by the Schwartz-Zippel Lemma (Proposition 9), $\det(I - K)$ is nonzero with probability at least

$$1 - 2m/p \geq 1 - 1/m^3$$

by setting $p \geq 2m^4$. ◀

Suppose from now on that $\det(I-K) \neq 0$ (by Lemma 10, this occurs with high probability). Then with this assumption, we can solve for F in Equation (11) to get

$$F = H_s(I - K)^{-1} = \frac{H_s\left(\operatorname{adj}(I - K)\right)}{\det(I - K)}. \tag{12}$$

This equation will allow us to relate ranks of collections of flow vectors to connectivity values in the input graph.

▶ **Lemma 11.** *For any vertex* t *in* G*, with probability at least* $1 - 2/m^3$*, we have*

$$\operatorname{rank} F[*, E_{\text{in}}(t)] \leq \lambda(s, t).$$

Proof. Abbreviate $\lambda = \lambda(s, t)$. Conceptually, this proof works by arguing that the flow vectors assigned to all edges entering t are linear combinations of the flow vectors assigned to edges in a minimum (s, t)-cut of G.

Let C be a minimum (s, t)-cut. By Menger's theorem, $|C| = \lambda$.

Let S be the set of nodes reachable from s without using an edge in C, and let T be the set of nodes which can reach t without using an edge in C. By definition of an (s, t)-cut, S and T partition the vertices in G.

Now, let E' be the set of edges $e = (u, v)$ with $v \in T$. Set $K' = K[E', E']$ and $F' = F[*, E']$. Finally, let H' be a matrix whose columns are indexed by edges in E', such that the column associated with an edge $e \in C$ is $\vec{e}$, and all other columns are equal to $\vec{0}$.

Then by Equations (9) and (10), we have

$$F' = F'K' + H'.$$

Indeed, for any edge $e = (u, v) \in E'$, if $u \in S$ then $e \in C$ so $H'[*, e] = \vec{e}$, and there can be no edge $f \in E'$ entering u, so $(F'K')[*, e] = \vec{0}$. If instead $u \in T$, then $H'[*, e] = \vec{0}$, but every edge f entering u is in E', so by Equation (10), we have $(F'K')[*, e] = F'[*, e]$ as desired.

Using similar reasoning to the proof of Lemma 10, we have $\det(I-K') \neq 0$ with probability at least $1 - 1/m^3$. If this event occurs, we can solve for F' in the previous equation to get

$$F' = H'(I - K')^{-1}.$$

Since H' has at most λ nonzero columns, $\operatorname{rank} H \leq \lambda$. So by the above equation, $\operatorname{rank} F' \leq \lambda$. By definition, $E_{\text{in}}(t) \subseteq E'$. Thus $F[*, E_{\text{in}}(t)]$ is a submatrix of F'. Combining this with the previous results, we see that $\operatorname{rank} F[*, E_{\text{in}}(t)] \leq \lambda$, as desired. The claimed probability bound follows by a union bound over the events that $I - K$ and $I - K'$ are both invertible. ◀

▶ **Lemma 12.** *For any vertex t in G, with probability at least $1 - 2/m^3$, we have*

$$\operatorname{rank} F[*, E_{\text{in}}(t)] \geq \min(k, \lambda(s,t)).$$

Proof. Abbreviate $\lambda = \min(k, \lambda(s,t))$. Intuitively, our proof will argue that the presence of edge-disjoint paths from s to t will lead to certain edges in $E_{\text{in}}(t)$ being assigned linearly independent flow vectors (with high probability), which will then imply the desired rank lower bound.

By Menger's theorem, G contains λ edge-disjoint paths $P_1, \ldots, P_\lambda$ from s to t.

Consider the following assignment to the variables of the symbolic matrices X_i and Y_i. For each index $i \leq \lambda$ and edge $e = (u, v)$, we set variable $x_{i,ev} = 1$ if e is an edge in P_i. Similarly, for each index $i \leq \lambda$ and edge $f = (u, v)$, we set variable $y_{i,uf} = 1$ if f is an edge in P_i. All other variables are set to zero. In particular, if $i > \lambda$, then X_i and Y_i have all their entries set to zero. With respect to this assignment, the matrix X_iY_i (whose rows and columns are indexed by edges in the graph) has the property that $(X_iY_i)[e, f] = 1$ if f is the edge following e on path P_i, and all other entries are set to zero.

Then by Equation (6), we see that under this assignment, $Z[e, f] = 1$ if e and f are consecutive edges in some path P_i, and all other entries of Z are set to zero. For this particular assignment, because the P_i are edge-disjoint paths, Equations (9) and (10) imply that the last edge of each path P_i is assigned a distinct d-dimensional unit vector. These vectors are independent, so, $\operatorname{rank} F[*, E_{\text{in}}(t)] = \lambda$ in this case.

With respect to this assignment, this means that $F[*, E_{\text{in}}(t)]$ contains a $\lambda \times \lambda$ full-rank submatrix. Let F' be a submatrix of $F[*, E_{\text{in}}(t)]$ with this property. Since F' has full rank, we have $\det F' \neq 0$ for the assignment described above.

Now, before assigning values to variables, each entry of $\operatorname{adj}(I - Z)$ is a polynomial of degree at most $2m$. So by Equation (12), $\det F'$ is equal to some polynomial P of degree at most $2\lambda m$, divided by $(\det(I - Z))^\lambda$. We know P is a nonzero polynomial, because we saw above that $\det F'$ is nonzero for some assignment of values to the variables (and if P were the zero polynomial, then $\det F'$ would evaluate to zero under every assignment).

By Lemma 10, with probability at least $1 - 1/m^3$, a random evaluation to the variables will have $\det(I - Z)$ evaluate to a nonzero value. Assuming this event occurs, by Schwartz-Zippel Lemma (Proposition 9), a random evaluation to the variables in Z will have $\det F' \neq 0$ with probability at least $1 - (2\lambda m)/p \geq 1 - 1/m^3$ by setting $p \geq 2m^5$.

So by union bound, a particular $\lambda \times \lambda$ submatrix of $F[*, E_{\text{in}}(t)]$ will be full rank with probability at least $1 - 2/m^3$. This proves the desired result. ◀

▶ **Lemma 13.** *Fix vertices s and t. Define $\lambda = \operatorname{rank} (I - K)^{-1}[E_{\text{out}}(s), E_{\text{in}}(t)]$. With probability at least $1 - 4/m^3$, we have $\min(k, \lambda) = \min(k, \lambda(s,t))$.*

Proof. The definition of H_s together with Equation (12) implies that

$$F[*, E_{\text{in}}(t)] = (I - K)^{-1}[E_{\text{out}}(s), E_{\text{in}}(t)]. \tag{13}$$

By union bound over Lemmas 11 and 12, with probability at least $1 - 4/m^3$ the inequalities

$$\lambda = \operatorname{rank} (I - K)^{-1}[E_{\text{out}}(s), E_{\text{in}}(t)] = \operatorname{rank} F[*, E_{\text{in}}(t)] \leq \lambda(s,t)$$

and

$$\lambda = \operatorname{rank} (I - K)^{-1}[E_{\text{out}}(s), E_{\text{in}}(t)] = \operatorname{rank} F[*, E_{\text{in}}(t)] \geq \min(k, \lambda(s,t))$$

simultaneously hold. The desired result follows. ◀

5 Connectivity Algorithm

In this section, we present our algorithm for k-APC.

We begin by modifying the input graph G as follows. For every vertex v in G, we introduce two new nodes v_{out} and v_{in}. We replace each edge (u, v) originally in G is by the edge $(u_{\text{out}}, v_{\text{in}})$. We add k parallel edges from v to v_{out}, and k parallel edges from v_{in} to v, for all u and v. We call vertices present in the graph before modification the *original vertices*.

Suppose G originally had n nodes and m edges. Then the modified graph has $n_{\text{new}} = 3n$ nodes and $m_{\text{new}} = m + 2kn$ edges. For any original vertices s and t, edge-disjoint paths from s to t in the new graph correspond to edge disjoint paths from s to t in the original graph. Moreover, for any integer $\ell \leq k$, if the original graph contained ℓ edge-disjoint paths from s to t, then the new graph contains ℓ edge-disjoint paths from s to t as well.

Thus, for any original vertices s and t, the value of $\min(k, \lambda(s, t))$ remains the same in the old graph and the new graph. So, it suffices to solve k-APC on the new graph. In this new graph, the indegrees and outdegrees of every original vertex are equal to k. Moreover, sets $E_{\text{out}}(s)$ and $E_{\text{in}}(t)$ are pairwise disjoint, over all original vertices s and t.

We make use of the matrices defined in Section 4, except now these matrices are defined with respect to the modified graph. In particular, K, L, and R are now matrices with dimensions $m_{\text{new}} \times m_{\text{new}}$, $m_{\text{new}} \times n_{\text{new}}$, and $n_{\text{new}} \times m_{\text{new}}$ respectively.

Define $\tilde{L}$ to be the $kn \times n_{\text{new}}$ matrix obtained by vertically concatenating $L[E_{\text{out}}(s), *]$ over all original vertices s. Similarly, define $\tilde{R}$ to be the $n_{\text{new}} \times kn$ matrix obtained by horizontally concatenating $R[*, E_{\text{in}}(t)]$ over all original vertices t.

The Algorithm

Using the above definitions, we present our approach for solving k-APC in Algorithm 1.

Algorithm 1 Our algorithm for solving k-APC.

1: Compute the $n_{\text{new}} \times n_{\text{new}}$ matrix $(I - RL)^{-1}$.
2: Compute the $kn_{\text{new}} \times kn_{\text{new}}$ matrix $M = \tilde{L}(I - RL)^{-1}\tilde{R}$.
3: For each pair (s, t) of original vertices, compute

$$\operatorname{rank} M[E_{\text{out}}(s), E_{\text{in}}(t)]$$

and output this as the value for $\min(k, \lambda(s, t))$.

▶ **Theorem 14.** *With probability at least $1 - 5/(m_{\text{new}})$, Algorithm 1 correctly solves k-APC.*

Proof. By Lemma 10, with probability at least $1 - 1/(m_{\text{new}})^4$ the matrix $I - K$ is invertible. Going forward, we assume that $I - K$ is invertible.

By Lemma 13, with probability at least $1 - 4/(m_{\text{new}})^3$, we have

$$\operatorname{rank}(I - K)^{-1}[E_{\text{out}}(s), E_{\text{in}}(t)] = \min(k, \lambda(s, t)) \tag{14}$$

for any given original vertices s and t. By union bound over all $n^2 \leq (m_{\text{new}})^2$ pairs of original vertices (s, t), we see that Equation (14) holds for all original vertices s and t with probability at least $1 - 4/(m_{\text{new}})$.

Since $I - K$ is invertible, by Equation (8) and Proposition 6 we have

$$(I - K)^{-1} = (I - LR)^{-1} = I + L(I - RL)^{-1}R.$$

Using the above equation in Equation (14) shows that for original vertices s and t, the quantity $\min(k, \lambda(s,t))$ is equal to the rank of

$$(I + L(I - RL)^{-1}R)[E_{\text{out}}(s), E_{\text{in}}(t)] = L[E_{\text{out}}(s), *](I - RL)^{-1}R[*, E_{\text{in}}(t)]$$

where we use the fact that $I[E_{\text{out}}(s), E_{\text{in}}(t)]$ is the all zeroes matrix, since in the modified graph, $E_{\text{out}}(s)$ and $E_{\text{in}}(t)$ are disjoint sets for all pairs of original vertices (s,t).

Then by definition of $\tilde{L}$ and $\tilde{R}$, the above equation and discussion imply that

$$\min(k, \lambda(s,t)) = \text{rank } (\tilde{L}(I - RL)^{-1}\tilde{R})[E_{\text{out}}(s), E_{\text{in}}(t)] = \text{rank } M[E_{\text{out}}(s), E_{\text{in}}(t)]$$

which proves that Algorithm 1 outputs the correct answers.

A union bound over the events that $I - K$ is invertible and that Equation (14) holds for all (s,t), shows that Algorithm 1 is correct with probability at least $1 - 5/(m_{\text{new}})$. ◀

We are now ready to prove our main result.

▶ **Theorem 4.** *For any positive integer k, k-APC can be solved in $\tilde{O}((kn)^\omega)$ time.*

Proof. By Theorem 14, Algorithm 1 correctly solves the k-APC problem. We now argue that Algorithm 1 can be implemented to run in $\tilde{O}((kn)^\omega)$ time.

In step 1 of Algorithm 1, we need to compute $(I - RL)^{-1}$.

From the definitions of R and L, we see that to compute RL, it suffices to compute the products R_iL_j for each pair of indices $(i,j) \in [k]^2$. The matrix R_iL_j is $n_{\text{new}} \times n_{\text{new}}$, and its rows and columns are indexed by vertices in the graph. Given vertices u and v, let $E(u,v)$ denote the set of parallel edges from u to v. From the definitions of the R_i and L_j matrices, we see that for any vertices u and v, we have

$$(R_iL_j)[u,v] = \sum_{e \in E(u,v)} R_i[u,e]L_j[e,v]. \tag{15}$$

As noted in Section 2, for all vertices u and v we may assume that $|E(u,v)| \leq k$.

For each vertex u, define the $k \times \deg_{\text{out}}(u)$ matrix R'_u, with rows indexed by $[k]$ and columns indexed by edges exiting u, by setting

$$R'_u[i,e] = R_i[u,e]$$

for all $i \in [k]$ and $e \in E_{\text{out}}(u)$.

Similarly, for each vertex v, define the $\deg_{\text{in}}(v) \times k$ matrix L'_v by setting

$$L'_v[e,j] = L_j[e,v]$$

for all $e \in E_{\text{in}}(v)$ and $j \in [k]$.

Finally, for each pair (u,v) of vertices, define $R'_{uv} = R'_u[*, E(u,v)]$ and $L'_{uv} = L'_v[E(u,v), *]$. Then by Equation (15), we have

$$(R_iL_j)[u,v] = R'_{uv}L'_{uv}[i,j].$$

Thus, to compute the R_iL_j products, it suffices to build the R'_u and L'_v matrices in $O(km_{\text{new}})$ time, and then compute the $R'_{uv}L'_{uv}$ products. We can do this by computing $(n_{\text{new}})^2$ products of pairs of $k \times k$ matrices. Since for every pair of vertices (u,v), there are at most k parallel edges from u to v, $km_{\text{new}} \leq k^2n^2$, we can compute all the R_iL_j products, and hence the entire matrix RL, in $\tilde{O}(n^2k^\omega)$ time.

We can then compute $I - RL$ by modifying $O(kn)$ entries of RL. Finally, by Proposition 7 we can compute $(I - RL)^{-1}$ in $\tilde{O}((kn)^\omega)$ time.

So overall, step 1 of Algorithm 1 takes $\tilde{O}((kn)^\omega)$ time.

In step 2 of Algorithm 1, we need to compute $M = \tilde{L}(I - RL)^{-1}\tilde{R}$.

Recall that $\tilde{L}$ is a $kn \times n_{\text{new}}$ matrix. By definition, each row of $\tilde{L}$ has a single nonzero entry. Similarly, $\tilde{R}$ is an $n_{\text{new}} \times kn$ matrix, with a single nonzero entry in each column.

Thus we can compute M, and complete step 2 of Algorithm 1 in $\tilde{O}((kn)^2)$ time.

Finally, in step 3 of Algorithm 1, we need to compute

$$\operatorname{rank} M[E_{\text{out}}(s), E_{\text{in}}(t)] \tag{16}$$

for each pair of original vertices (s, t) in the graph. In the modified graph, each original vertex has indegree and outdegree k, so each $M[E_{\text{out}}(s), E_{\text{in}}(t)]$ is a $k \times k$ matrix. For any fixed (s, t), by Proposition 8 we can compute the rank of $M[E_{\text{out}}(s), E_{\text{in}}(t)]$ in $\tilde{O}(k^\omega)$ time.

So we can compute the ranks from Equation (16) for all n^2 pairs of original vertices (s, t) and complete step 3 of Algorithm 1 in $\tilde{O}(k^\omega n^2)$ time.

Thus we can solve k-APC in $\tilde{O}((kn)^\omega)$ time overall, as claimed. ◀

6 Encoding Vertex Connectivities

Take a prime $p = \tilde{\Theta}(n^5)$. Let K be an $n \times n$ matrix, whose rows and columns are indexed by vertices of G. For each pair (u, v) of vertices, if (u, v) is an edge in G, we set $K[u, v]$ to be a uniform random element of $\mathbb{F}_p$. Otherwise, $K[u, v] = 0$.

Recall from Section 2 that given a vertex v in G, we let $V_{\text{in}}[v] = V_{\text{in}}(v) \cup \{v\}$ be the set consisting of v and all in-neighbors of v, and $V_{\text{out}}[v] = V_{\text{out}}(v) \cup \{v\}$ be the set consisting of v and all out-neighbors of v. The following proposition[7] is based off ideas from [7, Section 2]. A proof of this result can be found in the full version of this paper [3, Appendix B.2].

▶ **Proposition 15.** *For any vertices s and t in G, with probability at least $1 - 3/n^3$, the matrix $(I - K)$ is invertible and we have*

$$\operatorname{rank}\ (I - K)^{-1}[V_{\text{out}}[s], V_{\text{in}}[t]] = \begin{cases} \nu(s,t) + 1 & \text{if } (s,t) \text{ is an edge} \\ \nu(s,t) & \text{otherwise.} \end{cases}$$

Proposition 15 shows that we can compute vertex connectivities in G simply by computing ranks of certain submatrices of $(I - K)^{-1}$. However, these submatrices could potentially be quite large, which is bad if we want to compute the vertex connectivities quickly. To overcome this issue, we show how to decrease the size of $(I - K)^{-1}$ while still preserving relevant information about the value of $\nu(s, t)$.

▶ **Lemma 16.** *Let M be an $a \times b$ matrix over $\mathbb{F}_p$. Let Γ be a $(k+1) \times a$ matrix with uniform random entries from $\mathbb{F}_p$. Then with probability at least $1 - (k+1)/p$, we have*

$$\operatorname{rank} \Gamma M = \min(k + 1, \operatorname{rank} M).$$

[7] The result stated here differs from a similar claim used in [1, Section 5]. See the full version of this paper [3, Appendix B.1] for a comparison of these arguments.

Proof. Since ΓM has $k+1$ rows, $\operatorname{rank}(\Gamma M) \leq k+1$.

Similarly, since ΓM has M as a factor, $\operatorname{rank}(\Gamma M) \leq \operatorname{rank} M$. Thus

$$\operatorname{rank} \Gamma M \leq \min(k+1, \operatorname{rank} M). \tag{17}$$

So, it suffices to show that ΓM has rank at least $\min(k+1, \operatorname{rank} M)$.

Set $r = \min(k+1, \operatorname{rank} M)$. Then there exist subsets S and T of row and column indices respectively, such that $|S| = |T| = r$ and $M[S,T]$ has rank r. Now, let U be an arbitrary set of r rows in Γ. Consider the matrix $M' = (\Gamma M)[U,T]$.

We can view each entry of M' as a polynomial of degree at most 1 in the entries of Γ. This means that $\det M'$ is a polynomial of degree at most r in the entries of Γ. Moreover, if the submatrix $\Gamma[U,T] = I$ happens to be the identity matrix, then $M' = M[S,T]$. This implies that $\det M'$ is a nonzero polynomial in the entries of Γ, because for some assignment of values to the entries of Γ, this polynomial has nonzero evaluation $\det M[S,T] \neq 0$ (where we are using the fact that $M[S,T]$ has full rank).

So by the Schwartz-Zippel Lemma (Proposition 9), the matrix ΓM has rank at least r, with probability at least $1 - r/p$.

Together with Equation (17), this implies the desired result. ◀

Now, to each vertex u in the graph, we assign a $(k+1)$-dimensional column vector $\vec{b}_u$ and a $(k+1)$-dimensional row vector $\vec{c}_u$.

Let B be the $(k+1) \times n$ matrix formed by concatenating all of the $\vec{b}_u$ vectors horizontally, and let C be the $n \times (k+1)$ matrix formed by concatenating all of the $\vec{c}_u$ vectors vertically. For each pair of distinct vertices (s,t), define the $(k+1) \times (k+1)$ matrix

$$M_{s,t} = B[*, V_{\text{out}}[s]] \left((I-K)^{-1}[V_{\text{out}}[s], V_{\text{in}}[t]] \right) C[V_{\text{in}}[t], *]. \tag{18}$$

The following result is the basis of our algorithm for k-APVC.

▶ **Lemma 17.** *For any vertices s and t in G, with probability at least $1 - 5/n^3$, we have*

$$\operatorname{rank} M_{s,t} = \begin{cases} \min(k+1, \nu(s,t)+1) & \text{if } (s,t) \text{ is an edge} \\ \min(k+1, \nu(s,t)) & \text{otherwise.} \end{cases}$$

Proof. Fix vertices s and t. Then, by Proposition 15, we have

$$\operatorname{rank}\ (I-K)^{-1}[V_{\text{out}}[s], V_{\text{in}}[t]] = \begin{cases} \nu(s,t)+1 & \text{if } (s,t) \text{ is an edge} \\ \nu(s,t) & \text{otherwise} \end{cases}$$

with probability at least $1 - 3/n^3$. Assume the above equation holds.

Then, by setting $\Gamma = B[*, V_{\text{out}}[s]]$ and $M = (I-K)^{-1}[V_{\text{out}}[s], V_{\text{in}}[t]]$ in Lemma 16, we see that with probability at least $1 - 1/n^3$ we have

$$\operatorname{rank} B[*, V_{\text{out}}[s]](I-K)^{-1}[V_{\text{out}}[s], V_{\text{in}}(t)] = \begin{cases} \min(k+1, \nu(s,t)+1) & \text{if } (s,t) \text{ is an edge} \\ \min(k+1, \nu(s,t)) & \text{otherwise.} \end{cases}.$$

Assume the above equation holds.

Finally, by setting $\Gamma = C^\top[*, V_{\text{in}}(t)]$ and $M = (B[*, V_{\text{out}}[s]](I-K)^{-1}[V_{\text{out}}[s], V_{\text{in}}(t)])^\top$ in Lemma 16 and transposition, we see that with probability at least $1 - 1/n^3$ we have

$$\operatorname{rank} B[*, V_{\text{out}}[s]] \left((I-K)^{-1}[V_{\text{out}}[s], V_{\text{in}}(t)] \right) C[V_{\text{in}}(t), *] = \min(k+1, \nu(s,t)+1)$$

if there is an edge from s to t, and

$$\operatorname{rank} B[*, V_{\text{out}}[s]] \left((I-K)^{-1}[V_{\text{out}}[s], V_{\text{in}}(t)] \right) C[V_{\text{in}}(t), *] = \min(k+1, \nu(s,t))$$

otherwise. So by union bound, the desired result holds with probability at least $1 - 5/n^3$. ◀

7 Vertex Connectivity Algorithm

Let A be the adjacency matrix of the graph G with self-loops. That is, A is the $n \times n$ matrix whose rows and columns are indexed by vertices of G, and for every pair (u, v) of vertices, $A[u, v] = 1$ if $v \in V_{\text{out}}[u]$ (equivalently, $u \in V_{\text{in}}[v]$), and $A[u, v] = 0$ otherwise.

Recall the definitions of the $\vec{b}_u$ and $\vec{c}_u$ vectors, and the K, B, C and $M_{s,t}$ matrices from Section 6. For each $i \in [k+1]$, let P_i be the $n \times n$ diagonal matrix, with rows and columns indexed by vertices of G, such that $P_i[u, u] = \vec{b}_u[i]$. Similarly, let Q_i be the $n \times n$ diagonal matrix, with rows and columns indexed by vertices of G, such that $Q_i[u, u] = \vec{c}_u[i]$.

With these definitions, we present our approach for solving k-APVC in Algorithm 2.

Algorithm 2 Our algorithm for solving k-APVC.

1: Compute the $n \times n$ matrix $(I - K)^{-1}$.
2: For each pair $(i, j) \in [k+1]^2$ of indices, compute the $n \times n$ matrix

$$D_{ij} = AP_i(I - K)^{-1}Q_jA^\top.$$

3: For each pair (s, t) of vertices, let $F_{s,t}$ be the $(k+1) \times (k+1)$ matrix whose (i, j) entry is equal to $D_{ij}[s, t]$. If (s, t) is an edge, output $(\operatorname{rank} F_{s,t}) - 1$ as the value for $\min(k, \nu(s, t))$. Otherwise, output $\min(k, \operatorname{rank} F_{s,t})$ as the value for $\min(k, \nu(s, t))$.

The main idea of Algorithm 2 is to use Lemma 17 to reduce computing $\min(k, \nu(s, t))$ for a given pair of vertices (s, t) to computing the rank of a corresponding $(k+1) \times (k+1)$ matrix, $M_{s,t}$. To make this approach efficient, we compute the entries of all $M_{s,t}$ matrices simultaneously, using a somewhat indirect argument.

▶ **Theorem 18.** *With probability at least $1 - 5/n$, Algorithm 2 correctly solves k-APVC.*

Proof. We prove correctness of Algorithm 2 using the following claim.

▷ Claim 19. For all pairs of indices $(i, j) \in [k+1]^2$ and all pairs of vertices (s, t), we have

$$M_{s,t}[i, j] = D_{ij}[s, t],$$

where D_{ij} is the matrix computed in step 2 of Algorithm 2.

Proof. By expanding out the expression for D_{ij} from step 2 of Algorithm 2, we have

$$D_{ij}[s, t] = \sum_{u,v} A[s, u]P_i[u, u]\left((I - K)^{-1}[u, v]\right)Q_j[v, v]A[v, t],$$

where the sum is over all vertices u, v in the graph (here, we use the fact that P_i and Q_j are diagonal matrices). By the definitions of A, the P_i, and the Q_j matrices, we have

$$D_{ij}[s, t] = \sum_{\substack{u \in V_{\text{out}}[s] \\ v \in V_{\text{in}}[t]}} \vec{b}_u[i]\left((I - K)^{-1}[u, v]\right)\vec{c}_v[j]. \tag{19}$$

On the other hand, the definition of $M_{s,t}$ from Equation (18) implies that

$$M_{s,t}[i, j] = \sum_{\substack{u \in V_{\text{out}}[s] \\ v \in V_{\text{in}}[t]}} B[i, u]\left((I - K)^{-1}[u, v]\right)C[v, j].$$

Since $B[i,u] = \vec{b}_u[i]$ and $C[v,j] = \vec{c}_v[j]$, the above equation and Equation (19) imply that

$$M_{s,t}[i,j] = D_{ij}[s,t]$$

for all (i,j) and (s,t), as desired. ◁

By Claim 19, the matrix $F_{s,t}$ computed in step 3 of Algorithm 2 is equal to $M_{s,t}$. So by Lemma 17, for any fixed pair (s,t) of vertices we have

$$\operatorname{rank} F_{s,t} = \begin{cases} \min(k+1, \nu(s,t)+1) & \text{if } (s,t) \text{ is an edge} \\ \min(k+1, \nu(s,t)) & \text{otherwise.} \end{cases} \tag{20}$$

with probability at least $1 - 5/n^3$. Then by a union bound over all pairs of vertices (s,t), we see that Equation (20) holds for all pairs (s,t), with probability at least $1 - 5/n$.

Assume this event occurs. Then if (s,t) is an edge, by Equation (20) we correctly return

$$(\operatorname{rank} F_{s,t}) - 1 = \min(k+1, \nu(s,t)+1) - 1 = \min(k, \nu(s,t))$$

as our answer for this pair.

Similarly, if (s,t) is not an edge, by Equation (20) we correctly return

$$\min(k, \operatorname{rank} F_{s,t}) = \min(k, k+1, \nu(s,t)) = \min(k, \nu(s,t))$$

as our answer for this pair. This proves the desired result. ◀

With Theorem 18 established, we can prove our result for vertex connectivities.

▶ **Theorem 5.** *For any positive integer k, k-APVC can be solved in $\tilde{O}(k^2 n^\omega)$ time.*

Proof. By Theorem 18, Algorithm 2 correctly solves the k-APVC problem. We now argue that Algorithm 2 can be implemented to run in $\tilde{O}(k^2 n^\omega)$ time.

In step 1 of Algorithm 2, we need to compute $(I-K)^{-1}$. Since K is an $n \times n$ matrix, by Proposition 7 we can complete this step in $\tilde{O}(n^\omega)$ time.

In step 2 of Algorithm 2, we need to compute D_{ij} for each pair $(i,j) \in [k+1]^2$. For each fixed pair (i,j), the D_{ij} matrix is defined as a product of five $n \times n$ matrices whose entries we know, so this step takes $\tilde{O}(k^2 n^\omega)$ time overall.

In step 3 of Algorithm 2, we need to construct each F_{st} matrix, and compute its rank. Since each F_{st} matrix has dimensions $(k+1) \times (k+1)$ and its entries can be filled in simply by reading entries of the D_{ij} matrices we have already computed, by Proposition 8 this step can be completed in $\tilde{O}(k^\omega n^2)$ time.

By adding up the runtimes for each of the steps and noting that $k \leq n$, we see that Algorithm 2 solves k-APVC in $\tilde{O}(k^2 n^\omega)$ time, as claimed. ◀

References

1 Amir Abboud, Loukas Georgiadis, Giuseppe F. Italiano, Robert Krauthgamer, Nikos Parotsidis, Ohad Trabelsi, Przemysław Uznański, and Daniel Wolleb-Graf. Faster algorithms for all-pairs bounded min-cuts. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 7:1–7:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.7.

2 Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. APMF < APSP? Gomory-Hu tree for unweighted graphs in almost-quadratic time. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1135–1146. IEEE, 2021. doi:10.1109/FOCS52979.2021.00112.

3 Shyan Akmal and Ce Jin. An efficient algorithm for all-pairs bounded edge connectivity, 2023. arXiv:2305.02132.

4 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10–13, 2021*, pages 522–539. SIAM, 2021. doi:10.1137/1.9781611976465.32.

5 Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 612–623, 2022. doi:10.1109/FOCS54457.2022.00064.

6 Ho Yee Cheung, Tsz Chiu Kwok, and Lap Chi Lau. Fast matrix rank algorithms and applications. *Journal of the ACM*, 60(5):1–25, October 2013. doi:10.1145/2528404.

7 Ho Yee Cheung, Lap Chi Lau, and Kai Man Leung. Graph connectivities, network coding, and expander graphs. *SIAM Journal on Computing*, 42(3):733–751, January 2013. doi:10.1137/110844970.

8 M. J. Fischer and A. R. Meyer. Boolean matrix multiplication and transitive closure. In *12th Annual Symposium on Switching and Automata Theory (SWAT 1971)*. IEEE, October 1971. doi:10.1109/swat.1971.4.

9 François Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the Coppersmith-Winograd tensor. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 1029–1046, USA, 2018. Society for Industrial and Applied Mathematics.

10 Loukas Georgiadis, Daniel Graf, Giuseppe F. Italiano, Nikos Parotsidis, and Przemysław Uznański. All-Pairs 2-Reachability in $O(n^{\omega} log n)$ Time. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 74:1–74:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2017.74.

11 Zhiyi Huang, Yaowei Long, Thatchaphol Saranurak, and Benyu Wang. Tight conditional lower bounds for vertex connectivity problems, 2022. doi:10.48550/arXiv.2212.00359.

12 Robert Krauthgamer and Ohad Trabelsi. Conditional lower bounds for all-pairs max-flow. *ACM Trans. Algorithms*, 14(4), August 2018. doi:10.1145/3212510.

13 Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, August 1995. doi:10.1017/cbo9780511814075.

14 Ohad Trabelsi. (Almost) ruling out SETH lower bounds for all-pairs max-flow, 2023. doi:10.48550/arXiv.2304.04667.

15 Xiaowei Wu and Chenzi Zhang. Efficient algorithm for computing all low s-t edge connectivities in directed graphs. In *Mathematical Foundations of Computer Science 2015*, pages 577–588. Springer Berlin Heidelberg, 2015. doi:10.1007/978-3-662-48054-0_48.

Low-Depth Arithmetic Circuit Lower Bounds: Bypassing Set-Multilinearization

Prashanth Amireddy
Harvard University, Cambridge, MA, USA

Ankit Garg
Microsoft Research, Bangalore, India

Neeraj Kayal
Microsoft Research, Bangalore, India

Chandan Saha
Indian Institute of Science, Bangalore, India

Bhargav Thankey
Indian Institute of Science, Bangalore, India

Abstract

A recent breakthrough work of Limaye, Srinivasan and Tavenas [29] proved superpolynomial lower bounds for low-depth arithmetic circuits via a "hardness escalation" approach: they proved lower bounds for low-depth *set-multilinear* circuits and then lifted the bounds to low-depth general circuits. In this work, we prove superpolynomial lower bounds for low-depth circuits by bypassing the hardness escalation, i.e., the set-multilinearization, step. As set-multilinearization comes with an exponential blow-up in circuit size, our direct proof opens up the possibility of proving an exponential lower bound for low-depth homogeneous circuits by evading a crucial bottleneck. Our bounds hold for the iterated matrix multiplication and the Nisan-Wigderson design polynomials. We also define a subclass of unrestricted depth homogeneous formulas which we call *unique parse tree* (UPT) formulas, and prove superpolynomial lower bounds for these. This significantly generalizes the superpolynomial lower bounds for *regular* formulas [6, 19].

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases arithmetic circuits, low-depth circuits, lower bounds, shifted partials

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.12

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2022/151/`

Funding *Prashanth Amireddy*: Supported in part by a Simons Investigator Award and NSF Award CCF 2152413 to Madhu Sudan. A part of this work was done while the author was a research fellow at Microsoft Research, India.
Chandan Saha: Partially supported by a MATRICS grant of the Science and Engineering Research Board, DST, India.
Bhargav Thankey: Supported by the Prime Minister's Research Fellowship, India.

Acknowledgements We would like to thank the anonymous reviewers for their valuable feedback.

1 Introduction

Arithmetic circuits are a natural model for computing polynomials using the basic operations of addition and multiplication. One of the most fundamental questions about arithmetic circuits is about finding a family of explicit polynomials (if they exist) that cannot be computed by polynomial-sized arithmetic circuits. The existence of such explicit polynomials was conjectured by Valiant in 1979 [40] and is the famed VP vs VNP conjecture. Arithmetic

circuit lower bounds are expected to be easier than Boolean circuit lower bounds. Among many reasons, one is due to the phenomenon of depth reduction. Arithmetic circuits can be converted into low-depth circuits preserving the output polynomial and not blowing up the size too much [1, 10, 22, 39, 41]. Due to this, strong enough lower bounds even for restrictive models of computation like depth-3 circuits or homogeneous depth-4 circuits can lead to superpolynomial arithmetic circuit lower bounds.

Arithmetic formulas are an important subclass of arithmetic circuits where the out-degree of every gate is at most 1. For constant-depth, formulas and circuits are polynomially related. Also, all our results deal with formulas. So we will only refer to formulas from here on. We consider (families of) polynomials having degree at most polynomial in n, the number of variables. One of the first results studying low-depth arithmetic formulas was that of [32], who proved lower bounds for homogeneous depth-3 formulas. Progress on homogeneous formula lower bound was stalled for a while, and then various lower bounds for homogeneous depth-4 formulas were proven in a series of works [6, 9, 13, 14, 19, 25, 26]. There was limited progress for higher-depth formulas, and lower bounds remained open even for depth-5 formulas. In a recent breakthrough work, [29] proved superpolynomial lower bounds for constant-depth arithmetic formulas. Their lower bounds are of the form $n^{\Omega(\log(n)^{c_\Delta})}$ for a constant $0 < c_\Delta < 1$ depending on the depth Δ of the formula. The following two open problems naturally emerge out of their work.

▶ **Open Problem 1.** *Prove superpolynomial lower bounds for* general *formulas or even homogeneous formulas. (A formula is homogeneous if every gate computes a homogeneous polynomial.)*

▶ **Open Problem 2.** *Prove* exponential *lower bounds for constant-depth arithmetic formulas. This is interesting even for homogeneous depth-*5 *formulas.*

Towards answering Open Problems 1.1 and 1.2, let us examine the lower bound proof in [29] at a high level. Their proof has two main steps: First, they reduce the problem of proving lower bounds for low-depth formulas to the problem of proving lower bounds for low-depth *set-multilinear* formulas; set-multilinear formulas are special homogeneous formulas with an underlying partition of the variables into subsets. [29] calls such reductions "hardness escalation". Second, they use an interesting adaptation of the rank of the partial derivatives matrix measure [31] to prove a lower bound for low-depth set-multilinear formulas. They call this measure *relative rank* (relrk). The effectiveness of the relrk measure crucially depends on a certain "imbalance" between the sizes of the sets used to define set-multilinear polynomials. The proof in [29] raises two natural questions:

Question 1: Can we bypass the hardness escalation, i.e., the set-multilinearization, step?

Question 2: Can we design a measure that exploits some weakness of homogeneous (but not necessarily set-multilinear) formulas directly?

Motivations for studying Question 1. Set-multilinear circuits form a natural circuit class as most interesting polynomial families, such as the determinant, permanent, iterated matrix multiplication, etc., are set-multilinear. However, set-multilinearization comes with an exponential blow up in size – a homogeneous, depth-Δ formula computing a set-multilinear polynomial of degree d can be converted to a set-multilinear formula of depth Δ and size $d^{O(d)} \cdot s$ (see [29]). So, an exponential lower bound for low-depth set-multilinear formulas does not imply an exponential lower bound for low-depth homogeneous formulas since we are restricted to work with $d \leq \frac{\log n}{\log \log n}$. Indeed, it is possible to strengthen and refine the argument in [29] to get an *exponential* lower bound for low-depth set-multilinear formulas (see [2]). An approach that evades the hardness escalation step, which is a critical bottleneck,

and directly works with homogeneous formulas has the potential to avoid the $d^{O(d)}$ loss and give an exponential lower bound for low-depth homogeneous formulas. For instance, the direct arguments in [14,26] yield exponential lower bounds for homogeneous depth-4 formulas. If we go via the hardness escalation approach, we get a quasi-polynomial lower bound for the same model. Besides, a direct argument can also be used to prove lower bounds for polynomials that do not have a non-trivial set-multilinear component, see the full version of this article [2] for more details. The hardness escalation approach of [29] can not yield such a lower bound. Furthermore, it is conceivable that a direct argument can also be used to obtain functional lower bounds for low-depth formulas which might be useful in proof complexity.

Motivations for studying Question 2. Typical measures used for proving lower bounds for arithmetic circuits include the partial derivatives measure (PD) [32,38], the rank of the partial derivatives matrix measure (a.k.a. evaluation dimension) [31, 34, 36], the shifted partials measure (SP) and its variants [9,14,19], the affine projections of partials measure (APP) [7,15], etc. All these measures are defined for *any* polynomial, which is not necessarily set-multilinear. Whereas the relrk measure used in [29], although very effective, is defined for set-multilinear polynomials. Measures such as PD, SP, and APP have the geometrically appealing property that they are invariant under the application of invertible linear transformations on the variables. Since low-depth formulas, as well as low-depth homogeneous formulas, are closed under linear transformations, it is natural to look for measures that do not blow up much on applying linear transformations. Another important motivation for studying Question 2 is to learn low-depth homogeneous formulas. While the "hardness escalation" paradigm of reducing to the set-multilinear case works for proving lower bounds, it is not clear how to exploit it to design learning algorithms for low-depth formulas. Lower bounds for arithmetic circuits are intimately connected to learning [5, 7, 18, 42]. Hence if we have a lower bound measure that directly exploits the weakness of low-depth homogeneous formulas, it opens up the possibility of new learning algorithms for such models.

1.1 Our results

We answer Questions 1 and 2 by giving a direct lower bound for low-depth homogeneous formulas via the SP measure which was used in the series of works on homogeneous depth-4 exponential lower bounds. While our proof also yields lower bounds only in the low-degree setting, the hope is that it could potentially lead to a stronger lower bound in the future.

Consider the *shifted partials* measure: $\mathsf{SP}_{k,\ell}(f) := \dim\langle \mathbf{x}^{\ell} \cdot \boldsymbol{\partial}^{k}(f)\rangle$, where f is a polynomial. That is, $\mathsf{SP}_{k,\ell}(f)$ is the dimension of the space spanned by the polynomials obtained by multiplying degree ℓ monomials to partial derivatives of f of order k. Also, for convenience, let us denote by $M(n,k) := \binom{n+k-1}{k}$ the number of monomials of degree k in n variables. Then note that for a homogeneous polynomial f of degree d, $\mathsf{SP}_{k,\ell}(f) \leq \min\{M(n,k)M(n,\ell), M(n,d-k+\ell)\}$.

We show that for polynomials computed by low-depth homogeneous formulas, the shifted partials measure with an appropriate setting of k and ℓ is substantially smaller than the above upper bound. At the same time, we exhibit explicit "hard" polynomials for which the shifted partials measure is close to the above bound, hence yielding a lower bound.

▶ **Theorem 3** (Lower bound for low-depth homogeneous formulas via shifted partials)**.** *Let C be a homogeneous formula of size s and product-depth Δ that computes a polynomial of degree d in n variables. Then for appropriate values of k and ℓ,*

$$\mathsf{SP}_{k,\ell}(C) \leq \frac{s\, 2^{O(d)}}{n^{\Omega(d^{2^{1-\Delta}})}} \min\{M(n,k)M(n,\ell), M(n,d-k+\ell)\}.$$

At the same time, there are homogeneous polynomials f of degree d in n variables (e.g., an appropriate projection of iterated matrix multiplication polynomial, Nisan-Wigderson design polynomial, etc.) such that

$$\mathsf{SP}_{k,\ell}(f) \geq 2^{-O(d)} \min\{M(n,k)M(n,\ell), M(n,d-k+\ell)\}.$$

This gives a lower bound of $\frac{n^{\Omega(d^{2^{1-\Delta}})}}{2^{O(d)}}$ on the size of homogeneous product-depth Δ formulas for f.

▶ Remark 4.

1. At the end of this section, we briefly remark why it is surprising that we are able to obtain the above lower bound using shifted partials. We also show that the lower bound can be derived using the *affine projections of partials* (APP) measure (Lemma 19).
2. The above lower bound is slightly better than the bound of [29]. Instead of the $d^{O(d)}$ loss incurred due to converting homogeneous to set-multilinear formulas, our analysis incurs a $2^{O(d)}$ loss; in fact, this loss can be brought down to $2^{O(k)}$, but we ignore this distinction as we set $k = \Theta(d)$ in the analysis. So, for example, for homogeneous product-depth 2 formulas, our superpolynomial lower bound continues to hold for a higher degree ($\log^2(n)$ vs $(\log(n)/\log\log(n))^2$ in [29]). While the improvement may be insignificant, this hints at something interesting going on with the direct approach (see Section 1.2).

Lower bounds for general-depth arithmetic formulas are expected to be easier than arithmetic circuit lower bounds. However, despite several approaches and attempts (e.g. via tensor rank lower bounds [35]), we still do not have superpolynomial arithmetic formula lower bounds. There has been some success though in proving lower bounds for some natural restricted models (apart from the depth restrictions considered above). For example, [19] considered the model of *regular* arithmetic formulas. These are formulas which consist of alternating layers of addition ($+$) and multiplication ($\times$) gates such that the fanin of all gates in any fixed layer is the same. This is a natural model and the best-known formulas for many interesting polynomial families like determinant, permanent, iterated matrix multiplication, etc. are all regular. [19] proved a superpolynomial lower bound on the size of regular formulas for an explicit polynomial and later [6] proved a tight lower bound for the iterated matrix multiplication polynomial.

We prove superpolynomial lower bounds for a more general model.[1] Consider a model of homogeneous arithmetic formulas consisting of alternating layers of addition ($+$) and multiplication ($\times$) gates such that the fanin of all addition gates can be arbitrary but fanin of product gates in any fixed layer is the same. We call these *product-regular*. We prove super-polynomial lower bounds for homogeneous product-regular formulas. Previously we did not know of lower bounds for even a much simpler model where the fanins of all the product gates are fixed to 2.

In fact, we prove lower bounds for an even more general model which we call *Unique Parse Tree* (UPT) formulas. A parse tree of a formula is a tree where for every $+$ gate, one picks exactly one child and for every product gate, we pick all the children. Then we "short circuit" all the addition gates. Parse trees capture the way monomials are generated in a formula. We say that a formula is UPT if all its parse trees are isomorphic. A product-regular formula is clearly UPT. In the theorem below, $IMM_{n,\log n}$ is the iterated multiplication polynomial of degree $\log n$.

[1] The model in [6, 19] allowed slight non-homogeneity with the formal degree upper bounded by a small constant times the actual degree. However, we only work with homogeneous formulas.

▶ **Theorem 5.** *Any UPT formula computing $IMM_{n,\log(n)}$ has size at least $n^{\Omega(\log\log(n))}$. A similar lower bound holds for the Nisan-Wigderson design polynomial.*

▶ Remark 6.

1. While homogeneous product-regular formulas are restricted to compute polynomials with only certain degrees (e.g., higher product-depth cannot compute prime degrees), homogeneous UPT formulas do not suffer from this restriction.
2. While this result (which is obtained using the SP and the APP measures) could possibly also be obtained by defining a similar model in the set-multilinear world, proving a lower bound there and then transporting it back to the homogeneous world, our framework has fewer number of moving parts and hence makes it easier to derive such results.

Challenges to using the SP measure. Let us remark briefly why it is surprising that we are able to prove low-depth lower bounds via shifted partials. [8, 37] showed that the PD measure of the polynomial $(x_1^2 + \cdots + x_n^2)^{\frac{d}{2}}$ is the maximum possible when the order of derivatives, k, is at most $\frac{d}{2}$. Notice that $(x_1^2 + \cdots + x_n^2)^{\frac{d}{2}}$ can be computed by a homogeneous depth-4 formula of size $O(nd)$. So, it is not possible to prove super-polynomial lower bounds for low-depth homogeneous formulas using the PD measure as it is. One may ask if the SP measure also has a similar limitation. Some of the finer separation results in [23, 24] indicate that the SP measure (and some of its variants) can be fairly large for homogeneous depth-4 and depth-5 formulas for the choices of k used in prior work. Also, the exponential lower bounds for homogeneous depth-4 circuits in [14, 26] use random restrictions along with a variant of the SP measure. It is not clear how to leverage random restrictions for even homogeneous depth-5 circuits – this is also pointed out in [29]. Fortunately, [23, 24] do not rule out the possibility of using SP for all choices of parameters, like, say, $k \approx \frac{d}{2}$, to prove lower bounds for low-depth homogeneous formulas. But, the original intuition from algebraic geometry that led to the development of the SP measure (see [9] Section 2.1) breaks down completely when k is so large (see [2]). Despite these apparent hurdles, and to our surprise, we overcome these challenges and are able to use SP with $k \approx \frac{d}{2}$ to prove super-polynomial lower bounds for low-depth homogeneous formulas. To the best of our knowledge, no previous work uses SP with this high a value of k.

1.2 Techniques and proof overview

In this section, we explain the proof idea and compare it with that in [29]. A lot of lower bounds in arithmetic complexity follow the following outline.

Step 1: Depth reduction. One first shows that if $f(\mathbf{x})$ is computed by a *small* circuit from some restricted subclass of circuits, then there is a corresponding subclass of depth-4 circuits such that $f(\mathbf{x})$ is also computed by a *relatively small* circuit from this subclass[2]. The resulting subclass is of the form: $f(\mathbf{x}) = \sum_{i=1}^{s} \prod_{j=1}^{t_i} Q_{i,j}$. Usually there are simple restrictions on the degrees of $Q_{i,j}$'s. For example, they could be upper bounded by some number.

[2] Some major results in the area such as [29, 33] did not originally proceed via a depth reduction but instead analysed formulas directly. These results can however be restated as first doing a depth reduction and then applying the appropriate measure.

Step 2: Employing a suitable set of linear maps. Let $\mathbb{F}[\mathbf{x}]^{=d}$ be the space of homogeneous polynomials of degree d, W be a suitable vector space, and $\mathrm{Lin}(\mathbb{F}[\mathbf{x}]^{=d}, W)$ be the space of linear maps from $\mathbb{F}[\mathbf{x}]^{=d}$ to W. We choose a suitable set of linear maps $\mathcal{L} \subseteq \mathrm{Lin}(\mathbb{F}[\mathbf{x}]^{=d}, W)$ that define a complexity measure $\mu_{\mathcal{L}}(f) := \dim(\mathcal{L}(f))$, where $\mathcal{L}(f) := \langle \{L(f) \ : \ L \in \mathcal{L}\} \rangle$.

We would like to choose $\mathcal{L}$ so that it identifies some weakness of the terms $\prod_{j=1}^{t} Q_j$ in the depth-4 circuit. That is, $\mu_{\mathcal{L}}\left(\prod_{j=1}^{t} Q_j\right)$ should be much smaller than $\mu_{\mathcal{L}}(f)$ for a generic f. For e.g., if Q_j's are all linear polynomials, we can choose $\mathcal{L}$ to be the partial derivatives of order k, $\boldsymbol{\partial}^k$. Then, $\mu_{\mathcal{L}}\left(\prod_{j=1}^{t} Q_j\right) \leq \binom{t}{k} \ll \binom{n+k-1}{k}$ which is the value for a generic f (for $k \leq t/2$). This is the basis of the homogeneous depth-3 formula lower bound in [32].

For proving lower bounds for bounded bottom fan-in depth-4 circuits (i.e., when degree of Q_j's is upper bounded by some number), [9, 13] introduced the SP measure and used the linear maps $\mathcal{L} = \mathbf{x}^{\ell} \cdot \boldsymbol{\partial}^k$. The main insight in their proof was that if we apply a partial derivative of order k on $\prod_{j=1}^{t} Q_j$ and use the product rule, then at least $t-k$ of the Q_j's remain untouched. This structure can then be exploited by the shifts to get a lower bound. This intuition however completely breaks down for $k \geq t$ (see [2]). Due to this, progress remain stalled for higher depth arithmetic circuit lower bounds via SP.

In a major breakthrough, [29] gets around the above obstacle by working with set-multilinear circuits which entails working with polynomials over d sets of variables $(\mathbf{x}_1, \ldots, \mathbf{x}_d)$, $|\mathbf{x}_i| = n$. Let us use the shorthand $\mathbf{x}_S = (\mathbf{x}_i)_{i \in S}$. The products they deal with are of the form $\prod_{j=1}^{t} Q_j(\mathbf{x}_{S_j})$, where $S_1, S_2, \ldots, S_t$ form a partition of $[d]$. The set of linear maps they use are $\mathcal{L} = \Pi \circ \partial_{\mathbf{x}_A}$ for a subset $A \subseteq [d]$. Here, Π is a map that sets $n - n_0$ variables in each of the variable sets in $\mathbf{x}_{[d]\setminus A}$ to 0. They observe (for the appropriate choice of n_0) that $\mu_{\mathcal{L}}\left(\prod_{j=1}^{t} Q_j(\mathbf{x}_{S_j})\right) \leq \frac{n^{|A|}}{2^{\frac{1}{2}\sum_{j=1}^{t} \mathrm{imbalance}_j}}$.

Here, $\mathrm{imbalance}_j = ||A \cap S_j| \log(n) - |S_j \setminus A| \log(n_0)|$. For the appropriate choice of n_0, a generic set-multilinear f satisfies $\mu_{\mathcal{L}}(f) = n^{|A|}$, so that lower bound (on the number of summands) obtained is exponential in the total imbalance $\sum_{j=1}^{t} \mathrm{imbalance}_j$. [29] observe that this quantity is *somewhat large* for the depth-4 circuits that they consider.

The core of the above derivatives-based argument allows us to unravel some structure in partial derivatives of order k applied on $\prod_{j=1}^{t} Q_j$ for values of $k \gg t$. We use this to derive a structure for the partial derivative space of a product $\prod_{j=1}^{t} Q_j(\mathbf{x})$. Consider a partial derivative operator of order k indexed by a multiset α of size k. Using the chain rule,

$$\partial_{\alpha} \prod_{j=1}^{t} Q_j = \sum_{\alpha_1, \ldots, \alpha_t : \sum_{i=1}^{t} \alpha_i = \alpha} c^{\alpha}_{\alpha_1, \ldots, \alpha_t} \prod_{j=1}^{t} \partial_{\alpha_j} Q_j$$

for appropriate constants $c^{\alpha}_{\alpha_1, \ldots, \alpha_t}$'s. In the product $\prod_{j=1}^{t} \partial_{\alpha_j} Q_j$, we can try to club terms into two groups depending on if the size of $|\alpha_j|$ is small or large. It turns out that the right threshold for $|\alpha_j|$ is $k \deg(Q_j)/d$ (i.e., if we divide the order of the derivatives proportional to the degrees of the terms). Let $S := \{j : |\alpha_j| \leq k \deg(Q_j)/d\}$. Define $k_0 := \sum_{j \in S} |\alpha_j|$ and $\ell_0 := \sum_{j \in \overline{S}} (\deg(Q_j) - |\alpha_j|)$. Notice that we can write the product $\prod_{j=1}^{t} \partial_{\alpha_j} Q_j$ as $P \prod_{j \in S} \partial_{\alpha_j} Q_j$, for a degree ℓ_0 polynomial P. Hence, $\partial_{\alpha} \prod_{j=1}^{t} Q_j$ is a sum of terms of this form. While it is not immediate (due to the condition on α_j's in S), with a bit more work, one can combine the product of partials into a single partial.

What can we say about k_0 and ℓ_0? It turns out that the quantity that comes up in the calculations is $k_0 + \frac{k}{d-k}\ell_0$ and it satisfies $k_0 + \frac{k}{d-k}\ell_0 \leq k$. Note that k_0 is between 0 and k, and ℓ_0 between 0 and $d-k$. So the normalization brings ℓ_0 to the right "scale".

It turns out we can give a better bound in terms of a quantity we call *residue* defined as

$$\mathsf{residue}_k(d_1,\ldots,d_t) := \frac{1}{2}\cdot \min_{k_1,\ldots,k_t\in\mathbb{Z}} \sum_{j=1}^{t}\left|k_j - \frac{k}{d}\cdot d_j\right|.$$

and having the property that:

▶ **Proposition 7.** *Let k_0 and ℓ_0 be defined as above. Then, $k_0 + \frac{k}{d-k}\ell_0 \le k - \mathsf{residue}_k(d_1,\ldots,d_t)$, where $d_j = deg(Q_j)$.*

We want to spread the derivatives equally among all terms but cannot due to *integrality issues*. The residue captures this quantitatively and as described below, is what gives us our lower bounds. While the proof in [29] also relies on an integrality issue, there it originates from an imbalance between the sizes of the variable sets involved in a set-multilinear partition (as the map Π sets some variables in certain sets to 0). In contrast, we show that the integrality issue arising directly from the derivatives can be leveraged without involving set-multilinearity. In this sense, our approach is *conceptually* direct and simpler. Combined with the above discussion, we get the following structural lemma about the derivative space of $\prod_{j=1}^{t} Q_j$.

▶ **Lemma 8.**

$$\left\langle \boldsymbol{\partial}^k\left(Q_1\cdots Q_t\right)\right\rangle \subseteq \sum_{\substack{S\subseteq[t],\ k_0\in[0..k],\ \ell_0\in[0..(d-k)],\\ k_0+\frac{k}{d-k}\cdot\ell_0\ \le\ k-\mathsf{residue}_k(d_1,\ldots,d_t)}} \left\langle \mathbf{x}^{\ell_0}\cdot\boldsymbol{\partial}^{k_0}\left(\prod_{j\in S} Q_j\right)\right\rangle.$$

Now we have the choice to utilize the above structure using an additional set of linear maps. Both shifts and projections give similar lower bounds, so let us explain shifts here. Note that there is an intriguing possibility of getting even better lower bounds (in terms of dependence on d) using other sets of linear maps! From the above structural result, we have

$$\left\langle \mathbf{x}^{\ell}\cdot\boldsymbol{\partial}^k\left(Q_1\cdots Q_t\right)\right\rangle \subseteq \sum_{\substack{S\subseteq[t],\ k_0\in[0..k],\ \ell_0\in[0..(d-k)],\\ k_0+\frac{k}{d-k}\cdot\ell_0\ \le\ k-\mathsf{residue}_k(d_1,\ldots,d_t)}} \left\langle \mathbf{x}^{\ell+\ell_0}\cdot\boldsymbol{\partial}^{k_0}\left(\prod_{j\in S} Q_j\right)\right\rangle.$$

Thus we can upper bound,

$$\begin{aligned}\mathsf{SP}_{k,\ell}((Q_1\cdots Q_t)) &\le 2^t\cdot d^2\cdot \max_{\substack{k_0,\ell_0\ge 0\\ k_0+\frac{k}{d-k}\cdot\ell_0\ \le\ k-\mathsf{residue}_k(d_1,\ldots,d_t)}} M(n,k_0)\cdot M(n,\ell_0+\ell)\\ &\le 2^t\cdot d^2\frac{2^{O(d)}}{n^{\mathsf{residue}_k(d_1,\ldots,d_t)}}\ \min\{M(n,k)M(n,\ell), M(n,d-k+\ell)\},\end{aligned}$$

where the second inequality follows from elementary calculations.

Now to upper bound the shifted partial dimension of polynomials computed by low-depth formulas, we give a decomposition for such formulas into sums of products of polynomials (Lemma 16) where the degree sequences are carefully chosen so that that the residues can be simultaneously lower bounded for all the terms (Lemma 17). While in a different context, these calculations do bear similarity with related calculations in [29].

Step 3: Lower bounding $\dim(\mathcal{L}(f))$ for an explicit f. As a last step, one shows that for some explicit candidate hard polynomial $\dim(\mathcal{L}(f))$ is large and thereby obtains a lower bound. This is another step where bypassing set-multilinearity helps as one is not constrained

to pick a set-multilinear hard polynomial. Indeed, using a straightforward analysis we show that the APP measure is high for an explicit non-set-multilinear polynomial (see Remark 23). We also show that the measures are high for more standard polynomial families such as the iterated matrix multiplication polynomials and the Nisan-Wigderson design polynomials.

Application to UPT formulas. We observe here that for the subclass of homogeneous formulas that we call UPT formulas, one can do a depth-reduction to obtain a depth-4 formula in which all the summands have the same factorization pattern (i.e. the sequence of degrees of the factors in all the summands is that same) - see Lemma 30. We further observe (Lemma 31) that for any fixed sequence of degrees, there exists a suitable value of the parameter k such that the residue is *sufficiently large*. This gives us the superpolynomial lower bound for UPT formulas as stated in Theorem 5.

Despite the conceptual directness and simplicity of our approach, in bypassing set-multilinearity, some of the calculations in the analysis become evidently more involved than that in [29]. This is primarily due to the delicate choice of parameters in ratios involving binomial coefficients; this is also the case in several prior exponential lower bound proofs using SP and its variants [14, 16, 26]. Nevertheless, we think that by circumventing a critical bottleneck, the analysis opens up the possibility of an exponential lower bound for low-depth arithmetic circuits. Some of the ideas may indeed yield stronger bounds in the future.

Organization. After describing preliminaries in Section 2, we present a structural theorem about the derivative space of a product of homogeneous polynomials in Section 3. This result is then directly used to upper bound both the SP and APP measures of a product of polynomials. Using this result and a decomposition result for low-depth formulas, we obtain lower bounds for low-depth formulas in Section 4. Finally, we prove lower bounds for UPT formulas in Section 5.

2 Preliminaries

In this section, we give the essential notations and definitions necessary to follow the article.

Let a, b, c be real numbers. Then we define the sets $[a..b] := \{x \in \mathbb{Z} : x \in [a, b]\}$ and $[a] := [1..a]$. For a constant $c \geq 1$ and $b \geq 0$, we say $a \approx_c b$ if $a \in [b/c, b]$. We write $a \approx b$ if $a \approx_c b$ for some (unspecified) constant c. All logarithms have base 2 unless specified otherwise. We denote the fractional part of a by $\{a\} := a - \lfloor a \rfloor$ and the nearest integer of a by $\lfloor a \rceil$. The following quantity will be crucially used in the proofs of our lower bounds. Here we think of $d_1, \ldots, d_t$ as degrees of certain homogeneous polynomials, d as the degree of the product of those polynomials, and k is the order of partial derivatives used for the complexity measures.

▶ **Definition 9** (residue). *For non-negative integers $d_1, \ldots, d_t$ such that $d := \sum_{i=1}^{t} d_i \geq 1$ and $k \in [0..(d-1)]$, we define* $\mathsf{residue}_k(d_1, \ldots, d_t) := \frac{1}{2} \cdot \min_{k_1, \ldots, k_t \in \mathbb{Z}} \sum_{i=1}^{t} \left| k_i - \frac{k}{d} \cdot d_i \right|$.

The factor of half has been included in the definition just to make the statements of some of the lemmas in our analysis simple. It is easy to show that $\mathsf{residue}_k(d_1, \ldots, d_t) \leq \frac{k}{2}$. The minimum is attained when for all $i \in [t]$, $k_i = \left\lfloor \frac{k}{d} \cdot d_i \right\rceil$. When we use residue in the analysis of complexity measures, we would also have the following additional constraints that $k_i \geq 0$ and $k_i \leq d_i$, $k_1 + \cdots + k_n = k$, where k shall be the order of derivatives. As the value of residue can not decrease when we impose these constraints, we omit them.

Let n and n_0 be positive integers. Define variable sets $\mathbf{x} := \{x_1, \ldots, x_n\}$ and $\mathbf{z} := \{z_1, \ldots, z_{n_0}\}$. For a monic monomial m and a $P \in \mathbb{F}[\mathbf{x}]$, we define $\partial_m P \in \mathbb{F}[\mathbf{x}]$ to be the polynomial obtained by successively taking partial derivatives with respect to all the variables of m (counted with their multiplicities). For an integer $\ell \geq 0$, $\mathbf{x}^{\ell} := \{{x_1}^{e_1} \cdots {x_n}^{e_n} : e_1, \ldots, e_n \in \mathbb{Z}_{\geq 0} \text{ and } \sum_{i \in [n]} e_i = \ell\}$. For an integer $k \geq 0$ and $P \in \mathbb{F}[\mathbf{x}]$, $\boldsymbol{\partial}^k P := \left\{\partial_m P : m \in \mathbf{x}^k\right\}$. For a $P \in \mathbb{F}[\mathbf{x}]$, a map $L : \mathbf{x} \to \langle \mathbf{z} \rangle$, and $\mathcal{S} \subseteq \mathbb{F}[\mathbf{x}]$, $\pi_L(P) \in \mathbb{F}[\mathbf{z}]$ and $\pi_L(\mathcal{S}) \subseteq \mathbb{F}[\mathbf{z}]$ are defined as $\pi_L(P) := P(L(x_1), \ldots, L(x_n))$ and $\pi_L(\mathcal{S}) := \{\pi_L(P) : P \in \mathcal{S}\}$, respectively.

For $\mathcal{S}, \mathcal{T} \subseteq \mathbb{F}[\mathbf{x}]$, $\mathcal{S} \cdot \mathcal{T} := \{P \cdot Q : P \in \mathcal{S} \text{ and } Q \in \mathcal{T}\}$ and $\mathcal{S} + \mathcal{T} := \{P + Q : P \in \mathcal{S} \text{ and } Q \in \mathcal{T}\}$. For a $\mathcal{S} \subseteq \mathbb{F}[\mathbf{x}]$, we define its *span* as $\langle \mathcal{S} \rangle \subseteq \mathbb{F}[\mathbf{x}]$ to be the set of all polynomials which can be expressed as $\mathbb{F}$-linear combinations of elements in $\mathcal{S}$. For a $\mathcal{S} \subseteq \mathbb{F}[\mathbf{x}]$, its *dimension*, denoted by $\dim \mathcal{S}$, refers to the maximum number of *linearly independent* polynomials in $\mathcal{S}$. We can now define the complexity measures for polynomials that we use to prove our lower bounds: the *shifted partials* (SP) measure and the *affine projections of partials* (APP) measure.

▶ **Definition 10** (SP and APP measures). *For a polynomial $P \in \mathbb{F}[\mathbf{x}]$, non-negative integers k, ℓ, and $n_0 \in [n]$, we define $\mathsf{SP}_{k,\ell}(P) := \dim \left\langle \mathbf{x}^{\ell} \cdot \boldsymbol{\partial}^k P \right\rangle$ and $\mathsf{APP}_{k,n_0}(P) := \max_{L:\mathbf{x} \to \langle \mathbf{z} \rangle} \dim \left\langle \pi_L\left(\boldsymbol{\partial}^k P\right) \right\rangle$.*

SP and APP are *sub-additive*. APP is related to the *skewed partials* and relrk measures used in [15] and [29], respectively. For a comparison, see [2].

Next, we define a subclass of homogeneous formulas which we call *UPT formulas*[3].

▶ **Definition 11.** *A homogeneous formula C is said to be a* unique-parse-tree formula *if all of its parse trees are isomorphic to each other as directed graphs.*

For a UPT formula C, we define its *canonical parse tree* to be some fixed tree among all the parse trees (this is a binary tree without loss of generality). For a detailed definition of (canonical) parse tree, we refer the reader to the full version of this article [2].

Iterated Matrix Multiplication. The iterated matrix multiplication, $IMM_{n,d}$ is a polynomial in $N = d \cdot n^2$ variables defined as the $(1,1)$-th entry of the matrix product of d many $n \times n$ matrices whose entries are distinct variables. To prove a lower bound for IMM, we analyze the SP and APP for a projection of IMM, $P_{\mathbf{w}}$ that was introduced in [29].

▶ **Definition 12** (Word polynomial $P_{\mathbf{w}}$ [29]). *Given a word $\mathbf{w} = (w_1, \ldots, w_d) \in \mathbb{Z}^d$, let $\mathbf{x}(\mathbf{w})$ be a tuple of d pairwise disjoint sets of variables $(\mathbf{x}_1(\mathbf{w}), \ldots, \mathbf{x}_d(\mathbf{w}))$ with $|\mathbf{x}_i(\mathbf{w})| = 2^{|w_i|}$ for all $i \in [d]$. $\mathbf{x}_i(\mathbf{w})$ will be called negative if $w_i < 0$ and positive otherwise. As the set sizes are powers of 2, we can map the variables in a set $\mathbf{x}_i(\mathbf{w})$ to Boolean strings of length $|w_i|$. Let $\sigma : \mathbf{x} \to \{0,1\}^*$ be such a mapping.*[4] *We extend the definition of σ from variables to set-multilinear monomials as follows: Let $X = x_1 \cdots x_r$ be a set-multilinear monomial where $x_i \in \mathbf{x}_{\phi(i)}(\mathbf{w})$ and $\phi : [r] \to [d]$ be an increasing function. Then, we define a Boolean string $\sigma(X) := \sigma(x_1) \circ \cdots \circ \sigma(x_r)$, where $\circ$ denotes the concatenation of bits. Let $\mathcal{M}_+(\mathbf{w})$ and $\mathcal{M}_-(\mathbf{w})$ denote the set of all (monic) set-multilinear monomials over all the positive sets*

[3] Our definition for UPT formulas is more general than the model considered in a recent paper by Limaye, Srinivasan and Tavenas [30] as we do not impose set-multilinearity.

[4] Note that σ may map a variable from $\mathbf{x}_i(\mathbf{w})$ and a variable from $\mathbf{x}_j(\mathbf{w})$ to the same string if $i \neq j$.

and all the negative sets, respectively. For two Boolean strings a, b, we say $a \sim b$ if a is a prefix of b or vice versa. For a word $\mathbf{w}$, the corresponding word polynomial $P_{\mathbf{w}}$ is defined as $P_{\mathbf{w}} := \sum_{\substack{m_+ \in \mathcal{M}_+(\mathbf{w}),\ m_- \in \mathcal{M}_-(\mathbf{w}) \\ \sigma(m_+) \sim \sigma(m_-)}} m_+ \cdot m_-$.

We will make use of the following lemma from [29] which shows that computing IMM is at least as hard as computing $P_{\mathbf{w}}$. For this, we recall the notion of *unbiased*-ness of $\mathbf{w} = (w_1, \ldots, w_d)$ from [29] – we say that $\mathbf{w}$ is *h-unbiased* if $\max_{i \in [d]} |w_1 + \cdots + w_i| \leq h$.

▶ **Lemma 13** (Lemma 7 in [29]). *Let $\mathbf{w} \in [-h..h]^d$ be h-unbiased. If for some $n \geq 2^h$, $IMM_{n,d}$ has a formula C of product-depth[5] Δ and size s, then $P_{\mathbf{w}}$ has a formula C' of product-depth at most Δ and size at most s. Moreover, if C is homogeneous, then so is C' and if C is UPT, then so is C' with the same canonical parse tree.*[6]

Nisan-Wigderson design polynomial. For a prime power q and $d \in \mathbb{N}$, let $\mathbf{x} = \{x_{1,1}, \ldots, x_{1,q}, \ldots, x_{d,1}, \ldots, x_{d,q}\}$. For any $k \in [d]$, the Nisan-Wigderson design polynomial on qd variables, denoted by $NW_{q,d,k}$ or simply NW, is defined as follows:

$$NW_{q,d,k} = \sum_{\substack{h(z) \in \mathbb{F}_q[z]: \\ \deg(h) < k}} \prod_{i \in [d]} x_{i,h(i)}.$$

The IMM and the NW polynomials, and their variants, have been extensively used to prove various circuit lower bounds [3, 4, 11, 14, 16, 19–21, 23, 26, 27, 29, 32].

3 Structure of the space of partials of a product

In this section, we bound the partial derivative space of a product of homogeneous polynomials. In the following lemma, we show that the space of k-th order partial derivatives of a product of polynomials is contained in a sum of shifted partial spaces with shift ℓ_0 and order of derivatives k_0 such that $k_0 + \frac{k}{d-k} \cdot \ell_0$ is "small". Using this lemma, we upper bound the SP and APP measures of a product of homogeneous polynomials. These bounds are then used in Sections 4 and 5 for proving lower bounds for low-depth homogeneous formulas and UPT formulas respectively. Missing proofs from this section can be found in the full version of this article [2],

▶ **Lemma 14** (Upper bounding the partials of a product). *Let n and t be positive integers and $Q_1, \ldots, Q_t$ be non-constant, homogeneous polynomials in $\mathbb{F}[\mathbf{x}]$ with degrees $d_1, \ldots, d_t$ respectively. Let $d := \deg(Q_1 \cdots Q_t) = \sum_{i=1}^{t} d_i$ and $k < d$ be a non-negative integer. Then,*

$$\left\langle \boldsymbol{\partial}^k (Q_1 \cdots Q_t) \right\rangle \subseteq \sum_{\substack{S \subseteq [t],\ k_0 \in [0..k],\ \ell_0 \in [0..(d-k)], \\ k_0 + \frac{k}{d-k} \cdot \ell_0 \ \leq\ k - \mathsf{residue}_k(d_1, \ldots, d_t)}} \left\langle \mathbf{x}^{\ell_0} \cdot \boldsymbol{\partial}^{k_0} \left(\prod_{i \in S} Q_i \right) \right\rangle.$$

We now use the above lemma to upper bound the shifted partials and affine projections of partials measures of a product of polynomials.

[5] The product-depth of a formula is the maximum number of product gates on any path from the root to a leaf in the formula.

[6] Although the lemma in [29] is stated for set-multilinear circuits, it also applies to homogeneous formulas and UPT formulas (albeit with a mild blow-up in size) by the same argument.

▶ **Lemma 15** (Upper bounding SP and APP of a product). *Let $Q = Q_1 \cdots Q_t$ be a homogeneous polynomial in $\mathbb{F}[x_1, \ldots, x_n]$ of degree $d = d_1 + \cdots + d_t \geq 1$, where Q_i is homogeneous and $d_i := \deg(Q_i)$ for $i \in [t]$. Then, for any non-negative integers $k < d$, $\ell \geq 0$, and $n_0 \leq n$,*

1.
$$\mathsf{SP}_{k,\ell}(Q) \leq 2^t \cdot d^2 \cdot \max_{\substack{k_0,\ell_0 \geq 0 \\ k_0 + \frac{k}{d-k}\cdot\ell_0 \;\leq\; k-\mathsf{residue}_k(d_1,\ldots,d_t)}} M(n,k_0) \cdot M(n,\ell_0+\ell),$$

2.
$$\mathsf{APP}_{k,n_0}(Q) \leq 2^t \cdot d^2 \cdot \max_{\substack{k_0,\ell_0 \geq 0 \\ k_0 + \frac{k}{d-k}\cdot\ell_0 \;\leq\; k-\mathsf{residue}_k(d_1,\ldots,d_t)}} M(n,k_0) \cdot M(n_0,\ell_0).$$

4 Lower bound for low-depth homogeneous formulas

In this section, we present a superpolynomial lower bound for "low-depth" homogeneous formulas computing the IMM and NW polynomials. We begin by proving a structural result for homogeneous formulas. Missing proofs from this section can be found in the full version of this article [2].

4.1 Decomposition of low-depth formulas

We show that any homogeneous formula can be decomposed as a sum of products of homogeneous polynomials of lower degrees, where the number of summands is bounded by the number of gates in the original formula. The decomposition lemma given below bears some resemblance to a decomposition of homogeneous formulas in [12]. In the decomposition in [12], the degrees of the factors of every summand roughly form a geometric sequence, and hence each summand is a product of a "large" number of factors. Here we show that each summand has "many" low-degree factors. While the lower bound argument in [29] does not explicitly make use of such a decomposition, their inductive argument can be formulated as a depth-reduction or decomposition lemma (with slightly different thresholds for the degrees).

▶ **Lemma 16** (Decomposition of low-depth formulas). *Suppose C is a homogeneous formula of product-depth $\Delta \geq 1$ computing a homogeneous polynomial in $\mathbb{F}[x_1, \ldots, x_n]$ of degree* at least *$d > 0$. Then, there exist homogeneous polynomials $\{Q_{i,j}\}_{i,j}$ in $\mathbb{F}[x_1, \ldots, x_n]$ such that*

1. *$C = \sum_{i=1}^{s} Q_{i,1} \cdots Q_{i,t_i}$, for some $s \leq \mathrm{size}(C)$, and*
2. *for all $i \in [s]$, either*

$$|\{j \in [t_i] : \deg(Q_{i,j}) = 1\}| \geq d^{2^{1-\Delta}} \text{ , or}$$

$$\left|\left\{j \in [t_i] : \deg(Q_{i,j}) \approx_2 d^{2^{1-\delta}}\right\}\right| \geq d^{2^{1-\delta}} - 1 \text{ , for some } \delta \in [2..\Delta].$$

4.2 Low-depth formulas have high residue

The following lemma gives us a value for the order of derivatives k with respect to which low-depth formulas yield high residue. Its proof uses Lemma 16.

▶ **Lemma 17** (Low-depth formulas have high residue). *Suppose C is a homogeneous formula of product-depth $\Delta \geq 1$ computing a polynomial in $\mathbb{F}[x_1, \ldots, x_n]$ of degree d, where $d^{2^{1-\Delta}} = \omega(1)$. Then, there exist homogeneous polynomials $\{Q_{i,j}\}_{i,j}$ in $\mathbb{F}[x_1, \ldots, x_n]$ such that $C = \sum_{i=1}^{s} Q_{i,1} \cdots Q_{i,t_i}$, for some $s \leq \text{size}(C)$. Fixing an arbitrary $i \in [s]$, let $t := t_i$ and define $d_j := \deg(Q_{i,j})$ for $j \in [t]$. Then, $\mathsf{residue}_k(d_1, \ldots, d_t) \geq \Omega\left(d^{2^{1-\Delta}}\right)$, where $k := \left\lfloor \frac{\alpha \cdot d}{1+\alpha} \right\rfloor$, $\alpha := \sum_{\nu=0}^{\Delta-1} \frac{(-1)^\nu}{\tau^{2^\nu - 1}}$, and $\tau := \left\lfloor d^{2^{1-\Delta}} \right\rfloor$.*

4.3 High residue implies lower bounds

For a "random" homogeneous degree-d polynomial in $\mathbb{F}[x_1, \ldots, x_n]$, if the shift ℓ is not too large, we expect the SP measure to be close to the maximum number of operators used to construct the shifted partials space, i.e., $M(n,k) \cdot M(n,\ell)$. Explicit examples of such polynomials are given in Section 4.4. In the lemma below, we derive a lower bound corresponding to the decompositions established above. The main step is to show that the SP measure of a high-residue-decomposition is small.

▶ **Lemma 18** (High residue implies lower bounds). *Let $P = \sum_{i=1}^{s} Q_{i,1} \cdots Q_{i,t_i}$ be a homogeneous n-variate polynomial of degree d where $\{Q_{i,j}\}_{i,j}$ are homogeneous and $\mathsf{SP}_{k,\ell}(P) \geq 2^{-O(d)} \cdot M(n,k) \cdot M(n,\ell)$ for some $1 \leq k < \frac{d}{2}, n_0 \leq n$ and $\ell = \left\lfloor \frac{n \cdot d}{n_0} \right\rfloor$ such that $d \leq n_0 \approx 2(d-k) \cdot \left(\frac{n}{k}\right)^{\frac{k}{d-k}}$. If there is a $\gamma > 0$ such that for all $i \in [s]$, $\mathsf{residue}_k(\deg(Q_{i,1}), \ldots, \deg(Q_{i,t_i})) \geq \gamma$, then $s \geq 2^{-O(d)} \left(\frac{n}{d}\right)^{\Omega(\gamma)}$.*

We state an analogous lemma with APP instead of SP.

▶ **Lemma 19** (High residue implies lower bounds, using APP). *Let $P = \sum_{i=1}^{s} Q_{i,1} \cdots Q_{i,t_i}$ be a homogeneous n-variate polynomial of degree d where $\{Q_{i,j}\}_{i,j}$ are homogeneous and $\mathsf{APP}_{k,n_0}(P) \geq 2^{-O(d)} \cdot M(n,k)$ for some $1 \leq k < \frac{d}{2}, n_0 \leq n$ such that $d \leq n_0 \approx 2(d-k).\left(\frac{n}{k}\right)^{\frac{k}{d-k}}$. If there is a $\gamma > 0$ such that for all $i \in [s]$, $\mathsf{residue}_k(\deg(Q_{i,1}), \ldots, \deg(Q_{i,t_i})) \geq \gamma$, then $s \geq 2^{-O(d)} \cdot \left(\frac{n}{d}\right)^{\Omega(\gamma)}$.*

▶ Remark 20. In the above lemmas, although our lower bound appears as $2^{-O(d)} \cdot n^{\Omega(\gamma)}$, similar calculations actually give a lower bound of $2^{-O(k)} \cdot n^{\Omega(\gamma)}$ for any choice of k and an appropriate choice of ℓ (or n_0 in the case of APP). We do not differentiate between the two, as for our applications (i.e., low-depth circuits and UPT formulas), the value of k we choose is $\Theta(d)$. Moreover, we observe that the factor of $2^{-O(k)}$ in our lower bounds is likely unavoidable for any choice of k and ℓ (or n_0 in the case of APP) *using our current estimates for the complexity measures*. We refer the reader to the full version of this article [2] for more details.

4.4 The hard polynomials

We shall prove our lower bound for the word polynomial $P_{\mathbf{w}}$ introduced in [29] as well as for the Nisan-Wigderson design polynomial. In order to do this, we show that the SP and APP measures of $P_{\mathbf{w}}$ and the SP measure of NW are large for suitable choices of k, ℓ and n_0.

▶ **Lemma 21** ($P_{\mathbf{w}}$ as a hard polynomial). *For integers h, d such that $h > 100$ and any $k \in \left[\frac{d}{30}, \frac{d}{2}\right]$, there exists an h-*unbiased *word $\mathbf{w} \in [-h..h]^d$, integers $n_0 \leq n$, $\ell = \left\lfloor \frac{n \cdot d}{n_0} \right\rfloor$ such that $n_0 \approx 2(d-k) \cdot \left(\frac{n}{k}\right)^{\frac{k}{d-k}}$ and the following bounds hold: $\mathsf{SP}_{k,\ell}(P_{\mathbf{w}}) \geq 2^{-O(d)} \cdot M(n,k) \cdot M(n,\ell)$ and $\mathsf{APP}_{k,n_0}(P_{\mathbf{w}}) \geq 2^{-O(d)} \cdot M(n,k)$. Here n refers to the number of variables in $P_{\mathbf{w}}$, i.e., $n = \sum_{i \in [d]} 2^{|w_i|}$.*

The following lemma shows that the SP measure of the Nisan-Wigderson design polynomial is "large" for k as high as $\Theta(d)$, if ℓ is chosen suitably.

▶ **Lemma 22** (NW as a hard polynomial). *For $n, d \in \mathbb{N}$ such that $120 \leq d \leq \frac{1}{150}\left(\frac{\log n}{\log\log n}\right)^2$, let q be the largest prime number between $\left\lfloor \frac{n}{2d} \right\rfloor$ and $\left\lfloor \frac{n}{d} \right\rfloor$. For parameters $k \in \left[\frac{d}{30}, \frac{d}{2} - \frac{\sqrt{d}}{8}\right]$ and $\ell = \left\lfloor \frac{qd^2}{n_0} \right\rfloor$, where $n_0 = 2(d-k) \cdot \left(\frac{qd}{k}\right)^{\frac{k}{d-k}}$, $\mathsf{SP}_{k,\ell}(NW_{q,d,k}) \geq 2^{-O(d)} \cdot M(qd,k) \cdot M(qd,\ell)$.*

▶ Remark 23. An advantage of directly analysing the complexity measures for homogeneous formulas instead of for set-multilinear formulas is that our hard polynomial need not be set multilinear. In the full version of this article [2], we describe an explicit non set-multilinear polynomial P (in VNP) with a large APP measure; the construction is similar to a polynomial in [7]. The proof that APP of P is large is considerably simpler than the proofs of the above lemmas.

4.5 Putting everything together: the low-depth lower bound

▶ **Theorem 24** (Low-depth homogeneous formula lower bound for IMM). *For any d, n, Δ such that $n = \omega(d)$, any homogeneous formula of product-depth at most Δ computing $IMM_{n,d}$ over any field $\mathbb{F}$ has size at least $2^{-O(d)} \cdot n^{\Omega\left(d^{2^{1-\Delta}}\right)}$. In particular, when $d = O(\log n)$, we get a lower bound of $n^{\Omega\left(d^{2^{1-\Delta}}\right)}$.*

▶ **Theorem 25** (Low-depth homogeneous formula lower bound for NW). *Let n, d, Δ be positive integers. If $\Delta = 1$, let $d = n^{1-\epsilon}$ for any constant $\epsilon > 0$ and $k = \left\lfloor \frac{d-1}{2} \right\rfloor$. Otherwise, let $d \leq \frac{1}{150}\left(\frac{\log n}{\log\log n}\right)^2$, let $\tau = \left\lfloor d^{2^{1-\Delta}} \right\rfloor$, $\alpha = \sum_{\nu=0}^{\Delta-1} \frac{(-1)^\nu}{\tau^{2^\nu - 1}}$, and $k = \left\lfloor \frac{\alpha \cdot d}{1+\alpha} \right\rfloor$. In both cases, let q be the largest prime between $\left\lfloor \frac{n}{2d} \right\rfloor$ and $\left\lfloor \frac{n}{d} \right\rfloor$. Then, any homogeneous formula of product-depth at most Δ computing $NW_{q,d,k}$ over any field $\mathbb{F}$ has size at least $2^{-O(d)} \cdot n^{\Omega\left(d^{2^{1-\Delta}}\right)}$. In particular, when $d = O(\log n)$, we get a lower bound of $n^{\Omega\left(d^{2^{1-\Delta}}\right)}$.*

▶ Remark 26. Notice that in the above theorem, as k depends on the product-depth Δ, the polynomial $NW_{q,d,k}$ may be different for different values of Δ. However, much like in [19], there is a way to "stitch" all the different NW polynomials for different values of Δ into a single polynomial P such that any homogeneous formula of product-depth Δ computing P has size at least $2^{-O(d)} n^{\Omega\left(d^{2^{1-\Delta}}\right)}$. See Theorem 34 for more details.

In [29], the authors showed how to convert a circuit of product-depth Δ computing a homogeneous polynomial to a homogeneous formula of product-depth 2Δ without much increase in the size. Combining Lemma 11 from [29] with Theorems 24 and 25, we get:

▶ **Corollary 27** (Low-depth circuit lower bound for IMM). *For any positive integers d, n, Δ such that $n = \omega(d)$, any circuit of product-depth at most Δ computing $IMM_{n,d}$ over any field $\mathbb{F}$ with characteristic 0 or more than d has size at least $2^{-O(d)} \cdot n^{\Omega\left(\frac{d^{2^{1-2\Delta}}}{\Delta}\right)}$.*

In particular, when $d = O(\log n)$, we get a lower bound of $n^{\Omega\left(\frac{d^{2^{1-2\Delta}}}{\Delta}\right)}$.

▶ **Corollary 28** (Low-depth circuit lower bound for NW). *Let n, d, Δ be positive integers. If $\Delta = 1$, let $d = n^{1-\epsilon}$ for any constant $\epsilon > 0$ and $k = \left\lfloor \frac{d-1}{2} \right\rfloor$. Otherwise, let $d \leq \frac{1}{150}\left(\frac{\log n}{\log\log n}\right)^2$, let $\tau = \left\lfloor d^{2^{1-\Delta}} \right\rfloor$, $\alpha = \sum_{\nu=0}^{\Delta-1} \frac{(-1)^\nu}{\tau^{2^\nu - 1}}$, and $k = \left\lfloor \frac{\alpha \cdot d}{1+\alpha} \right\rfloor$. In both cases, let q be the largest prime number between $\left\lfloor \frac{n}{2d} \right\rfloor$ and $\left\lfloor \frac{n}{d} \right\rfloor$. Then, any circuit of product-depth at most Δ computing $NW_{q,d,k}$ over any field $\mathbb{F}$ of characteristic 0 or more than d has size at least $2^{-O(d)} \cdot n^{\Omega\left(\frac{d^{2^{1-2\Delta}}}{\Delta}\right)}$.*

In particular, when $d = O(\log n)$, we get a lower bound of $n^{\Omega\left(\frac{d^{2^{1-2\Delta}}}{\Delta}\right)}$.

We note that our lower bounds quantitatively improve on the original homogeneous formula lower bound of [29] in terms of the dependence on the degree. While [29] gives a lower bound of $d^{O(-d)} \cdot n^{\Omega\left(d^{1/2^\Delta - 1}\right)}$ (as the conversion from homogeneous to set-multilinear formulas increases the size by a factor of $d^{O(d)}$), our lower bound is $2^{-O(d)} \cdot n^{\Omega\left(d^{2^{1-\Delta}}\right)}$. Thus, we get slight improvement both in the multiplicative factor (from $d^{O(d)}$ to $2^{O(d)}$) and in the exponent of n (from $d^{\frac{1}{2^\Delta - 1}}$ to $d^{\frac{1}{2^{(\Delta-1)}}}$). We point out what these improvements mean for smaller depths: For $\Delta = 2$, our lower bound for homogeneous formulas computing IMM is superpolynomial as long as $d \leq \epsilon \cdot \log^2 n$ for a small enough positive constant ϵ, whereas the lower bound in [29] does not work beyond $d = O\left(\left(\frac{\log n}{\log\log n}\right)^2\right)$. In particular, we obtain a lower bound of $n^{\Omega(\log n)}$ for the size of homogeneous depth-5 formulas computing $IMM_{n,d}$ when $d = \Theta(\log^2 n)$. Finally, for $\Delta = 3$ and $d \leq \epsilon \cdot \log^{4/3} n$, we get a lower bound of $n^{\Omega\left(d^{1/4}\right)}$, as compared to $n^{\Omega\left(d^{1/7}\right)}$ from [29].

5 Lower bound for unique-parse-tree formulas

In this section, we show that UPT formulas computing IMM must have a "large" size. We begin by giving a decomposition for such formulas. Missing proofs from this section can be found in the full version of this article [2].

5.1 Decomposition of UPT formulas

In order to upper bound the SP (or APP) measure of a UPT formula, we need certain results about binary trees and UPT formulas. For a given canonical parse tree $\mathcal{T}$ with d leaves, we define its *degree sequence* $(d_1, \ldots, d_t)$ using the function Deg-seq described in Algorithm 1.

We prove the following lemma in the full version of this article [2]. The idea here is to "break" the tree at various nodes so that the successive sizes of the smaller trees are far from each other.

Algorithm 1 Degree sequence of a right-heavy binary tree.

```
function DEG-SEQ(𝒯)
    v_0 ← root node of 𝒯.
    if v_0 is a leaf then
        return (1).
    end if
    d ← leaves(v_0), i ← 0.
    while v_i is not a leaf do
        v_{i+1} ← right child of v_i, i ← i + 1.
    end while
    v ← v_j corresponding to the largest index j such that leaves(v_j) > d/3.
    d_1 ← d − leaves(v).
    return (d_1, DEG-SEQ(𝒯_v)).
end function
```

▶ **Lemma 29.** *For a given canonical parse tree $\mathcal{T}$ with $d \geq 1$ leaves, let $(d_1, \ldots, d_t) :=$ DEG-SEQ$(\mathcal{T})$, where the function* DEG-SEQ *is given in Algorithm 1. Also let $e_i := d - \sum_{j=1}^{i} d_j$ for $i \in [t]$ and $e_0 := d$. Then, for all $i \in [t-1]$, $e_i \in \left(\frac{e_{i-1}}{3}, \frac{2 \cdot e_{i-1}}{3}\right]$. Additionally, $d_t = 1$, $e_t = 0$, and $\log_3 d + 1 \leq t \leq \log_{3/2} d + 1$.*

As mentioned in Section 4.1, it was shown in [12] that a homogeneous formula can be expressed as a "small" sum of products of homogeneous polynomials such that in each summand, the degrees of the factors roughly form a geometric sequence. We observe that this result can be strengthened for UPT formulas; in particular, we show that for UPT formulas, the "degree sequences" of all the summands are identical.

▶ **Lemma 30** (Log-product decomposition of UPT formulas)**.** *Let $f \in \mathbb{F}[\mathbf{x}]$ be a homogeneous polynomial of degree $d \geq 1$ computed by a UPT formula C with canonical parse tree $\mathcal{T}(C)$. Let $(d_1, \ldots, d_t) :=$ DEG-SEQ$(\mathcal{T}(C))$. Then there exist an integer $s \leq \text{size}(C)$ and homogeneous polynomials $\{Q_{i,j}\}_{i,j}$ where $\deg(Q_{i,j}) = d_j$ for $i \in [s]$, $j \in [t]$, such that*

$$f = \sum_{i=1}^{s} Q_{i,1} \cdots Q_{i,t}.$$

5.2 UPT formulas have high residue

Now we show that there exists a value of k that has high residue with respect to the degrees of the factors given by the above log-product lemma.

▶ **Lemma 31** (High residue for a degree sequence)**.** *For any given canonical parse tree $\mathcal{T}$ with $d \geq 1$ leaves, let $(d_1, \ldots, d_t) :=$ DEG-SEQ$(\mathcal{T})$ and $k :=$ UPT-K$(d_1, \ldots, d_t)$ where the function* UPT-K *is described in Algorithm 2. Then*

$$\text{residue}_k(d_1, \ldots, d_t) \geq \frac{\log_3 d - 10}{216}.$$

Algorithm 2 The value of k for a given sequence of degrees.

function UPT-K$(d_1, \ldots, d_t)$
  /* Returns k which shall be the order of derivatives for the SP and APP measures. */
  $d = d_1 + \cdots + d_t$.
  for $i \in [0..t]$ **do**
    $e_i \leftarrow d - \sum_{j=1}^{i} d_j$.
  end for
  $m \leftarrow \left\lfloor \frac{\log_3 d - 1}{3} \right\rfloor$.
  /* Defining a function $\mathcal{J} : [3m] \to [t-2]$. */
  for $i \in [3m]$ **do**
    $\mathcal{J}(i) \leftarrow \min\left\{ j \in [0..t] : e_j \leq 3^i \right\}$.
  end for
  $(a_1, \ldots, a_m) \leftarrow$ *undefined*.
  for $i \in [m]$ **do**
    $j \leftarrow \mathcal{J}(3i)$.
    $b_0 \leftarrow \left(\sum_{p=1}^{i-1} \frac{a_p}{3^{3p}} \right) \cdot d_{j+1}$.
  /* b_1 defined below is not used in the algorithm but will be useful in the analysis. */
    $b_1 \leftarrow \left(\sum_{p=1}^{i-1} \frac{a_p}{3^{3p}} + \frac{1}{3^{3i}} \right) \cdot d_{j+1}$.
    if $\{b_0\} \in \left[\frac{1}{18}, \frac{17}{18}\right]$ **then**
      $a_i \leftarrow 0$.
    else
      $a_i \leftarrow 1$.
    end if
  end for
  $\alpha \leftarrow \sum_{p=1}^{m} \frac{a_p}{3^{3p}}$
  $k \leftarrow \lfloor \alpha \cdot d \rfloor$
  return k.
end function

5.3 Putting everything together: the UPT formula lower bound

In this section, we state our lower bounds for UPT formulas.

▶ **Theorem 32** (UPT formula lower bound for IMM)**.** *For $n \in \mathbb{N}$ and $d \leq \epsilon \cdot \log n \cdot \log\log n$, where $\epsilon > 0$ is a small enough constant, any UPT formula computing $IMM_{n,d}$ over any field $\mathbb{F}$ has size $n^{\Omega(\log d)}$.*

▶ Remark 33. The above theorem can also be derived by using the complexity measure studied in [29] along with the observation that the *unbounded-depth* set-multilinearization due to [35] (which increases the size by a factor of $2^{O(d)}$) preserves parse trees.

We also get an analogous theorem for a polynomial related to the NW polynomial.

▶ **Theorem 34.** *Let $n \in \mathbb{N}$, $d \leq \epsilon \cdot \log n \cdot \log\log n$, where $\epsilon > 0$ is a small enough constant, and q be the largest prime number between $\left\lfloor \frac{n}{2d} \right\rfloor$ and $\left\lfloor \frac{n}{d} \right\rfloor$. Then, any UPT formula computing $P = \sum_{i=\lfloor d/30 \rfloor}^{\lceil d/2 \rceil} y_i \cdot NW_{q,d,i}$ (where the y variables are distinct from the $\mathbf{x}$ variables), over any field $\mathbb{F}$ has size $n^{\Omega(\log d)}$.*

6 Conclusion

Recently, [29] made remarkable progress on arithmetic circuit lower bounds by giving the first super-polynomial lower bound for low-depth formulas. They achieve this by a hardness escalation approach via set-multilinearization. But, set-multilinearization is an inherently expensive process that seems to restrict us from obtaining an exponential lower bound for even homogeneous low-depth formulas. In this work, we take the vital first step of sidestepping set-multilinearization and showing a super-polynomial lower bound for low-depth formulas via a direct approach. A direct approach does not seem to incur an *inherent* exponential loss. So, it might be possible to prove stronger lower bounds for low-depth homogeneous formulas or other related models using this approach or an adaptation of it.

Problem 1. Prove exponential lower bounds for low-depth homogeneous arithmetic formulas. Prove exponential lower bounds for low-depth, *multi-r-ic* formulas.

A formula is said to be multi-r-ic, if the formal degree of every gate with respect to every variable is at most r [17, 21]. The UPT formula lower bound proved in this work is for formulas computing polynomials of degree at most $O(\log n \cdot \log\log n)$. It would be interesting to increase the range of degrees for which our bound works. In the non-commutative setting, exponential lower bounds are known for formulas with exponentially many parse trees [28].

Problem 2. Prove an $n^{\Omega(\log d)}$ lower bound for UPT formulas for $d = n^{O(1)}$. Prove a superpolynomial lower bound for formulas with "many" parse trees.

Our work also raises the prospect of learning low-depth homogeneous formulas given black-box access using the "learning from lower bounds" paradigm proposed in [7, 18].

Problem 3. Obtain learning algorithms for random low-depth homogeneous formulas.

To upper bound SP or APP of a homogeneous formula C, we first show in Section 3 that the space of partial derivatives of C has some structure and then exploit this structure using shifts or affine projections. There might be a better way to exploit this structure, say by going modulo an appropriately chosen ideal or using random restrictions along with shifts as done in [14, 26]. Exploring this possibility is also an interesting direction for future work.

References

1 Manindra Agrawal and V. Vinay. Arithmetic Circuits: A Chasm at Depth Four. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 67–75. IEEE Computer Society, 2008.

2 Prashanth Amireddy, Ankit Garg, Neeraj Kayal, Chandan Saha, and Bhargav Thankey. Low-depth arithmetic circuit lower bounds via shifted partials. *Electron. Colloquium Comput. Complex.*, TR22-151, 2022. URL: `https://eccc.weizmann.ac.il/report/2022/151`.

3 Suryajith Chillara, Nutan Limaye, and Srikanth Srinivasan. Small-depth multilinear formula lower bounds for iterated matrix multiplication with applications. *SIAM J. Comput.*, 48(1):70–92, 2019. Conference version appeared in the proceedings of STACS 2018. `doi:10.1137/18M1191567`.

4 Michael A. Forbes, Mrinal Kumar, and Ramprasad Saptharishi. Functional lower bounds for arithmetic circuits and connections to boolean circuit complexity. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 33:1–33:19, 2016.

5 Lance Fortnow and Adam R. Klivans. Efficient learning algorithms yield circuit lower bounds. *J. Comput. Syst. Sci.*, 75(1):27–36, 2009.

6 Hervé Fournier, Nutan Limaye, Guillaume Malod, and Srikanth Srinivasan. Lower Bounds for Depth-4 Formulas Computing Iterated Matrix Multiplication. *SIAM J. Comput.*, 44(5):1173–1201, 2015. Conference version appeared in the proceedings of STOC 2014.

7 Ankit Garg, Neeraj Kayal, and Chandan Saha. Learning sums of powers of low-degree polynomials in the non-degenerate case. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 889–899. IEEE, 2020.

8 Fulvio Gesmundo and Joseph M. Landsberg. Explicit polynomial sequences with maximal spaces of partial derivatives and a question of k. mulmuley. *Theory Comput.*, 15:1–24, 2019. doi:10.4086/toc.2019.v015a003.

9 Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Approaching the Chasm at Depth Four. *J. ACM*, 61(6):33:1–33:16, 2014. Conference version appeared in the proceedings of CCC 2013.

10 Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Arithmetic Circuits: A Chasm at Depth 3. *SIAM J. Comput.*, 45(3):1064–1079, 2016. Conference version appeared in the proceedings of FOCS 2013.

11 Nikhil Gupta, Chandan Saha, and Bhargav Thankey. A super-quadratic lower bound for depth four arithmetic circuits. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 23:1–23:31. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CCC.2020.23.

12 Pavel Hrubes and Amir Yehudayoff. Homogeneous formulas and symmetric polynomials. *Comput. Complex.*, 20(3):559–578, 2011. doi:10.1007/s00037-011-0007-3.

13 Neeraj Kayal. An exponential lower bound for the sum of powers of bounded degree polynomials. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:81, 2012. URL: http://eccc.hpi-web.de/report/2012/081.

14 Neeraj Kayal, Nutan Limaye, Chandan Saha, and Srikanth Srinivasan. An Exponential Lower Bound for Homogeneous Depth Four Arithmetic Formulas. *SIAM J. Comput.*, 46(1):307–335, 2017. Conference version appeared in the proceedings of FOCS 2014.

15 Neeraj Kayal, Vineet Nair, and Chandan Saha. Separation Between Read-once Oblivious Algebraic Branching Programs (ROABPs) and Multilinear Depth-three Circuits. *ACM Trans. Comput. Theory*, 12(1):2:1–2:27, 2020. Conference version appeared in the proceedings of STACS 2016.

16 Neeraj Kayal and Chandan Saha. Lower Bounds for Depth-Three Arithmetic Circuits with small bottom fanin. *Computational Complexity*, 25(2):419–454, 2016. Conference version appeared in the proceedings of CCC 2015.

17 Neeraj Kayal and Chandan Saha. Multi-k-ic depth three circuit lower bound. *Theory Comput. Syst.*, 61(4):1237–1251, 2017. The conference version appeared in the proceedings of STACS, 2015. doi:10.1007/s00224-016-9742-9.

18 Neeraj Kayal and Chandan Saha. Reconstruction of non-degenerate homogeneous depth three circuits. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 413–424. ACM, 2019.

19 Neeraj Kayal, Chandan Saha, and Ramprasad Saptharishi. A super-polynomial lower bound for regular arithmetic formulas. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 146–153, 2014.

20 Neeraj Kayal, Chandan Saha, and Sébastien Tavenas. An Almost Cubic Lower Bound for Depth Three Arithmetic Circuits. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 33:1–33:15, 2016.

21 Neeraj Kayal, Chandan Saha, and Sébastien Tavenas. On the size of homogeneous and of depth four formulas with low individual degree. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 626–632, 2016.

22 Pascal Koiran. Arithmetic circuits: The chasm at depth four gets wider. *Theor. Comput. Sci.*, 448:56–65, 2012.

23 Mrinal Kumar and Ramprasad Saptharishi. The computational power of depth five arithmetic circuits. *SIAM J. Comput.*, 48(1):144–180, 2019. doi:10.1137/18M1173319.

24 Mrinal Kumar and Shubhangi Saraf. The limits of depth reduction for arithmetic formulas: it's all about the top fan-in. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 136–145, 2014.

25 Mrinal Kumar and Shubhangi Saraf. Superpolynomial lower bounds for general homogeneous depth 4 arithmetic circuits. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 751–762, 2014.

26 Mrinal Kumar and Shubhangi Saraf. On the Power of Homogeneous Depth 4 Arithmetic Circuits. *SIAM J. Comput.*, 46(1):336–387, 2017. Conference version appeared in the proceedings of FOCS 2014.

27 Deepanshu Kush and Shubhangi Saraf. Improved low-depth set-multilinear circuit lower bounds. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPIcs*, pages 38:1–38:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CCC.2022.38.

28 Guillaume Lagarde, Nutan Limaye, and Srikanth Srinivasan. Lower Bounds and PIT for Non-commutative Arithmetic Circuits with Restricted Parse Trees. *Computational Complexity*, 28(3):471–542, 2019.

29 Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. Superpolynomial Lower Bounds Against Low-Depth Algebraic Circuits. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 804–814. IEEE, 2021. A full version of the paper can be found at https://eccc.weizmann.ac.il/report/2021/081.

30 Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. On the partial derivative method applied to lopsided set-multilinear polynomials. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPIcs*, pages 32:1–32:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CCC.2022.32.

31 Noam Nisan. Lower Bounds for Non-Commutative Computation (Extended Abstract). In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 410–418. ACM, 1991.

32 Noam Nisan and Avi Wigderson. Lower Bounds on Arithmetic Circuits Via Partial Derivatives. *Computational Complexity*, 6(3):217–234, 1997. Conference version appeared in the proceedings of FOCS 1995.

33 Ran Raz. On the Complexity of Matrix Product. *SIAM J. Comput.*, 32(5):1356–1369, 2003. Conference version appeared in the proceedings of STOC 2002.

34 Ran Raz. Separation of Multilinear Circuit and Formula Size. *Theory of Computing*, 2(6):121–135, 2006. Conference version appeared in the proceedings of FOCS 2004.

35 Ran Raz. Tensor-Rank and Lower Bounds for Arithmetic Formulas. *J. ACM*, 60(6):40:1–40:15, 2013. Conference version appeared in the proceedings of STOC 2010.

36 Ran Raz and Amir Yehudayoff. Lower Bounds and Separations for Constant Depth Multilinear Circuits. *Computational Complexity*, 18(2):171–207, 2009. Conference version appeared in the proceedings of CCC 2008.

37 Bruce Reznick. Sums of even powers of real linear forms. *Memoirs of the AMS*, 96:463, 1992.

38 Amir Shpilka and Avi Wigderson. Depth-3 arithmetic circuits over fields of characteristic zero. *Computational Complexity*, 10(1):1–27, 2001. Conference version appeared in the proceedings of CCC 1999.

39 Sébastien Tavenas. Improved bounds for reduction to depth 4 and depth 3. *Inf. Comput.*, 240:2–11, 2015. Conference version appeared in the proceedings of MFCS 2013.

40 Leslie G. Valiant. Completeness Classes in Algebra. In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 249–261, 1979.

41 Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. Fast Parallel Computation of Polynomials Using Few Processors. *SIAM J. Comput.*, 12(4):641–644, 1983.

42 Ilya Volkovich. A Guide to Learning Arithmetic Circuits. In *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, pages 1540–1561, 2016.

Multi Layer Peeling for Linear Arrangement and Hierarchical Clustering

Yossi Azar
School of Computer Science, Tel-Aviv University, Israel

Danny Vainstein
School of Computer Science, Tel-Aviv University, Israel

Abstract

We present a new multi-layer peeling technique to cluster points in a metric space. A well-known non-parametric objective is to embed the metric space into a simpler structured metric space such as a line (i.e., Linear Arrangement) or a binary tree (i.e., Hierarchical Clustering). Points which are close in the metric space should be mapped to close points/leaves in the line/tree; similarly, points which are far in the metric space should be far in the line or on the tree. In particular we consider the Maximum Linear Arrangement problem [20] and the Maximum Hierarchical Clustering problem [12] applied to metrics.

We design approximation schemes ($1 - \epsilon$ approximation for any constant $\epsilon > 0$) for these objectives. In particular this shows that by considering metrics one may significantly improve former approximations (0.5 for Max Linear Arrangement and 0.74 for Max Hierarchical Clustering). Our main technique, which is called multi-layer peeling, consists of recursively peeling off points which are far from the "core" of the metric space. The recursion ends once the core becomes a sufficiently densely weighted metric space (i.e. the average distance is at least a constant times the diameter) or once it becomes negligible with respect to its inner contribution to the objective. Interestingly, the algorithm in the Linear Arrangement case is much more involved than that in the Hierarchical Clustering case, and uses a significantly more delicate peeling.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Hierarchical clustering, Linear arrangements, Metric embeddings

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.13

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2305.01367`

Funding *Yossi Azar*: Research supported in part by the Israel Science Foundation (grant No. 2304/20).

1 Introduction

Unsupervised learning plays a major role in the field of machine learning. Arguably the most prominent type of unsupervised learning is done through clustering. Abstractly, in this setting we are given a set of data points with some notion of pairwise relations which is captured via a metric space (such that closer points are more similar). In order to better understand the data, the goal is to embed this space into a simpler structured space while preserving the original pairwise relationships. A prevalent solution in this domain is to build a flat clustering (or partition) of the data (e.g., by using the k-means algorithm). However, these types of solutions ultimately fail to capture all pairwise relations (e.g., intra-cluster relations). To overcome this difficulty, often the metric space is mapped to structures that may capture all pairwise relations - in our case into a Linear Arrangement (LA) or a Hierarchical Clustering (HC).

The idea of embedding spaces by using a Linear Arrangement or Hierarchical Clustering structure is not new. These types of solutions have been extensively used in practice (e.g., see [11, 30, 5, 6, 26]) and have also been extensively researched from a theoretical point of view (e.g., see [13, 12, 24, 10, 17, 20]). Notably, the Linear Arrangement type objectives were first considered by Hansen [19] who considered the embedding of graphs into 2-dimensional and higher planes. On the other hand, the study of Hierarchical Clustering type objectives was initiated by Dasgupta [13] - spurring a fruitful line of work resulting in many novel algorithms. In practice, more often than not, the data considered adheres to the triangle inequality (in particular guaranteeing that if point a is similar, equivalently close, to points b and c then so are b and c) and thus may be captured by a metric (e.g., see [9, 25, 26]).
The first objective we consider is the Max Linear Arrangement objective.

▶ **Definition 1.** *Let $G = (V, w)$ denote a metric (specifically, w satisfies the triangle inequality) with $|V| = n$. In the **Max Linear Arrangement problem** our goal is to return a 1-1 mapping $y : V \to [n]$ so as to maximize $\sum_{i,j} w_{i,j} y_{i,j}$, where $y_{i,j} = |y_i - y_j|$.*

The second objective we consider is the Max Hierarchical Clustering objective.

▶ **Definition 2.** *Let $G = (V, w)$ denote a metric (specifically, w satisfies the triangle inequality). In the **Max Hierarchical Clustering problem** our goal is to return a binary HC tree T such that its leaves are in a 1-1 correspondence with V. Furthermore, we would like to return T so as to maximize $\sum_{i,j} w_{i,j} |T_{i,j}|$, where T_{ij} is the subtree rooted at the lowest-common-ancestor of the leaves i and j in the Hierarchical Clustering tree T and $|T_{i,j}|$ is the number of leaves in $T_{i,j}$.*

These objectives were first considered by Hassin and Rubinstein [20] and Cohen-Addad et al. [12] (respectively) with respect to the non-metric case. For these (non-metric) objectives the best known approximation ratios are 0.5 for the Linear Arrangement objective [20] and 0.74 for the Hierarchical Clustering objective [25]). The former was achieved by was achieved by bisecting the data points randomly and thereafter greedily arranging each set and the latter was achieved by approximating the Balanced Max-2-SAT problem.

As stated earlier, more often than not, the data considered in practical applications adheres to the triangle inequality. Therefore, our results' merits are two fold. First, we offer a generalized framework to tackle these types of embedding objectives. Second, our results show that by applying this natural assumption we may significantly improve former best known approximations (from 0.5 (LA) and 0.74 (HC) to $1 - \epsilon$ for any constant $\epsilon > 0$).

Our Results

We provide the following results.

- We design a general framework in order to tackle the embedding of metric spaces into simpler structured spaces (see Algorithm 1). We then concretely apply our framework to both the Linear Arrangement and Hierarchical Clustering settings. For an extended discussion see Our Techniques.
- We apply our framework to the Linear Arrangement case. In this case we prove that our applied algorithm (2) is an EPRAS (see Definition 8) - i.e., for any constant $\epsilon > 0$ it yields a $1 - \epsilon$ approximation.
- We apply our framework to the Hierarchical Clustering case. In this case we prove that our applied algorithm (4) is an EPRAS (see Definition 8) - i.e., for any constant $\epsilon > 0$ it yields a $1 - \epsilon$ approximation.

Our Techniques

Our generic multi-layer peeling approach appears in Algorithm 1. We begin by checking whether the metric space is sufficiently densely weighted (i.e., whether the average distance is at least a constant times the diameter, or equivalently the metric's weighted density (see Definition 3) is constant). If this is the case then we apply a specific algorithm that handles such instances. In the LA case we devise our own algorithm (see Algorithm 3). Algorithm 3 leverages the General Graph Partitioning algorithm of Goldreich et al. [18] in order to "guess" an optimal graph partition that induces an almost optimal linear arrangement. In the HC case we leverage the work of Vainstein et al. [31].

If, however, the metric is not sufficiently densely weighted, then we observe that it must contain a **core** - a subset of nodes containing almost all data points with a diameter significantly smaller than the original metric's. Our general algorithm then peels off data points *far* from the core (in the LA setting) or *not in* the core (in the HC setting). We then embed these peeled off points; by placing them on one of the extreme sides of the line (in the LA setting) or by arranging them in a ladder structure (in the HC case; see Definition 11). Thus, we are left with handling the core (in the HC setting) or the extended core (in the LA setting).

Once again we consider two cases - either the total weight within the (extended) core is small enough, in which case we embed the core arbitrarily. Otherwise, we recurse on the instance induced by these data points. We claim that in every recursion step the density of the (extended) core increases significantly until eventually the recursion ends either when the (extended) core is sufficiently densely weighted or the total weight within the (extended) core is small enough.

Our proof is based on several claims. First, we consider the metric's (extended) core compared to the peeled off layer. Since our algorithm embeds the two sets separately, we need to bound the resulting loss in objective value. We show that the weights within the peeled off layer contribute negligibly towards the objective while the weights between the peeled off layer and the (extended) core, contribute significantly. Hence, it makes sense then to peel off this layer in order to maximize the gain in objective value.

While the aforementioned is enough to bound the loss in a single recursion step, it is not enough. The number of recursion steps may not be constant which, in principle, may cause a blow up of the error. Nevertheless, we show that the error in each level is bounded by a geometric sequence and hence is dominated by the error of the deepest recursion step. Consequently, we manage to upper bound the total accumulated error by a constant that we may take to be as small as we wish.

While at large this describes our proof techniques, the algorithm and analysis of LA objective is a bit more nuanced as we will be considering 3 sets: the metric's core, the peeled off layer, and any remaining points which together with the core are labeled as the extended core. In this case, to be able to justify peeling off a layer, we must choose the layer more aggressively. Specifically, we define this layer as points that are sufficiently far from the core (rather than any point outside the core, as in the HC case). Fortunately, this defined layer (see Algorithm 2) fits our criteria (of our general algorithm, Algorithm 1).

Related Work

While the concept of hierarchical clustering has been around for a long time, the HC objective is relatively recent. In their seminal work, Dasgupta [13] considered the problem of HC from an optimization view point. Thereafter, Cohen-Addad et al. [12] were the

first to consider the objective we use in our manuscript. In their work they showed that the well known Average-Linkage algorithm yields an approximation of $\frac{2}{3}$. Subsequently, Charikar et al. [8] improved upon this result through the use of semidefinite programming - resulting in a 0.6671 approximation. Finally, Naumov et al. [25] improved this to 0.74 by approximating the Balanced Max-2-SAT problem. With respect to the Max LA objective, Hassin and Rubinstein [20] were first to consider the problem. Through an approach of bisection and then greedily arranging the points, Hassin and Rubinstein managed to achieve a 0.5 approximation. We note that the previous mentioned results all hold for arbitrary weights, while our main contribution is showing that by assuming the triangle inequality (i.e., metric-based dissimilarity weights) we may achieve PTAS's for both objectives. We further note that with respect to metric-based dissimilarity weights, specifically an L1 metric, Rajagopalan et al. [26] proved a 0.9 approximation through the use of random cut trees.

Both objectives have been originally studied with respect to their minimization variants. The minimum LA setting was first considered by Hansen [19]. Hansen leveraged the work of Leighton and Rao [23] on balanced separators in order to approximate the minimum linear arrangement objective to facor of $O(\log^2 n)$. Following several works improving upon this result, both Charikar et al. [10] and Feige and Lee [17] leveraged the novel work of Arora et al. [4] on rounding of semidefinite programs, and combined this with the rounding algorithm of Rao and Reicha [27] in order to show a $O(\sqrt{\log n}\log\log n)$ approximation. For further reading on these are related types of objectives see [16, 27, 29, 28]. On the other hand, as mentioned earlier the minimum HC setting was introduced by Dasgupta [13] and extensively studied as well (e.g., see [13, 12, 7, 8, 1, 2, 31]).

Most related to our work is that of de la Vega and Kenyon [15]. In their work they provide a PTAS for the Max Cut problem given a metric. The algorithm works by first creating a graph of clones (wherein each original vertex is cloned a number of times that is based on its outgoing weight in the original metric) with the property of being dense. It thereafter solves the problem in this new graph by applying the algorithm of de la Vega and Karpinski [14]. For our objectives (HC and LA) such an approach seems to fail - specifically due to the fact that our objectives take into consideration the number of nodes in every induced cut and the cloned graph inflates the number of nodes which in turn inflates our objective values. Thus, for our considered types of objectives we need the more intricate process of iterative peeling (and subsequently terminating the process with more suited algorithms that leverage the General Graph Partitioning algorithm of Goldreich et al. [18]). It is worth while mentioning that there has also been an extensive study of closely related objectives with respect to dense instances (e.g. see [22, 3, 21]). However these types of approaches seem to fall short since our considered metrics need not be dense.

2 Multi-Layer Peeling Framework

Before defining our algorithms we need the following definitions.

▶ **Definition 3.** *Let $G = (V, w)$ denote a metric and $U \subset V$ denote a subset of its nodes. We introduce the following notations: (1) let $\mathbf{D_U} = \max_{i,j\in U} w_{i,j}$ denote U's **diameter**, (2) let $\mathbf{W_U} = \sum_{i,j\in U} w_{i,j}$ denote U's **sum of weights**, (3) let $\mathbf{n_U} = |U|$ denote U's **size** and (4) let $\rho_{\mathbf{U}} = \frac{W_U}{n_U^2 D_U}$ denote U's **weighted density**[1].*

[1] Typically the density is defined with respect to $\binom{n}{2}$. For ease of presentation, we chose to define it with respect to n^2 - the proofs remain the same using the former definition.

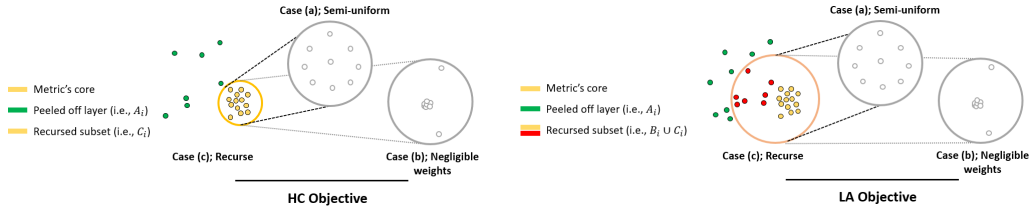


Figure 1 A recursion step (case (c)) and the two possible halting steps (cases (a) and (b)). The yellow points define the metric's core. In the HC case we peel off both red and green points in a single step, while in the LA we must be more delicate and only peel off the green points.

All our algorithms will make use of the following simple yet useful structural lemma that states that for small-density instances there exists a large cluster of nodes with a small diameter. The proof appears in the full version.

▶ **Lemma 4.** *For any metric $G = (V, w)$ there exists a set $U \subset V$ such that $D_U \leq 4D_V\sqrt{\rho_V}$ and $n_U \geq n_V(1 - \sqrt{\rho_V})$.*

▶ **Definition 5.** *Given a metric $G = (V, w)$ we denote $U \subset V$ as guaranteed by Lemma 4 as a metric's* ***core****.*

Note that the core can be found algorithmicaly simply through brute force (while the core need not be unique, our algorithms will choose one arbitrarily).

Throughout our paper we consider different metric-based objectives. In order to solve them, we apply the same recipe - if the instance is sufficiently densely weighted, apply an algorithm for these types of instances. Otherwise, the algorithm detects the metric's core (which is a small-diameter subset containing almost all nodes) and peel off (and subsequently embed) a layer of data points that are far from the core. The algorithm then considers the core; if it is sufficiently small (in terms of inner weights) then we embed the core arbitrarily and halt. Otherwise, we recurse on the core. Our algorithms for both objectives (LA and HC) will follow the same structure as defined in Algorithm 1.

Algorithm 1 General Algorithm.

if *the instance is sufficiently densely weighted* **then** `// case (a)`
  Solve it using ALG_{d-w}.
else
  Let C denote the metric's core (as defined by Definition 5).
  Define the layer to peel off $A \subset V \setminus C$ appropriately.
  Embed A.
  if $W_{V \setminus A}$ *is negligible* **then** Embed $V \setminus A$ arbitrarily and return. ; `// case (b)`
  else Continue recursively on $V \setminus A$; `// case (c)`

We denote by cases (a) and (b) the different cases for which the algorithm may terminate and by case (c) the recursive step. We further denote by ALG_{d-w} an auxiliary algorithm that will handle sufficiently densely weighted instances. (These algorithms will differ according to the different objectives).

Henceforth, given an algorithm ALG and metric G we denote by $ALG(G)$ the algorithm's returned embedding. We note that when clear from context we overload the notation and denote $ALG(G)$ as the embedding's value under the respective objectives. Equivalently, we will use the term $OPT(G)$ for the optimal embedding.

Our different algorithms will be similarly defined and thus so will their analyses. Thus, we introduce a general scheme for analyzing such algorithms. Let k denote the number of recursive calls our algorithm performs. Furthermore, let G_i denote the instance the algorithm is called upon in step i for $i = 0, 1, \ldots, k$. (I.e., $G = G_0$ and $ALG(G_k)$ does not perform a recursive step, meaning that it terminates with case (a) or (b)). We first observe that by applying a simple averaging argument we get the following useful observation.

▶ **Observation 6.** *If there exist $\alpha_i, \beta_i, \gamma_i > 0$ such that $ALG(G_i) \geq \alpha_i + ALG(G_{i+1})$ and $OPT(G_i) \leq \beta_i + \gamma_i OPT(G_{i+1})$ for all $i = 0, \ldots, k-1$ then*

$$\begin{aligned}\frac{ALG(G)}{OPT(G)} &\geq \frac{\sum_{i=0}^{k-1} \alpha_i + ALG(G_k)}{\sum_{i=0}^{k-1} \left(\beta_i \Pi_{j=0}^{i-1} \gamma_j\right) + (\Pi_{i=0}^{k-1} \gamma_i) OPT(G_k)} \\ &\geq \min\{\min_i\{\frac{\alpha_i}{\beta_i \Pi_{j=0}^{i-1} \gamma_j}\}, \frac{ALG(G_k)}{(\Pi_{i=0}^{k-1} \gamma_i) OPT(G_k)}\}.\end{aligned}$$

Thus, in order to analyze a given algorithm, it will be enough to set the values of α_i, β_i and γ_i, and further analyze the approximation ratio of $\frac{ALG(G_k)}{OPT(G_k)}$ for the different terminating cases (cases (a) and (b)).

3 Notations and Preliminaries

We introduce the following notation to ease our presentation later on.

▶ **Definition 7.** *Given a metric $G = (V, w)$, a solution $\mathbf{SOL(G)}$ for the LA objective and disjoints sets $A, B \subset V$ we define: $\mathbf{SOL(G)}|_{\mathbf{A}} = \sum_{i,j \in A} w_{i,j} y_{i,j}$ and $\mathbf{SOL(G)}|_{\mathbf{A,B}} = \sum_{i \in A, j \in B} w_{i,j} y_{i,j}$. For the HC objective the notations are defined symmetrically by replacing $y_{i,j}$ with $|T_{i,j}|$.*

We will make use of algorithms belonging to the following class of algorithms.

▶ **Definition 8.** *An algorithm is considered an Efficient Polytime Randomized Approximation Scheme (EPRAS) if for any $\epsilon > 0$ the algorithm has expected running time of $f(\frac{1}{\epsilon}) n^{O(1)}$ and approximates the optimal solution's value up to a factor of $1 - \epsilon$.*

We will frequently use the following (simple) observations and thus we state them here.

▶ **Observation 9.** *Given values $\alpha_i \geq 0$, $\alpha \in (0, \frac{1}{k(k+1)})$ and $k \in \mathbb{N}$ we have: **(1)** $\Pi_i(1-\alpha_i) \geq 1 - \sum_i \alpha_i$, **(2)** $1 + k\alpha < \frac{1}{1-k\alpha} < 1 + (k+1)\alpha$ and **(3)** $1 + k\alpha < e^{k\alpha} < 1 + (k+1)\alpha$.*

The following facts will prove useful in our subsequent proofs and are therefore stated here.

▶ **Fact 10.** *Given a metric G, if the optimal linear arrangement under the LA objective is $OPT_{LA}(G)$ and the optimal hierarchical clustering under the HC objective is $OPT_{HC}(G)$ then we have $OPT_{LA}(G) \geq \frac{1}{3} n \sum_{i,j} w_{i,j} y_{i,j}$ and $OPT_{HC}(G) \geq \frac{2}{3} n \sum_{i,j} w_{i,j} |T_{i,j}|$.*

We note that the HC portion of Fact 10 has been used widely in the literature (e.g., see proof in [12]). The LA portion of Fact 10 is mentioned in Hassin and Rubinstein [20]. Finally, in the HC section we make use of "ladder" HC trees. We define them here.

▶ **Definition 11.** *We define a "ladder" as an HC tree that cuts a single data point from the rest at every cut (or internal node).*

4 The Linear Arrangement Objective

We will outline the section as follows. We begin by presenting our algorithms (first the algorithm that handles case (a) and thereafter the general algorithm). We will then bound the algorithm's approximation guarantee (by following the bounding scheme of Observation 6). Finally, we will analyze the algorithm's running time.

4.1 Defining the Algorithms

Here we begin by applying our general algorithm to the linear arrangement problem (which we will denote simply as ALG). The algorithm uses, as a subroutine, an algorithm to handle case (a). We denote this subroutine as ALG_{d-w} and define it following the definition of ALG.

4.1.1 Defining ALG

Here we apply our general algorithm (Algorithm 1) to the linear arrangement setting. In order to do so, roughly speaking, we define the layer to peel off A as the set of all points which are "far" from the metric's core. We also introduce a subroutine to handle densely weighted instances, ALG_{d-w}.

Algorithm 2 Linear Arrangement Algorithm (ALG).

if $\rho \geq \epsilon^6$ **then** solve it using ALG_{d-w}. ; `// case (a)`
else
  Let C denote the metric's core (as defined by Lemma 4).
  Let A denote all data points that are of distance $\geq \epsilon^2 D_V$ from C.
  Place A to the left of $V \setminus A$. Arrange A arbitrarily.
  if $W_{V \setminus A} < \epsilon W_V$ **then** Arrange $V \setminus A$ arbitrarily and return. ; `// case (b)`
  else Continue recursively on $V \setminus A$. ; `// case (c)`

The set $V \setminus \{A \cup C\}$ will be used frequently in the upcoming proofs and thus we give it its own notation.

▶ **Definition 12.** *Denote $B = V \setminus \{A \cup C\}$ where A and C are defined as in Algorithm 2.*

4.1.2 Defining ALG_{d-w}

Here we will introduce an algorithm to handle case (a) type instances. Before formally defining the algorithm, we will first provide some intuition. Towards that end we first introduce the following definition.

▶ **Definition 13.** *Consider $OPT(G_k)$'s embedding into the line, $[n]$. Partition $[n]$ into $\frac{1}{\epsilon}$ consecutive sets each of size ϵn and let P_i^* denote the points embedded by $OPT(G_k)$ into the i'th consecutive set. Furthermore, denote by $P^* = \{P_i^*\}$ the induced partition of the metric.*

Later on, we will show that $OPT(G_k)$'s objective value is closely approximated by the value generated solely from inter-partition-set edges (i.e., any (u, v) where u, v lie in different partition sets of P^*). While $OPT(G_k)$ cannot be found algorithmically, assuming the above holds, it is enough for ALG_{d-w} to guess the partition P^*. Indeed, that is exactly what we will do, by using the general graph partitioning algorithm of Goldreich et al. [18].

We denote the General Graph Partitioning algorithm of Goldreich et al. [18] as $PT(G, \Phi, \epsilon_{err})$. See Definition 21 for a definition of Φ and ϵ_{err} (these will be defined by ALG_{d-w} as well) and see Theorem 22 for the tester's guarantees. We are now ready to define our algorithm that handles sufficiently densely weighted instances (Algorithm 3).

Algorithm 3 LA Algorithm for Sufficiently Densely Weighted Instances (ALG_{d-w}).

Let $k = \frac{1}{\epsilon}$ denote the size of the partition.
for $\{\mu_{j,j'}\}_{j \leq k, j' \leq k, j \neq j'} \subset \{i\epsilon^9 n^2 D_V : i \in \mathbb{N} \wedge i \leq \frac{1}{\epsilon^7}\}$ **do**
  Let $\Phi = \{\epsilon n, \epsilon n\}_{j=1}^{k} \cup \{\mu_{j,j'}, \mu_{j,j'}\}_{j,j'=1}^{k}$.
  Run $PT(G, \Phi, \epsilon_{err} = \epsilon^9)$. Let P denote the output partition (if succeeded).
  Let $\widehat{y}$ denote the linear arrangement obtained from embedding P consecutively on the line (and arbitrarily within the partition sets).
  Compute the value $\sum_e w_e \widehat{y_e}$ for P.
Return the partition with maximum $\sum_e w_e \widehat{y_e}$ value.

4.2 Analyzing the Approximation Ratio of ALG

Now that we have defined ALG we are ready to analyze its approximation ratio. Recall that by Observation 6 it is enough to analyze the approximation ratio of cases (a), (b) and the total loss incurred by the recursion steps (i.e., by setting α_i, β_i and γ_i).

4.2.1 Structural Lemmas

Recall that we defined k to be the number of recursion steps used by ALG and that G_i is the instance that ALG is applied to at recursion step i. Further recall that given G_i, $ALG(G_i)$ partitioned the instance into A_i, B_i and C_i and that, informally, by Lemma 4 n_{C_i} contains the majority of the data points and D_{C_i} is relatively small compared to D_{V_i}.

By the definition of C_i, A_i could be considered as a set of outliers. Therefore, intuitively it makes sense to split A_i from C_i. In order to prove our algorithm's approximation ratio we will show that in fact one does not lose too much compared to optimal solution, by splitting A_i from C_i. In order to do so we will show that in fact, both the values of ALG and OPT will be roughly equal to $\frac{1}{2} n W_{A_i,C_i}$ (which makes sense intuitively since C_i is of low diameter and contains many points and A_i are the points that are far from this cluster).

The following lemmas consider 2 types of algorithms - algorithms that split A_i and C_i and algorithms that do not. Furthermore, they show that in fact, by the structural properties of A_i and C_i, if we consider the values generated by these 2 types of algorithms restricted to the objective value generated by the inter-weights W_{A_i,C_i}, are approximately equal. We begin by lower bounding the value generated by algorithms that split A_i and C_i. Due to lack of space, we defer the following proofs to the full version.

▶ **Lemma 14.** *Given the two disjoint sets C_i and A_i and a linear arrangement y that places all nodes in A_i to the left of all nodes in C_i we are guaranteed that*

$$\sum_{c \in C_i, a \in A_i} w_{a,c} y_{a,c} \geq \frac{n_{C_i}}{2} (W_{C_i,A_i} - n_{C_i} n_{A_i} D_{C_i}).$$

Due to the fact that C_i is a small cluster containing most of the data points the above lemma reduces to the following corollary.

▶ **Corollary 15.** *Given any linear arrangement y that places all nodes in A_i to the left of all nodes in C_i we are guaranteed that*

$$\sum_{a\in A_i, c\in C_i} w_{a,c}y_{a,c} \geq \frac{1}{2}nW_{A_i,C_i}(1-\frac{5\sqrt{\rho}}{\epsilon^2})$$

Now that we have lower bounded algorithms that split A_i and C_i we will upper bound algorithms that do not have this restriction. (Note that we begin by handling the case where one of the disjoint sets is a single data point and thereafter generalize it to two disjoint sets).

▶ **Lemma 16.** *Given a set C_i and a point $p \notin C_i$, we are guaranteed that*

$$\sum_{c\in C_i} w_{p,c}y_{p,c} \leq (W_{p,C_i} + n_{C_i}D_{C_i})(n-\frac{n_{C_i}}{2}).$$

We are now ready to upper bound the inter-objective-value of two sets of disjoint points.

▶ **Lemma 17.** *Given the two disjoint sets C_i and A_i and any linear arrangement y we are guaranteed that*

$$\sum_{c\in C_i, a\in A_i} w_{a,c}y_{a,c} \leq (n-\frac{n_{C_i}}{2})(W_{C_i,A_i} + n_{C_i}n_{A_i}D_{C_i}).$$

Due to the fact that C_i is a small cluster containing most of the data points the lemma reduces to the following corollary.

▶ **Corollary 18.** *Given any linear arrangement y we are guaranteed that*

$$\sum_{a\in A_i, c\in C_i} w_{a,c}y_{a,c} \leq \frac{1}{2}nW_{A_i,C_i}(1+\frac{9\sqrt{\rho}}{\epsilon^2}).$$

We will want to show that the objective values of both ALG and OPT (and some other intermediate values that will be defined later on) are approximately determined by their value on the inter-weights of W_{A_i,C_i}. In order to do so, we first introduce the following structural lemma that will help us explain this behaviour.

▶ **Lemma 19.** *Given an instance G and sets A, B and C as defined by $ALG(G)$ we have $W_A + W_{A,B} \leq 2\frac{\sqrt{\rho}}{\epsilon^2}W_{A,C}$.*

4.2.2 Analyzing the Approximation Ratio of Case (a) of ALG

We first give an overview the approximation ratio analysis. Recall the definition of P^* (Definition 13). The first step towards our proof, is to show that instead of trying to approximate $OPT(G_k)$, it will be enough to consider its value restricted to intra-partition-set weights with respect to P^*. Even more, for such weights $w_{u,v}$, incident to P^*_i and P^*_{i+j}, it will be enough to assume that their generated value towards the objective (i.e., the value $y_{u,v}$) is only $(j-1)\epsilon n$ (while it may be as large as $(j+1)\epsilon n$). Formally, this will be done in Lemma 20 (whose proof is deferred to the full version).

Next, recall that ALG_{d-w} tries to guess the partition P^* (up to some additive error) and let P denote the partition guessed by ALG_{d-w}. Observe that if guessed correctly, the value generated towards ALG's objective for any intra-partition-set weight crossing between P_i and P_{i+j} is at least $|P_{i+1}| + \cdots |P_{i+j-1}|$ and if we managed to guess the set sizes as well then this value is exactly $(j-1)\epsilon n$ (equivalent to that of OPT's). This will be done in Proposition 23.

▶ **Lemma 20.** *Given the balanced line partition of set sizes ϵn, denoted as P^*, we have*

$$OPT(G_k) \leq (1+13\epsilon) \sum_{\substack{1\leq i\leq k-1\\ 1\leq j\leq k-i}} W_{P_i^*,P_{i+j}^*}(|P_{i+1}^*|+\cdots+|P_{i+j-1}^*|).$$

Before proving Proposition 23 we state the properties of the general graph partitioning algorithm of Goldreich et al. [18].

▶ **Definition 21** ([18])**.** *Let $\Phi = \{\lambda_j^{LB},\lambda_j^{UB}\}_{j=1}^k \cup \{\mu_{j,j'}^{LB},\mu_{j,j'}^{UB}\}_{j,j'=1}^k$ denote a set of non-negative values such that $\lambda_j^{LB} \leq \lambda_j^{UB}$ and $\mu_{j,j'}^{LB} \leq \mu_{j,j'}^{UB}$. We define $\mathcal{GP}_\Phi$ the set of graphs G on n vertices that have a k partition $(V_1,\ldots,V_k)$ upholding the following constraints*

$$\forall j:\ \lambda_j^{LB} \leq \frac{|V_j|}{n} \leq \lambda_j^{UB};\quad \forall j,j':\ \mu_{j,j'}^{LB} \leq \frac{W_{V_j,V_{j'}}}{n^2} \leq \mu_{j,j'}^{UB}.$$

▶ **Theorem 22** ([18])**.** *Given inputs $G=(V,w)$ with $|V|=n$ and $w: V\times V\to[0,1]$ describing the graph and Φ describing bounds on the wanted partition, ϵ_{err}, the algorithm $PT(G,\Phi,\epsilon_{err})$ has expected running time*[2] *of*

$$\exp\left(\log(\frac{1}{\epsilon_{err}})\cdot(\frac{O(1)}{\epsilon_{err}})^{k+1}\right) + O(\frac{\log\frac{k}{\epsilon_{err}}}{\epsilon_{err}^2})\cdot n.$$

Furthermore, if $G\in\mathcal{GP}_\Phi$ as in Definition 21 then the algorithm outputs a partition satisfying

- $\forall j:\ \lambda_j^{LB}-\epsilon_{err} \leq \frac{|V_j|}{n} \leq \lambda_j^{UB}+\epsilon_{err}$,
- $\forall j,j':\ \mu_{j,j'}^{LB}-\epsilon_{err} \leq \frac{W_{V_j,V_{j'}}}{n^2} \leq \mu_{j,j'}^{UB}+\epsilon_{err}$.

We are now ready to prove Proposition 23.

▶ **Proposition 23.** *If ALG terminates in case (a) then $\frac{ALG_{d-w}(G_k)}{OPT(G_k)} = \frac{ALG(G_k)}{OPT(G_k)} \geq 1-20\epsilon$.*

Proof. Let $P=\{P_i\}$ denote the partition returned by $PT(G_k,\Phi,\epsilon_{err})$ and recall that its number of sets is $k=\frac{1}{\epsilon}$ and that $\epsilon_{err}=\epsilon^9$. We first observe that by Theorem 22 we are guaranteed that the error in $|P_i|$ compared to $|P_i^*|=\epsilon n$ is at most $|P_i|\geq \epsilon n-\epsilon_{err}n$ (due to the fact that in Φ we requested sets of size exactly ϵn). Therefore

$$ALG_{d-w} \geq \sum_{\substack{1\leq i\leq k-1\\ 1\leq j\leq k-i}} W_{P_i,P_{i+j}}(|P_{i+1}|+\cdots+|P_{i+j-1}|) \geq \sum_{\substack{1\leq i\leq k-1\\ 1\leq j\leq k-i}} (j-1)(\epsilon n-\epsilon_{err}n)W_{P_i,P_{i+j}}, \quad (1)$$

where $W_{P_i,P_{i+j}}$ denotes the weight crossing between P_i and P_{i+j}. For ease of presentation we will remove the subscript in the summation henceforth.

Consider the difference between the cut size of $W_{P_i,P_{i+j}}$ and $W_{P_i^*,P_{i+j}^*}$. Their difference originates from two errors: (1) the error that incurred by the PT algorithm (see Theorem 22) and (2) the error ALG_{d-w} incurred in order to guess the partition of $OPT(G_k)$ (see Algorithm 3). Therefore,

$$W_{P_i,P_{i+j}} \geq W_{P_i^*,P_{i+j}^*} - \epsilon_{err}n^2D_V - \epsilon^9n^2D_V = W_{P_i^*,P_{i+j}^*} - 2\epsilon^9n^2D_V$$

where the last equality is since $\epsilon_{err}=\epsilon^9$. Combining this with inequality 1 yields

$$\begin{aligned} ALG_{d-w} \geq (\epsilon n-\epsilon_{err}n)\cdot\sum(j-1)W_{P_i^*,P_{i+j}^*} - (\epsilon n-\epsilon_{err}n)\cdot 2(\epsilon^9n^2)D_V\sum(j-1) \geq \\ (\epsilon n-\epsilon_{err}n)\cdot\sum(j-1)W_{P_i^*,P_{i+j}^*} - 2n^3\epsilon^7D_V, \end{aligned} \quad (2)$$

[2] We remark that the original algorithm contains a probability of error δ, that appears in the running time. We disregard this error and bound the expected running time of the algorithm.

where the last inequality follows since $\epsilon_{err} > 0$ and $\sum(j-1) = \sum_{i=1}^{k}\sum_{j=i+1}^{k}(j-1) \leq k^3 = \epsilon^{-3}$.

Due to the fact that we are in case (a) we have that $\frac{W}{n^2 D_V} = \rho \geq \epsilon^6$. By Fact 10 we have that $OPT \geq \frac{1}{3}nW$ and therefore $2n^3\epsilon^7 D_V$ can be bounded by $2n^3\epsilon^7 D_V \leq 2\epsilon nW \leq 6\epsilon OPT$. Thus we get $2n^3\epsilon^7 D_V \leq 6\epsilon OPT(G_k)$. Combining this with inequality 2 yields

$$ALG_{d-w} \geq (\epsilon n - \epsilon_{err} n) \cdot \sum(j-1) W_{P_i^*, P_{i+j}^*} - 6\epsilon OPT(G_k). \tag{3}$$

On the other hand, recall that P^* denotes the balanced partition where all sets are of size ϵn. Therefore, by Lemma 20 we therefore get

$$\begin{aligned} &OPT(G_k) \leq (1+13\epsilon)\sum W_{P_i^*, P_{i+j}^*}(|P_{i+1}^*| + \cdots + |P_{i+j-1}^*|) = \\ &(1+13\epsilon)\sum(j-1)(\epsilon n) W_{P_i^*, P_{i+j}^*} = \epsilon n(1+13\epsilon) \cdot \sum(j-1) W_{P_i^*, P_{i+j}^*}. \end{aligned} \tag{4}$$

Combining inequalities 3 and 4 yields

$$\begin{aligned} &ALG_{d-w} \geq \frac{\epsilon n - \epsilon_{err} n}{\epsilon n(1+13\epsilon)} OPT(G_k) - 6\epsilon OPT(G_k) = \\ &\frac{1-\epsilon^8}{1+13\epsilon} OPT(G_k) - 6\epsilon OPT(G_k) \geq (1-20\epsilon) OPT(G_k), \end{aligned}$$

thereby concluding the proof. ◀

4.2.3 Analyzing the Approximation Ratio of Case (b) of *ALG*

Using our structural lemmas we will analyze the approximation ratio of ALG applied to G_k under the assumption that the algorithm terminated in case (b) (i.e., that $\rho < \epsilon^6$ and $W_{B \cup C} \leq \epsilon W_{G_k}$). The full proof is deferred to the full version.

▶ **Proposition 24.** *If ALG terminates in case (b) then* $\frac{ALG(G_k)}{OPT(G_k)} \geq 1 - 33\epsilon$.

Sketch. The proof follows the following path. Due to the fact that most of the instance's density is centered at the metric's core C, the majority of $OPT(G_k)$'s objective is derived from weights incident to C. Since we are case (b), the weight of $W_{B \cup C}$ is negligible and therefore we will show that in fact $OPT(G_k)$'s objective is defined by $OPT(G_k)|_{A,C}$. Thereafter, we show that in fact the best strategy to optimize for weights in $W_{A,C}$ is to place A at one extreme of the line and C at the other - which, fortunately, is what $ALG(G_k)$ (approximately) does - thereby approximating $OPT(G_k)$. ◀

4.2.4 Setting the Values α_i, β_i and γ_i

Due to lack of space, the following proofs are deferred to the full version.

▶ **Proposition 25.** *For A_i and C_i as defined by our algorithm applied to G_i and for $\alpha_i = \frac{1}{2}nW_{A,C}(1 - \frac{5\sqrt{\rho}}{\epsilon^2})$, we have $ALG(G_i) \geq \alpha_i + ALG(G_{i+1})$.*

▶ **Proposition 26.** *Let $G_i = (V_i, w_i)$ and $G_{i+1} = (V_{i+1}, w_{i+1})$ denote the instances defined by the i and $i+1$ recursion steps. Furthermore let $\beta_i = \frac{1}{2}n_{V_i}W_{A_i,C_i}(1 + \frac{13\sqrt{\rho}}{\epsilon^2})$ and $\gamma_i = 1 + 4\sqrt{\rho_i}$. Therefore, $OPT(G_i) \leq \beta_i + \gamma_i OPT(G_{i+1})$.*

Thus, we have managed to set the values of α_i, β_i and γ_i as follows.

▶ **Definition 27.** *We define the values α_i, β_i and γ_i as follows*

$$\alpha_i = \frac{1}{2}nW_{A_i,C_i}(1 - \frac{5\sqrt{\rho_i}}{\epsilon^2}); \quad \beta_i = \frac{1}{2}n_{V_i}W_{A_i,C_i}(1 + \frac{13\sqrt{\rho_i}}{\epsilon^2}); \quad \gamma_i = 1 + 4\sqrt{\rho_i}. \tag{5}$$

4.2.5 Putting it all Together

Now that we have analyzed the terminal cases of the algorithm (cases (a) and (b)) and that we have set the values of α_i, β_i and γ_i we will to combine these results to prove ALG's approximation ratio (as in Observation 9). In order to so we must therefore bound the values $\min_i\{\frac{\alpha_i}{\beta_i\Pi_{j=0}^{i-1}\gamma_j}\}$ and $\frac{ALG(G_k)}{(\Pi_{i=0}^{k-1}\gamma_i)OPT(G_k)}$. However, before doing so we will first show that $\Pi_{j=0}^{i-1}\gamma_j$ converges. Recall that $\gamma_i = 1+4\sqrt{\rho_i}$. The following lemma shows that the instances' densities (ρ_i) increase at a fast enough rate (exponentially) in order for $\Pi_{j=0}^{i-1}\gamma_j$ to converge.

▶ **Lemma 28.** *For all $i = 1,\ldots,k-1$ we are guaranteed that $\rho_{i+1} \geq 4\rho_i$.*

Proof. Let V denote the set of nodes of G_i. Recall the notations A, B and C defined by our algorithm applied to V (in particular, the set of nodes of G_{i+1} is exactly $B \cup C$). Therefore, if we denote by D_{B-C} the largest distance between any point in B and its closest point in C, then $D_{B\cup C} \leq 2D_{B-C} + D_C \leq 2\epsilon^2 D_V + 4D_V\sqrt{\rho_i}$, where the first inequality follow from the triangle inequality and the second follows due to the fact that B is defined as the set of all points of distance at most ϵ^2 from C. Therefore,

$$\rho_{i+1} = \frac{W_{B\cup C}}{n_{B\cup C}^2 \cdot D_{B\cup C}} \geq \frac{W_V}{n_V^2 \cdot D_V}\left(\frac{\epsilon}{2\epsilon^2+4\sqrt{\rho_i}}\right) = \rho_i\left(\frac{\epsilon}{2\epsilon^2+4\sqrt{\rho_i}}\right), \tag{6}$$

where the equalities follows by the definition of ρ_i and the inequality follows due to the fact that $W_{B\cup C} \geq \epsilon W_V$ (which follows due to the fact that we are in case (c)), $n_{B\cup C} \leq n_V$ and $D_{B\cup C} \leq (2\epsilon^2+4\sqrt{\rho_i})D_V$ (as stated above). Since we are in case (c), we are guaranteed that $\rho_i \leq \epsilon^6$ and therefore

$$\frac{\epsilon}{2\epsilon^2+4\sqrt{\rho_i}} \geq \frac{\epsilon}{2\epsilon^2+4\epsilon^3} \geq \frac{1}{3\epsilon}, \tag{7}$$

since $\epsilon \leq 10^{-2}$. Combining inequalities 6 and 7, and since $\epsilon < 10^{-2}$ yields $\rho_{i+1} \geq \rho_i(\frac{\epsilon}{2\epsilon^2+4\sqrt{\rho_i}}) \geq \frac{\rho_i}{3\epsilon} \geq 4\rho_i$, thereby concluding the proof. ◀

We are now ready to show that $\Pi_{j=0}^{i-1}\gamma_j$ converges.

▶ **Lemma 29.** *For $\gamma_i = 1+4\sqrt{\rho_i}$ we have $\Pi_{j=0}^{i-1}\gamma_j \leq 1+5\sqrt{\rho_i}$.*

Proof. Observe that $\Pi_{j=0}^{i-1}(1+4\sqrt{\rho_j}) \leq e^{4\cdot\sum_j\sqrt{\rho_j}} \leq e^{4\sqrt{\rho_i}} \leq 1+5\sqrt{\rho_i}$, where the first inequality follows from Observation 9, the second follows since $\sqrt{\rho_j}$ are exponentially increasing (see full version) and the third inequality follows again by Observation 9 combined with the fact that $\rho < \epsilon^2$ and $\epsilon < 10^{-2}$. ◀

Next we leverage the former lemma to bound $\min_i\{\frac{\alpha_i}{\beta_i\Pi_{j=0}^{i-1}\gamma_j}\}$ and $\frac{ALG(G_k)}{(\Pi_{i=0}^{k-1}\gamma_i)OPT(G_k)}$.

▶ **Proposition 30.** *For α_i, β_i and γ_i as in Definition 27, we have $\min_i\{\frac{\alpha_i}{\beta_i\Pi_{j=0}^{i-1}\gamma_j}\} \geq 1-23\epsilon$.*

Proof. We first bound $\frac{\alpha_i}{\beta_i}$. By the definitions of α_i and β_i we have

$$\frac{\alpha_i}{\beta_i} = \frac{1-\frac{5\sqrt{\rho_i}}{\epsilon^2}}{1+\frac{13\sqrt{\rho_i}}{\epsilon^2}} \geq \left(1-\frac{5\sqrt{\rho_i}}{\epsilon^2}\right)\left(1-\frac{13\sqrt{\rho_i}}{\epsilon^2}\right) \geq 1-\frac{18\sqrt{\rho_i}}{\epsilon^2}, \tag{8}$$

where the first inequality follows from the definitions of α_i and β_i and the rest of the inequalities follow since $\epsilon < 10^2$ and $\rho < \epsilon^6$.

By Lemma 29 we are guaranteed that $\Pi_{j=0}^{i-1}\gamma_j \leq 1+5\sqrt{\rho_i}$. Combining this with inequality 8 yields

$$\frac{\alpha_i}{\beta_i\Pi_{j=0}^{i-1}\gamma_j} \geq \frac{1-\frac{18\sqrt{\rho_i}}{\epsilon^2}}{1+5\sqrt{\rho_i}} \geq (1-\frac{18}{\epsilon^2}\sqrt{\rho_i})(1-5\sqrt{\rho_i}) \geq 1-\frac{23}{\epsilon^2}\sqrt{\rho_i},$$

and since ρ_i only increases and $\rho_{k-1} \leq \epsilon^6$ we have $\min_i\{\frac{\alpha_i}{\beta_i\Pi_{j=0}^{i-1}\gamma_j}\} \geq 1-\frac{23}{\epsilon^2}\sqrt{\rho_{k-1}} \geq 1-23\epsilon$, thereby concluding the proof. ◀

▶ **Proposition 31.** *For $\gamma_i = 1+4\sqrt{\rho_i}$ we have $\frac{ALG(G_k)}{(\Pi_{i=0}^{k-1}\gamma_i)OPT(G_k)} \geq 1-34\epsilon$.*

Proof. By Propositions 23 and 24 we are guaranteed that $\frac{ALG(G_k)}{OPT(G_k)} \geq 1-33\epsilon$. On the other hand by by Lemma 29 we are guaranteed that $\Pi_{i=0}^{k-2}\gamma_i \leq 1+5\sqrt{\rho_{k-1}}$. Therefore, if $k=1$ then $\frac{ALG(G_k)}{(\Pi_{i=0}^{k-1}\gamma_i)OPT(G_k)} = \frac{ALG(G_k)}{OPT(G_k)} \geq 1-33\epsilon$. Otherwise, we have

$$\frac{ALG(G_k)}{(\Pi_{i=0}^{k-1}\gamma_i)OPT(G_k)} \geq \frac{1-33\epsilon}{(1+4\sqrt{\rho_{k-1}})(1+5\sqrt{\rho_{k-1}})} \geq \frac{1-33\epsilon}{(1+4\epsilon^3)(1+5\epsilon^3)} \geq 1-34\epsilon,$$

where the second inequality follows since $\rho_{k-1} < \epsilon^6$ (since we recursed to step k) and the subsequent inequalities follow since $\epsilon < 10^{-3}$ - thereby concluding the proof. ◀

Finally, we combine Propositions 30 and 31 to bound ALG's approximation ratio.

▶ **Theorem 32.** *For any metric G, $\frac{ALG(G)}{OPT(G)} \geq 1-34\epsilon$.*

4.3 Analyzing the Running Time of *ALG*

Consider the definition of ALG. We observe that in each recursion step, the algorithm finds the layer to peel off, A, and then recurses. Therefore the running time is defined by the sum of these recursion steps, plus the terminating cases (i.e., either case (a) or case (b)). Recall that case (a) applies ALG_{d-w} on the instance, while case (b) arranges the instance arbitrarily. Therefore, a bound on cases (a) and (b) is simply a bound on the running time of ALG_{d-w} which is given by Lemma 33 (whose proof appears in the full version).

▶ **Lemma 33.** *Given an instance G, the running time of $ALG_{d-w}(G)$ is at most $(\frac{1}{\epsilon^7})^{\frac{1}{\epsilon^2}} \cdot O(n^2)$.*

▶ Remark 34. A bi-product of Lemma 28 is that the number of recursion steps is bounded by $O(\log n)$. The proof follows similarly to the proof of Lemma 48 substituting the inequality $\rho_{i+1} \geq 4\epsilon\sqrt{\rho_i}$ with $\rho_{i+1} \geq 4\sqrt{\rho_i}$ (which holds due to Lemma 28).

We are now ready to analyze the running time of ALG. (The proof is deferred to the full version.)

▶ **Theorem 35.** *The algorithm ALG is an EPRAS (with running time $O(n^2 \log n)$ plus the running time of ALG_{d-w}).*

▶ Remark 36. We remark that one may improve the running time by replacing ALG_{d-w} with any faster algorithm while slightly degrading the quality of the approximation.

5 The Hierarchical Clustering Objective

The section is outlined as follows. We begin by presenting our algorithms (first the algorithm to handle case (a) and subsequently the general algorithm). Thereafter we will bound the algorithm's approximation guarantee (by following the bounding scheme of Observation 6). Finally, we will analyze the algorithm's running time.

5.1 Defining the Algorithms

As in the linear arragement setting, we will begin by applying our general algorithm to the linear arrangement problem (which we will denote simply as ALG). The algorithm uses, as a subroutine, an algorithm to handle case (a). We denote this subroutine as ALG_{d-w} and define it following the definition of ALG.

5.1.1 Defining ALG

Here we apply our general algorithm (Algorithm 1) to the hierarchical clustering setting. In order to do so, roughly speaking, we define the layer to peel off A as all points outside of the metric's core.

Algorithm 4 Hierarchical Clustering Algorithm (ALG).

if $\rho \geq \epsilon^2$ **then** Solve the instance using ALG_{d-w}. ; `// case (a)`
else
  Let C denote the metric's core (as defined by Lemma 4).
  Let $A = V \setminus C$ denote the rest of the points.
  Arrange A as a (arbitrary) ladder and denote the tree by T_A.
  if $W_C < 16\epsilon \cdot W_V$ **then** `// case (b)`
    Arrange C arbitrarily and denote the resulting tree by T_C.
    Attach T_C's root as a child of the bottom most internal node of T_A and return.
  else `// case (c)`
    Continue recursively on C and denote the resulting tree by T_C.
    Attach T_C's root as a child of the bottom most internal node of T_A and return.

▶ Remark 37. Note that Algorithm 4 conforms to the general Algorithm 1 since $C = V \setminus A$.

5.1.2 Defining ALG_{d-w}

We will use the algorithm of Vainstein et al. [31] as ALG_{d-w}. As part of their algorithm they make use of the general graph partitioning algorithm of Goldreich et al. [18] which is denoted by $PT(\cdot)$. Since we will use $PT(\cdot)$ to devise our own algorithm for the LA objective we refer the reader to Definition 21 and Theorem 22 for a more in-depth explanation of the $PT(\cdot)$ algorithm. We restate ALG_{d-w} in Algorithm 5 as defined in Vainstein et al. [31].

Algorithm 5 HC Algorithm for Sufficiently Densely Weighted Instances (ALG_{d-w}).

Enumerate over all trees T with $k = \frac{1}{\epsilon}$ internal nodes.
for *each such* T **do**
  for $\{\lambda_i\}_{i \leq k} \subset \{i\epsilon^2 n : i \in \mathbb{N} \wedge i \leq \frac{3}{\epsilon}\}$ **do**
    for $\{\mu_{j,j'}\}_{j \leq k, j' \leq k, j \neq j'} \subset \{i\epsilon^3 n^2 D_V : i \in \mathbb{N} \wedge i \leq \frac{9}{\epsilon}\}$ **do**
      Let $\Phi = \{\lambda_i, \lambda_i\}_{i=1}^{k} \cup \{\mu_{j,j'}, \mu_{j,j'}\}_{j,j'=1}^{k}$.
      Run $PT(G, \Phi, \epsilon_{err} = \epsilon^3)$. Let P denote the output partition (if succeeded).
      Compute the HC objective value based on T and P.
Return the partition P and tree T with maximal HC objective value.

5.2 Analyzing the Approximation Ratio of ALG

Now that we have defined ALG we are ready to analyze its approximation ratio. Recall that by Observation 6 it is enough to analyze the approximation ratio of cases (a), (b) and the total approximation loss generated by the recursion steps (i.e., by finding α_i, β_i and γ_i).

5.2.1 Analyzing the Approximation Ratio of Case (a) of ALG

In order to analyse the approximation ratio of ALG_{d-w} in our setting we must first recall the definition of instances with not-all-small-weights (as defined by Vainstein et al. [31]).

▶ **Definition 38.** *A metric G is said to have not all small weights if there exist constants (with respect to n_V) $c_0, c_1 < 1$ such that the fraction of weights smaller than $c_0 \cdot D_V$ is at most $1 - c_1$.*

The following theorem was presented in Vainstein et al. [31].

▶ **Theorem 39.** *For any constant $\xi > 0$ and any metric $G = (V, w)$ with not all small weights (with constants c_0 and c_1) we are guaranteed that $\frac{ALG_{d-w}(G)}{OPT(G)} \geq 1 - O(\frac{\xi}{c_0 \cdot c_1})$ and that ALG_{d-w}'s expected running time is at most $f(\frac{1}{\xi}) \cdot n^2$.*

Applying the above theorem with $\xi = \epsilon^5$ to our metric instance G_k yields Proposition 40 (whose proof is deferred to the full version).

▶ **Proposition 40.** *If ALG terminates in case (a) then $\frac{ALG_{d-w}(G_k)}{OPT(G_k)} = \frac{ALG(G_k)}{OPT(G_k)} \geq 1 - \epsilon$.*

5.2.2 Analyzing the Approximation Ratio of Case (b) of ALG

▶ **Proposition 41.** *If ALG terminates in case (b) then $\frac{ALG(G_k)}{OPT(G_k)} \geq 1 - 17\epsilon$.*

Proof. The proof appears in the full version. ◀

5.2.3 Setting the Values α_i, β_i and γ_i

Due to lack of space, we defer the following proofs to the full version.

▶ **Lemma 42.** *For A_i and C_i as defined by our algorithm applied to G_i and for $\alpha_i = n_{V_i}(W_{A_i} + W_{A_i,C_i})(1 - \sqrt{\rho_i})$ we have $ALG(G_i) \geq \alpha_i + ALG(G_{i+1})$.*

▶ **Lemma 43.** *Let $G_i = (V_i, w_i)$ and $G_{i+1} = (V_{i+1}, w_{i+1})$ denote the instances defined by the i and $i+1$ recursion steps. Furthermore, let $\beta_i = n_{V_i}(W_{A_i} + W_{A_i,C_i})$ and $\gamma_i = 1 + 2\sqrt{\rho_i}$. Therefore, $OPT(G_i) \leq \beta_i + \gamma_i OPT(G_{i+1})$.*

Thus, we combine these values in Definition 44.

▶ **Definition 44.** *We define the values α_i, β_i and γ_i as follows*

$$\alpha_i = n_{V_i}(W_{A_i} + W_{A_i,C_i})(1 - \sqrt{\rho_i}); \quad \beta_i = n_{V_i}(W_{A_i} + W_{A_i,C_i}); \quad \gamma_i = 1 + 2\sqrt{\rho_i}.$$

5.2.4 Putting it all Together

Now that we have analyzed the terminal cases of the algorithm (cases (a) and (b)) and that we have set the values of α_i, β_i and γ_i we will combine these results to prove ALG's approximation ratio (as in Observation 6). Due to lack of space we defer the proofs of this section to the full version.

▶ **Proposition 45.** *For α_i, β_i and γ_i as in Definition 44, we have* $\min_i\{\frac{\alpha_i}{\beta_i \Pi_{j=0}^{i-1}\gamma_j}\} \geq 1 - 4\epsilon$.

▶ **Proposition 46.** *For $\gamma_i = 1 + 2\sqrt{\rho_i}$ we have* $\frac{ALG(G_k)}{(\Pi_{i=0}^{k-1}\gamma_i)OPT(G_k)} \geq 1 - 23\epsilon$.

▶ **Theorem 47.** *For any metric G,* $\frac{ALG(G)}{OPT(G)} \geq 1 - 23\epsilon$.

5.3 Analyzing the Running Time of ALG

Consider the definition of ALG. In each recursion step, the algorithm finds the layer to peel off and then recurses. Therefore the running time is defined by the sum of these recursion steps, plus the terminating cases (i.e., either case (a) or case (b)). Recall that case (a) applies ALG_{d-w} on the instance, while case (b) arranges the instance arbitrarily. Therefore, a bound on cases (a) and (b) is simply a bound on the running time of ALG_{d-w} which is given by Theorem 39 [31]. In Lemma 48 we bound the number of recursion steps and subsequently prove Theorem 49 (the proofs of which appears in the full version).

▶ **Lemma 48.** *The number of recursion steps performed by Algorithm 4 is bounded by* $O(\log\log n)$.

▶ **Theorem 49.** *The algorithm ALG is an EPRAS (with running time $O(n^2 \log\log n)$ plus the running time of ALG_{d-w}).*

▶ Remark 50. We remark that one may improve the running time by replacing ALG_{d-w} with any faster algorithm while slightly degrading the quality of the approximation.

References

1 Sara Ahmadian, Vaggos Chatziafratis, Alessandro Epasto, Euiwoong Lee, Mohammad Mahdian, Konstantin Makarychev, and Grigory Yaroslavtsev. Bisect and conquer: Hierarchical clustering via max-uncut bisection. *CoRR*, abs/1912.06983, 2019. arXiv:1912.06983.

2 Noga Alon, Yossi Azar, and Danny Vainstein. Hierarchical clustering: A 0.585 revenue approximation. In Jacob D. Abernethy and Shivani Agarwal, editors, *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, volume 125 of *Proceedings of Machine Learning Research*, pages 153–162. PMLR, 2020. URL: http://proceedings.mlr.press/v125/alon20b.html.

3 Sanjeev Arora, David R. Karger, and Marek Karpinski. Polynomial time approximation schemes for dense instances of np-hard problems. *J. Comput. Syst. Sci.*, 58(1):193–210, 1999. doi:10.1006/jcss.1998.1605.

4 Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 222–231. ACM, 2004. doi:10.1145/1007352.1007355.

5 Kevin Aydin, MohammadHossein Bateni, and Vahab S. Mirrokni. Distributed balanced partitioning via linear embedding. *Algorithms*, 12(8):162, 2019. doi:10.3390/a12080162.

6 MohammadHossein Bateni, Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Raimondas Kiveris, Silvio Lattanzi, and Vahab S. Mirrokni. Affinity clustering: Hierarchical clustering at scale. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6864–6874, 2017. URL: https://proceedings.neurips.cc/paper/2017/hash/2e1b24a664f5e9c18f407b2f9c73e821-Abstract.html.

7 Moses Charikar and Vaggos Chatziafratis. Approximate hierarchical clustering via sparsest cut and spreading metrics. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 841–854, 2017.

8 Moses Charikar, Vaggos Chatziafratis, and Rad Niazadeh. Hierarchical clustering better than average-linkage. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2291–2304, 2019.

9 Moses Charikar, Vaggos Chatziafratis, Rad Niazadeh, and Grigory Yaroslavtsev. Hierarchical clustering for euclidean data. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, pages 2721–2730, 2019. URL: `http://proceedings.mlr.press/v89/charikar19a.html`.

10 Moses Charikar, Mohammad Taghi Hajiaghayi, Howard J. Karloff, and Satish Rao. l^2_2 spreading metrics for vertex ordering problems. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 1018–1027. ACM Press, 2006. URL: `http://dl.acm.org/citation.cfm?id=1109557.1109670`.

11 Gui Citovsky, Giulia DeSalvo, Claudio Gentile, Lazaros Karydas, Anand Rajagopalan, Afshin Rostamizadeh, and Sanjiv Kumar. Batch active learning at scale. *CoRR*, abs/2107.14263, 2021. `arXiv:2107.14263`.

12 Vincent Cohen-Addad, Varun Kanade, Frederik Mallmann-Trenn, and Claire Mathieu. Hierarchical clustering: Objective functions and algorithms. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 378–397, 2018.

13 Sanjoy Dasgupta. A cost function for similarity-based hierarchical clustering. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 118–127, 2016.

14 Wenceslas Fernandez de la Vega and Marek Karpinski. Polynomial time approximation of dense weighted instances of MAX-CUT. *Electron. Colloquium Comput. Complex.*, 64, 1998. URL: `https://eccc.weizmann.ac.il/eccc-reports/1998/TR98-064/index.html`, `arXiv:TR98-064`.

15 Wenceslas Fernandez de la Vega and Claire Kenyon. A randomized approximation scheme for metric MAX-CUT. In *39th Annual Symposium on Foundations of Computer Science, FOCS '98, November 8-11, 1998, Palo Alto, California, USA*, pages 468–471. IEEE Computer Society, 1998. `doi:10.1109/SFCS.1998.743497`.

16 Guy Even, Joseph Naor, Satish Rao, and Baruch Schieber. Divide-and-conquer approximation algorithms via spreading metrics (extended abstract). In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 62–71. IEEE Computer Society, 1995. `doi:10.1109/SFCS.1995.492463`.

17 Uriel Feige and James R. Lee. An improved approximation ratio for the minimum linear arrangement problem. *Inf. Process. Lett.*, 101(1):26–29, 2007. `doi:10.1016/j.ipl.2006.07.009`.

18 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.

19 Mark D. Hansen. Approximation algorithms for geometric embeddings in the plane with applications to parallel processing problems (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 604–609. IEEE Computer Society, 1989. `doi:10.1109/SFCS.1989.63542`.

20 Refael Hassin and Shlomi Rubinstein. Approximation algorithms for maximum linear arrangement. *Inf. Process. Lett.*, 80(4):171–177, 2001. `doi:10.1016/S0020-0190(01)00159-4`.

21 Marek Karpinski and Warren Schudy. Linear time approximation schemes for the gale-berlekamp game and related minimization problems. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 313–322. ACM, 2009. doi:10.1145/1536414.1536458.

22 Claire Kenyon-Mathieu and Warren Schudy. How to rank with few errors. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 95–103. ACM, 2007. doi:10.1145/1250790.1250806.

23 Frank Thomson Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999. doi:10.1145/331524.331526.

24 Benjamin Moseley and Joshua Wang. Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 3094–3103, 2017.

25 Stanislav Naumov, Grigory Yaroslavtsev, and Dmitrii Avdiukhin. Objective-based hierarchical clustering of deep embedding vectors. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 9055–9063. AAAI Press, 2021. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17094.

26 Anand Rajagopalan, Fabio Vitale, Danny Vainstein, Gui Citovsky, Cecilia M. Procopiuc, and Claudio Gentile. Hierarchical clustering of data streams: Scalable algorithms and approximation guarantees. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8799–8809. PMLR, 2021. URL: http://proceedings.mlr.press/v139/rajagopalan21a.html.

27 Satish Rao and Andréa W. Richa. New approximation techniques for some ordering problems. In Howard J. Karloff, editor, *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 25-27 January 1998, San Francisco, California, USA*, pages 211–218. ACM/SIAM, 1998. URL: http://dl.acm.org/citation.cfm?id=314613.314703.

28 R. Ravi, Ajit Agrawal, and Philip N. Klein. Ordering problems approximated: Single-processor scheduling and interval graph completion. In Javier Leach Albert, Burkhard Monien, and Mario Rodríguez-Artalejo, editors, *Automata, Languages and Programming, 18th International Colloquium, ICALP91, Madrid, Spain, July 8-12, 1991, Proceedings*, volume 510 of *Lecture Notes in Computer Science*, pages 751–762. Springer, 1991. doi:10.1007/3-540-54233-7_180.

29 Paul D. Seymour. Packing directed circuits fractionally. *Comb.*, 15(2):281–288, 1995. doi:10.1007/BF01200760.

30 Baris Sumengen, Anand Rajagopalan, Gui Citovsky, David Simcha, Olivier Bachem, Pradipta Mitra, Sam Blasiak, Mason Liang, and Sanjiv Kumar. Scaling hierarchical agglomerative clustering to billion-sized datasets. *CoRR*, abs/2105.11653, 2021. arXiv:2105.11653.

31 Danny Vainstein, Vaggos Chatziafratis, Gui Citovsky, Anand Rajagopalan, Mohammad Mahdian, and Yossi Azar. Hierarchical clustering via sketches and hierarchical correlation clustering. In Arindam Banerjee and Kenji Fukumizu, editors, *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 559–567. PMLR, 2021. URL: http://proceedings.mlr.press/v130/vainstein21a.html.

Robust Communication Complexity of Matching: EDCS Achieves 5/6 Approximation

Amir Azarmehr
Northeastern University, Boston, MA, USA

Soheil Behnezhad
Northeastern University, Boston, MA, USA

Abstract

We study the *robust communication complexity* of maximum matching. Edges of an *arbitrary* n-vertex graph G are *randomly* partitioned between Alice and Bob independently and uniformly. Alice has to send a single message to Bob such that Bob can find an (approximate) maximum matching of the whole graph G. We specifically study the best approximation ratio achievable via protocols where Alice communicates only $\widetilde{O}(n)$ bits to Bob.

There has been a growing interest on the robust communication model due to its connections to the random-order streaming model. An algorithm of Assadi and Behnezhad [ICALP'21] implies a $(2/3 + \varepsilon_0 \sim .667)$-approximation for a small constant $0 < \varepsilon_0 < 10^{-18}$, which remains the best-known approximation for general graphs. For bipartite graphs, Assadi and Behnezhad [Random'21] improved the approximation to .716 albeit with a computationally inefficient (i.e., exponential time) protocol.

In this paper, we study a natural and efficient protocol implied by a random-order streaming algorithm of Bernstein [ICALP'20] which is based on *edge-degree constrained subgraphs* (EDCS) [Bernstein and Stein; ICALP'15]. The result of Bernstein immediately implies that this protocol achieves an (almost) $(2/3 \sim .666)$-approximation in the robust communication model. We present a new analysis, proving that it achieves a much better (almost) $(5/6 \sim .833)$-approximation. This significantly improves previous approximations both for general and bipartite graphs. We also prove that our analysis of Bernstein's protocol is tight.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Random order and robust communication complexity

Keywords and phrases Maximum Matching, Robust Communication Complexity, Edge Degree Constrained Subgraph

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.14

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2305.01070` [4]

Acknowledgements The second author thanks David Wajc for enlightening discussions about going beyond 2/3-approximations via EDCS.

1 Introduction

Given an n-vertex graph $G = (V, E)$, a *matching* is a collection of vertex disjoint edges in G and a *maximum matching* is the matching with the maximum size. In this paper, we study matchings in Yao's (one-way) communication model [14]. The edge-set E is partitioned between two players Alice and Bob. Alice has to send a single message to Bob such that Bob can find an (approximate) maximum matching of the whole graph G. We are particularly interested in the trade-off between the size of the message sent by Alice and the approximation ratio of the output solution. Besides being a natural problem, this communication model is closely related to streaming algorithms and has thus been studied extensively over the years [11, 13, 9, 1, 2].

In order to obtain an *exact* maximum matching, it is known that $\Omega(n^2)$ bits of communication are needed [10]. That is, the trivial protocol where Alice sends her whole input to Bob is optimal. The situation is more interesting for approximate solutions. It is clear that $\Omega(n)$ words of communication are needed for any approximation as the whole matching can be given to Alice. A natural question, therefore, studied in numerous prior works [11, 12, 13, 1, 2], is the best approximation achievable via protocols that have a near-optimal communication complexity of $\widetilde{O}(n) = O(n \operatorname{poly} \log)$.

It is not hard to see that if Alice sends a maximum matching of her input to Bob, then Bob can find a 1/2-approximate matching. There is, however, a more sophisticated approach based on the powerful *edge-degree constrained subgraph (EDCS)* of Bernstein and Stein [7] that achieves an (almost) 2/3-approximation (see the paper of Assadi and Bernstein [3]). This turns out to be the right approximation under an adversarial partitioning of edges. In their seminal paper, Goel, Kapralov, and Khanna [11] proved that obtaining a better than 2/3-approximation requires $n^{1+1/(\log \log n)} \gg n \operatorname{poly} \log n$ communication.

The communication model discussed above is *doubly worst-case* in that both the input graph and the edge partitioning are chosen by an adversary. In this paper, we study the so called *robust communication* model – à la Chakrabarti, Cormode, and McGregor [9] – where the graph G is still chosen by an adversary but its edges are now *randomly* partitioned between Alice and Bob (i.e., each edge is uniformly given either to Alice or Bob independently). This model goes beyond the doubly worst-case scenario discussed above and sheds light on whether the hardness of a problem is inherent to the input graph or rather a pathological partitioning of its edges. Another motivation behind the study of the robust communication model is its connections to *random-order* streams. In particular, almost all known lower bounds for random-order streams are proved in this robust communication model.

While existing protocols for adversarial partitionings already imply an (almost) 2/3-approximation in the robust communication model, a random-order streaming algorithm of Assadi and Behnezhad [1] implies a better bound. Their algorithm starts with an EDCS-based algorithm of Bernstein [6], and then augments it with a number of short *augmenting paths*, achieving a $(2/3 + \varepsilon_0)$-approximation for some fixed constant $0 < \varepsilon_0 < 10^{-18}$. This remains the best-known approximation in general graphs. For bipartite graphs, an entirely different approach of Assadi and Behnezhad [2] achieves a larger .716-approximtaion although their protocol runs in doubly exponential time.

In this paper, we give a new analysis for the EDCS-based protocol of Bernstein [6] showing that, without any augmentation, it already achieves a much better than 2/3-approximation.

▶ **Theorem 1.1.** *Bernstein's protocol [6] with high probability achieves a* $(1-\varepsilon)5/6 \sim .833$ *approximation in the robust communication model using* $O(n \cdot \log n \cdot \operatorname{poly}(1/\varepsilon))$ *words of communication.*

Theorem 1.1 improves, rather significantly, the state-of-the-art approximation for both general and bipartite graphs from .667 [1] and .716 [2] respectively to .833. We note that Bernstein's protocol runs in linear time in the input size; hence Theorem 1.1, in addition to improving approximation, also improves the running time of the algorithm of [2] from doubly exponential to linear. Besides these quantitative improvements, we believe that a more important qualitative implication of Theorem 1.1 is that EDCS, which has been used in the literature to only obtain 2/3 or slightly-larger-than-2/3 approximations in various models, can be used to obtain a significantly better approximation in the robust communication model.

Our analysis can be applied to the more general *multi-party* one-way robust communication model where instead of two players Alice and Bob, the input is randomly partitioned between k players (see Section 3 for the formal definition of the model). This communication model is particularly of interest since any lower bound in it, for any choice of k, also implies a lower bound for random-order streams. We show the following, which generalizes Theorem 1.1:

▶ **Theorem 1.2.** *For any $k \geq 2$ and any $\varepsilon > 0$, Bernstein's protocol [6] in the k-party one-way robust communication model achieves a $(1-\varepsilon)(\frac{2}{3}+\frac{1}{3k})$-approximation of maximum matching using messages of length $O(n \cdot \log n \cdot \text{poly}(1/\varepsilon))$.*

We note that the current best approximation known in the random-order streaming setting for maximum matching is $(2/3 + 10^{-18})$ by Assadi and Behnezhad [1]. Theorem 1.2 implies that either there is a better random-order streaming algorithm for maximum matching (which likely is the case), or else to prove a tight lower bound via the multi-party communication model, one has to consider at least $k \geq 10^{18}/3$ parties!

Finally, we show that our guarantees of Theorems 1.1 and 1.2 are tight for Bernstein's protocol. That is, we show that:

▶ **Theorem 1.3.** *For any $k \geq 2$, there exist an infinite family of graphs G such that the expected approximation ratio of Bernstein's protocol in the k-party one-way robust communication model is at most $(\frac{2}{3} + \frac{1}{3k})$.*

2 Technical Overview

Bernstein's protocol constructs two subgraphs H and U of size $O(n \log n)$ both of which will be communicated to Bob. Subgraph H is constructed solely by Alice who does so by revealing only ε fraction of her input graph. The construction guarantees that for some sufficiently large constant $\beta \geq 1$, every edge $(u, v) \in H$ satisfies $\deg_H(u) + \deg_H(v) \leq \beta$. That is, H has edge-degree upper bounded by β. This already implies that H has at most $O(n\beta) = O(n)$ edges. The subgraph U is simply the set of all the remaining edges (u, v) in the graph G (given either to Alice or Bob) for which $\deg_H(u) + \deg_H(v) \leq \beta - 1$. In other words, all the remaining "underfull" edges whose edge-degree is less than β are added to U. While it is not at all clear that H can be constructed in such a way that guarantees $|U| = O(n \log n)$, Bernstein [6] showed this is indeed possible. At the end, Bob returns a maximum matching of all the edges that he receives.

The subgraph $H \cup U$ can be shown to include an *edge-degree constrained subgraph* (EDCS) of G, which is known to include a $(2/3 - O(\varepsilon))$-approximate maximum matching of the base graph G for $\beta \geq 1/\varepsilon$ [7, 5]. This already implies an (almost) 2/3-approximation in our model. This guarantee is in fact tight for the maximum matching contained in $H \cup U$ as illustrated in Figure 1. In the example of Figure 1, the missed (red dashed) edges have edge-degree β in H, and so they do not belong to U. While the graph in the example of Figure 1 has a perfect matching, any matching in $H \cup U$ can only match 2/3-fraction of vertices.

The crucial insight is that although $H \cup U$ may only include a 2/3-approximate matching of the graph, Bob in addition will also have access to the set E_B of the edges originally given to him in the random partitioning. So instead of $H \cup U$, we need to focus on the size of the maximum matching contained in $H \cup U \cup E_B$. Let us now revisit the example of Figure 1. As we discussed, the set H is only constructed using a small ε fraction of the edges. Moreover, conditioned on H, the subgraph U will also be fully determined regardless

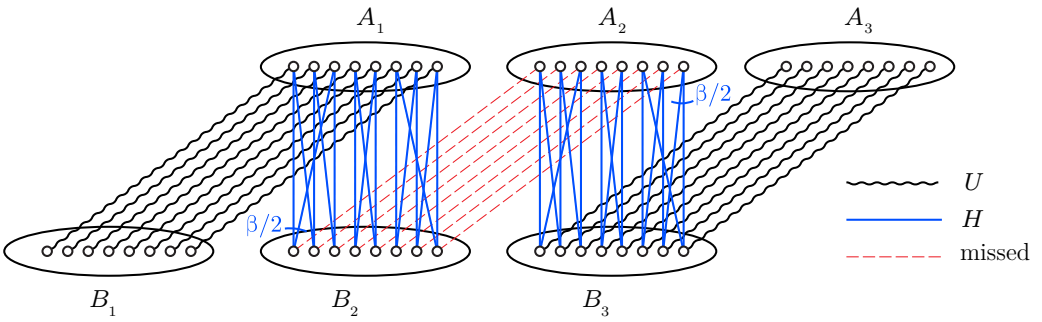


Figure 1 An example where subgraph $H \cup U$ in Bernstein's protocol does not include a better than 2/3-approximation.

of how the edges are partitioned between Alice and Bob. This implies that, *even conditioned on the outcome of H and U*, each dashed edge is given to Bob with probability (almost) 1/2. This results in an (almost) 5/6-approximation in the example of Figure 1: We can combine the 2/3-approximate black matching in U with half of the dashed edges, obtaining an (almost) $\frac{2}{3} + \frac{1}{2} \cdot \frac{1}{3} = \frac{5}{6}$ approximation. We remark that this example already shows that our 5/6-approximation guarantee of Theorem 1.1 is tight for Bernstein's protocol (see Theorem 1.3 for the formal proof).

The nice property of the example of Figure 1 is that subgraph $H \cup U$ includes a 2/3-approximate matching M (the black matching in U) where removing its vertices from the graph still leaves a 1/3-approximate matching in G (the dashed red edges). If we prove that this holds for every graph, then we immediately get an (almost) 5/6-approximation analysis for Bernstein's protocol. Unfortunately, however, this property does not hold for all graphs. In Section 6, we provide examples of H, U such that for *every* matching M in $H \cup U$, it holds that

$$|M| + \frac{1}{2}\mu(G - V(M)) \leq 0.75\mu(G),$$

where $\mu(G - V(M))$ here is the size of maximum matching remained in graph G after removing vertices of M. This implies that this idea is not sufficient to guarantee an (almost) 5/6-approximation for Bernstein's protocol.

In our analysis, instead of first committing to a 2/3-approximate matching in $H \cup U$ and then augmenting it using the edges in E_B, we first commit to a smaller 1/2-approximate matching by fixing an arbitrary maximum matching M^* and taking half of its edges that are given to Bob. The advantage of this smaller 1/2-approximate matching is that it can be augmented much better. Specifically, we show that this 1/2-approximate matching, in expectation, can be augmented by a matching of size (almost) $\mu(G)/3$ using the edges in $H \cup U$, achieving overall a matching of size (almost) $\frac{1}{2}\mu(G) + \frac{1}{3}\mu(G) = \frac{5}{6}\mu(G)$. The proof of why a matching of size $\mu(G)/3$ can be found within the available vertices is the crux of our analysis and is formalized via fractional matchings.

3 Preliminaries

We start by formally defining the robust communication model for maximum matching.

▶ **Definition 3.1.** *In the k-party one-way robust communication model, each edge is assigned independently and uniformly to one of the parties. The i-th party, supplied with the assigned edges and a message m_i from the $(i-1)$-th party, decides what message to send to the $(i+1)$-th*

party. The k-th and last party is responsible for reporting a matching. The communication complexity of a protocol in this model, is defined as the maximum number of words in the messages communicated between the parties, i.e. $\max_i |m_i|$, where $|m_i|$ denotes the number of words in m_i.

In case $k = 2$, we refer to the first party as Alice and to the second party as Bob.

We use $\mu(G)$ to denote the size of the maximum matching in graph G. For an edge e, we define its edge-degree as the sum of the degrees of its endpoints.

3.1 Background on Matching Theory

▶ **Proposition 3.2** (folklore). *Let G be any graph, and let x be a fractional matching on G, such that for every vertex set $S \subseteq V$ that $|S|$ is smaller than $\frac{1}{\varepsilon}$, we have*

$$\sum_{e \in G[S]} x_e \leq \left\lfloor \frac{|S|}{2} \right\rfloor.$$

Then, it holds that $\mu(G) \geq (1-\varepsilon) \sum_e x_e$.

Proof sketch. Let z be another fractional matching where $z_e = (1-\varepsilon)x_e$. If the x satisfies the blossom inequality, i.e. $\sum_{e \in G[S]} x_e \leq \left\lfloor \frac{|S|}{2} \right\rfloor$, for all S of size at most $\frac{1}{\varepsilon}$, Then z satisfies it for all S. To see this, let S be an odd-sized vertex set size at least $\frac{1}{\varepsilon}$ such that $\sum_{e \in G[S]} x_e \geq \left\lfloor \frac{|S|}{2} \right\rfloor$. Then it holds:

$$\sum_{e \in G[S]} z_e = (1-\varepsilon) \sum_{e \in G[S]} x_e \leq \sum_{e \in G[S]} x_e - \frac{1}{2} \leq \frac{|S|}{2} - \frac{1}{2} \leq \left\lfloor \frac{|S|}{2} \right\rfloor.$$

Hence there exists an integral matching of size at least $\sum_e z_e = (1-\varepsilon) \sum_e x_e$. ◀

The following definitions were introduced by Bernstein [6]. The proposition, from the same paper, plays a key role in our analysis.

▶ **Definition 3.3.** *A graph H has bounded edge-degree β, if for all edges $(u,v) \in E_H$ it holds that $d_H(u) + d_H(v) \leq \beta$.*

▶ **Definition 3.4.** *Given a graph G, and a subgraph $H \subseteq G$ an edge $(u,v) \in E_G \setminus E_H$ is (H, β, λ)-underfull if $d_H(u) + d_H(v) < (1-\lambda)\beta$.*

▶ **Proposition 3.5** (Lemma 3.1 from [6]). *Fix any $\varepsilon \in \left[0, \frac{1}{2}\right]$, let λ, β be parameters such that $\lambda \leq \frac{\varepsilon}{384}$, $\beta \geq 50\lambda^{-2} \log\left(\frac{1}{\lambda}\right)$. Consider any graph G, and any subgraph H with bounded edge-degree β. Let U contain all the $(H, \beta, 3\lambda)$-underfull edges in $G \setminus H$. Then $\mu(H \cup U) \geq \left(\frac{2}{3} - \varepsilon\right) \mu(G)$.*

3.2 Concentration Inequalities

We use the following concentration inequalities in our proofs.

▶ **Proposition 3.6** (Chernoff bound). *Let $X_1, \ldots, X_n$ be independent random variables taking values in $[0,1]$. Let $X = \sum X_i$ and let $\mu = \mathbb{E}[X]$. Then, for any $0 < \delta \leq 1$ and $0 < a \leq \mu$, we have*

$$\Pr(X \geq (1+\delta)\mu) \leq \exp\left(-\frac{\delta^2 \mu}{3}\right) \qquad \text{and} \qquad \Pr(X \geq \mu + a) \leq \exp\left(-\frac{a^2}{3\mu}\right).$$

▶ **Definition 3.7** ([8]). *A function $f : \{0,1\}^n \to \mathbb{N}$ is self-bounding if there exist functions $f_1, \ldots, f_n : \{0,1\}^{n-1} \to \mathbb{N}$ such that for all $x \in \{0,1\}^n$ satisfy*

$$0 \leq f(x) - f_i(x^{(i)}) \leq 1 \qquad \forall i \in [n],$$

and

$$\sum_{i=1}^{n} \left(f(x) - f_i(x^{(i)}) \right) \leq f(x).$$

Where $x^{(i)}$ is obtained by dropping the i-th component of x.

▶ **Proposition 3.8** ([8]). *Take a self-bounding function $f : \{0,1\}^n \to \mathbb{N}$, and independent $0-1$ variables $X_1, \ldots, X_n$. Define $Z = f(X_1, \ldots, X_n)$. Then, it holds that*

$$\Pr\left(Z \leq \mathbb{E}\, Z - t\right) \leq \exp\left(\frac{-t^2}{2\,\mathbb{E}\, Z}\right).$$

4 A New Analysis of Bernstein's Protocol

This section is devoted to the proof of Theorems 1.1 and 1.2. We will provide an analysis of Bernstein's protocol (Protocol 1) in the two-party model. To make this analysis applicable to the multi-party model, we assume that each edge is assigned to Bob independently with probability $p \leq \frac{1}{2}$.

We give a description of the protocol for the two-party model here. The multi-party protocol is rather similar and we describe it in the Proof of Theorem 2. Let E_A be the set of edges assigned to Alice, and E_B be the set of edges assigned to Bob. Also, fix a constant $\varepsilon \in \left[0, \frac{1}{2}\right]$ and let $\lambda = \frac{\varepsilon}{384}$, and $\beta = 50\lambda^{-4}$. The protocol is formalized as Protocol 1.

Protocol 1: Bernstein's protocol via EDCS in the two-party one-way robust communication model.

Alice:

1. Take a subsample E_s that includes each edge of E_A independently with probability $\frac{\varepsilon}{1-p}$.
2. Take a subgraph H of bounded edge-degree β from E_s, such that the number of (H, β, λ)-underfull edges in $E_r = E(G) \setminus E_s$ is $O(n \cdot \log n \cdot \text{poly}(1/\varepsilon))$ with high probability. (See Claim 4.1 for the existence of H.)
3. Find the (H, β, λ)-underfull edges of $E_A \setminus E_s$, call them U_A.
4. Communicate $H \cup U_A$ to Bob.

Bob:

1. Return the maximum matching in $E_B \cup H \cup U_A$.

Claim 4.1 shows that Alice can execute step 2, and that Protocol 1 has communication complexity $O(n \cdot \log n \cdot \text{poly}(1/\varepsilon))$. Note that taking into account the randomization in dividing the edges between Alice and Bob, E_s can be considered a uniform sample from the whole edge set that contains each edge with probability ε.

▷ Claim 4.1 (Lemma 4.1 in [6]). Alice, by looking only at the edges of E_s, can take a subgraph $H \subseteq E_s$ that has bounded degree β, and with high probability $E_r = E(G) \setminus E_s$ has at most $O(n \cdot \log n \cdot \text{poly}(1/\varepsilon))$ many (H, β, λ)-underfull edges.

For the analysis, we first construct a fractional matching x of expected size $\left(\frac{2}{3} - O(\varepsilon)\right)\mu(G)$, the support of which is contained in $H \cup U$. Then we show that a fractional matching y can be obtained from x, such that its support is contained in $E_B \cup H \cup U_A$ and has expected size at least $\left(\frac{2}{3} + \frac{p}{3} - O(\varepsilon)\right)\mu(G)$. Finally, we use the structure of y to show that its existence implies $E_B \cup H \cup U_A$ has an integral matching almost as large as the size of y. In Section 5, we show that the approximation ratio is also achieved with high probability.

First, we describe how to obtain the fractional matching x given $H \cup U$, where recall that U is the set of (H, β, λ)-underfull edges in E_r. Fix a maximum matching M^* in E_r. Let M_{in} be the edges of M^* that appear in $H \cup U$, i.e. $M_{\text{in}} = M^* \cap (H \cup U)$, and let $M_{\text{out}} = M^* \setminus M_{\text{in}}$.

1. Start with $H_1 = H$, and $U_1 = U$.
2. For $i = 1, \ldots, \lambda\beta$:
3. Let M_i be a maximum matching in $H_i \cup U_i$.
4. Let $H_{i+1} = H_i \setminus (M_i \setminus M_{\text{in}})$, $U_{i+1} = U_i \setminus (M_i \setminus M_{\text{in}})$.
5. For every edge e, let $x_e = \dfrac{|\{i : e \in M_i\}|}{\lambda\beta}$.

One can think of this process as, starting with $H \cup U$, taking a maximum matching M_i each time, and removing $M_i \setminus M_{\text{in}}$ from the graph, then, letting x_e equal to the fraction of matchings we have taken that include e. Note that the matchings $M_1, \ldots M_{\lambda\beta}$ can intersect only in M_{in}. We will use Proposition 3.5 to show that x has expected size at least $\left(\frac{2}{3} - O(\varepsilon)\right)\mu(G)$.

▶ **Lemma 4.2.** *It holds that* $\mathbb{E}\left[\sum_e x_e\right] \geq \left(\frac{2}{3} - \frac{5}{3}\varepsilon\right)\mu(G)$.

Proof. We apply Proposition 3.5 to $G_i = (H \cup E_r) \setminus \left(\bigcup_{j<i} M_j \setminus M_{\text{in}}\right)$, H_i, and U_i. After removing a matching from H, for any edge, its degree in H will decrease by at most 2. Also, U contains all the (H, β, λ)-underfull edges of E_r. Hence, U_i contains all the edges of $G_i \setminus H_i$ that have H_i-degree smaller than $(1-\lambda)\beta - 2(i-1) \geq (1-3\lambda)\beta$. Therefore, Proposition 3.5 implies

$$|M_i| \geq \left(\frac{2}{3} - \varepsilon\right)\mu(G_i).$$

Also, notice that G_i always includes M^*, consequently it holds $\mu(G_i) = \mu(E_r)$, and we have:

$$\sum_e x_e \geq \frac{1}{\lambda\beta}\sum_i |M_i| \geq \left(\frac{2}{3} - \varepsilon\right)\mu(E_r)$$

Taking into account the fact that $\mathbb{E}\left[\mu(E_R)\right] \geq (1-\varepsilon)\mu(G)$, we get:

$$\mathbb{E}\left[\sum_e x_e\right] \geq \left(\frac{2}{3} - \varepsilon\right)(1-\varepsilon)\mu(G) \geq \left(\frac{2}{3} - \frac{5}{3}\varepsilon\right)\mu(G). \qquad \blacktriangleleft$$

To describe how y is obtained, we condition on E_s, thereby fixing H, U, and x. The support of y is included in $E_B \cup H \cup U_A$, i.e. the edges that Bob will have access to in the end. We show that y has expected size at least about $p \cdot \mu(E_r) + (1-p)\sum_e x_e$, where the randomness is over how the remaining edges E_r are divided between Alice and Bob. Lifting the condition on E_s, the expectation of this value, by Lemma 4.2, is larger than $\left(\frac{2}{3} + \frac{p}{3} - O(\varepsilon)\right)\mu(G)$.

After drawing E_B, take a matching M', which includes each edge of M_{in} independently with probability p, and includes each edge of $M_{\text{out}} \cap E_B$ independently with probability $1-\varepsilon$. Note, that conditioned on E_s, each edge of M_{out} is assigned to Bob with probability $\frac{p}{1-\varepsilon}$. Hence, each edge of M_{out} ends up in M' with probability $\frac{p}{1-\varepsilon}(1-\varepsilon) = p$, i.e. M' includes each edge of M^* independently with probability p.

For any edge $e \notin M^*$, define p_e as the probability of e not being adjacent to any edge in M'. Notice, p_e is simply equal to $(1-p)$ to the power of the number of edges in M' that are adjacent to e. We define matching $\hat{y}$ as follows:

$$\hat{y}_e = \begin{cases} 1 & \text{if } e \in M', \\ x_e & \text{if } e \in M^* \setminus M', \\ 0 & \text{if } e \notin M^* \text{ and } e \text{ is adjacent to an edge of } M', \\ (1-p) \cdot \dfrac{x_e}{p_e} & \text{otherwise.} \end{cases}$$

We then scale down $\hat{y}$ by a factor of $1+\varepsilon$, and zero out some edges to obtain a fractional matching. Formally, we let:

$$y_{(u,v)} = \begin{cases} 0 & \text{if } \hat{y}_u/(1+\varepsilon) > 1 \text{ or } \hat{y}_v/(1+\varepsilon) > 1, \\ \dfrac{\hat{y}_{(u,v)}}{1+\varepsilon} & \text{otherwise.} \end{cases}$$

▶ **Lemma 4.3.** *Conditioned on E_s, it holds that*

$$\mathbb{E}\left[\sum_e y_e\right] \geq (1-3\varepsilon)p \cdot \mu(E_r) + (1-3\varepsilon)(1-p)\sum_e x_e - 2\varepsilon\mu(G).$$

Proof. All the arguments made in this proof are conditioned on E_s.

▷ Claim 4.4. For every vertex u, it holds that $\mathbb{E}[\hat{y}_u] = p \cdot \chi_{M^*}(u) + (1-p)x_u$, where $\chi_{M^*}(u)$ is equal to 1 if u is covered by M^* and zero otherwise.

Proof. First, consider a vertex u that is covered by M^*, say by edge $e^* \in M^*$. When e^* appears in M', we have $\hat{y}_{e^*} = 1$, and for all the other edges e adjacent to u, the value of $\hat{y}_e$ is equal to zero. Thus, we will have $\hat{y}_u = 1$, i.e. $\mathbb{E}[\hat{y}_u \mid e^* \in M^*] = 1$.

Now, we condition on $e^* \notin M'$. In this case, we will have $\hat{y}_{e^*} = x_{e^*}$. Also, for any other edge e adjacent to u, the probability that e is not adjacent to any edge in M' is equal to $\frac{p_e}{1-p}$. Thus with probability $\frac{p_e}{1-p}$ it holds that $\hat{y}_e = (1-p)\cdot\frac{x_e}{p_e}$, and we will have $\hat{y}_e = 0$ otherwise. Hence, we can write:

$$\mathbb{E}[\hat{y}_u \mid e^* \notin M'] = x_{e^*} + \sum_{\substack{e \ni u \\ e \neq e^*}} \frac{p_e}{1-p} \cdot \left((1-p)\cdot\frac{x_e}{p_e}\right) = \sum_{e \ni u} x_e = x_u.$$

Therefore, for a vertex u that is covered by M^*, it holds that

$$\mathbb{E}[\hat{y}_u] = p \cdot \mathbb{E}[\hat{y}_u \mid e^* \in M'] + (1-p)\cdot\mathbb{E}[\hat{y}_u \mid e^* \notin M'] = p + (1-p)x_u.$$

The case where the vertex u is not covered by M^* follows similarly. For each edge e adjacent to u we have $\hat{y}_e = (1-p)\cdot\frac{x_e}{p_e}$ with probability p_e, and we have $\hat{y}_e = 0$ otherwise. Thus

$$\mathbb{E}[\hat{y}_u] = \sum_{e \ni u} p_e \cdot \left((1-p)\cdot\frac{x_e}{p_e}\right) = (1-p)x_u. \qquad \triangleleft$$

The following claim helps us show that we do not lose much of $\hat{y}$ when we scale it down and zero out some of the edges.

▷ Claim 4.5. For every vertex u, we have $\hat{y}_u \leq 1$ if u is not covered by M^* or $x_u \leq \frac{1}{2}$, otherwise it holds that $\Pr(\hat{y}_u > 1+\varepsilon) \leq \varepsilon$.

Proof. Consider a vertex u not covered by M^*. For each edge e adjacent to u it holds that $p_e \geq 1-p$ because e has at most one neighbouring edge in M^*. Hence we have:

$$\hat{y}_u \leq \sum_{e \ni u} (1-p) \cdot \frac{x_e}{p_e} \leq \sum_{e \ni u} x_e \leq 1.$$

Now take a vertex u that is covered by M^*, say by edge $e^* \in M^*$. If $e^* \in M'$, then we have $\hat{y}_u = 1$. Therefore we assume $e^* \notin M'$, and accordingly $\hat{y}_e = x_e$. For any other edge e it holds that $p_e \geq (1-p)^2$. Hence we have:

$$\hat{y}_u \leq x_{e^*} + \sum_{\substack{e \ni u \\ e \neq e^*}} (1-p) \cdot \frac{x_e}{p_e} \leq x_{e^*} + \sum_{\substack{e \ni u \\ e \neq e^*}} \frac{x_e}{1-p} \leq 2 \sum_{e \ni u} x_e = 2x_u.$$

Thus, if it holds that $x_u \leq \frac{1}{2}$, then it follows $\hat{y}_u \leq 1$.

For the other cases, we use the Chernoff bound to show that with high probability $\hat{y}_u$ is not much larger than 1. As mentioned before, if e^* appears in M', it holds that $\hat{y}_u = 1$. Therefore, we condition on $e^* \notin M'$. We express $X = y_u - y_{e^*}$ as a sum of independent random variables that take values in $[0, 4/\lambda\beta]$. Note that since the edges outside M_{in} appear in at most one M_i, for $e \notin M^*$ we have $x_e \leq \frac{1}{\lambda\beta}$.

Take an edge $e = (u,v) \neq e^*$. If v is not matched in M^* to another neighbour of u, then the value of y_e is independent of the value of the other edges adjacent to u. It is equal to $(1-p) \cdot \frac{x_e}{p_e} \leq \frac{2}{\lambda\beta}$ with probability $\frac{p_e}{1-p}$, and zero otherwise.

If v is matched in M^* to another neighbour v' of u. Let $e' = (u,v')$. Then the value of $y_e + y'_e$ is independent of the value of the other edges adjacent to u. It is equal to $(1-p) \cdot \frac{x_e + x'_e}{(1-p)^2} \leq \frac{4}{\lambda\beta}$, with probability $(1-p)$, and zero otherwise.

Thus, by pairing the edges that are matched together, we can express X as a sum of independent random variables in $[0, 4/\lambda\beta]$. The expectation of X, as calculated in Claim 4.4, is equal to $x_u - x_{e^*} \leq 1$. From the Chernoff bound we get:

$$\begin{aligned}
\Pr(\hat{y}_u > 1+\varepsilon) &\leq \Pr(X > \mathbb{E}\,X + \varepsilon) \\
&\leq \Pr\left(X \cdot \frac{\lambda\beta}{4} > \mu\frac{\lambda\beta}{4} + \varepsilon\frac{\lambda\beta}{4}\right) \\
&\leq \exp\left(-\frac{\varepsilon^2\lambda^2\beta^2/16}{3\mu\lambda\beta/4}\right) \\
&\text{(By Chernoff bound, noting that } X\lambda\beta/4 \text{ is a sum of independent random variables in } [0,1].) \\
&= \exp\left(-\frac{\varepsilon^2\lambda\beta}{12}\right) \\
&< \varepsilon, \qquad \text{(Since } \lambda = \tfrac{\varepsilon}{384} \text{ and } \beta = 50\lambda^{-4})
\end{aligned}$$

concluding the proof. ◁

We analyze $\mathbb{E}[y_u]$. Consider generating y_u, in two steps. First, for every edge (u,v), we let $y_{(u,v)}$ be equal to $\frac{1}{1+\varepsilon}\hat{y}_{(u,v)}$ if $\hat{y}_v \leq 1+\varepsilon$, and zero otherwise. Then, we zero out y for all the edges adjacent to u if $\hat{y}_u > 1+\varepsilon$.

In the first step, we lose a factor $(1+\varepsilon)$ when we scale $\hat{y}_u$ down. Also, by Claim 4.5, when we zero out edge (u,v) because $y_v > 1+\varepsilon$, we lose an ε-fraction from each edge, and consequently from $\mathbb{E}[\hat{y}_u]$. In the second step, again by Claim 4.5, if $x_u \le \frac{1}{2}$ or u is not covered by M^* we lose nothing. Otherwise, we zero out all the edges with probability at most ε. We have $\hat{y}_u \le 2x_u \le 2$, hence we lose an additive factor of 2ε. Overall for any vertex u we get:

$$\mathbb{E}[y_u] \ge (1-\varepsilon)\frac{\mathbb{E}[\hat{y}_u]}{1+\varepsilon} - 2\varepsilon \ge (1-3\varepsilon)\left(p \cdot \chi_{M^*}(u) + (1-p)x_u\right) - 2\varepsilon,$$

and for any vertex u with $x_u \le \frac{1}{2}$, we get:

$$\mathbb{E}[y_u] \ge (1-3\varepsilon)\left(p \cdot \chi_{M^*}(u) + (1-p)x_u\right).$$

Notice that since the sum of the components of x is at most $\mu(G)$, there are at most $2\mu(G)$ vertices with $x_u \ge \frac{1}{2}$. Thus, by summing the last two equations over u we get:

$$\begin{aligned}
\sum_u \mathbb{E}[y_u] &= \sum_{u\in V(M^*)} \mathbb{E}[y_u] + \sum_{u\notin V(M^*)} \mathbb{E}[y_u] \\
&\ge \left((1-3\varepsilon)\sum_{u\in V(M^*)} p + (1-p)x_u\right) + \left((1-3\varepsilon)\sum_{u\notin V(M^*)} (1-p)x_u\right) - 2\varepsilon\cdot 2\mu(G) \\
&= (1-3\varepsilon)2p\cdot|M^*| + (1-3\varepsilon)(1-p)\sum_u x_u - 4\varepsilon\mu(G).
\end{aligned}$$

Recall that $|M^*| = \mu(E_r)$. Finally, by dividing both sides by 2, we get:

$$\sum_e \mathbb{E}[y_e] \ge (1-3\varepsilon)p\cdot\mu(E_r) + (1-3\varepsilon)(1-p)\sum_e x_e - 2\varepsilon\mu(G). \qquad \blacktriangleleft$$

Now we lift the condition on E_s.

▶ **Lemma 4.6.** *It holds that* $\mathbb{E}\left[\sum_e y_e\right] \ge \left(\frac{2}{3} + \frac{p}{3} - 6\varepsilon\right)\mu(G)$.

Proof. We have:

$$\begin{aligned}
\mathbb{E}\left[\sum_e y_e\right] &= \mathbb{E}\left[\mathbb{E}\left[\sum_e y_e \;\middle|\; E_s\right]\right] \\
&\ge \mathbb{E}\left[(1-3\varepsilon)p\cdot\mu(E_r) + (1-3\varepsilon)(1-p)\sum_e x_e - 2\varepsilon\mu(G)\right] \\
&\ge (1-3\varepsilon)p\cdot(1-\varepsilon)\mu(G) + (1-3\varepsilon)(1-p)\left(\frac{2}{3} - \frac{5}{3}\varepsilon\right)\mu(G) - 2\varepsilon\mu(G) \\
&\qquad\qquad \text{(by } \mathbb{E}[\mu(E_r)] = (1-\varepsilon)\mu(G) \text{ and Lemma 4.2)} \\
&\ge \left(\frac{2}{3} + \frac{p}{3} - \left(\frac{7}{6}p + \frac{29}{6}\right)\varepsilon\right)\mu(G) \\
&\ge \left(\frac{2}{3} + \frac{p}{3} - 6\varepsilon\right)\mu(G).
\end{aligned}$$

We show that $E_B \cup H \cup U_A$ has an integral matching almost as large as the size of y.

▶ **Lemma 4.7.** *There exists a matching of size* $(1-3\varepsilon)\sum_e y_e$ *in* $E_B \cup H \cup U_A$.

Proof. Notice that for every edge e, except the edges of M^* which is a matching, it holds that $y_e \leq \frac{4}{\lambda\beta} \leq \varepsilon^3$. Therefore, for any vertex set $S \subseteq V$ that $|S|$ is smaller than $\frac{1}{\varepsilon}$, we have:

$$\sum_{e \in G[S]} x_e = \sum_{e \in G[S] \cap M^*} x_e + \sum_{e \in G[S] \setminus M^*} x_e \leq |G[S] \cap M^*| + \frac{1}{\varepsilon^2}\varepsilon^3 \leq \left\lfloor \frac{|S|}{2} \right\rfloor + \varepsilon.$$

Hence, we can apply Proposition 3.2 to $(1-2\varepsilon)y$, to get $\mu(E_B \cup H \cup U_A) \geq (1-3\varepsilon)\sum_e y_e$. ◀

Proof of Theorem 1.1. By Claim 4.1, Bernstein's protocol (Protocol 1) is implementable using only $O(n \cdot \log n \cdot \text{poly}(1/\varepsilon))$ words of communication.

By Lemma 4.6 there exists a fractional matching y of expected size $\left(\frac{2}{3} + \frac{p}{3} - 6\varepsilon\right)\mu(G)$. Putting this together with Lemma 4.7, we can conclude Protocol 1 achieves a $\left(\frac{2}{3} + \frac{p}{3} - 9\varepsilon\right)$ approximation ratio. To see this approximation ratio is also achieved with high probability, refer to Section 5. Finally, letting $p = \frac{1}{2}$ and rescaling ε proves the theorem. ◀

Proof of Theorem 1.2. We need to adjust Protocol 1 for the k-party model. The first party will sample each of its edges independently with probability $\varepsilon/(1-1/k)$ to obtain E_s. It will then construct the subgraph $H \subseteq E_s$ with bounded edge-degree, and send it to the next party along with the (H, β, λ)-underfull edges. Each of the next parties, except the last, communicates the (H, β, λ)-underfull edges it has been assigned along with the edges in the message it has received, to the next party. Finally, the last party will report the maximum matching in the graph consisting of all the edges to which it has access.

This way, setting $p = \frac{1}{k}$, the first $k-1$ parties will act as Alice in our analysis, and the last party acts as Bob. Hence, by a similar argument as in the Proof of Theorem 1, Protocol 1 achieves a $(\frac{2}{3} + \frac{1}{3k} - 9\varepsilon)$ approximation ratio, and a rescaling of ε proves the theorem. ◀

5 From Expectation to High Probability

In this section, we show that with a slight modification, Bernstein's protocol (Protocol 1) achieves the $\frac{5}{6}$-approximation with high probability. To do so, Alice should send all the edges to Bob when the number of edges is too small.

▷ Claim 5.1. Without loss of generality, we can assume $\mu(G) = \Omega(\log n)$.

Proof. A charging argument can be used to show that the number of edges in G is less than $2n\mu(G)$. Fix a maximum matching M in G. For any edge e, charge a unit to an edge of M that is adjacent to e. Such an edge must exist since M is a maximum matching. This way, we charge once for every edge in G, and every edge of M is charged at most n times through each of its endpoints.

To see why the claim is true, note that in case $\mu(G)$ is too small, i.e. $\mu(G) = O(\log n)$, the number of edges in the graph will be $O(n \log n)$ and Alice can send all of its edges to Bob. ◁

▶ **Lemma 5.2.** *Assuming that $\mu(G) = \Omega(\log n)$, whatever approximation ratio Protocol 1 achieves in expectation, it will achieve with high probability.*

Proof. We condition on the sample edge set E_s, thereby fixing H and U. Bob will have access to the edges of $H \cup U$ because they are either assigned to Bob, or they are communicated to Bob by Alice. Let $e_1, \ldots, e_k$ be the other edges, i.e. the edges of $E_r \setminus U$. Each of these edges is assigned to Bob independently with probability $\frac{1/2}{1-\varepsilon}$.

We define a self-bounding function $f : \{0,1\}^k \to \mathbb{Z}$. For $x \in \{0,1\}^k$, the value of $f(x)$ is equal to the maximum matching of $H \cup U \cup E_x$, where $E_x = \{e_i \mid x_i = 1\}$. Equivalently, $f(x)$ is the size of the output matching when the edges $\{e_i \mid x_i = 1\}$ are assigned to Bob, i.e. $E_B \cup H \cup U_A$ is equal to $E_x \cup H \cup U$. Also, let $f_i(x^{(i)}) = f(x_1, \dots x_{i-1}, 0, x_{i+1}, \dots x_k)$.

Take any $x \in \{0,1\}^k$. Notice that $f_i(x^{(i)})$ is equal to $\mu(E_x \setminus e_i)$, and removing an edge from a graph, will decrease its maximum matching by at most 1. Therefore, it holds:

$$0 \le f(x) - f_i(x^{(i)}) \le 1, \qquad \forall i : 1 \le i \le k.$$

Take the maximum matching M in $H \cup U \cup E_x$, and let I be the indices of the edges in $E_x \cap M$, i.e. $I = \{i \mid x_i = 1 \text{ and } e_i \in M\}$. For any $i \notin I$, the edge set $E_x \setminus e_i$ includes M. Therefore, $f_i(x^{(i)})$ is equal to $f(x)$, and we have:

$$\sum_{i=1}^{k} \Big(f(x) - f_i(x^{(i)}) \Big) \le |I| = f(x).$$

Thus, f is a self-bounding function.

We can now apply Proposition 3.8. Let X_i be the indicator variable that e_i is assigned to Bob, i.e. X_i is equal to 1 when $e_i \in E_B$. Let $Z = f(X_1, \dots, X_k)$, and $\mu = \mathbb{E}[Z] = r\mu(G)$. That is, Z is the size of the output matching, and r is the approximation ratio that Protocol 1 achieves in expectation. By Proposition 3.8, we have:

$$\Pr\left(Z \le r\mu(G) - \sqrt{2\mu(G)\log n}\right) \le \exp\left(-\frac{2\mu(G)\log n}{2\mu(G)}\right) = \frac{1}{n}.$$

Thus, with high probability Protocol 1 outputs a matching of size $(1 - o(1))r\mu(G)$. Note that the deviation is $o(1)$ by the assumption that $\mu(G) = \Omega(\log n)$. ◀

6 Some Instances for Bernstein's Protocol

In this section, we first prove Theorem 1.3 that our analysis of Bernstein's protocol in Theorems 1.1 and 1.2 are tight. Then, we formalize a remark we made in Section 2.

Proof of Theorem 1.3. Consider a bipartite graph $G(L, R)$, such that $|L| = |R|$. Where L consists of three equally-sized groups of vertices A_1, A_2, and A_3, and similarly R consists of B_1, B_2, and B_3. The induced subgraphs $M_1 = G[A_1, B_1]$, $M_2 = G[A_2, B_2]$, and $M_3 = G[A_3, B_3]$ are perfect matchings. The induced subgraphs $K_1 = G[A_1, B_2]$ and $K_2 = G[A_2, B_3]$ are complete bipartite graphs, and there are no other edges in the graph (see Figure 1). Note that G has a perfect matching, i.e. the size of the maximum matching is equal to $|L| = |R|$. Let $\beta \le \frac{|V(G)|}{12k}$ which is $O(|V(G)|)$.

Let E_L be the set of edges assigned to the last party. To upper bound the approximation ratio of the multi-party protocol, we construct a vertex cover for $E_L \cup H \cup U$. It is a well-known fact that the size of the minimum vertex cover is larger than the size of the maximum matching. With high probability, the first party can take H to be completely inside $K_1 \cup K_2$, so that U will be equal the $M_1 \cup M_3$, and no edges of M_2 will appear in $H \cup U$.

Let $X = V(M_2 \cap E_L) \cap A_2$, i.e. X includes one endpoint from every edge of M_2 that is assigned to the last party. We claim $A_1 \cup B_3 \cup X$ is a vertex cover for $E_L \cup H \cup U$. This is true because A_1 covers the edges of M_1 and K_1, B_3 covers the edges of M_3 and K_2, and X covers all the remaining edges, which is $E_L \cap M_2$.

Conditioned on the H as described above, each edge of M_2 will be assigned to Bob with probability $\frac{1/k}{1-\varepsilon}$. Thus the expected size of the vertex cover is equal to

$$|A_1| + |B_3| + \frac{1/k}{1-\varepsilon}|A_2| = \left(\frac{1}{3} + \frac{1}{3} + \frac{1/k}{1-\varepsilon} \cdot \frac{1}{3}\right)\mu(G) \leq (1+2\varepsilon)\left(\frac{2}{3} + \frac{1}{3k}\right)\mu(G).$$

Letting ε be arbitrarily small proves the theorem. ◀

As mentioned in Section 2, the graph discussed in Theorem 1.3 (see Figure 1) has a nice property, i.e. there exists a large matching M such that $G - V(M)$ also has a large matching. As the output of Bernstein's protocol has expected size of at least $|M| + \frac{1}{2}\mu(G - V(M))$, which in this case is equal to $\frac{5}{6}\mu(G)$, this property might seem useful to analyze the protocol. However, the following claim shows that such an M does not generally exist.

▷ Claim 6.1. There exists a graph G, with arbitrarily large number of vertices, such that for $\beta \leq |V(G)|/4$, there is a choice of H and U, such that every matching M in $H \cup U$ satisfies $|M| + \frac{1}{2}\mu(G - V(M)) \leq 0.75\mu(G)$. Where here H is subgraph with bounded edge-degree β, and U is the set of the underfull edges in $G \setminus H$, i.e. the edges of $G \setminus H$ with H-degree smaller than $\beta - 1$.

Proof. Let $G(L, R)$ be a bipratite graph, such that $|L| = |R|$. Where L consists of four equally-sized groups of vertices A_1, A_2, A_3, and A_4, and similarly R consists of B_1, B_2, B_3, and B_4. The induced subgraphs $M_1 = G[A_1, B_1]$, $M_2 = G[A_2, B_2]$, $M_3 = G[A_3, B_3]$, and $M_4 = G[A_4, B_4]$ are perfect matchings. The induced subgraphs $K_1 = G[A_1, B_2]$, $K_2 = G[A_2, B_3]$, and $K_3 = G[A_3, B_4]$ are complete bipartite graphs, and there are no other edges in the graph. Note that G has a perfect matching (see Figure 2).

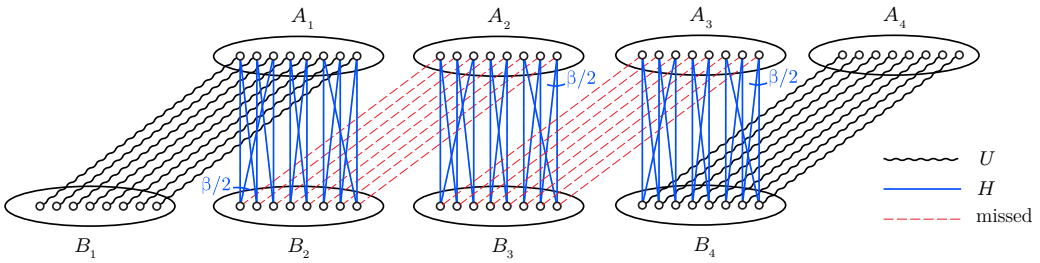


Figure 2 An example where every matching M satisfies $|M| + \frac{1}{2}\mu(G - V(M)) \leq 0.75\mu(G)$.

Let H be a $(\beta/2)$-regular subgraph of $K_1 \cup K_2 \cup K_3$. The corresponding U is equal to $M_1 \cup M_4$, and none of the edges in $M_2 \cup M_3$ appear in $H \cup U$. We prove that $\max_M |M| + \frac{1}{2}\mu(G - V(M)) \leq 0.75\mu(G)$, where M ranges over all the matchings in $H \cup U$. We say that a matching is optimal if it achieves the maximum possible value for $|M| + \frac{1}{2}\mu(G - V(M))$.

First, we prove there exists an optimal matching that includes all of $M_1 \cup M_4$. To see this, take an optimal matching M. Take any vertex u in A_1, and let its adjacent edge in M_1 be e. If u is covered by some edge $e' \in M$, then removing e' from M and adding e does not decrease $|M| + \frac{1}{2}\mu(G - V(M))$. Because this would not change $|M|$ and can only increase $\mu(G - V(M))$. Also, when u is not covered by M, adding the e to M will cause $|M|$ to grow by one, and $\mu(G - V(M))$ to decrease by at most one. A similar argument holds for the vertices in B_4. Hence, by repeatedly adding such edges, we can obtain an optimal matching containing $M_1 \cup M_4$.

Now we can restrict our attention to $A_2 \cup A_3 \cup B_2 \cup B_3$. We claim no matter what the rest of M (a.k.a. $M \setminus (M_1 \cup M_2) = M \cap K_2$) is, the value of $|M| + \frac{1}{2}\mu(G - V(M))$ would be the same. Because if $|M \cap K_2|$ is equal to k, it holds that $\mu(G - V(M)) = \frac{1}{2}\mu(G) - 2k$.

Hence the optimal value of $|M| + \frac{1}{2}\mu(G - V(M))$ is equal to

$$|M_1| + |M_2| + k + \frac{1}{2}\left(\frac{1}{2}\mu(G) - 2k\right) = \frac{3}{4}\mu(G).$$

To see why $\mu(G - V(M)) = \frac{1}{2}\mu(G) - 2k$, note that since $M_1 \cup M_4 \subseteq M$, any vertex of $B_2 \cup A_3$ is a singleton in $G - V(M)$. Hence, a maximum matching in $G - V(M)$ is the set of edges in $M_2 \cup M_3$ that are not adjacent to an edge of M, which has size $\frac{1}{2}\mu(G) - 2k$. ◁

References

1. Sepehr Assadi and Soheil Behnezhad. Beating Two-Thirds For Random-Order Streaming Matching. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, pages 19:1–19:13, 2021. doi:10.4230/LIPIcs.ICALP.2021.19.
2. Sepehr Assadi and Soheil Behnezhad. On the Robust Communication Complexity of Bipartite Matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, pages 48:1–48:17, 2021. doi:10.4230/LIPIcs.APPROX/RANDOM.2021.48.
3. Sepehr Assadi and Aaron Bernstein. Towards a Unified Theory of Sparsification for Matching Problems. In *2nd Symposium on Simplicity in Algorithms, SOSA 2019, January 8-9, 2019, San Diego, CA, USA*, pages 11:1–11:20, 2019. doi:10.4230/OASIcs.SOSA.2019.11.
4. Amir Azarmehr and Soheil Behnezhad. Robust Communication Complexity of Matching: EDCS Achieves 5/6 Approximation, 2023. arXiv:2305.01070.
5. Soheil Behnezhad. Improved Analysis of EDCS via Gallai-Edmonds Decomposition. *CoRR*, abs/2110.05746, 2021. arXiv:2110.05746.
6. Aaron Bernstein. Improved Bounds for Matching in Random-Order Streams. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 12:1–12:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.12.
7. Aaron Bernstein and Cliff Stein. Fully Dynamic Matching in Bipartite Graphs. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 167–179. Springer, 2015. doi:10.1007/978-3-662-47672-7_14.
8. Stephane Boucheron, Gabor Lugosi, and Pascal Massart. On concentration of self-bounding functions. *Electronic Journal of Probability*, 14(none):1884–1899, 2009.
9. Amit Chakrabarti, Graham Cormode, and Andrew McGregor. Robust lower bounds for communication and stream computation. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 641–650, 2008. doi:10.1145/1374376.1374470.
10. Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005. doi:10.1016/j.tcs.2005.09.013.
11. Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 468–485, 2012. doi:10.1137/1.9781611973099.41.
12. Michael Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1679–1697. SIAM, 2013. doi:10.1137/1.9781611973105.121.

13 Michael Kapralov. Space Lower Bounds for Approximating Maximum Matching in the Edge Arrival Model. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1874–1893. SIAM, 2021. doi:10.1137/1.9781611976465.112.

14 Andrew Chi-Chih Yao. Some Complexity Questions Related to Distributive Computing (Preliminary Report). In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 209–213. ACM, 1979. doi:10.1145/800135.804414.

Improved Approximation Algorithms by Generalizing the Primal-Dual Method Beyond Uncrossable Functions

Ishan Bansal
Operations Research and Information Engineering, Cornell University, Ithaca, NY, USA

Joseph Cheriyan
Department of Combinatorics and Optimization, University of Waterloo, Canada

Logan Grout
Operations Research and Information Engineering, Cornell University, Ithaca, NY, USA

Sharat Ibrahimpur
Department of Mathematics, London School of Economics and Political Science, UK

Abstract

We address long-standing open questions raised by Williamson, Goemans, Vazirani and Mihail pertaining to the design of approximation algorithms for problems in network design via the primal-dual method (Combinatorica 15(3):435-454, 1995). Williamson et al. prove an approximation ratio of two for connectivity augmentation problems where the connectivity requirements can be specified by uncrossable functions. They state: "Extending our algorithm to handle non-uncrossable functions remains a challenging open problem. The key feature of uncrossable functions is that there exists an optimal dual solution which is laminar ... A larger open issue is to explore further the power of the primal-dual approach for obtaining approximation algorithms for other combinatorial optimization problems."

Our main result proves a 16-approximation ratio via the primal-dual method for a class of functions that generalizes the notion of an uncrossable function. There exist instances that can be handled by our methods where none of the optimal dual solutions have a laminar support.

We present applications of our main result to three network-design problems.

1. A 16-approximation algorithm for augmenting the family of small cuts of a graph G. The previous best approximation ratio was $O(\log |V(G)|)$.
2. A $16 \cdot \lceil k/u_{min} \rceil$-approximation algorithm for the Cap-k-ECSS problem which is as follows: Given an undirected graph $G = (V, E)$ with edge costs $c \in \mathbb{Q}^E_{\geq 0}$ and edge capacities $u \in \mathbb{Z}^E_{\geq 0}$, find a minimum cost subset of the edges $F \subseteq E$ such that the capacity across any cut in (V, F) is at least k; u_{min} (respectively, u_{max}) denote the minimum (respectively, maximum) capacity of an edge in E, and w.l.o.g. $u_{max} \leq k$. The previous best approximation ratio was $\min(O(\log |V|), k, 2u_{max})$.
3. A 20-approximation algorithm for the model of $(p, 2)$-Flexible Graph Connectivity. The previous best approximation ratio was $O(\log |V(G)|)$, where G denotes the input graph.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Approximation algorithms, Edge-connectivity of graphs, f-Connectivity problem, Flexible Graph Connectivity, Minimum cuts, Network design, Primal-dual method, Small cuts

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.15

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2209.11209v2 [3]

Funding *Ishan Bansal*: Partially supported by Office of Naval Research (ONR) Grant N00014-21-1-2575.
Joseph Cheriyan: Supported in part by NSERC, RGPIN-2019-04197.

Sharat Ibrahimpur: Received funding from the following sources: NSERC grant 327620-09 and an NSERC DAS Award, European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. ScaleOpt–757481), and Dutch Research Council NWO Vidi Grant 016.Vidi.189.087.

Acknowledgements We thank the anonymous reviewers and the ICALP PC for their comments. We are grateful to Cedric Koh and Madison Van Dyk for reading a preliminary version, and for their detailed comments and feedback.

1 Introduction

The primal-dual method is a well-known algorithmic discovery of the past century. Kuhn (1955) [25] presented a primal-dual algorithm for weighted bipartite matching, and Dantzig et al. (1957) [9] presented a generalization for solving linear programs. Primal-dual methods for problems in combinatorial optimization are based on linear programming (LP) relaxations; the associated linear programs (LPs) are crucial for the design and analysis of these algorithms. A key feature of the primal-dual method is that it does not require solving the underlying LPs, which makes it attractive for both theoretical studies and real-world applications. Several computational studies of some of the well-known primal-dual approximation algorithms have been conducted, and the consensus is that these algorithms work well in practice, see [19, Section 4.9], [16], [21], [27], [32].

Several decades after the pioneering work of Kuhn, Dantzig et al., the design of approximation algorithms for NP-hard problems emerged as an important area of research. Agrawal, Klein and Ravi [2] designed and analyzed a primal-dual approximation algorithm for the Steiner forest problem. Goemans and Williamson [18] generalized these algorithms to constrained forest problems. Subsequently, Williamson, Goemans, Vazirani and Mihail [33] (abbreviated WGMV) extended the methods of [18] to obtain a primal-dual 2-approximation algorithm for the problem of augmenting the connectivity of a graph to satisfy requirements specified by *uncrossable* functions. These functions are versatile tools for modeling several network-design problems.

Network design encompasses a wide class of problems that find applications in sectors like transportation, facility location, information security, and resource connectivity, to name a few. Due to its wide scope and usefulness, the area has been studied for decades and it has led to major algorithmic innovations. Most network-design problems are NP-Hard, and oftentimes even APX-hard, so researchers in the area have focused on designing good approximation algorithms, preferably with a small constant-factor approximation ratio. In the context of network design, many of the $O(1)$ approximation algorithms rely on a particular property called *uncrossability*, see the books by Lau, Ravi & Singh [26], Vazirani [31], and Williamson & Shmoys [34]. This property has been leveraged in various ways to obtain $O(1)$ approximation ratios for problems such as survivable network design [20], min-cost/min-size k-edge connected spanning subgraph [15, 14], min-cost 2-node connected spanning subgraph [11], $(p, 1)$-flexible graph connectivity [5], etc. The primal-dual method is one of the most successful algorithmic paradigms that leverages these uncrossability properties.

On the other hand, when the uncrossability property does not hold, most known techniques for designing $O(1)$ approximation algorithms fail to work. Indeed, only logarithmic approximation ratios are known for some of the problems where the uncrossability property does not hold. These logarithmic approximation ratios are usually obtained via a reduction to the set cover problem, for which a greedy strategy yields a logarithmic approximation. WGMV [33] conclude their paper with the following remark:

> Extending our algorithm to handle non-uncrossable functions remains a challenging open problem. The key feature of uncrossable functions is that there exists an optimal dual solution which is laminar ... A larger open issue is to explore further the power of the primal-dual approach for obtaining approximation algorithms for other combinatorial optimization problems. Handling all non-uncrossable functions is ruled out by the fact that there exist instances corresponding to non-uncrossable $\{0,1\}$ functions whose relative duality gap is larger than any constant.

Our main contribution in this work is a novel analysis of the WGMV primal-dual approximation algorithm applied to a class of functions that strictly contain the class of uncrossable functions; we show that the algorithm still yields an $O(1)$ approximation guarantee for this larger class. This new class of functions captures some well-studied network design problems. An application of our main result provides improved approximation ratios for the capacitated k-edge connected subgraph problem, some instances of the flexible graph connectivity problem, and the problem of augmenting all *small* cuts of a graph. A detailed discussion of our results can be found in Section 1.1. For the benefit of the reader, in Section 2.1 we give an overview of WGMV's primal-dual algorithm and its analysis.

The primal-dual algorithm for solving network design problems follows the common strategy of starting with a graph that has no edges and then iteratively buying (i.e., including) a subset of edges into the infeasible solution until feasibility is attained. Within each iteration, the algorithm's goal is to buy a cheap edge-set that fixes some or all of the infeasibility of the current solution. Let F denote the edge-set that has been bought until some step in the algorithm. A set of nodes S is said to be *violated* if the number of F-edges in the cut of S is less than the prespecified connectivity requirement of S. The algorithm deems an edge to be useful if it is in the cut of a violated set S. Clearly, the family of violated sets is important for the design and analysis of these algorithms, especially the inclusion-wise minimal violated sets. A family $\mathcal{F}$ of sets is called *uncrossable* if the following holds:

$$A, B \in \mathcal{F} \implies \text{either } A \cap B, A \cup B \in \mathcal{F} \text{ or } A \setminus B, B \setminus A \in \mathcal{F}.$$

Informally speaking, the uncrossability property ensures that the the minimal sets within the family can be considered independently. Formally, a minimal violated set A in an uncrossable family $\mathcal{F}$ cannot cross another set $S \in \mathcal{F}$; otherwise, we get a contradiction since $A, S \in \mathcal{F}$ implies that either $A \cap S$ or $A \setminus S$ is in $\mathcal{F}$. This key property is one of the levers used in the design of $O(1)$-approximation algorithms for some network-design problems. Unfortunately, there are important problems in network design where the family of violated sets does not form an uncrossable family. For instance, see the instance described in Appendix B. This leads us to define a new class of set families that contains all uncrossable families.

Call a family $\mathcal{F}$ *pliable* if the following holds:

$$A, B \in \mathcal{F} \implies \text{at least two of } A \cap B, A \cup B, A \setminus B, B \setminus A \text{ are in } \mathcal{F}.$$

In the full version of our paper, we show that the WGMV primal-dual algorithm has a super-constant approximation ratio for pliable families. Nevertheless, by enforcing an additional property on the given pliable family, we can establish that the WGMV algorithm yields an $O(1)$ approximation. We call this additional assumption property (γ); see Section 1.1.1 for the formal definition. From a structural standpoint, this property still allows a minimal violated set to cross another violated set, but, crucially, it does not allow them to cross an arbitrary number of violated sets in arbitrary ways. As we show later, the fact that disparate network design problems can be captured by pliable families with property (γ) hints that this property is "just right".

The above connectivity augmentation problems can be understood in a general framework called f-connectivity. In this problem, we are given an undirected graph $G = (V, E)$ on n vertices with nonnegative costs $c \in \mathbb{Q}^E_{\geq 0}$ on the edges and a requirement function $f : 2^V \to \{0, 1\}$ on subsets of vertices. We are interested in finding an edge-set $F \subseteq E$ with minimum cost $c(F) := \sum_{e \in F} c_e$ such that for all cuts $\delta(S),\ S \subseteq V$, we have $|\delta(S) \cap F| \geq f(S)$. This problem can be formulated as the following integer program where binary variables x_e model inclusion of edge e in F:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e & & \qquad (f\text{-IP}) \\ \text{subject to:} \quad & x(\delta(S)) \geq f(S) & \forall\ S \subseteq V \\ & x_e \in \{0, 1\} & \forall\ e \in E. \end{aligned}$$

We remark that in its most-general form, f-connectivity is hard to approximate within a logarithmic factor. This can be shown via a reduction from the hitting set problem.[1] Thus, research on f-connectivity has focused on instances where f has some nice structural properties.

▶ **Definition 1** ([33]). *A function $f : 2^V \to \{0, 1\}$ satisfying $f(V) = 0$ is called* uncrossable *if for any $A, B \subseteq V$ with $f(A) = f(B) = 1$, we have $f(A \cap B) = f(A \cup B) = 1$ or $f(A \setminus B) = f(B \setminus A) = 1$.*

▶ **Definition 2.** *A function $f : 2^V \to \{0, 1\}$ satisfying $f(V) = 0$ is called* pliable *if for any $A, B \subseteq V$ with $f(A) = f(B) = 1$, we have $f(A \cap B) + f(A \cup B) + f(A \setminus B) + f(B \setminus A) \geq 2$.*

Note that the problem of augmenting an uncrossable (pliable) family can be seen as an f-connectivity problem whose requirement function is an uncrossable (pliable) function.

1.1 Our Contributions

In this work, we introduce the class of pliable functions and study the approximation ratio of WGMV's algorithm on f-connectivity instances arising from pliable functions. To the best of our knowledge, we are the first to investigate the f-connectivity problem beyond uncrossable functions. As mentioned before, the algorithm of WGMV can perform poorly on an arbitrary instance with a pliable function f. In the full version [3, Section 6], we present an instance where the solution returned by the WGMV algorithm costs $\Omega(\sqrt{n})$ times the optimal cost.

1.1.1 Pliable Functions and Property (γ)

As alluded to in the introduction, the analysis of WGMV relies on the property that for any inclusion-wise minimal violated set C and any violated set S, either C is a subset of S or C is disjoint from S ([33, Lemma 5.1(3)]). This property does not hold when we apply the primal-dual method to augment a pliable function; see the instance described in Appendix B. Nevertheless, we carve out a subclass of pliable functions – still containing all uncrossable functions – for which the WGMV algorithm yields an $O(1)$-approximate solution. This subclass is characterized by the following structural property that allows for minimal violated sets to cross other violated sets, but in a limited way.

[1] Given a ground-set X and a family $\mathcal{S}$ of subsets of X to hit, we define $L := \{\ell_x : x \in X\}$, $R := \{r_x : x \in X\}$, and $E := \{e_x = \ell_x r_x : x \in X\}$. We then take $G = (L \sqcup R, E)$ to be a bipartite graph with a perfect matching E and f to be the indicator function of the family $\{\{\ell_x : x \in A\} : A \in \mathcal{S}\}$.

Property (γ) : For any edge-set $F \subseteq E$ and for any violated sets (w.r.t. f and F) C, S_1, S_2, with $S_1 \subsetneq S_2$, the following conditional proposition holds:
(C is inclusion-wise minimal) and (C crosses both S_1 and S_2)
$\implies S_2 \setminus (S_1 \cup C)$ is either empty or violated.

▶ **Theorem 3.** *Let $G = (V, E)$ be an undirected graph with nonnegative costs $c : E \to \mathbb{Q}_{\geq 0}$ on its edges, and let $f : 2^V \to \{0, 1\}$ be a pliable function satisfying property (γ). Suppose that there is a subroutine that, for any given $F \subseteq E$, computes all minimal violated sets w.r.t. f and F. Then, in polynomial time and using a polynomial number of calls to the subroutine, we can compute a 16-approximate solution to the given instance of the f-connectivity problem.*

In the next three sections, we introduce the network-design applications where Theorem 3 gives new/improved approximation algorithms. In each of these applications, we setup an f-connectivity problem where the function f is a pliable function with property (γ).

1.1.2 Application 1: Augmenting a Family of Small Cuts

Our first application is on finding a minimum-cost augmentation of a family of small cuts in a graph. Formally, in an instance of the AugSmallCuts problem we are given an undirected capacitated graph $G = (V, E)$ with edge-capacities $u \in \mathbb{Q}^E_{\geq 0}$, a set of links $L \subseteq \binom{V}{2}$ with costs $c \in \mathbb{Q}^L_{\geq 0}$, and a threshold $\widetilde{\lambda} \in \mathbb{Q}_{\geq 0}$. A subset $F \subseteq L$ of links is said to *augment* a node-set S if there exists a link $e \in F$ with exactly one end-node in S. The objective is to find a minimum-cost $F \subseteq L$ that augments all non-empty $S \subsetneq V$ with $u(\delta(S) \cap E) < \widetilde{\lambda}$.

We remark that some special cases of the AugSmallCuts problem have been studied previously, and, to the best of our knowledge, there is no previous publication on the general version of this problem. Let $\lambda(G)$ denote the minimum capacity of a cut of G, thus, $\lambda(G) := \min\{u(\delta(S) \cap E) \ : \ \emptyset \subsetneq S \subsetneq V\}$. Assuming u is integral and $\widetilde{\lambda} = \lambda(G) + 1$, we get the well-known connectivity augmentation problem for which constant-factor approximation algorithms are known [13, 23]. On the other hand, when $\widetilde{\lambda} = \infty$, a minimum-cost spanning tree of (V, L), if one exists, gives an optimal solution to the problem.

Our main result here is an $O(1)$-approximation algorithm for the AugSmallCuts problem that works for any choice of $\widetilde{\lambda}$. The proof of the following theorem is given in Section 4.

▶ **Theorem 4.** *There is a 16-approximation algorithm for the* AugSmallCuts *problem.*

As an aside, we refer the reader to Benczur & Goemans [4] and the references therein for results on the representations of the near-minimum cuts of graphs; they do not study the problem of augmenting the near-minimum cuts.

In Appendix B, we give a small instance of the AugSmallCuts problem that illustrates some of the technical challenges which arise while working with the f-connectivity problem for a pliable function with property (γ). The instance described has bizarre properties that do not arise when working with uncrossable functions. First, it has a minimal violated set which crosses another violated set. Second, none of the optimal solutions to the dual LP of the f-connectivity problem are supported on a laminar family. The latter was believed to be a major hindrance to developing constant-factor approximation algorithms for general network-design problems.

1.1.3 Application 2: Capacitated k-Edge-Connected Subgraph Problem

In the capacitated k-edge-connected subgraph problem (Cap-k-ECSS), we are given an undirected graph $G = (V, E)$ with edge costs $c \in \mathbb{Q}^E_{\geq 0}$ and edge capacities $u \in \mathbb{Z}^E_{\geq 0}$. The goal is to find a minimum-cost subset of the edges $F \subseteq E$ such that the capacity across any cut in (V, F) is at least k, i.e., $u(\delta_F(S)) \geq k$ for all non-empty sets $S \subsetneq V$. Let u_{max} and u_{min}, respectively, denote the maximum capacity of an edge in E and the minimum capacity of an edge in E. We may assume (w.l.o.g.) that $u_{max} \leq k$.

We mention that there are well-known 2-approximation algorithms for the special case of the Cap-k-ECSS problem with $u_{max} = u_{min} = 1$, which is the problem of finding a minimum-cost k-edge connected spanning subgraph. Khuller & Vishkin [24] presented a combinatorial 2-approximation algorithm and Jain [20] matched this approximation guarantee via the iterative rounding method.

Goemans et al. [17] gave a $2k$-approximation algorithm for the general Cap-k-ECSS problem. Chakrabarty et al. [6] gave a randomized $O(\log |V(G)|)$-approximation algorithm; note that this approximation guarantee is independent of k but does depend on the size of the underlying graph. Recently, Boyd et al. [5] improved on these results by providing a $\min(k, 2u_{max})$-approximation algorithm. In this work, we give a $(16 \cdot \lceil k/u_{\min} \rceil)$-approximation algorithm, which leads to improved approximation guarantees when both $u_{\min}$ and $u_{\max}$ are sufficiently large. In particular, in the regime when $k \geq u_{\max} \geq u_{\min} \geq 32$ and $u_{\min} \cdot u_{\max} \geq 16k$.

▶ **Theorem 5.** *There is a $16 \cdot \lceil k/u_{min} \rceil$-approximation algorithm for the Cap-k-ECSS problem.*

The proof of Theorem 5 can be found in Section 5.

1.1.4 Application 3: $(p, 2)$-Flexible Graph Connectivity

Adjiashvili, Hommelsheim and Mühlenthaler [1] introduced the model of Flexible Graph Connectivity that we denote by FGC. Boyd, Cheriyan, Haddadan and Ibrahimpur [5] introduced a generalization of FGC. Let $p \geq 1$ and $q \geq 0$ be integers. In an instance of the (p, q)-Flexible Graph Connectivity problem, denoted (p, q)-FGC, we are given an undirected graph $G = (V, E)$, a partition of E into a set of safe edges $\mathcal{S}$ and a set of unsafe edges $\mathcal{U}$, and nonnegative edge-costs $c \in \mathbb{Q}^E_{\geq 0}$. A subset $F \subseteq E$ of edges is feasible for the (p, q)-FGC problem if for any set F' consisting of at most q unsafe edges, the subgraph $(V, F \setminus F')$ is p-edge connected. The objective is to find a feasible solution F that minimizes $c(F) = \sum_{e \in F} c_e$.

Boyd et al. [5] presented a 4-approximation algorithm for $(p, 1)$-FGC based on the WGMV primal-dual method, and they gave an $O(q \log n)$-approximation algorithm for general (p, q)-FGC and a $(q + 1)$-approximation for $(1, q)$-FGC. Concurrently with our work, Chekuri and Jain [8] obtained $O(p)$-approximation algorithms for $(p, 2)$-FGC, $(p, 3)$-FGC and $(2p, 4)$-FGC; in particular, they present a $(2p + 4)$-approximation ratio for $(p, 2)$-FGC. Chekuri and Jain have several other results for network design in non-uniform fault models; [7] have results on the flexible graph connectivity problem that arises from the classical survivable network design problem, which they call (p, q)-Flex-SNDP.

Our main result here is an $O(1)$-approximation algorithm for the $(p, 2)$-FGC problem.

▶ **Theorem 6.** *There is a 20-approximation algorithm for the $(p, 2)$-FGC problem. Moreover, for even p, the approximation ratio is 6.*

Note that in comparison to [8], Theorem 6 yields a better approximation ratio when $p > 8$ or $p \in \{2, 4, 6, 8\}$. For $p = 1$, the approximation ratio of 3 from [5] is better than the guarantees given by [8] and Theorem 6. The proof of Theorem 6 can be found in Section 6.

1.2 Related work

Goemans & Williamson [18] introduced the notion of proper functions with the motivation of designing approximation algorithms for problems in network design. They formulated several of these problems as the f-connectivity problem where f is a proper function. A symmetric function $f : 2^V \to \mathbb{Z}_{>0}$ with $f(V) = 0$ is said to be *proper* if $f(A \cup B) \leq \max(f(A), f(B))$ for any pair of disjoint sets $A, B \subseteq V$.

Jain [20] designed the iterative rounding framework for the setting when f is weakly supermodular and presented a 2-approximation algorithm. A function f is said to be *weakly supermodular* if $f(A) + f(B) \leq \max(f(A \cap B) + f(A \cup B),\ f(A \setminus B) + f(B \setminus A))$ for any $A, B \subseteq V$. One can show that proper functions are weakly supermodular. We mention that there are examples of uncrossable functions that are not weakly supermodular, see [5].

2 Preliminaries

This section has definitions and preliminary results. Our notation and terms are consistent with [10, 30], and readers are referred to those texts for further information.

For a positive integer k, we use $[k]$ to denote the set $\{1, \ldots, k\}$. For a ground-set V and a subset S of V, the complement of S (w.r.t. V) is denoted $V \setminus S$. Sets $A, B \subseteq V$ are said to *cross*, denoted $A \bowtie B$, if each of the four sets $A \cap B$, $V \setminus (A \cup B)$, $A \setminus B$, $B \setminus A$ is non-empty; on the other hand, if A, B do not cross, then either $A \cup B = V$, or A, B are disjoint, or one of A, B is a subset of the other one. A family of sets $\mathcal{L} \subseteq 2^V$ is said to be *laminar* if for any two sets $A, B \in \mathcal{L}$ either A and B are disjoint or one of them is a subset of the other one.

We may use abbreviations for some standard terms, e.g., we may use "(p, q)-FGC" as an abbreviation for "the (p, q)-FGC problem". In some of our discussions, we may use the informal phrasing "we apply the primal-dual method to augment a pliable function" instead of the phrasing "we apply the primal-dual method to an f-connectivity problem where the function f is a pliable function".

Graphs, Subgraphs, and Related Notions

Let $G = (V, E)$ be an undirected multi-graph (possibly containing parallel edges but no loops) with non-negative costs $c \in \mathbb{R}^E_{\geq 0}$ on the edges. We take G to be the input graph, and we use n to denote $|V(G)|$. For a set of edges $F \subseteq E(G)$, $c(F) := \sum_{e \in F} c(e)$, and for a subgraph G' of G, $c(G') := c(E(G'))$. For any instance G, we use $\textsc{opt}(G)$ to denote the minimum cost of a feasible subgraph (i.e., a subgraph that satisfies the requirements of the problem). When there is no danger of ambiguity, we use $\textsc{opt}$ rather than $\textsc{opt}(G)$.

Let $G = (V, E)$ be any multi-graph, let $A, B \subseteq V$ be two disjoint node-sets, and let $F \subseteq V$ be an edge-set. We denote the multi-set of edges of G with exactly one end-node in each of A and B by $E(A, B)$, thus, $E(A, B) := \{e = uv : u \in A, v \in B\}$. Moreover, we use $\delta_E(A)$ or $\delta(A)$ to denote $E(A, V \setminus A)$. By a p-cut we mean a cut of size p. We use $G[A]$ to denote the subgraph of G induced by A, $G - A$ to denote the subgraph of G induced by $V \setminus A$, and $G - F$ to denote the graph $(V,\ E \setminus F)$. We may use relaxed notation for singleton sets, e.g., we may use $G - v$ instead of $G - \{v\}$, etc. A multi-graph G is called k-edge connected if $|V(G)| \geq 2$ and for every $F \subseteq E(G)$ of size $< k$, $G - F$ is connected.

We use the following observations.

▶ **Fact 7.** *Let $A, B \subseteq V$ be a pair of crossing sets. For any edge-set $F \subseteq \binom{V}{2}$ and any $S \in \{A \cap B, A \cup B, A \setminus B, B \setminus A\}$, we have $\delta_F(S) \subseteq \delta_F(A) \cup \delta_F(B)$.*

Proof. By examining cases, we can show that $e \in \delta_F(S) \implies e \in \delta_F(A)$ or $e \in \delta_F(B)$. ◀

For any function $f : 2^V \to \{0,1\}$ and any edge-set $F \subseteq E$, we say that $S \subseteq V$ is *violated w.r.t. f, F* if $|\delta_F(S)| < f(S)$, i.e., if $f(S) = 1$ and there are no F-edges in the cut $\delta(S)$. We drop f and F when they are clear from the context. The next observation states that the violated sets w.r.t. any pliable function f and any "augmenting" edge-set F form a pliable family.

▶ **Fact 8.** *Let $f : 2^V \to \{0,1\}$ be a pliable function and $F \subseteq E$ be an edge-set. Define the function $f' : 2^V \to \{0,1\}$ such that $f'(S) = 1$ if and only if both $f(S) = 1$ and $\delta_F(S) = \emptyset$ hold. Then, f' is also pliable.*

Proof. Consider $A, B \subsetneq V$ such that $f'(A) = 1 = f'(B)$. Clearly, $f(A) = 1 = f(B)$. Moreover, for any $S \in \{A \cap B, A \cup B, A \setminus B, B \setminus A\}$, we have $\delta_F(S) = \emptyset$, by Fact 7. Since f is pliable, there are at least two distinct sets $S_1, S_2 \in \{A \cap B, A \cup B, A \setminus B, B \setminus A\}$ with f-value one. Then, we have $f'(S_1) = 1 = f'(S_2)$ (since $\delta_F(S_1) = \emptyset = \delta_F(S_2)$). Hence, f' is pliable. ◀

2.1 The WGMV Primal-Dual Algorithm for Uncrossable Functions

In this section, we give a brief description of the primal-dual algorithm of Williamson et al. [33] that achieves approximation ratio 2 for an f-connectivity problem where the function f is an uncrossable function.

▶ **Theorem 9** (Lemma 2.1 in [33])**.** *Let $f : 2^V \to \{0,1\}$ be an uncrossable function. Suppose we have a subroutine that for any given $F \subseteq E$, computes all minimal violated sets w.r.t. f, F. Then, in polynomial time and using a polynomial number of calls to the subroutine, we can compute a 2-approximate solution to the given instance of the f-connectivity problem.*

The algorithm and its analysis are based on the following LP relaxation of (f-IP) (stated on the left) and its dual. Define $\mathcal{S} := \{S \subseteq V : f(S) = 1\}$.

Primal LP

$$\min \sum_{e \in E} c_e x_e$$

$$\text{subject to: } \sum_{e \in \delta(S)} x_e \geq 1 \quad \forall S \in \mathcal{S}$$

$$0 \leq x_e \leq 1 \quad \forall e \in E$$

Dual LP

$$\max \sum_{S \in \mathcal{S}} y_S$$

$$\text{subject to: } \sum_{S \in \mathcal{S} : e \in \delta(S)} y_S \leq c_e \quad \forall e \in E$$

$$y_S \geq 0 \quad \forall S \in \mathcal{S}$$

The algorithm starts with an infeasible primal solution $F = \emptyset$, which corresponds to $x = \chi^F = \mathbf{0} \in \{0,1\}^E$, and a feasible dual solution $y = \mathbf{0}$. At any time, we say that $S \in \mathcal{S}$ is *violated* if $\delta_F(S) = \emptyset$, i.e., the primal-covering constraint for S is not satisfied. We call inclusion-wise minimal violated sets as *active sets*. An edge $e \in E$ is said to be *tight* if $\sum_{S \in \mathcal{S} : e \in \delta(S)} y_S = c_e$, i.e., the dual-packing constraint for e is tight. Throughout the algorithm, the following conditions are maintained: (i) integrality of the primal solution; (ii) feasibility of the dual solution; (iii) y_S is never decreased for any S; and (iv) y_S may only be increased for $S \in \mathcal{S}$ that are active.

The algorithm has two stages. In the first stage, the algorithm iteratively improves primal feasibility by including tight edges in F that are incident to active sets. If no such edge exists, then the algorithm uniformly increases y_S for all active sets S until a new edge becomes tight. The first stage ends when $x = \chi^F$ becomes feasible. In the second stage, called *reverse delete*, the algorithm removes redundant edges from F. Initially $F' = F$. The algorithm examines edges picked in the first stage in reverse order, and discards edges from F' as long as feasibility is maintained. Note that F' is feasible if the subroutine in the hypothesis of Theorem 9 does not find any (minimal) violated sets.

The analysis of the 2-approximation ratio is based on showing that a relaxed form of the complementary slackness conditions hold on "average". Let F' and y be the primal and dual solutions returned by the algorithm. By the design of the algorithm, $\sum_{S \in \mathcal{S}: e \in \delta(S)} y_S = c_e$ holds for any edge $e \in F'$. Thus, the cost of F' can be written as $\sum_{e \in F'} \sum_{S \in \mathcal{S}: e \in \delta(S)} y_S = \sum_{S \in \mathcal{S}} y_S \cdot |\delta_{F'}(S)|$. Observe that the approximation ratio follows from showing that the algorithm always maintains the following inequality:

$$\sum_{S \in \mathcal{S}} y_S \cdot |\delta_{F'}(S)| \leq 2 \sum_{S \in \mathcal{S}} y_S. \tag{1}$$

Consider any iteration and recall that the dual variables corresponding to active sets were uniformly increased by an $\varepsilon > 0$ amount, until some edge became tight. Let $\mathcal{C}$ denote the collection of active sets during this iteration. During this iteration, the left-hand side of (1) increases by $\varepsilon \cdot \sum_{S \in \mathcal{C}} |\delta_{F'}(S)|$ and the right-hand side increases by $2 \cdot \varepsilon \cdot |\mathcal{C}|$. Thus, (1) is maintained if one can show that the average F'-degree of active sets in any iteration is ≤ 2, and this forms the crux of the WGMV analysis.

We refer the reader to [19] for a detailed discussion of the primal-dual method for network design problems.

3 Extending the WGMV Primal-Dual Method to Pliable functions

In this section, we prove our main result, Theorem 3: we show that the primal-dual algorithm outlined in Section 2.1 is a 16-approximation algorithm for the f-connectivity problem where f is a pliable function with property (γ). Our analysis follows the same high-level plan as that of Williamson et al. [33] which was outlined in Section 2.1. We will show that, in any iteration of the first stage of the primal-dual algorithm, $\sum_{C \in \mathcal{C}} |\delta_{F'}(C)| \leq 16|\mathcal{C}|$, where $\mathcal{C}$ is the collection of active sets in that iteration, and F' is the set of edges output by the algorithm at termination, after the reverse delete stage.

For the remainder of this proof we assume that the iteration, and thus $\mathcal{C}$, is fixed. We define $H := \cup_{C \in \mathcal{C}} \delta_{F'}(C)$. (Informally speaking, H is the subset of F' that is relevant for the analysis of our fixed iteration.) Additionally, to ease notation when discussing a laminar family of sets, we say that two sets A, B *overlap* if $A \setminus B, A \cap B$ and $B \setminus A$ are all non-empty. (Clearly, if A, B cross, then A, B overlap; if $A \cup B = V$, then A, B do not cross but A, B could overlap.)

We begin with a lemma which can be proved by the same arguments as in the proof of [33, Lemma 5.1].

▶ **Lemma 10.** *For any edge $e \in H := \cup_{C \in \mathcal{C}} \delta_{F'}(C)$, there exists a witness set $S_e \subseteq V$ with:*
(i) *$f(S_e) = 1$ and S_e is violated in the current iteration, and*
(ii) *$\delta_{F'}(S_e) = \{e\}$.*

Our proof of the following key lemma is presented in Appendix A.

▶ **Lemma 11.** *There exists a laminar family of witness sets.*

▶ **Lemma 12.** *The active sets in $\mathcal{C}$ are pair-wise disjoint.*

Proof. Suppose that two sets $C_1, C_2 \in \mathcal{C}$ intersect. Then due to property (i) of pliable functions, one of the sets $C_1 \cap C_2$, $C_1 \setminus C_2$, or $C_2 \setminus C_1$ is violated; thus, a proper subset of either C_1 or C_2 is violated. This is a contradiction because C_1 and C_2 are minimal violated sets and no proper subset of C_1 (respectively, C_2) is violated. ◀

Let $\mathcal{L}$ be the laminar family of witness sets together with the node-set V. Let $\mathcal{T}$ be a rooted tree that represents $\mathcal{L}$; for each set $S \in \mathcal{L}$, there is a node v_S in $\mathcal{T}$, and the node v_V is taken to be the root of $\mathcal{T}$. The edges of $\mathcal{T}$ are oriented away from the root; thus, $\mathcal{T}$ has an oriented edge (v_Q, v_S) iff Q is the smallest set of $\mathcal{L}$ that properly contains the set S of $\mathcal{L}$. Let ψ be a mapping from $\mathcal{C}$ to $\mathcal{L}$ that maps each active set C to the smallest set $S \in \mathcal{L}$ that contains it. If a node v_S of $\mathcal{T}$ has some active set mapped to its associated set S, then we call v_S *active* and we assign the color red to v_S. Moreover, we assign the color green to each of the non-active nodes of $\mathcal{T}$ that are incident to three or more edges of $\mathcal{T}$; thus, node v_S of $\mathcal{T}$ is green iff $\deg_{\mathcal{T}}(v_S) \geq 3$ and v_S is not active. Finally, we assign the color black to each of the remaining nodes of $\mathcal{T}$; thus, node v_S of $\mathcal{T}$ is black iff $\deg_{\mathcal{T}}(v_S) \leq 2$ and v_S is not active.

Let the number of red, green and black nodes of $\mathcal{T}$ be denoted by n_R, n_G and n_B, respectively. Clearly, $n_R + n_G + n_B = |\mathcal{T}| = |F'| + 1$. Let n_L denote the number of leaf nodes of $\mathcal{T}$.

▶ **Lemma 13.** *The following are true:*

(i) *Each leaf node of $\mathcal{T}$ is red.*

(ii) *We have $n_G \leq n_L \leq n_R$.*

Proof. The first claim follows by repeating the argument in [33, Lemma 5.3]. Next, by (i), we have $n_L \leq n_R$. Moreover, we have $n_G \leq n_L$ because the number of leaves in any tree is at least the number of nodes that are incident to three or more edges of the tree. ◀

Observe that each black node of $\mathcal{T}$ is incident to two edges of $\mathcal{T}$; thus, every black non-root node of $\mathcal{T}$ has a unique child.

Let us sketch our plan for proving Theorem 3. Clearly, the theorem would follow from the inequality $\sum_{C \in \mathcal{C}} |\delta_{F'}(C)| \leq O(1) \cdot |\mathcal{C}|$; thus, we need to prove an upper bound of $O(|\mathcal{C}|)$ on the number of "incidences" between the edges of F' and the cuts $\delta(C)$ of the active sets $C \in \mathcal{C}$. We start by assigning a token to $\mathcal{T}$ corresponding to each "incidence". In more detail, for each edge $e \in F'$ and cut $\delta(C)$ such that $C \in \mathcal{C}$ and $e \in \delta(C)$ we assign one token to the node v_{S_e} of $\mathcal{T}$ that represents the witness set S_e of the edge e. Thus, the total number of tokens assigned to $\mathcal{T}$ is $\sum_{C \in \mathcal{C}} |\delta_{F'}(C)|$; moreover, after the initial assignment, it can be seen that each node of $\mathcal{T}$ has ≤ 2 tokens (see Lemma 14 below). Then we redistribute the tokens according to a simple rule such that (after redistributing) each of the red/green nodes has ≤ 8 tokens and each of the black nodes has no tokens. Lemma 15 (below) proves this key claim by applying property (γ). The key claim implies that the total number of tokens assigned to $\mathcal{T}$ is $\leq 8n_R + 8n_G \leq 16n_R \leq 16|\mathcal{C}|$ (by Lemma 13). This concludes our sketch.

We apply the following two-phase scheme to assign tokens to the nodes of $\mathcal{T}$.

- In the first phase, for $C \in \mathcal{C}$ and $e \in \delta_{F'}(C)$, we assign a new token to the node v_{S_e} corresponding to the witness set S_e for the edge e. At the end of the first phase, observe that the root v_V of $\mathcal{T}$ has no tokens (since the set V cannot be a witness set).

- In the second phase, we apply a root-to-leaves scan of $\mathcal{T}$ (starting from the root v_V). Whenever we scan a black node, then we move all the tokens at that node to its unique child node. (There are no changes to the token distribution when we scan a red node or a green node.)

▶ **Lemma 14.** *At the end of the first phase, each node of $\mathcal{T}$ has ≤ 2 tokens.*

Proof. Consider a non-root node v_{S_e} of $\mathcal{T}$. This node corresponds to a witness set $S_e \in \mathcal{L}$ and e is the unique edge of F' in $\delta(S_e)$. The edge e is in ≤ 2 of the cuts $\delta(C), C \in \mathcal{C}$, because the active sets are pairwise disjoint (in other words, the number of "incidences" for e is ≤ 2). No other edge of F' can assign tokens to v_{S_e} during the first phase. ◀

▶ **Lemma 15.** *We have that:*

(i) *Any oriented path of $\mathcal{T} \setminus \{v_V\}$ with four nodes has at least one non-black node.*

(ii) *Hence, after token redistribution, each red or green node of $\mathcal{T}$ has ≤ 8 tokens and each black node of $\mathcal{T}$ has zero tokens.*

Proof. For the sake of contradiction, assume that there exists an oriented path of $\mathcal{T}$ that has four black nodes and that is not incident to the root v_V; let $v_{S_4} \to v_{S_3} \to v_{S_2} \to v_{S_1}$ be such an oriented path. Thus, $S_1 \subsetneq S_2 \subsetneq S_3 \subsetneq S_4$ are witness sets of $\mathcal{L}$. For $i \in \{1,2,3,4\}$, let S_i be the witness set of edge $e_i = \{a_i, b_i\} \in F'$; note that e_i has exactly one end-node in S_i, call it a_i. Clearly, for $i \in \{1,2,3\}$, both nodes a_i, b_i are in S_{i+1} (since e_{i+1} is the unique edge of F' in $\delta(S_{i+1})$).

Let $C \in \mathcal{C}$ be an active set such that $e_1 \in \delta(C)$.

▷ Claim 16. C is not a subset of S_1.

For the sake of contradiction, suppose that C is a subset of S_1. Since e_1 has (exactly) one end-node in C and $b_1 \notin S_1$, we have $a_1 \in C$. Let W be the smallest set in $\mathcal{L}$ that contains C. Then $W \subseteq S_1$, and, possibly, $W = S_1$. Thus, we have $a_1 \in W$ and $b_1 \notin W$, hence, $e_1 \in \delta(W)$. Then we must have $W = S_1$ (since e_1 is in exactly one of the cuts $\delta(S), S \in \mathcal{L}$). Then the mapping ψ from $\mathcal{C}$ to $\mathcal{L}$ maps C to $W = S_1$, hence, v_{S_1} is colored red. This is a contradiction.

▷ Claim 17. C crosses each of the sets S_2, S_3, S_4.

First, observe that e_1 has (exactly) one end-node in C and has both end-nodes in S_2. Hence, both $S_2 \cap C$ and $S_2 \setminus C$ are non-empty. Next, using Claim 16, we can prove that C is not a subset of S_2. (Otherwise, S_2 would be the smallest set in $\mathcal{L}$ that contains C, hence, v_{S_2} would be colored red.) Repeating the same argument, we can prove that C is not a subset of S_3, and, moreover, C is not a subset of S_4. Finally, note that $V \setminus (C \cup S_4)$ is non-empty. (Otherwise, at least one of $C \setminus S_4$ or $C \cap S_4$ would be violated, since f is a pliable function, and that would contradict the fact that C is a minimal violated set.) Observe that S_2 crosses C because all four sets $S_2 \cap C$, $S_2 \setminus C$, $C \setminus S_2$, $V \setminus (S_2 \cup C)$ are non-empty (in more detail, we have $|\{a_1, b_1\} \cap (S_2 \cap C)| = 1$, $|\{a_1, b_1\} \cap (S_2 \setminus C)| = 1$, $C \not\subseteq S_2 \implies C \setminus S_2 \neq \emptyset$, $V \setminus (C \cup S_2) \supseteq V \setminus (C \cup S_4) \neq \emptyset$). Similarly, it can be seen that S_3 crosses C, and S_4 crosses C.

▷ Claim 18. Either $S_3 \setminus (C \cup S_2)$ is non-empty or $S_4 \setminus (C \cup S_3)$ is non-empty.

For the sake of contradiction, suppose that both sets $S_3 \setminus (C \cup S_2)$, $S_4 \setminus (C \cup S_3)$ are empty. Then $C \supseteq S_4 \setminus S_3$ and $C \supseteq S_3 \setminus S_2$. Consequently, both end-nodes of e_3 are in C (since $a_3 \in S_3 \setminus S_2$ and $b_3 \in S_4 \setminus S_3$). This leads to a contradiction, since $e_3 \in F'$ is incident to an active set in $\mathcal{C}$, call it C_3 (i.e., $e_3 \in \delta(C_3)$), hence, one of the end-nodes of e_3 is in both C and C_3, whereas the active sets are pairwise disjoint.

To conclude the proof of the lemma, suppose that $S_4 \setminus (C \cup S_3)$ is non-empty (by Claim 18); the other case, namely, $S_3 \setminus (C \cup S_2) \neq \emptyset$, can be handled by the same arguments. Then, by property (γ), $S_4 \setminus (C \cup S_3)$ is a violated set, therefore, it contains a minimal violated set, call it $\widetilde{C}$. Clearly, the mapping ψ from $\mathcal{C}$ to $\mathcal{L}$ maps the active set $\widetilde{C}$ to a set $S_{\widetilde{C}}$. Either $S_{\widetilde{C}} = S_4$ or else $S_{\widetilde{C}}$ is a subset of of $S_4 \setminus S_3$. Both cases give contradictions; in the first case, S_4 is colored red, and in the second case, v_{S_4} has ≥ 2 children in $\mathcal{T}$ so that S_4 is colored either green or red. Thus, we have proved the first part of the lemma.

The second part of the lemma follows by Lemma 13 and the sketch given below Lemma 13. In more detail, at the start of the second phase, each node of $\mathcal{T}$ has ≤ 2 tokens, by Lemma 14. In the second phase, we redistribute the tokens such that each (non-root) black node ends up with no tokens, and each red/green node v_S receives ≤ 6 redistributed tokens because there are ≤ 3 black ancestor nodes of v_S that could send their tokens to v_S (by the first part of the lemma). Hence, each non-root non-black node has ≤ 8 tokens, after token redistribution. ◀

4 $O(1)$-Approximation Algorithm for Augmenting Small Cuts

In this section, we give a 16-approximation algorithm for the AugSmallCuts problem, thereby proving Theorem 4. Our algorithm for AugSmallCuts is based on a reduction to an instance of the f-connectivity problem on the graph $H = (V, L)$ for a pliable function f with property (γ).

Recall the AugSmallCuts problem: we are given an undirected graph $G = (V, E)$ with edge-capacities $u \in \mathbb{Q}^E_{\geq 0}$, a set of links $L \subseteq \binom{V}{2}$ with costs $c \in \mathbb{Q}^L_{\geq 0}$, and a threshold $\widetilde{\lambda} \in \mathbb{Q}_{\geq 0}$. A subset $F \subseteq L$ of links is said to *augment* a node-set S if there exists a link $e \in F$ with exactly one end-node in S. The objective is to find a minimum-cost $F \subseteq L$ that augments all non-empty $S \subsetneq V$ with $u(\delta_E(S)) < \widetilde{\lambda}$.

Proof of Theorem 4. Define $f : 2^V \to \{0, 1\}$ such that $f(S) = 1$ if and only if $S \notin \{\emptyset, V\}$ and $u(\delta_E(S)) < \widetilde{\lambda}$. We apply Theorem 3 for the f-connectivity problem on the graph $H = (V, L)$ with edge-costs $c \in \mathbb{Q}^L_{\geq 0}$ to obtain a 16-approximate solution $F \subseteq L$. By our choice of f, there is a one-to-one cost-preserving correspondence between feasible augmentations for AugSmallCuts and feasible solutions to the f-connectivity problem. Thus, it remains to argue that the assumptions of Theorem 3 hold.

First, we show that f is pliable. Note that f is symmetric and $f(V) = 0$. Consider sets $A, B \subseteq V$ with $f(A) = f(B) = 1$. By submodularity and symmetry of cuts in undirected graphs, we have: $\max\{u(\delta(A \cup B)) + u(\delta(A \cap B)),\ u(\delta(A \setminus B)) + u(\delta(B \setminus A))\} \leq u(\delta(A)) + u(\delta(B))$. Since the right hand side is strictly less than $2\widetilde{\lambda}$, we have $f(A \cap B) + f(A \cup B) \geq 1$ and $f(A \setminus B) + f(B \setminus A) \geq 1$, hence, f is pliable.

Second, we argue that f satisfies property (γ). Fix some edge-set $F \subseteq L$, and define $f' : 2^V \to \{0, 1\}$ such that $f'(S) = 1$ if and only if $f(S) = 1$ and $\delta_F(S) = \emptyset$. By Fact 8, f' is also pliable. Consider sets $C, S_1, S_2 \subseteq V$, $S_1 \subsetneq S_2$, that are violated w.r.t. f, F, i.e., $f'(C) = f'(S_1) = f'(S_2) = 1$. Further, suppose that C is minimally violated, and C crosses both S_1 and S_2. Suppose that $S_2 \setminus (S_1 \cup C)$ is non-empty (the other case is trivial). To show that $S_2 \setminus (S_1 \cup C)$ is violated w.r.t. f, F, we have to show that (i) $\delta_F(S_2 \setminus (S_1 \cup C))$ is empty and (ii) $u(\delta_E(S_2 \setminus (S_1 \cup C))) < \widetilde{\lambda}$. Observe that S_2 crosses $(S_1 \cup C)$. To show (i), we apply Fact 7 twice; first, we show that $\delta_F(S_1 \cup C)$ is empty (since $\delta_F(C)$, $\delta_F(S_1)$ are empty), and then we show that $\delta_F(S_2 \setminus (S_1 \cup C))$ is empty (since $\delta_F(S_2)$ is empty). To show (ii), observe that the multiset

$$\delta_E(S_2 \setminus (S_1 \cup C)) \cup \delta_E(C \setminus S_2) \text{ is a (multi-)subset of } \delta_E(S_2) \cup \delta_E(C \cup S_1).$$

(Note that for disjoint sets $A_1, A_2, A_3 \subsetneq V$, $\delta(A_1) \cup \delta(A_2)$ is a (multi-)subset of $\delta(A_1 \cup A_3) \cup \delta(A_2 \cup A_3)$.) Moreover, we claim that $u(\delta_E(C \cup S_1)) < \widetilde{\lambda}$ and $u(\delta_E(C \setminus S_2)) \geq \widetilde{\lambda}$. The two claims immediately imply (ii) (since $u(\delta_E(S_2)) < \widetilde{\lambda}$).

Next, we prove the two claims. Note that the sets $C \cap S_1, C \setminus S_1, S_1 \setminus C, V \setminus (C \cup S_1)$ are non-empty, and note that $f'(C \cap S_1) = 0 = f'(C \setminus S_1)$ since C is a minimal violated set. Since f' is pliable and $f'(C) = 1 = f'(S_1)$, we have $f'(C \cup S_1) = 1$. By Fact 7, $\delta_F(C \cup S_1) = \emptyset$, hence, $f(C \cup S_1) = 1$; equivalently, $u(\delta_E(C \cup S_1)) < \widetilde{\lambda}$. Since C is a minimal violated set, $f'(C \setminus S_2) = 0$. Moreover, $\delta_F(C \setminus S_2) = \emptyset$, by Fact 7. Hence, $f(C \setminus S_2) = 0$; equivalently, $u(\delta_E(C \setminus S_2)) \geq \widetilde{\lambda}$.

Last, we describe a polynomial-time subroutine that for any $F \subseteq L$ gives the collection of all minimal violated sets w.r.t. f, F. Assign a capacity of $\widetilde{\lambda}$ to all edges in F, and consider the graph $G' = (V, E')$ where $E' := E \cup F$. Then, the family of minimal violated sets is given by $\{\emptyset \subsetneq S \subsetneq V : u(\delta_{E'}(S)) < \widetilde{\lambda}, u(\delta_{E'}(A)) \geq \widetilde{\lambda} \; \forall \, \emptyset \subsetneq A \subsetneq S\}$. We use the notion of solid sets to find all such minimally violated sets; see Naor, Gusfield, and Martel [29] and see Frank's book [12]. A solid set of an undirected graph $H = (V, E'')$ with capacities $w \in \mathbb{R}^{E''}_{\geq 0}$ on its edges is a non-empty node-set $Z \subsetneq V$ such that $w(\delta_{E''}(X)) > w(\delta_{E''}(Z))$ for all non-empty $X \subsetneq Z$. Note that the family of minimal violated sets of interest to us is a sub-family of the family of solid sets of G'. The family of all solid sets of a graph can be listed in polynomial time, see [29] and [12, Chapter 7.3]. Hence, we can find all minimal violated sets w.r.t. f, F in polynomial time, by examining the list of solid sets to check (1) whether there is a solid set S that is violated, and (2) whether every proper subset of S that is a solid set is *not* violated. This completes the proof of the theorem. ◀

5 $O(k/u_{min})$-Approximation Algorithm for the Capacitated k-Edge-Connected Subgraph Problem

In this section, we give a $16 \cdot \lceil k/u_{\min} \rceil$-approximation algorithm for the Cap-k-ECSS problem, thereby proving Theorem 5. Our algorithm is based on repeated applications of Theorem 4.

Recall the capacitated k-edge-connected subgraph problem (Cap-k-ECSS): we are given an undirected graph $G = (V, E)$ with edge costs $c \in \mathbb{Q}^E_{\geq 0}$ and edge capacities $u \in \mathbb{Z}^E_{\geq 0}$. The goal is to find a minimum-cost subset of the edges $F \subseteq E$ such that the capacity across any cut in (V, F) is at least k, i.e., $u(\delta_F(S)) \geq k$ for all non-empty sets $S \subsetneq V$.

Proof of Theorem 5. The algorithm is as follows: Initialize $F := \emptyset$. While the minimum capacity of a cut $\delta(S)$, $\emptyset \neq S \subsetneq V$, in (V, F) is less than k, run the approximation algorithm from Theorem 4 with input $G = (V, F)$ and $L = E \setminus F$, to augment all cuts $\delta(S)$, $\emptyset \neq S \subsetneq V$, with $u(\delta(S)) < k$ and obtain a valid augmentation $F' \subseteq L$. Update F by adding F', that is, $F := F \cup F'$. On exiting the while loop, output the set of edges F.

At any step of the algorithm, let λ denote the minimum capacity of a cut in (V, F), i.e., $\lambda := \min\{u(\delta_F(S)) \; : \; \emptyset \subsetneq S \subsetneq V\}$.

The above algorithm outputs a feasible solution since, upon exiting the while loop, λ is at least k. Let $F^* \subseteq E$ be an optimal solution to the Cap-k-ECSS instance. Notice that $F^* \setminus F$ is a feasible choice for F' during any iteration of the while loop. Hence, by Theorem 4, $c(F') \leq 16 \cdot c(F^*)$. We claim that the above algorithm requires at most $\lceil \frac{k}{u_{min}} \rceil$ iterations of the while loop. This holds because each iteration of the while loop (except possibly the last iteration) raises λ by at least u_{min}. (At the start of the last iteration, $k - \lambda$ could be less than u_{min}, and, at the end of the last iteration, λ could be equal to k). Hence, at the end of the algorithm, $c(F) \leq 16 \cdot \lceil \frac{k}{u_{min}} \rceil c(F^*)$. This completes the proof. ◀

We remark that our new result (Theorem 4) is critical for the bound of $\lceil \frac{k}{u_{min}} \rceil$ on the number of iterations of this algorithm. Earlier methods only allowed augmentations of minimum cuts, so such methods may require as many as $\Omega(k)$ iterations. (In more detail, the earlier methods would augment the cuts of (V, F) of capacity λ but would not augment the cuts of capacity $\geq \lambda + 1$; thus, cuts of capacity $\lambda + 1$ could survive the augmentation step.)

6 $O(1)$-Approximation Algorithm for $(p, 2)$-FGC

In this section, we present a 20-approximation algorithm for $(p, 2)$-FGC, by applying our results from Section 3.

Recall (from Section 1) that the algorithmic goal in $(p, 2)$-FGC is to find a minimum-cost edge-set F such that for any pair of unsafe edges $e, f \in F \cap \mathcal{U}$, the subgraph $(V, F \setminus \{e, f\})$ is p-edge connected.

Our algorithm works in two stages. First, we compute a feasible edge-set F_1 for $(p, 1)$-FGC on the same input graph, by applying the 4-approximation algorithm of [5]. We then augment the subgraph (V, F_1) using additional edges. Since F_1 is a feasible edge-set for $(p, 1)$-FGC, any cut $\delta(S)$, $\emptyset \subsetneq S \subsetneq V$, in the subgraph (V, F_1) either (i) has at least p safe edges or (ii) has at least $p + 1$ edges (see below for a detailed argument). Thus the cuts that need to be augmented have exactly $p + 1$ edges and contain at least two unsafe edges. Let us call such cuts *deficient*. Augmenting all deficient cuts by at least one (safe or unsafe) edge will ensure that we have a feasible solution to $(p, 2)$-FGC.

The following example shows that when p is odd, then the function f in the f-connectivity problem associated with $(p, 2)$-FGC may *not* be an uncrossable function. In other words, the indicator function $f : 2^V \to \{0, 1\}$ of the sets S such that $\delta(S)$ is a deficient cut could violate the definition of an uncrossable function.

▶ **Example 19.** We construct the graph G by starting with a 4-cycle v_1, v_2, v_3, v_4, v_1 and then replacing each edge of the 4-cycle by a pair of parallel edges; thus, we have a 4-regular graph with 8 edges; we designate the following four edges as unsafe (and the other four edges are safe): both copies of edge $\{v_1, v_4\}$, one copy of edge $\{v_1, v_2\}$, and one copy of edge $\{v_3, v_4\}$. Clearly, G is a feasible instance of $(3, 1)$-FGC. On the other hand, G is infeasible for $(3, 2)$-FGC, and the cuts $\delta(\{v_1, v_2\})$ and $\delta(\{v_2, v_3\})$ are deficient. Note that the function $f : \{v_1, v_2, v_3, v_4\} \to \{0, 1\}$ that has $f(\{v_1, v_2\}) = f(\{v_2, v_3\}) = f(\{v_1\}) = f(\{v_4\}) = 1$ and $f(S) = 0$ for all other $S \subseteq V$ is not uncrossable (observe that the cuts $\delta(v_2)$ and $\delta(v_3)$ are not deficient). Moreover, observe that the minimal violated set $C = \{v_2, v_3\}$ crosses the violated set $S = \{v_1, v_2\}$.

Proof of Theorem 6. In the following, we use F to denote the set of edges picked by the algorithm at any step of the execution; we mention that our correctness arguments are valid despite this ambiguous notation; moreover, we use $\delta(S)$ rather than $\delta_F(S)$ to refer to a cut of the subgraph (V, F), where $\emptyset \neq S \subseteq V$.

Since F is a feasible edge-set for $(p, 1)$-FGC, any cut $\delta(S)$ (where $\emptyset \neq S \subseteq V$) either (i) has at least p safe edges or (ii) has at least $p + 1$ edges. Consider a node-set S that violates the requirements of the $(p, 2)$-FGC problem. We have $\emptyset \neq S \subsetneq V$ and there exist two unsafe edges $e, f \in \delta(S)$ such that $|\delta_F(S) \setminus \{e, f\}| \leq p - 1$. Since F is feasible for $(p, 1)$-FGC, we have $|\delta(S) \setminus \{e\}| \geq p$ and $|\delta(S) \setminus \{f\}| \geq p$. Thus, $|\delta_F(S)| = p + 1$. In other words, the node-sets S that need to be augmented have exactly $p + 1$ edges in $\delta(S)$, at least two of which are unsafe edges. Augmenting all such violated sets by at least one (safe or unsafe) edge will result in a feasible solution to $(p, 2)$-FGC. Let $f : 2^V \to \{0, 1\}$ be the indicator function of these violated sets. Observe that f is symmetric, that is, $f(S) = f(V \setminus S)$ for any

$S \subseteq V$; this additional property of f is useful for our arguments. We claim that f is a pliable function that satisfies property (γ), hence, we obtain an $O(1)$-approximation algorithm for $(p, 2)$-FGC, via the primal-dual method and Theorem 3.

Our proof of the following key lemma is presented in [3, Section 5].

▶ **Lemma 20.** *f is a pliable function that satisfies property (γ). Moreover, for even p, f is an uncrossable function.*

Lastly, we show that there is a polynomial-time subroutine for computing the minimal violated sets. Consider the graph (V, F). Note that size of a minimum cut of (V, F) is at least p since F is a feasible edge-set for $(p, 1)$-FGC. The violated sets are subsets $S \subseteq V$ such that $\delta(S)$ contains exactly $p+1$ edges, at least two of which are unsafe edges. Clearly, all the violated sets are contained in the family of sets S such that $\delta(S)$ is a 2-approximate min-cut of (V, F); in other words, $\{S \subsetneq V \ : \ p \leq |\delta(S)| \leq 2p\}$ contains all the violated sets. It is well known that the family of 2-approximate min-cuts in a graph can be listed in polynomial time, see [22, 28]. Hence, we can find all violated sets and all minimally violated sets in polynomial time.

Thus, we have a 20-approximation algorithm for $(p, 2)$-FGC via the primal-dual algorithm of [33] based on our results in Section 3. Furthermore, for even p, the approximation ratio is $6 (= 4 + 2)$ since the additive approximation-loss for the augmenting step is 2 when f is uncrossable (see Theorem 9). This completes the proof of Theorem 6. ◀

References

1 David Adjiashvili, Felix Hommelsheim, and Moritz Mühlenthaler. Flexible Graph Connectivity. *Mathematical Programming*, 192:409–441, 2022. doi:10.1007/s10107-021-01664-9.

2 Ajit Agrawal, Philip Klein, and R Ravi. When Trees Collide: An Approximation Algorithm for the Generalized Steiner Problem on Networks. *SIAM Journal on Computing*, 24(3):440–456, 1995. doi:10.1137/S0097539792236237.

3 Ishan Bansal, Joseph Cheriyan, Logan Grout, and Sharat Ibrahimpur. Improved Approximation Algorithms by Generalizing the Primal-Dual Method Beyond Uncrossable Functions. *CoRR*, abs/2209.11209v2, 2022. arXiv:2209.11209.

4 András A. Benczúr and Michel X. Goemans. Deformable Polygon Representation and Near-Mincuts. In *Building Bridges. Bolyai Society Mathematical Studies*, pages 103–135. Springer, 2008. doi:10.1007/978-3-540-85221-6_3.

5 Sylvia C. Boyd, Joseph Cheriyan, Arash Haddadan, and Sharat Ibrahimpur. Approximation algorithms for flexible graph connectivity. *Mathematical Programming*, 2023. doi:10.1007/s10107-023-01961-5.

6 Deeparnab Chakrabarty, Chandra Chekuri, Sanjeev Khanna, and Nitish Korula. Approximability of Capacitated Network Design. *Algorithmica*, 72(2):493–514, 2015. doi:10.1007/s00453-013-9862-4.

7 Chandra Chekuri and Rhea Jain. Approximating Flexible Graph Connectivity via Räcke Tree based Rounding. *CoRR*, abs/2211.08324, 2022. doi:10.48550/arXiv.2211.08324.

8 Chandra Chekuri and Rhea Jain. Augmentation based Approximation Algorithms for Flexible Network Design. *CoRR*, abs/2209.12273, 2022. doi:10.48550/arXiv.2209.12273.

9 George B. Dantzig, Lester R. Ford Jr., and Delbert R. Fulkerson. A Primal-Dual Algorithm for Linear Programs. In *Linear Inequalities and Related Systems*, volume 38 of *Annals of Mathematics Studies*, pages 171–182. Princeton University Press, 1957. doi:10.1515/9781400881987-008.

10 Reinhard Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer, 2017. doi:10.1007/978-3-662-53622-3.

11 Lisa Fleischer, Kamal Jain, and David P. Williamson. Iterative rounding 2-approximation algorithms for minimum-cost vertex connectivity problems. *Journal of Computer and System Sciences*, 72(5):838–867, 2006. doi:10.1016/j.jcss.2005.05.006.

12 András Frank. *Connections in Combinatorial Optimization*, volume 38 of *Oxford Lecture Series in Mathematics and Its Applications*. Oxford University Press, 2011.

13 Greg N. Frederickson and Joseph F. JáJá. Approximation Algorithms for Several Graph Augmentation Problems. *SIAM Journal on Computing*, 10(2):270–283, 1981. doi:10.1137/0210019.

14 Harold N. Gabow and Suzanne Gallagher. Iterated Rounding Algorithms for the Smallest k-Edge Connected Spanning Subgraph. *SIAM Journal on Computing*, 41(1):61–103, 2012. doi:10.1137/080732572.

15 Harold N. Gabow, Michel X. Goemans, Éva Tardos, and David P. Williamson. Approximating the Smallest k-Edge Connected Spanning Subgraph by LP-Rounding. *Networks*, 53(4):345–357, 2009. doi:10.1002/net.20289.

16 Harold N. Gabow, Michel X. Goemans, and David P. Williamson. An efficient approximation algorithm for the survivable network design problem. *Mathematical Programming*, 82:13–40, 1998. doi:10.1007/BF01585864.

17 Michel X. Goemans, Andrew V. Goldberg, Serge A. Plotkin, David B. Shmoys, Éva Tardos, and David P. Williamson. Improved Approximation Algorithms for Network Design Problems. In *Proceedings of the 5th Symposium on Discrete Algorithms*, pages 223–232, 1994.

18 Michel X. Goemans and David P. Williamson. A General Approximation Technique for Constrained Forest Problems. *SIAM Journal on Computing*, 24(2):296–317, 1995. doi:10.1137/S0097539793242618.

19 Michel X. Goemans and David P. Williamson. The Primal-Dual Method for Approximation Algorithms and Its Application to Network Design Problems. In *Approximation Algorithms for NP-Hard Problems*, chapter 4, pages 144–191. PWS Publishing Company, 1996. URL: https://math.mit.edu/~goemans/PAPERS/book-ch4.pdf.

20 Kamal Jain. A Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem. *Combinatorica*, 21(1):39–60, 2001. doi:10.1007/s004930170004.

21 David S. Johnson, Maria Minkoff, and Steven Phillips. The Prize Collecting Steiner Tree Problem: Theory and Practice. In David B. Shmoys, editor, *Proceedings of the 11th Symposium on Discrete Algorithms*, pages 760–769, 2000. URL: https://dl.acm.org/doi/10.5555/338219.338637.

22 David R. Karger and Clifford Stein. A New Approach to the Minimum Cut Problem. *Journal of the ACM*, 43(4):601–640, 1996. doi:10.1145/234533.234534.

23 Samir Khuller and Ramakrishna Thurimella. Approximation Algorithms for Graph Augmentation. *Journal of Algorithms*, 14(2):214–225, 1993. doi:10.1006/jagm.1993.1010.

24 Samir Khuller and Uzi Vishkin. Biconnectivity Approximations and Graph Carvings. *Journal of the ACM*, 41(2):214–235, 1994. doi:10.1145/174652.174654.

25 Harold W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. doi:10.1002/nav.3800020109.

26 Lap Chi Lau, R. Ravi, and Mohit Singh. *Iterative Methods in Combinatorial Optimization*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2011. doi:10.1017/CBO9780511977152.

27 Milena Mihail, David Shallcross, Nate Dean, and Marco Mostrel. A Commercial Application of Survivable Network Design: ITP/INPLANS CCS Network Topology Analyzer. In *Proceedings of the 7th Symposium on Discrete Algorithms*, pages 279–287, 1996. URL: https://dl.acm.org/doi/10.5555/313852.314074.

28 Hiroshi Nagamochi, Kazuhiro Nishimura, and Toshihide Ibaraki. Computing All Small Cuts in an Undirected Network. *SIAM Journal on Discrete Mathematics*, 10(3):469–481, 1997. doi:10.1137/S0895480194271323.

29 Dalit Naor, Dan Gusfield, and Charles Martel. A Fast Algorithm for Optimally Increasing the Edge Connectivity. *SIAM Journal on Computing*, 26(4):1139–1165, 1997. doi:10.1137/S0097539792234226.

30 Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, 2003.
31 Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2003. doi:10.1007/978-3-662-04565-7.
32 David P. Williamson and Michel X. Goemans. Computational Experience with an Approximation Algorithm on Large-Scale Euclidean Matching Instances. *INFORMS Journal on Computing*, 8(1):29–40, 1996. doi:10.1287/ijoc.8.1.29.
33 David P. Williamson, Michel X. Goemans, Milena Mihail, and Vijay V. Vazirani. A Primal-Dual Approximation Algorithm for Generalized Steiner Network Problems. *Combinatorica*, 15(3):435–454, 1995. doi:10.1007/BF01299747.
34 David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

A Missing Proofs from Section 3

This section has several lemmas and proofs from Section 3 that are used to prove our main result, Theorem 3.

▶ **Lemma 21.** *Suppose S_1 is a witness for edge e_1 and S_2 is a witness for edge e_2 such that S_1 overlaps S_2. Then there exist S_1' and S_2' satisfying the following properties:*

(i) *S_1' is a valid witness for edge e_1, S_2' is a valid witness for edge e_2, and S_1' does not overlap S_2'.*
(ii) *$S_1', S_2' \in \{S_1, S_2, S_1 \cup S_2, S_1 \cap S_2, S_1 \setminus S_2, S_2 \setminus S_1\}$.*
(iii) *either $S_1' = S_1$ or $S_2' = S_2$.*

Proof. We perform an exhaustive case analysis to check that the lemma is true. Note that at least two of the four sets $S_1 \cup S_2, S_1 \cap S_2, S_1 \setminus S_2, S_2 \setminus S_1$ must be violated in the current iteration. We consider the following cases.

1. If $S_1 \cup S_2$ and $S_1 \cap S_2$ are violated or $S_1 \setminus S_2$ and $S_2 \setminus S_1$ are violated, then the proof of Lemma 5.2 in [33] can be applied.
2. If $S_1 \cup S_2$ and $S_1 \setminus S_2$ are violated, then consider where the end-nodes of the edges e_1 and e_2 lie. If $e_1 \in E(S_1 \setminus S_2, V \setminus (S_1 \cup S_2))$ and $e_2 \in E(S_1 \setminus S_2, S_1 \cap S_2)$, then we can set $S_1' = S_1 \cup S_2$ and $S_2' = S_2$. The other possibilities for e_1 and e_2 are handled similarly.
3. If $S_1 \cap S_2$ and $S_1 \setminus S_2$ are violated, again consider where the end-nodes of the edges e_1 and e_2 lie. If $e_1 \in E(S_1 \setminus S_2, V \setminus (S_1 \cup S_2))$ and $e_2 \in E(S_1 \setminus S_2, S_1 \cap S_2)$, then we can set $S_1' = S_1 \cap S_2$ and $S_2' = S_2$. The other possibilities for e_1 and e_2 are handled similarly.

This completes the proof of the lemma. ◀

▶ **Lemma 22.** *Suppose a set A_1 overlaps a set A_2 and a third set A_3 does not overlap A_1 nor A_2. Then A_3 does not overlap any of the sets $A_1 \cup A_2, A_1 \cap A_2, A_1 \setminus A_2, A_2 \setminus A_1$.*

Proof. Note that since A_3 does not overlap A_1 (or A_2), they are either disjoint or one contains the other. We consider the following cases.

1. Suppose $A_3 \cap A_1 = \emptyset$. Then $A_2 \not\subseteq A_3$ since $A_1 \cap A_2 \neq \emptyset$. If $A_3 \cap A_2 = \emptyset$, then $A_3 \subseteq V \setminus A_1 \cup A_2$ and we are done. Finally if $A_3 \subseteq A_2$, then $A_3 \subseteq A_2 \setminus A_1$ and we are done.
2. Suppose $A_1 \subseteq A_3$. Then $A_3 \cap A_2 \neq \emptyset$ since $A_1 \cap A_2 \neq \emptyset$. Also, $A_3 \not\subseteq A_2$ since $A_1 \not\subseteq A_2$. If $A_2 \subseteq A_3$, then $A_1 \cup A_2 \subseteq A_3$ and we are done.
3. Suppose $A_3 \subseteq A_1$. Then $A_2 \not\subseteq A_3$ since $A_2 \setminus A_1 \neq \emptyset$. If $A_3 \subseteq A_2$, then $A_3 \subseteq A_1 \cap A_2$ and we are done. Finally if $A_3 \cap A_2 = \emptyset$, then $A_3 \subseteq A_1 \setminus A_2$ and we are done. ◀

▶ **Lemma 11.** *There exists a laminar family of witness sets.*

Proof. We show that any witness family can be uncrossed and made laminar. We prove this by induction on the size of the witness family ℓ.

Base Case: Suppose $\ell = 2$, then one application of Lemma 21 is sufficient.

Inductive Hypothesis: If $S_1, \ldots, S_\ell$ are witness sets for edges $e_1, \ldots, e_\ell$ respectively with $\ell \leq k$, then, by repeatedly applying Lemma 21, one can construct witness sets $S'_1, \ldots, S'_\ell$ for the edges $e_1, \ldots, e_\ell$ respectively such that $S'_1, \ldots, S'_\ell$ is a laminar family.

Inductive Step: Consider $k+1$ witness sets $S_1, \ldots, S_{k+1}$. By the inductive hypothesis, we can uncross all the witness sets $S_1, \ldots, S_k$ to obtain witness sets $S'_1, \ldots, S'_k$ that form a laminar family. We now consider the following cases.

1. If S_{k+1} does not overlap some S'_i, say S'_1, then we can apply the inductive hypothesis to the k sets $S'_2, \ldots, S'_k, S_{k+1}$ and we obtain a laminar family of witness sets, none of which overlap S'_1 either (by Lemma 22) and so we are done.
2. Suppose S_{k+1} overlaps all the sets $S'_1, \ldots, S'_k$ and for some S'_i, say S'_1, applying Lemma 21 to the pair S'_1, S_{k+1} gives S'_1, S'_{k+1}. Then S'_1 does not overlap any of the witness sets $S'_2, \ldots, S'_{k+1}$, hence, applying the inductive hypothesis to these k sets gives us a laminar family of witness sets $S''_2, \ldots, S''_k$. By Lemma 22, S'_1 does not overlap any of the sets $S''_2, \ldots, S''_k$ and so we are done.
3. Suppose S_{k+1} overlaps all the sets $S'_1, \ldots, S'_k$ and, for every S'_i, applying Lemma 21 to the pair S'_i, S_{k+1} gives S''_i, S_{k+1}. Then after doing this for every S'_i, we end up with the witness family $S''_1, \ldots, S''_k, S_{k+1}$ with the property that S_{k+1} does not overlap any of the other sets. Applying the inductive hypothesis to the k sets $S''_1, \ldots, S''_k$ gives us a laminar family of witness sets $S'''_1, \ldots, S'''_k$. By Lemma 22, S_{k+1} does not overlap any of the sets $S'''_1, \ldots, S'''_k$ and so we are done. ◀

B Optimal Dual Solutions with Non-Laminar Supports

In this section, we describe an instance of the AugSmallCuts problem where none of the optimal dual solutions (to the dual LP given in (2.1), Section 2) have a laminar support. Recall that the connectivity requirement function f for the AugSmallCuts problem is pliable and satisfies property (γ), as seen in the proof of Theorem 4.

Consider the graph $G = (V, E)$ (shown in Figure 1 below using solid edges) which is a cycle on 4 nodes $1, 2, 3, 4$, in that order. Edge-capacities are given by $u_{12} = 3, u_{23} = 4, u_{34} = 2, u_{41} = 1$. The link-set (shown using dashed edges) is $L = \{12, 23, 34, 41\}$, a disjoint copy of E. Link-costs are given by $c_{12} = c_{23} = c_{34} = 1$ and $c_{41} = 2$.

Consider the AugSmallCuts instance that arises when we choose $\widetilde{\lambda} = 6$. The family of small cuts (with capacity strictly less than $\widetilde{\lambda}$) is given by $\bigcup_{S \in \mathcal{A}} \{S, V \setminus S\}$, where

$$\mathcal{A} = \{\{1\}, \{1,2\}, \{2,3\}, \{1,2,3\}\}.$$

The associated pliable function f satisfies $f(S) = 1$ if and only if $S \in \mathcal{A}$ or $V \setminus S \in \mathcal{A}$ holds. Observe that f is *not* uncrossable since $f(\{1,2\}) = 1 = f(\{2,3\})$, but $f(\{1,2\} \cap \{2,3\}) = f(\{2\}) = 0$ and $f(\{2,3\} \setminus \{1,2\}) = f(\{3\}) = 0$. Also note that the minimal violated set $\{2,3\}$ (w.r.t. $F = \emptyset$) crosses the violated set $\{1,2\}$.

It can be seen that there are three inclusion-wise minimal link-sets that are feasible for the above instance and these are given by

$$\mathcal{C} := \{\{12, 23, 34\}, \{12, 41\}, \{34, 41\}\}.$$

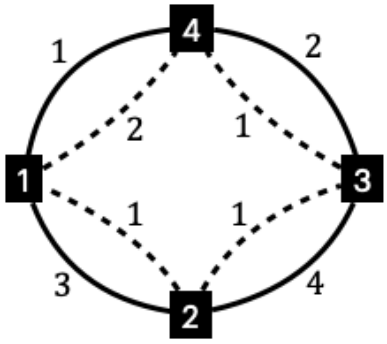


Figure 1 An instance of the AugSmallCuts problem where none of the optimal dual solutions have a laminar support.

Since each $F \in \mathcal{C}$ has cost 3, the optimal value for the instance is 3. Next, since L contains at least two links from every nontrivial cut, the vector $x \in [0,1]^L$ with $x_e = \frac{1}{2}$, $\forall e \in L$ is a feasible augmentation for the fractional version of the instance, i.e., x is feasible for the primal LP given in (2.1), Section 2. Therefore, the optimal value of the primal LP is at most $\frac{5}{2}$.

Now, consider the dual LP, which is explicitly stated below. The dual packing-constraints are listed according to the following ordering of the links: $12, 23, 34, 41$. For notational convenience, we use the shorthand y_1 to denote the dual variable $y_{\{1\}}$ corresponding to the set $\{1\}$. We use similar shorthand to refer to the dual variables of the other sets; thus, y_{234} refers to the dual variable $y_{\{2,3,4\}}$, etc.

$$
\begin{array}{lccccccccc}
\max & (y_1 + y_{234}) & + & (y_{12} + y_{34}) & + & (y_{23} + y_{14}) & + & (y_{123} + y_4) & & \\
\text{subject to:} & (y_1 + y_{234}) & & & + & (y_{23} + y_{14}) & & & \leq & 1 \\
& & & (y_{12} + y_{34}) & & & & & \leq & 1 \\
& & & & & (y_{23} + y_{14}) & + & (y_{123} + y_4) & \leq & 1 \\
& (y_1 + y_{234}) & + & (y_{12} + y_{34}) & & & + & (y_{123} + y_4) & \leq & 2 \\
& & & & & & & y & \geq & 0.
\end{array}
$$

Observe that adding all packing constraints gives $2 \cdot \sum_{S \in \mathcal{A}} (y_S + y_{V \setminus S}) \leq 5$, hence, the optimal value of the dual LP is at most $5/2$. Moreover, a feasible dual solution with objective $5/2$ must satisfy the following conditions:

$$y_1 + y_{234} = y_{23} + y_{14} = y_{123} + y_4 = \frac{1}{2} \quad \text{and} \quad y_{12} + y_{34} = 1.$$

Clearly, there is at least one solution to the above set of equations, hence, by LP duality, the optimal value of both the primal LP and the dual LP is $5/2$.

Furthermore, any optimal dual solution y^* satisfies $\max(y^*_S, y^*_{V \setminus S}) > 0$ for all $S \in \mathcal{A}$ (by the above set of equations). We conclude by arguing that for any optimal dual solution y^*, its support $\mathcal{S}(y^*) = \{S \subseteq V : y^*_S > 0\}$ is non-laminar, because some two sets $A, B \in \mathcal{S}(y^*)$ cross. Since the relation A crosses B is closed under taking set-complements (w.r.t. the ground-set V), we may assume w.l.o.g. that the support contains each set in $\mathcal{A} = \{\{1\}, \{1,2\}, \{2,3\}, \{1,2,3\}\}$. The support of y^* is not laminar because $\{1,2\}$ and $\{2,3\}$ cross.

Approximation Algorithms for Envy-Free Cake Division with Connected Pieces

Siddharth Barman
Indian Institute of Science, Bangalore, India

Pooja Kulkarni
University of Illinois at Urbana-Champaign, IL, USA

Abstract

Cake cutting is a classic model for studying fair division of a heterogeneous, divisible resource among agents with individual preferences. Addressing cake division under a typical requirement that each agent must receive a connected piece of the cake, we develop approximation algorithms for finding envy-free (fair) cake divisions. In particular, this work improves the state-of-the-art additive approximation bound for this fundamental problem. Our results hold for general cake division instances in which the agents' valuations satisfy basic assumptions and are normalized (to have value 1 for the cake). Furthermore, the developed algorithms execute in polynomial time under the standard Robertson-Webb query model.

Prior work has shown that one can efficiently compute a cake division (with connected pieces) in which the additive envy of any agent is at most 1/3. An efficient algorithm is also known for finding connected cake divisions that are (almost) 1/2-multiplicatively envy-free. Improving the additive approximation guarantee and maintaining the multiplicative one, we develop a polynomial-time algorithm that computes a connected cake division that is both $\left(\frac{1}{4} + o(1)\right)$-additively envy-free and $\left(\frac{1}{2} - o(1)\right)$-multiplicatively envy-free. Our algorithm is based on the ideas of interval growing and envy-cycle elimination.

In addition, we study cake division instances in which the number of distinct valuations across the agents is parametrically bounded. We show that such cake division instances admit a fully polynomial-time approximation scheme for connected envy-free cake division.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory; Theory of computation → Algorithmic game theory and mechanism design

Keywords and phrases Fair Division, Envy-Freeness, Envy-Cycle Elimination

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.16

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2208.08670` [6]

Funding *Siddharth Barman*: Siddharth Barman gratefully acknowledges the support of a SERB Core research grant (CRG/2021/006165).

1 Introduction

Cake cutting is an exemplar of fair division literature [9, 22, 21]. Since the foundational work of Steinhaus, Banach, and Knaster [24], fair cake division has been extensively studied over decades, and it continues to inspire research, including algorithmic breakthroughs [4], deep mathematical connections [17, 20], and applicable variants [16]. This fair-division model captures resource-allocation domains in which a divisible, heterogeneous resource (metaphorically, the cake) needs to be fairly divided among agents with individual, distinct preferences. For instance, cake division has been studied in the context of border negotiations [9] and fair electricity division [5]. The predominant fairness notion of envy-freeness was also defined in

the cake-division context [14]. This solution concept deems a cake division to be fair if each agent values the piece assigned to her over that of any other agent, i.e., if the agents are not envious of each other.

Formally, the cake is modeled as the interval $[0, 1]$ and the cardinal preferences of the n participating agents (over pieces of the cake) are expressed via valuation functions $v_1, \ldots, v_n$; in particular, $v_i(I) \in \mathbb{R}_+$ denotes the value that agent i has for any interval $I \subseteq [0, 1]$. In this work, we address cake division under the requirement that every agent must receive a connected piece (i.e., an interval) of the cake. That is, our goal is to partition the cake $[0, 1]$ into exactly n pairwise-disjoint intervals and assign them among the n agents. This connectivity requirement is standard in literature and is motivated by practical settings wherein each agent must receive a contiguous part of the resource; consider, for instance, division of land or non-preemptive scheduling. Hence, in this setup, an envy-free (i.e., fair) division corresponds to a partition of $[0, 1]$ into n pairwise-disjoint intervals, $I_1, I_2, \ldots, I_n$, such that assigning each interval I_i to agent $i \in [n]$ results in no envy, i.e., $v_i(I_i) \geq v_i(I_j)$, for all agents $i, j \in [n]$.

The significance of envy-freeness is elevated by universal existential guarantees: under benign assumptions on agents' valuations, an envy-free cake division, in which each agent receives a connected piece, is guaranteed to exist [25, 23, 27]. Here, the elegant proof of Su [27] is considered a foundational result across all of fair division. These strong existential results, however, do not have an algorithmic counterpart. Stromquist [26] has shown that even a finite-time algorithm does not exist for computing an envy-free cake division with connected pieces; this negative result holds in a model where the valuations are specified via an (adversarial) oracle. Furthermore, it is known that, under ordinal preferences, achieving envy-freeness with connected pieces is PPAD-hard [12].

These algorithmic barriers in route to finding exact envy-free cake divisions necessitate the study of approximation guarantees. The current paper contributes to this research thread by developing algorithms for finding connected cake divisions that are approximately envy-free. In particular, this work improves the state-of-the-art additive approximation bound for this fundamental fair division problem.

Our Results and Techniques. Our algorithmic results hold for general cake division instances in which the agents' valuations satisfy basic assumptions and are normalized, such that the value for the entire cake for every agent is equal to one, i.e., $v_i([0, 1]) = 1$ for all agents $i \in [n]$. Furthermore, the developed algorithms execute in the standard Robertson-Webb query model [21].

We address both additive and multiplicative approximations of envy-freeness. Specifically, for parameter $\varepsilon \in (0, 1)$, a connected cake division $I_1, \ldots I_n$ (in which interval I_i is assigned to agent $i \in [n]$) is said to be ε-envy-free (ε-EF) iff no agent has more than ε envy towards any other agent, i.e., $v_i(I_i) \geq v_i(I_j) - \varepsilon$ for all agents $i, j \in [n]$. Analogously, an α-multiplicatively envy-free (α-mult-EF) cake division $I_1, \ldots I_n$ is one in which the envy is multiplicatively bounded within a factor of α, i.e., $v_i(I_i) \geq \alpha v_i(I_j)$ for all agents $i, j \in [n]$; here parameter $\alpha \in (0, 1]$.

Our main result is a polynomial-time algorithm that computes a cake division (with connected pieces) that is simultaneously $\left(\frac{1}{4} + c\right)$-EF and $\left(\frac{1}{2} - c'\right)$-mult-EF (Theorems 13 and 14); here, c and c' are polynomially small (in n) terms. For instance, our algorithm can be used to efficiently find a cake division that is 0.251-EF and 0.499-mult-EF.

Our result improves upon the previously best known additive approximation guarantee. Specifically, prior work of Goldberg et al. [15] provides an efficient algorithm for computing a $\frac{1}{3}$-EF cake division (with connected pieces); here, the computed allocation can leave some

agents with no cake allocated to them and, hence, incur unbounded multiplicative envy. On the multiplicative front, for a lower order term κ, Arunachaleswaran et al. [3] obtain a $\left(\frac{1}{2} - \kappa\right)$-mult-EF guarantee, in conjunction with an additive envy bound close to $\frac{1}{3}$. Therefore, for envy-free cake division, we improve the additive approximation guarantee from $\frac{1}{3}$ to (almost) $\frac{1}{4}$, while maintaining the best known multiplicative one.

Our algorithm extends the interval-growing method of [3] with the idea of bifurcating intervals (see Definition 3). Such intervals satisfy the property that if an agent i receives an interval that is bifurcating with respect to v_i, then irrespective of how the rest of the cake is assigned, agent i's envy towards others remains bounded. For the algorithm's design and analysis, we modify each agent's valuation to have a preference for bifurcating intervals. With these modified valuations, we build upon the idea of interval growing. In particular, we first obtain an allocation of (pairwise disjoint) intervals that might partially cover the entire cake, though induce bounded envy among the agents and against the unassigned intervals. Then, we use an envy-cycle-elimination idea to further allocate small pieces till at most n unassigned intervals remain. Envy-graphs and the cycle-elimination method have been extensively utilized in fair division; see, e.g., [19]. However, their use for *contiguous* cake cutting (i.e., division under the contiguity requirement) is novel. We employ cycle elimination in such a way that envy remains bounded as we allocate more and more of the cake. Finally, we have n assigned and at most n unassigned intervals. We pair up adjacent assigned and unassigned intervals to overall obtain a complete partition of the cake that has bounded envy; see Section 3.1 for a detailed description of the algorithm.

It is relevant to note the technical distinctions between the algorithm of Arunachaleswaran et al. [3] and the current one. In contrast to the prior work, the current algorithm executes with a novel modification of the valuations (to incorporate preferences towards bifurcating intervals). Moreover, the current analysis is more involved; in particular, the analysis requires multiple new lemmas and consideration of intricate cases (see, e.g., Lemmas 6 to 11 and the case analysis in the proof of Theorem 13).

Our second result addresses cake division instances in which the number of distinct valuations is bounded. Specifically, we consider instances in which, for a parameter $\varepsilon \in (0, 1)$ and across the n agents, the number of *distinct* valuations is at most $(\varepsilon n - 1)$. For such instances with bounded heterogeneity, we provide an algorithm that computes ε-EF allocations in time polynomial in n and $\frac{1}{\varepsilon}$ (Theorem 16). Note that such settings naturally generalize the case of identical valuations. Fair division algorithms under identical valuations have been developed in many contexts (beyond cake division). Our result shows that, under this natural generalization, a strong additive approximation guarantee can be obtained for connected envy-free cake division; see Section 4.[1]

Additional Related Work. Prior works in (connected) cake division have also studied improved approximation guarantees for specific valuation classes. For instance, it is shown in [7] that a connected cake division with arbitrarily small envy can be computed efficiently if the agents' value densities satisfy the monotone likelihood ratios property. Another studied valuation class is that of single-block valuations; in particular, these correspond to valuations in which the agents have a constant density over some (agent-specific) interval of cake and zero everywhere else. The work of Alijani et al. [1] provides an efficient algorithm for finding (exact) envy-free cake division under single-block valuations that satisfy an ordering property.

[1] We also detail at the end of the Section 4 that achieving multiplicative approximation bounds for envy under bounded heterogeneity is as hard as it is in the general case.

For arbitrary single-block valuations (without the ordering property), Goldberg et al. [15] obtain a $\frac{1}{4}$-EF guarantee. They also obtain NP-hardness results for connected envy-free cake division under additional constraints, such as conforming to a given cut point.

Focussing on query complexity, Brânzei and Nisan [10] show that an ε-EF cake division (with connected pieces) can be computed in a query efficient manner but the algorithm runs in time exponential in n and $\frac{1}{\varepsilon}$. By contrast, we develop polynomial-time algorithms.

The study of approximation guarantees – to bypass computational or existential barriers – is an established research paradigm in theoretical computer science. For instance, in discrete fair division, (multiplicative) approximation bounds for the maximin share has received significant attention in recent years; see [2] and multiple references therein. Also, in algorithmic game theory, approximation guarantees for Nash equilibria in two-player games have been extensively studied; see, e.g., [11, 28, 18]. Our work contributes to this thematic thread with a focus on cake division.

Non-Contiguous Cake Division. Algorithmic aspects of (exact) envy-free cake division remain challenging even without the connectivity requirement. In fact, the existence of a finite-time algorithm for noncontiguous envy-free cake division remained open until the work of Brams and Taylor [8]. For noncontiguous envy-free cake divisions, an explicit runtime bound – albeit a hyper-exponential one – was obtained in the notable work of Aziz and Mackenzie [4]. Prior works have also addressed non-contiguous envy-free cake division for special valuation classes: [1] obtains a polynomial-time algorithm for finding (exact, but not necessarily contiguous) envy-free cake divisions under single-block valuations. Also, [29] develops a polynomial-time algorithm for computing non-contiguous envy-free cake divisions under single-peaked preferences. For the non-contiguous setting, [19] provides a fully polynomial-time approximation scheme for computing approximate envy-free cake divisions.

2 Notation and Preliminaries

We consider fair division of a divisible, heterogeneous good – i.e., a cake – among n agents. The cake is modeled as the interval $[0, 1]$, and the cardinal preferences of the agents $i \in [n]$ over the cake are expressed via valuation functions v_i. In particular, $v_i(I) \in \mathbb{R}_+$ denotes the valuation that agent $i \in [n]$ has for any interval $I = [x, y] \subseteq [0, 1]$; here, $0 \leq x \leq y \leq 1$. As in prior works (see, e.g., [21]), we will address valuations $\{v_i\}_{i=1}^n$ that are (i) nonnegative: $v_i(I) \geq 0$ for all intervals $I \subseteq [0, 1]$, (ii) normalized: $v_i([0, 1]) = 1$ for all agents i, (iii) divisible: for any interval $[x, y] \subseteq [0, 1]$ and scalar $\lambda \in [0, 1]$, there exists a point $z \in [x, y]$ with the property that $v_i([x, z]) = \lambda v_i([x, y])$, and (iv) additive: $v_i(I \cup J) = v_i(I) + v_i(J)$ for any pair of disjoint intervals $I, J \subseteq [0, 1]$.

These properties ensure that for all agents $i \in [n]$ and each interval $I \subseteq [0, 1]$ we have $0 \leq v_i(I) \leq 1$. Also, note that, since the valuations v_i are divisible, they are non-atomic: $v_i([x, x]) = 0$ for all points $x \in [0, 1]$. Relying on this property, we will throughout regard, as a convention, two intervals to be disjoint even if they intersect exactly at an endpoint. Our algorithms efficiently execute in the standard Robertson-Webb query model [22], that provides access to the agents' valuations via the following queries:

- **(i)** Evaluation queries, $\mathsf{Eval}_i(x, y)$: Given points $0 \leq x \leq y \leq 1$, the oracle returns the value that agent i has for the interval $[x, y]$, i.e., returns $v_i([x, y])$.
- **(ii)** Cut queries, $\mathsf{Cut}_i(x, \nu)$: Given an initial point $x \in [0, 1]$ and a value $\nu \in (0, 1)$, the oracle returns the leftmost point $y \in [x, 1]$ with the property that $v_i([x, y]) \geq \nu$. If no such y exists, the response to the query is 1.

Allocations. The current work addresses fair cake division under the requirement that each agent must receive a connected piece. That is, we focus solely on assigning to each agent a single sub-interval of $[0,1]$. Specifically, in a cake division instance with n agents, an allocation is defined as an n-tuple of pairwise-disjoint intervals, $\mathcal{I} = (I_1, I_2, \ldots, I_n)$, where interval I_i is assigned to agent $i \in [n]$ and $\bigcup_{i=1}^{n} I_i = [0,1]$. In addition, we will use the term partial allocation to refer to an n-tuple of pairwise-disjoint intervals $\mathcal{P} = (P_1, \ldots, P_n)$ that do not necessarily cover the entire cake, $\cup_{i\in[n]} P_i \subsetneq [0,1]$. Note that, in an allocation $\mathcal{J} = (J_1, J_2, \ldots, J_n)$, partial or complete, each interval J_i is indexed to identify the agent i that owns the interval, and not how the intervals are ordered within $[0,1]$.

Furthermore, for a partial allocation $\mathcal{P} = (P_1, \ldots, P_n)$, write $\mathcal{U}_{\mathcal{P}} = \{U_1, \ldots, U_t\}$ to denote the collection of unassigned intervals that remain after the assigned ones (i.e., P_is) are removed from $[0,1]$. Formally, $\mathcal{U}_{\mathcal{P}} = \{U_1, \ldots, U_t\}$ is the minimum-cardinality collection of disjoint intervals that satisfy $\bigcup_i U_i = [0,1] \setminus \left(\bigcup_{j=1}^{n} P_j\right)$.

Approximate Envy-Freeness. The fairness notions studied in this work are defined next. An allocation $\mathcal{E} = (E_1, \ldots, E_n)$ is said to be envy free (EF) iff each agent prefers the interval assigned to her over that of any other agent, $v_i(E_i) \geq v_i(E_j)$ for all agents $i, j \in [n]$. This paper addresses both additive and multiplicative approximations of envy-freeness.

▶ **Definition 1** (ε-EF). *In a cake division instance with n agents and for a parameter $\varepsilon \in (0,1)$, an (partial) allocation $\mathcal{I} = (I_1, I_2, \ldots, I_n)$ is said to be ε-additively envy-free (ε-EF) iff $v_i(I_i) \geq v_i(I_j) - \varepsilon$, for all agents $i, j \in [n]$.*

▶ **Definition 2** (α-mult-EF). *For a parameter $\alpha \in (0,1)$, an (partial) allocation $\mathcal{I} = (I_1, I_2, \ldots, I_n)$ is said to be α-multiplicatively envy-free (α-mult-EF) iff, for all agents $i, j \in [n]$, we have $v_i(I_i) \geq \alpha\ v_i(I_j)$.*

3 Approximation Algorithm for Envy-Free Cake Division

This section develops an algorithm for efficiently computing a cake division (with connected pieces) that is $\left(\frac{1}{4} + o(1)\right)$-EF and $\left(\frac{1}{2} - o(1)\right)$-mult-EF.

For the design of the algorithm, we will use the notion of bifurcating intervals. For an agent i, a bifurcating interval X satisfies the property that, if i is assigned interval X, then one can divide the rest of the cake in any way and still agent i will have at most $1/4$ envy towards any other agent. Formally,

▶ **Definition 3** (Bifurcating Intervals). *An interval $[x,y] \subseteq [0,1]$ is said to be a bifurcating interval for an agent $i \in [n]$ iff*

$$v_i([x,y]) \geq \frac{1}{4}, \qquad v_i([0,x]) \leq \frac{1}{2}, \qquad \text{and} \quad v_i([y,1]) \leq \frac{1}{2}.$$

For each agent $i \in [n]$, we extend the valuation v_i to a function $\widehat{v}_i$ which codifies a preference towards bifurcating intervals. Formally, for each agent $i \in [n]$ and any interval $X \subseteq [0,1]$, define

$$\widehat{v}_i(X) \coloneqq \begin{cases} 1 & \text{if } X \text{ is bifurcating for } i. \\ v_i(X) & \text{if } X \text{ is not bifurcating for } i. \end{cases} \tag{1}$$

We note that, in contrast to the valuation v_i, the function $\widehat{v}_i$ is not additive.[2] However, analogous to v_i, the function $\widehat{v}_i$ is monotonic, normalized, and nonnegative. In addition, given access to $\mathsf{Eval}_i()$ queries, we can efficiently compute $\widehat{v}_i(X)$ for any interval $X \subseteq [0,1]$. We will show in the analysis that the algorithm's steps involving $\widehat{v}_i$s can be implemented efficiently, given Robertson-Webb query access to the underlying valuations. The claim below provides a bound on the value of non-bifurcating intervals.

▷ Claim 4. For any agent $i \in [n]$, if $Y \subseteq [0,1]$ is *not* a bifurcating interval, then, $\widehat{v}_i(Y) = v_i(Y) < \frac{1}{2}$.

Proof. All intervals $H \subseteq [0,1]$ of value $v_i(H) \geq \frac{1}{2}$ are bifurcating; see Definition 3 and recall that the agents' valuations are normalized, $v_i([0,1]) = 1$. Hence, for any non-bifurcating interval $Y \subseteq [0,1]$ we have $v_i(Y) < \frac{1}{2}$, i.e., $\widehat{v}_i(Y) = v_i(Y) < \frac{1}{2}$ (see equation (1)). ◁

We will also use the construct of an envy-graph. Specifically, for a partial allocation $\mathcal{P} = (P_1, \ldots, P_n)$, an envy-graph $G_{\mathcal{P}}$ is a directed graph over n vertices. Here, the vertices represent the n agents and a directed edge, from vertex i to vertex j, is included in the graph iff $\widehat{v}_i(P_i) < \widehat{v}_i(P_j)$. Envy-graphs and the cycle-elimination algorithm (detailed next) have been extensively utilized in discrete fair division; see, e.g., [19]. However, their use for *contiguous* cake cutting is novel.

We will next show that, if for any partial allocation $\mathcal{P} = (P_1, \ldots, P_n)$, the envy-graph $G_{\mathcal{P}}$ contains a cycle, then we can in fact resolve the cycle – by reassigning the intervals – and eventually obtain a partial allocation $\mathcal{Q} = (Q_1, \ldots, Q_n)$ whose envy-graph $G_{\mathcal{Q}}$ is acyclic.[3]

▶ **Lemma 5.** *Given any partial allocation $\mathcal{P} = (P_1, \ldots, P_n)$, one can reassign the intervals P_is among the agents and efficiently find another partial allocation $\mathcal{Q} = (Q_1, \ldots, Q_n)$ with the properties that*

(i) *The envy-graph $G_{\mathcal{Q}}$ is acyclic.*

(ii) *The value $\widehat{v}_i(Q_i) \geq \widehat{v}_i(P_i)$, for all agents $i \in [n]$.*

The proof of Lemma 5 is standard and, for completeness, is provided in Appendix A. Recall that, for any partial allocation $\mathcal{P} = (P_1, \ldots, P_n)$, the set $\mathcal{U}_{\mathcal{P}} = \{U_1, \ldots, U_t\}$ denotes the collection of unassigned intervals that remain after the intervals P_is are removed from $[0,1]$. Also, note that for any partial allocation $\mathcal{P} = (P_1, \ldots, P_n)$, we have $|\mathcal{U}_{\mathcal{P}}| \leq n + 1$.

3.1 Interval Growing and Cycle Elimination

Our algorithm (Algorithm 1) consists of two phases. In Phase I (Lines 2 to 6 in Algorithm 1), which we call the interval growing phase, the algorithm starts with empty intervals – i.e., $P_i = \emptyset$ for all i – and iteratively grows these intervals while maintaining bounded envy among the agents. In particular, to extend a partial allocation $\mathcal{P} = (P_1, \ldots, P_n)$, we first judiciously select an unassigned interval $U \in \mathcal{U}_{\mathcal{P}}$ and then assign an inclusion-wise minimal sub-interval of U to an agent a. The sub-interval of U and agent a are selected such that the function value, $\widehat{v}_a$, increases appropriately and, at the same time, the envy towards a (from any other agents) remains bounded. Note that, in this phase, the cake might not be allocated completely, but the invariant of bounded envy is maintained throughout. Phase I terminates with a partial allocation $\overline{\mathcal{P}} = (\overline{P}_1, \ldots, \overline{P}_n)$ under which each agent $i \in [n]$ has bounded envy towards the other agents and towards all the unassigned intervals $U \in \mathcal{U}_{\overline{\mathcal{P}}}$ (see Lemma 8).

[2] Also, the function $\widehat{v}_i$ is not divisible.

[3] Here, the reassignment of the intervals implies that there exists a permutation $\pi \in \mathbb{S}_n$ such that $Q_i = P_{\pi(i)}$ for all agents i.

Algorithm 1 Approximation Algorithm for Connected Cake Division.

Input: A cake division instance with oracle access to the valuations $\{v_i\}_{i=1}^n$ of the n agents and a fixed constant $\delta \in (0,1)$.

Output: A complete allocation $(I_1, \ldots, I_n)$.

1: Initialize partial allocation $\mathcal{P} = (P_1, \ldots, P_n) = (\emptyset, \ldots, \emptyset)$ and $\mathcal{U}_\mathcal{P} = \{[0,1]\}$.
2: **while** there exists an unassigned interval $U = [\ell, r] \in \mathcal{U}_\mathcal{P}$ and an agent $i \in [n]$ such that $\widehat{v}_i(U) \geq \widehat{v}_i(P_i) + \frac{\delta}{n}$ **do**
3: Let $C \coloneqq \left\{i \in [n] \ : \widehat{v}_i(U) \geq \widehat{v}_i(P_i) + \frac{\delta}{n}\right\}$ and, for every agent $i \in C$, set $r_i \in [\ell, r]$ to be the leftmost point such that $\widehat{v}_i([\ell, r_i]) \geq \widehat{v}_i(P_i) + \frac{\delta}{n}$.
4: Select agent $a \in \arg\min_{i \in C} \ r_i$ and update the partial allocation $\mathcal{P}$: assign $P_a \leftarrow [\ell, r_a]$ and keep the interval assignment of all other agents unchanged.
5: Update $\mathcal{U}_\mathcal{P}$ to be the collection of intervals that are left unassigned under the current partial allocation $\mathcal{P}$.
6: **end while**
7: **while** $|\mathcal{U}_\mathcal{P}| > n$ **do**
8: Update $\mathcal{P} = (P_1, \ldots, P_n)$ following Lemma 5 to ensure that the envy-graph $G_\mathcal{P}$ is acyclic.
9: Let $s \in [n]$ be a source vertex in the graph $G_\mathcal{P}$, with assigned interval $P_s = [\ell_s, r_s]$.
10: Let $\widetilde{U} = [r_s, \widetilde{r}] \in \mathcal{U}_\mathcal{P}$ be the unassigned interval that is adjacent (on the right) to P_s. {Since $|\mathcal{U}_\mathcal{P}| > n$, such an interval $\widetilde{U}$ is guaranteed to exist.}
11: Write $x \in [r_s, \widetilde{r}]$ to be the point with the property that $v_i([r_s, x]) \leq \frac{\delta}{n}$ for all agents i and this inequality is tight for at least one agent. Append $P_s \leftarrow P_s \cup [r_s, x]$. {If for all agents the value of $\widetilde{U}$ is at most $\frac{\delta}{n}$, then append $P_s \leftarrow P_s \cup \widetilde{U}$.}
12: **end while**
13: Index the unassigned intervals $U_j \in \mathcal{U}_\mathcal{P}$ such that each U_j is adjacent to a *distinct* interval P_j, for all j. {Since $|\mathcal{U}_\mathcal{P}| \leq n$, such an indexing is possible.}
14: For all agents i, set interval $I_i = P_i \cup U_i$. {If an unassigned interval is not associated with P_i, then set $I_i = P_i$.}
15: **return** allocation $\mathcal{I} = (I_1, \ldots, I_n)$.

At the end of Phase I, it is possible that the number of unassigned intervals is $n+1$. The objective of Phase II (Lines 7 to 12 in the algorithm) is to reduce the number of unassigned intervals, while maintaining bounded envy between the agents and against the unassigned intervals. Towards this, we use the cycle-elimination method (Lemma 5) to first ensure that for the maintained partial allocation $\mathcal{P}$ the envy-graph $G_\mathcal{P}$ is acyclic. Now, given that the directed graph $G_\mathcal{P}$ is acyclic, it necessarily admits a source vertex $s \in [n]$, i.e., a vertex s with no incoming edges. Furthermore, by the definition of the envy graph, we get that no agent has sufficiently high envy towards the source vertex $s \in [n]$. With this guarantee in hand, we enlarge the interval assigned to s (i.e., enlarge P_s) while maintaining bounded envy overall. Specifically, we append to P_s a piece of small enough value from the unassigned interval $\widetilde{U}$ adjacent to P_s. Since $|\mathcal{U}_\mathcal{P}| = n+1$, an unassigned interval, adjacent to P_s, is guaranteed to exist. Also, note that this extension ensures that P_s continues to be a connected piece of the cake, i.e., agent s continues to receive a single interval. Performing such updates, Phase II efficiently finds a partial allocation $\mathcal{P}$ with the property that $|\mathcal{U}_\mathcal{P}| \leq n$. Since at the end of Phase II the number of unassigned intervals is at most n, we can associate each unassigned interval $U \in \mathcal{U}_\mathcal{P}$ with a *distinct* assigned interval P_j that is adjacent to U. We merge each assigned interval P_i with the associated and adjacent unassigned interval U_i (if any) to obtain

the interval I_i for each agent $i \in [n]$. The intervals $I_1, I_2, \ldots, I_n$ completely partition the cake $[0, 1]$ and constitute the returned allocation $\mathcal{I} = (I_1, I_2, \ldots, I_n)$. In Appendix A.1, we will prove that the two phases run in polynomial time under the Robertson-Webb query model. We now establish the approximation guarantee of the Algorithm.

3.2 Approximation Guarantee

We first note a monotonicity property with respect to the function values, $\widehat{v}_i$s, satisfied during the execution of the algorithm.

▶ **Lemma 6.** *For any agent $i \in [n]$, the function values $\widehat{v}_i$ of the assigned intervals, P_is, are nondecreasing through the execution of Algorithm 1.*

Proof. To establish the monotonicity under $\widehat{v}_i$ in Phase I, consider any iteration for the first while-loop. Here, for the selected agent $a \in [n]$, the value of the assigned interval, under $\widehat{v}_a$, in fact increases and for all the other agents it continues to be the same. Hence, the lemma holds throughout Phase I. The monotonicity is also maintained during the execution of Phase II: the value under $\widehat{v}_i$ does not decrease in Line 8 (Lemma 5) or in Line 11. Therefore, the lemma stands proved. ◀

Next, we assert that, throughout the execution of the algorithm, the assigned intervals satisfy an inclusion-wise minimality property. Note that in the following lemma we evaluate agent i's assigned interval under the function $\widehat{v}_i$ and evaluate the compared interval X under the valuation v_i.

▶ **Lemma 7.** *Let $\mathcal{P}' = (P'_1, \ldots, P'_n)$ be any partial allocation considered during the execution of Algorithm 1. Then, for any two assigned intervals P'_i and $P'_j = [\ell'_j, r'_j]$ along with any strict subset $X = [\ell'_j, x] \subsetneq P'_j$ (i.e., $x < r'_j$), we have $v_i(X) < \widehat{v}_i(P'_i) + \frac{\delta}{n}$.*

Proof. We establish the lemma via an inductive argument. Indeed, the initial partial allocation $(\emptyset, \ldots, \emptyset)$ satisfies the desired property. Now, consider any iteration of the first-while loop, and write $\mathcal{P}'' = (P''_1, \ldots, P''_n)$ to be the partial allocation that gets updated (in this iteration) to $\mathcal{P}' = (P'_1, \ldots, P'_n)$. In particular, let a be the agent selected in Line 4. Note that for all the other agents $i \neq a$, the assigned interval remains unchanged, $P'_i = P''_i$. Also, the induction hypothesis implies that $\mathcal{P}''$ satisfies the lemma. Hence, for all the agents $i, j \neq a$ (whose assigned intervals have not changed), the desired property continues to hold. Furthermore, agent a receives an interval of higher function value, $\widehat{v}_a(P'_a) \geq \widehat{v}_a(P''_a) + \delta/n$. Hence, we have the lemma from agent a against any other agent j.

It remains to show that the lemma holds between P'_i and $P'_a = [\ell'_a, r'_a]$. Assume, towards a contradiction, that there exists a strict subset $X = [\ell'_a, x] \subsetneq P'_a$ such that $v_i(X) \geq \widehat{v}_i(P'_i) + \delta/n$. Since $P'_i = P''_i$ and $\widehat{v}_i(X) \geq v_i(X)$, we obtain $\widehat{v}_i(X) \geq \widehat{v}_i(P''_i) + \delta/n$. This, however, contradicts the selection criterion in Lines 3 and 4. In particular, this bound implies $r_i < r_a$ (see Line 3) and, hence, a would not be the selected agent in Line 4. Therefore, by way of contradiction, we have that the property holds with respect to P'_a as well.

The above-mentioned arguments prove that the lemma holds for all allocations considered in Phase I. Next, we show that it continues to hold through Phase II.

Consider any iteration of the second while-loop, and write $\mathcal{P}'' = (P''_1, \ldots, P''_n)$ to be the partial allocation that gets updated in this iteration. The induction hypothesis gives us that $\mathcal{P}''$ satisfies the desired property. In Line 8 the intervals are reassigned among the agents (i.e., the collection of intervals remains unchanged) and for each agent i, the value, under $\widehat{v}_i$, of the assigned interval does not decrease; see Lemma 5. Hence, the property continues

to hold after Line 8. For analyzing the rest of the iteration, let $s \in [n]$ denote the (source) agent that gets selected in Line 9 and P'_s be the updated interval for agent s; in particular, interval P'_s is obtained by appending a piece to P''_s. Since s is the only agent whose interval got updated here, the lemma continues to hold between all other agents $i, j \neq s$. Also, the property is maintained from agent s's perspective, since $\widehat{v}_s(P'_s) \geq \widehat{v}_s(P''_s)$. To complete the proof we will next show that the property is upheld between P'_i and P'_s, for any $i \in [n]$.

Note that, for agent $i \neq s$, the assigned interval remains unchanged during the current update, $P'_i = P''_i$. Furthermore, the fact that s is a source vertex gives us

$$\widehat{v}_i(P''_i) \geq \widehat{v}_i(P''_s) \geq v_i(P''_s) \tag{2}$$

The extension of P''_s to P'_s (performed in Line 11) ensures that $v_i(P'_s) \leq v_i(P''_s) + \delta/n$. Hence, inequality (2) gives us $v_i(P'_s) \leq \widehat{v}_i(P'_i) + \delta/n$. That is, there does not exist an $X \subsetneq P'_s$ with the property that $v_i(X) \geq \widehat{v}_i(P'_i) + \delta/n$. This completes the proof. ◀

The next lemma provides a bounded envy guarantee for the partial allocation $\overline{\mathcal{P}} = (\overline{P}_1, \ldots, \overline{P}_n)$ computed by Phase I. Note that in this lemma, while considering envy from agent i to agent j, we evaluate $\overline{P}_i$ with respect to $\widehat{v}_i$ and evaluate $\overline{P}_j$ under v_i.

▶ **Lemma 8.** *Let $\overline{\mathcal{P}} = (\overline{P}_1, \ldots, \overline{P}_n)$ be the partial allocation maintained by Algorithm 1 at the end of Phase* I *(i.e., at the termination of the first while-loop). Then, for all agents $i, j \in [n]$ and all unassigned intervals $U \in \mathcal{U}_{\overline{\mathcal{P}}}$, we have*

$$\widehat{v}_i(\overline{P}_i) \geq v_i(\overline{P}_j) - \frac{\delta}{n} \qquad \text{and} \qquad \widehat{v}_i(\overline{P}_i) \geq v_i(U) - \frac{\delta}{n}.$$

Proof. Fix an arbitrary agent $i \in [n]$ and consider any unassigned interval $U \in \mathcal{U}_{\overline{\mathcal{P}}}$. The execution condition of the first while-loop ensures that at termination it holds that $\widehat{v}_i(\overline{P}_i) \geq \widehat{v}_i(U) - \frac{\delta}{n} \geq v_i(U) - \frac{\delta}{n}$; the last inequality directly follows from the definition of $\widehat{v}_i$. This establishes the desired inequalities with respect to the unassigned intervals.

Next, for any assigned interval $\overline{P}_j = [\overline{\ell}_j, \overline{r}_j]$, assume, towards a contradiction, that $v_i(\overline{P}_j) > \widehat{v}_i(\overline{P}_i) + \delta/n$. Since the valuation v_i is divisible (see Section 2),[4] there exists a strict subset $X = [\overline{\ell}_j, x] \subsetneq \overline{P}_j$ with the property that $v_i(X) = \widehat{v}_i(\overline{P}_i) + \delta/n$. This, however, contradicts Lemma 7 (instantiated with $\mathcal{P}' = \overline{\mathcal{P}}$). The lemma stands proved. ◀

Next, we show that the bounded envy guarantee obtained at the end of Phase I (as stated in Lemma 8) continues to hold in Phase II.

▶ **Lemma 9.** *Let $\mathcal{P} = (P_1, \ldots, P_n)$ be the partial allocation maintained by Algorithm 1 at the end of Phase* II*. Then, for all agents $i, j \in [n]$ and all unassigned intervals $U \in \mathcal{U}_{\mathcal{P}}$, we have*

$$\widehat{v}_i(P_i) \geq v_i(P_j) - \frac{\delta}{n} \qquad \text{and} \qquad \widehat{v}_i(P_i) \geq v_i(U) - \frac{\delta}{n}.$$

Proof. Let $\overline{\mathcal{P}} = (\overline{P}_1, \ldots, \overline{P}_n)$ be the partial allocation maintained by Algorithm 1 at the end of Phase I. Note that $\mathcal{U}_{\overline{\mathcal{P}}}$ and $\mathcal{U}_{\mathcal{P}}$ denote the collection of unassigned intervals left at the end of Phase I and Phase II, respectively. We observe that, for all unassigned intervals $U \in \mathcal{U}_{\mathcal{P}}$, there exists an unassigned interval $\overline{U} \in \mathcal{U}_{\overline{\mathcal{P}}}$ such that $U \subseteq \overline{U}$. These containments

[4] Here, we invoke divisibility of v_i with factor $\alpha = \frac{\widehat{v}_i(\overline{P}_i) + \delta/n}{v_i(\overline{P}_j)} \in (0, 1)$. Also, note that, in contrast to v_i, the function $\widehat{v}_i$ is not divisible.

follow from the fact that in each iteration of the second while-loop (i.e., in Phase II) we either reassign the allocated intervals (which preserves the collection of the unassigned ones) or we enlarge a chosen assigned interval (Line 11); under such an enlargement, one of the unassigned intervals gets reduced and the others remain unchanged.

Furthermore, Lemma 6 gives us $\widehat{v}_i(P_i) \geq \widehat{v}_i(\overline{P}_i) \geq v_i(\overline{U}) - \delta/n$, for any interval $\overline{U} \in \mathcal{U}_{\overline{P}}$; here, the last inequality follows from Lemma 8. Using this bound and the above-mentioned containment of unassigned intervals we get $\widehat{v}_i(P_i) \geq v_i(U) - \delta/n$, for all $U \in \mathcal{U}_{\mathcal{P}}$. This establishes the desired inequalities with respect to the unassigned intervals.

Next, for any assigned interval $P_j = [\ell_j, r_j]$, assume, towards a contradiction, that $v_i(P_j) > \widehat{v}_i(P_i) + \delta/n$. Since the valuation v_i is divisible, there exists a strict subset $X = [\ell_j, x] \subsetneq P_j$ with the property that $v_i(X) = \widehat{v}_i(P_i) + \delta/n$. This, however, contradicts Lemma 7 (instantiated with $\mathcal{P}' = \mathcal{P}$). The lemma stands proved. ◀

The following lemma shows that, at the end of Phase I, if an agent i does not receive a bifurcating interval but some other agent j does, then j's interval cannot be bifurcating (for i) with an additional margin of $\frac{\delta}{n}$. Formally,[5]

▶ **Lemma 10.** *Let $\overline{\mathcal{P}} = (\overline{P}_1, \ldots, \overline{P}_n)$ be the partial allocation maintained by Algorithm 1 at the end of Phase* I*. If, for an agent $i \in [n]$, the assigned interval $\overline{P}_i$ is* not *bifurcating (for i), but interval $\overline{P}_j = [\overline{\ell}_j, \overline{r}_j]$ is bifurcating (for i). Then, at least one of the following inequalities holds:*

$$v_i(\overline{P}_j) < \frac{1}{4} + \frac{\delta}{n} \qquad \text{or} \qquad v_i([\overline{r}_j, 1]) > \frac{1}{2} - \frac{\delta}{n}.$$

Proof. Assume, towards a contradiction, that the bifurcating interval $\overline{P}_j = [\overline{\ell}_j, \overline{r}_j]$ has value $v_i(\overline{P}_j) \geq \frac{1}{4} + \frac{\delta}{n}$ and $v_i([\overline{r}_j, 1]) \leq \frac{1}{2} - \frac{\delta}{n}$. These properties in fact imply the existence of a strict subset $X \subsetneq \overline{P}_j$ that is bifurcating for i: write $x \in [\overline{\ell}_j, \overline{r}_j]$ to denote the leftmost point that satisfies $v_i([\overline{\ell}_j, x]) = v_i(\overline{P}_j) - \frac{\delta}{n}$ and set $X = [\overline{\ell}_j, x]$. Since valuation v_i is divisible, such a point x exists and we have $x < \overline{r}_j$. Furthermore, the lower bound on the value of $\overline{P}_j$ gives us $v_i(X) \geq \frac{1}{4}$. In addition, note that $v_i([0, \overline{\ell}_j]) \leq 1/2$, since $\overline{P}_j = [\overline{\ell}_j, \overline{r}_j]$ is bifurcating. Also, using the inequality $v_i([\overline{r}_j, 1]) \leq \frac{1}{2} - \frac{\delta}{n}$ and the additivity of the valuation v_i, we get that $v_i([x, 1]) \leq \frac{1}{2}$. Indeed, these bounds ensure that X is a strict subset of $\overline{P}_j$ and is bifurcating for agent i.

The existence of X contradicts the selection criterion in Lines 3 and 4. In particular, consider the iteration in which $\overline{P}_j$ was assigned and write P'_i to denote the interval assigned to agent i during that iteration. We note that

$$\begin{aligned} \widehat{v}_i(P'_i) &\leq \widehat{v}_i(\overline{P}_i) && \text{(via Lemma 6)} \\ &< \frac{1}{2} && (\overline{P}_i \text{ is non-bifurcating \& Claim 4}) \end{aligned}$$

On the other hand, $\widehat{v}_i(X) = 1$, for the interval X identified above. Hence, j would not be the selected agent in Line 4. This contradiction establishes the lemma. ◀

We next prove that a guarantee, analogous to Lemma 10, holds for Phase II as well.[6]

[5] Note that, in contrast to Lemmas 8 and 9, here we have an absolute bound on the value of the compared interval $\overline{P}_j$.

[6] As in Lemma 10, here we have an absolute bound on the value of the compared interval P_j.

▶ **Lemma 11.** *Let $\mathcal{P} = (P_1, \ldots, P_n)$ be the partial allocation maintained by Algorithm 1 at the end of Phase* II*. If, for an agent $i \in [n]$, the assigned interval P_i is* not *bifurcating (for i), but interval $P_j = [\ell_j, r_j]$ is bifurcating (for i). Then, at least one of the following inequalities holds:*

$$v_i(P_j) < \frac{1}{4} + \frac{\delta}{n} \qquad \text{or} \qquad v_i([r_j, 1]) > \frac{1}{2} - \frac{\delta}{n}.$$

Proof. Write $\overline{\mathcal{P}} = (\overline{P}_1, \ldots, \overline{P}_n)$ to denote the partial allocation at the end of Phase I. Note that, throughout Phase II, the algorithm either reassigns the intervals (Line 8) or appends (unassigned) pieces to them (Line 11). Hence, for the interval P_j, assigned to agent j at the end of Phase II, there exists $\overline{P}_k$, for some $k \in [n]$, such that $P_j \supseteq \overline{P}_k$.

We assume, towards a contradiction, that the bifurcating interval $P_j = [\ell_j, r_j]$ has value $v_i(P_j) \geq \frac{1}{4} + \frac{\delta}{n}$ and $v_i([r_j, 1]) \leq \frac{1}{2} - \frac{\delta}{n}$. It cannot be the case that $P_j = \overline{P}_k$ (for an interval $\overline{P}_k$ assigned at the end of Phase I), since this would contradict Lemma 10. Hence, in the remainder of the proof we address the complementary case wherein P_j was obtained by appending to an interval, say P'_s, in an iteration of the second while-loop (specifically, Line 11). Also, write P'_i to denote the interval assigned to agent i during that iteration. Lemma 6 and the fact that P_i is non-bifurcating for i (Claim 4) give us $\widehat{v}_i(P'_i) \leq \widehat{v}_i(P_i) < 1/2$. Using this inequality and the fact that s was a source agent during the iteration under consideration, we get

$$\widehat{v}_i(P'_s) \leq \widehat{v}_i(P'_i) < 1/2 \tag{3}$$

However, the assumptions on the values of $P_j = [\ell_j, r_j]$ and $[r_j, 1]$ contradict inequality (3): Let z denote the right endpoint of P'_s, i.e., $P'_s = [\ell_j, z]$. Since a piece of bounded value is appended in Line 11, we have $v_i(P'_s) \geq v_i(P_j) - \frac{\delta}{n} \geq \frac{1}{4}$. Furthermore, using the inequality $v_i([r_j, 1]) \leq \frac{1}{2} - \frac{\delta}{n}$, we obtain $v_i([z, 1]) \leq \frac{1}{2}$. In addition, the fact that $P_j = [\ell_j, r_j]$ is a bifurcating interval gives us $v_i([0, \ell_j)) \leq \frac{1}{2}$, i.e., the value to the left of $P'_s = [\ell_j, z]$ is at most $1/2$. These observations imply that P'_s is a bifurcating interval for i; in particular, $\widehat{v}_i(P'_s) = 1$. This bound contradicts inequality (3) and completes the proof. ◀

Using Lemma 9, we next obtain a relevant envy bound for the allocation $\mathcal{I}$ returned by the algorithm.

▶ **Lemma 12.** *The allocation $\mathcal{I} = (I_1, \ldots, I_n)$ computed by Algorithm 1 satisfies $v_i(I_i) \geq \frac{1}{2} v_i(I_j) - \frac{\delta}{n}$, for all agents $i, j \in [n]$.*

Proof. Write $\mathcal{P} = (P_1, \ldots, P_n)$ to denote the partial allocation of the algorithm at the end of Phase II. The execution condition of the second while-loop ensures that $|\mathcal{U}_{\mathcal{P}}| \leq n$. Also, at the end of the algorithm, for each unassigned interval $U \in \mathcal{U}_{\mathcal{P}}$, we select a distinct and adjacent interval P_j and associate U with P_j. In particular, let U_j be the unassigned interval associated with P_j. If P_j is not associated with any unassigned interval, then set $U_j = \emptyset$. Indeed, $I_j = P_j \cup U_j$ is a connected piece of the cake, i.e., an interval.

For any agent $i \in [n]$, if interval I_i is bifurcating, then $v_i(I_i) \geq \frac{1}{4}$. Furthermore, any other assigned interval I_j is either completely to the left of I_i or to the right of I_i. In either case, by the definition of bifurcating intervals, we have $v_i(I_j) \leq \frac{1}{2}$. Hence, for agents i that receive a bifurcating interval I_i, we have the stated inequality, $v_i(I_i) \geq \frac{1}{2} v_i(I_j)$.

It remains to show that the lemma holds for agents i for whom I_i is not bifurcating. For such agents, the interval $P_i \subseteq I_i$ is also non-bifurcating and, hence, $v_i(P_i) = \widehat{v}_i(P_i)$. Therefore,

$$v_i(I_i) \geq v_i(P_i) \geq v_i(P_j) - \frac{\delta}{n} \quad \text{and} \quad v_i(I_i) \geq v_i(P_i) \geq v_i(U_j) - \frac{\delta}{n} \quad \text{(via Lemma 9)}$$

Summing we get

$$2v_i(I_i) \geq v_i(P_j) + v_i(U_j) - \frac{2\delta}{n} = v_i(I_j) - \frac{2\delta}{n} \qquad \text{(since } v_i \text{ is additive)}$$

Hence, we obtain the stated inequality, $v_i(I_i) \geq \frac{1}{2}v_i(I_j) - \frac{\delta}{n}$. This completes the proof. ◀

We now establish the main result of this section.

▶ **Theorem 13.** *Given any cake division instance – with Robertson-Webb query access to the valuations of the n agents – and parameter $\delta \in (0, 1)$, Algorithm 1 computes a connected cake division (i.e., an allocation) $\mathcal{I} = (I_1, \ldots, I_n)$ that is $\left(\frac{1}{4} + \frac{2\delta}{n}\right)$-EF. The algorithm executes in time that is polynomial in n and $\frac{1}{\delta}$.*

Proof. Fix any agent $i \in [n]$. We establish the theorem by considering three complementary and exhaustive cases, based on the interval I_i (assigned to agent i):
Case 1: Interval I_i is bifurcating,
Case 2: Value $v_i(I_i) < \frac{1}{4}$,
Case 3: Interval I_i is not bifurcating and $v_i(I_i) \geq \frac{1}{4}$.

In Case 1, since interval I_i is bifurcating for agent i, we have $v_i(I_i) \geq \frac{1}{4}$ and, for any other interval I_j (either to the left of I_i or to its right), we have $v_i(I_j) \leq \frac{1}{2}$. Therefore, in this case, the stated approximation bound on envy holds, $v_i(I_i) \geq v_i(I_j) - \frac{1}{4}$.

In Case 2, value $v_i(I_i) < \frac{1}{4}$. Note that, Lemma 12 gives us $v_i(I_i) \geq \frac{1}{2}v_i(I_j) - \frac{\delta}{n}$, for any other agent $j \in [n]$. Multiplying both sides of this inequality by 2 and simplifying we obtain

$$v_i(I_i) \geq v_i(I_j) - v_i(I_i) - \frac{2\delta}{n} \geq v_i(I_j) - \frac{1}{4} - \frac{2\delta}{n} \qquad \text{(since } v_i(I_i) < \tfrac{1}{4}\text{)}$$

Therefore, in Case 2 as well, for agent i the envy is additively at most $\left(\frac{1}{4} + \frac{2\delta}{n}\right)$.

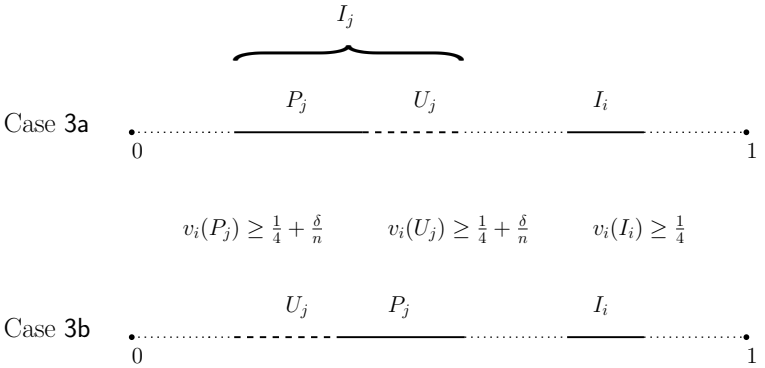


Figure 1 Placement of intervals P_j and U_j in Case 3.

Finally, in Case 3, interval I_i is not bifurcating and $v_i(I_i) \geq \frac{1}{4}$. Write $\mathcal{P} = (P_1, \ldots, P_n)$ to denote the partial allocation at the end of Phase II and recall that $I_k = P_k \cup U_k$, for each agent $k \in [n]$ and the associated unassigned interval $U_k \in \mathcal{U}_{\mathcal{P}}$. For analyzing this case, assume, towards a contradiction, that there exists an interval I_j that violates the stated approximate envy-freeness bound, i.e.,

$$v_i(I_j) > v_i(I_i) + \frac{1}{4} + \frac{2\delta}{n}. \tag{4}$$

Since, in the current case, $v_i(I_i) \geq \frac{1}{4}$, inequality (4) reduces to $v_i(I_j) > \frac{1}{2} + \frac{2\delta}{n}$. We will further show that for interval I_j the composing sub-intervals P_j and U_j are each of value (under v_i) at least $\frac{1}{4} + \frac{\delta}{n}$. Towards this, note that, in the current case, since I_i is not bifurcating for i, neither is $P_i \subseteq I_i$. Furthermore,

$$\begin{aligned} v_i(I_i) &\geq v_i(P_i) && (v_i \text{ is monotonic}) \\ &= \widehat{v}_i(P_i) && (\text{since } P_i \text{ is not bifurcating for } i) \\ &\geq v_i(P_j) - \frac{\delta}{n} && (5) \end{aligned}$$

The last inequality follows from Lemma 9. A similar application of the lemma also gives us

$$v_i(I_i) \geq v_i(U_j) - \frac{\delta}{n} \tag{6}$$

Inequalities (4), (5), and (6) imply that the values of both P_j and U_j are at least $\frac{1}{4} + \frac{\delta}{n}$. Otherwise, say $v_i(P_j) < \frac{1}{4} + \frac{\delta}{n}$. Then,

$$v_i(I_j) = v_i(P_j) + v_i(U_j) < \frac{1}{4} + \frac{\delta}{n} + v_i(U_j) \leq v_i(I_i) + \frac{1}{4} + \frac{2\delta}{n} \qquad (\text{via inequality (6)})$$

Since the last inequality contradicts assumption (4), we have $v_i(P_j) \geq \frac{1}{4} + \frac{\delta}{n}$. Similarly, $v_i(U_j) \geq \frac{1}{4} + \frac{\delta}{n}$.

As mentioned previously, $v_i(I_j) > \frac{1}{2} + \frac{2\delta}{n}$. Hence, the values to the left and to the right of $I_j = [\ell_j, r_j]$ are upper bounded as follows:[7] $v_i([0, \ell_j]) \leq \frac{1}{2} - \frac{2\delta}{n}$ and $v_i([r_j, 1]) \leq \frac{1}{2} - \frac{2\delta}{n}$.

For the subsequent analysis, we also assume that interval I_j is on the left of I_i (see Figure 1); the proof for the other configuration (of I_j being to the right of I_i) follows analogously. Now, there are two sub-cases to consider:

Case 3a: Interval P_j is to the left of U_j (i.e., U_j lies between P_j and I_i).

Case 3b: Interval P_j is to the right of U_j (i.e., P_j lies between U_j and I_i).

In Case 3a, we note that the interval $U_j \in \mathcal{U}_{\mathcal{P}}$ is bifurcating for agent i: As observed above, $v_i(U_j) \geq \frac{1}{4} + \frac{\delta}{n}$ and the value (in the cake) to the right of U_j is equal to $v_i([r_j, 1]) \leq \frac{1}{2} - \frac{2\delta}{n}$. In addition, the value to the left of U_j is at most $1 - (v_i(U_j) + v_i(I_i)) \leq 1 - \frac{1}{4} - \frac{1}{4} - \frac{\delta}{n} = \frac{1}{2} - \frac{\delta}{n}$; interval I_i is to the right to I_j and, hence, to the right of U_j. Hence, $U_j \in \mathcal{U}_{\mathcal{P}}$ is bifurcating for agent i.

Also, the design of Phase II ensures that, for the interval U_j, there exists an unassigned interval $\overline{U} \in \mathcal{U}_{\overline{\mathcal{P}}}$ such that $\overline{U} \supseteq U_j$; here $\overline{\mathcal{P}} = (\overline{P}_1, \ldots, \overline{P}_n)$ denotes the partial allocation at the end of Phase I. Since U_j is bifurcating for i, so is $\overline{U}$. By contrast, in the current case (Case 3), the interval P_i is not bifurcating for i and, hence, neither is $\overline{P}_i$ (Lemma 6). That is, $\widehat{v}_i(\overline{P}_i) < \frac{1}{2} < 1 = \widehat{v}_i(\overline{U})$. The bound, however, contradicts the termination of the first while-loop. Therefore, by way of contradiction, we get that assumption (4) cannot hold in Case 3a. This completes the analysis of this sub-case.

In Case 3b, we note that the interval P_j is bifurcating for agent i, with a margin of $\frac{\delta}{n}$: As observed above, $v_i(P_j) \geq \frac{1}{4} + \frac{\delta}{n}$ and the value to the right of P_j is equal to $v_i([r_j, 1]) \leq \frac{1}{2} - \frac{2\delta}{n}$. In addition, the value to the left of P_j is at most $1 - (v_i(P_j) + v_i(I_i)) \leq 1 - \frac{1}{4} - \frac{1}{4} - \frac{\delta}{n} = \frac{1}{2} - \frac{\delta}{n}$. The existence of such a bifurcating interval P_j contradicts Lemma 11. Hence, even in Case 3b, we must have $v_i(I_j) \leq v_i(I_i) + \frac{1}{4} + \frac{2\delta}{n}$, i.e., the stated bound on envy holds.

This completes the analysis for all the cases, and the theorem stands proved. ◀

[7] Recall that the value of the entire cake is normalized, $v_i([0, 1]) = 1$.

Complementing the additive envy-freeness guarantee obtained in Theorem 13, the next result establishes that, in the computed allocation $\mathcal{I}$, the envy is within a factor of $(2+c)$, where parameter $c \in (0,1)$ is polynomially small (in n).

▶ **Theorem 14.** *Given any cake division instance – with Robertson-Webb query access to the valuations of the n agents – and parameter $c \in (0,1)$, we can compute (in time polynomial in n and $1/c$) a connected cake division (i.e., an allocation) $\mathcal{I} = (I_1, \ldots, I_n)$ that is $\frac{1}{2+c}$-mult-EF.*

Proof. The theorem directly follows from Lemma 12. In particular, we execute Algorithm 1 with parameter $\delta = \frac{c}{8}$, for a sufficiently small $c \in (0,1)$,[8] and note that, for the computed allocation $\mathcal{I}$, Lemma 12 gives us $v_i(I_i) \geq \frac{1}{2} v_i(I_j) - \frac{\delta}{n} = \frac{1}{2} v_i(I_j) - \frac{c}{8n}$, for agents $i, j \in [n]$. Summing over j, we obtain

$$n\ v_i(I_i) \geq \frac{1}{2} \sum_{j=1}^{n} v_i(I_j) - \frac{c}{8} = \frac{1}{2} - \frac{c}{8} \tag{7}$$

The last equality follows from the fact that $I_1, \ldots, I_n$ constitute a complete partition of the cake, with value $v_i([0,1]) = 1$. Since constant $c \leq 1$, inequality (7) reduces to $v_i(I_i) \geq \frac{1}{4n}$, for all agents $i \in [n]$. Therefore, the bound obtained in Lemma 12 can be expressed as

$$v_i(I_i) \geq \frac{1}{2} v_i(I_j) - \frac{c}{8n} \geq \frac{1}{2} v_i(I_j) - \frac{c}{2} v_i(I_i).$$

Simplifying we obtain $(2+c)\, v_i(I_i) \geq v_i(I_j)$, for all agents $i, j \in [n]$. Therefore, the computed allocation is $\frac{1}{2+c}$-mult-EF ◀

4 An ε-EF Algorithm under Bounded Heterogeneity

This section addresses cake division instances in which, for a parameter $\varepsilon \in (0,1)$ and across the n agents, the number of distinct valuations is at most $(\varepsilon n - 1)$. Our algorithm (Algorithm 2) for finding ε-EF allocations in such instances in detailed next.

We first show in Lemma 15 below that the collection of intervals computed by Algorithm 2 cover the entire cake. We will then use this lemma to establish the approximate envy-freeness guarantee in Theorem 16.

▶ **Lemma 15.** *Given any cake division instance in which, across the n agents, the number of distinct valuations is at most $(\varepsilon n - 1)$, Algorithm 2's output $\mathcal{I} = (I_1, \ldots, I_n)$ is a complete allocation, i.e., I_is are pairwise disjoint and $\cup_{i \in [n]} I_i = [0,1]$.*

Proof. By construction, the set of intervals $\mathcal{F}$ populated in Line 4 of Algorithm 2 are pairwise disjoint and cover the entire cake. We will show that the number of intervals in $\mathcal{F}$ is at most n, i.e., $|\mathcal{F}| \leq n$. Since the assigned intervals, I_is, are selected from the set $\mathcal{F}$ (see the for-loop in the algorithm), the cardinality bound implies that no interval in $\mathcal{F}$ remains unassigned. Hence, $\cup_{i \in [n]} I_i = [0,1]$. Also, given that the intervals in $\mathcal{F}$ are pairwise disjoint, so are the I_is. Therefore, $\mathcal{I} = (I_1, \ldots, I_n)$ is a complete allocation.

We complete the proof by establishing that $|\mathcal{F}| \leq n$. Towards this it suffices to show that $|Z| \leq n+1$; see Line 4 and note that $|\mathcal{F}| = |Z| - 1$. In Line 2, for each agent $i \in [n]$, we consider $T+1$ cut points $0 = x_0^i < x_1^i < x_2^i < \ldots < x_{T-1}^i < x_T^i = 1$. The end points of the cake, 0 and 1, are considered for every agent. Moreover, for any two agents, $i, j \in [n]$, with

[8] With this parameter choice, the algorithm executes in time that is polynomial in n and $1/c$.

Algorithm 2 ε-EF under bounded heterogeneity.

Input: A cake division instance with oracle access to the valuations $\{v_i\}_{i=1}^n$ of the n agents along with parameter $\varepsilon \in (0,1)$.
Output: A complete allocation $(I_1, \ldots, I_n)$.

1: Set T to be the smallest integer such that $T\varepsilon \geq 1$, i.e., $T \coloneqq \lceil \frac{1}{\varepsilon} \rceil$.
2: For each agent $i \in [n]$, let $0 = x_0^i < x_1^i < x_2^i < \ldots < x_{T-1}^i < x_T^i = 1$ be the collection of $(T+1)$ cut points that satisfy $v_i([x_{t-1}^i, x_t^i]) = \varepsilon$, for all $1 \leq t \leq T-1$, and $v_i([x_{T-1}^i, x_T^i]) \leq \varepsilon$.
3: Let Z be the union of these cut points $Z \coloneqq \bigcup_{i\in[n]} \{x_0^i, x_1^i, \ldots, x_{T-1}^i, x_T^i\}$
$\{Z$ is not a multiset, i.e., multiple instances of same cut point are not repeated in $Z.\}$
4: Index the points in $Z = \{z_0, z_1, z_2, \ldots, z_r\}$ such that $0 = z_0 < z_1 < z_2 < \ldots < z_r = 1$ and define the collection of intervals $\mathcal{F} \coloneqq \left\{[z_t, z_{t+1}]\right\}_{t=0}^{r-1}$
5: **for** agents $i = 1$ to n **do**
6: If $\mathcal{F} = \emptyset$, then set interval $I_i = \emptyset$. Otherwise, if $\mathcal{F} \neq \emptyset$, then set $I_i = \arg\max_{F\in\mathcal{F}} v_i(F)$ and update $\mathcal{F} \leftarrow \mathcal{F} \setminus \{I_i\}$.
7: **end for**
8: **return** allocation $\mathcal{I} = (I_1, \ldots, I_n)$.

identical valuations, $v_i = v_j$, even the remaining $(T-1)$ points are the same: $x_t^i = x_t^j$ for all $1 \leq t \leq T-1$. Since the number of distinct valuations is at most $(\varepsilon n - 1)$, there are at most $(\varepsilon n - 1)(T-1)$ cut points in Z that are strictly between 0 and 1. Including the endpoints of the cake in the count, we get $|Z| \leq (\varepsilon n - 1)(T-1) + 2 \leq (\varepsilon n - 1)\frac{1}{\varepsilon} + 2$. The last inequality follows from the definition of T; in particular, $T - 1 < \frac{1}{\varepsilon}$. Simplifying we obtain $|Z| \leq n - \frac{1}{\varepsilon} + 2 \leq n + 1$; recall that $\varepsilon \leq 1$. Therefore, $|\mathcal{F}| \leq n$ and the lemma stands proved. ◀

The following theorem establishes that Algorithm 2 finds an allocation $\mathcal{I} = (I_1, \ldots, I_n)$ that satisfies $v_i(I_i) \geq v_i(I_j) - \varepsilon$ for all agents $i, j \in [n]$.

▶ **Theorem 16.** *Given any cake division instance in which, across the n agents, the number of distinct valuations is at most $(\varepsilon n - 1)$, Algorithm 2 (with Robertson-Webb query access to the valuations) computes an ε-EF allocation in polynomial time.*

Proof. The runtime analysis of the algorithm is direct. Also, via Lemma 15, we have that the the returned tuple $\mathcal{I} = (I_1, \ldots, I_n)$ is indeed a complete allocation.

For proving that the algorithm achieves an ε-EF guarantee, consider any agent $i \in [n]$ and interval $I_j = [z_t, z_{t+1}]$, where z_t and z_{t+1} are successive points in the set Z; see Line 4. If $I_j = \emptyset$, then in fact i does not envy j. We will show that $v_i(I_j) \leq \varepsilon$ and, hence, obtain the desired bound: $v_i(I_i) \geq v_i(I_j) - \varepsilon$.

For agent i, write x_s^i to be the largest (rightmost) cut point considered in Line 2 that satisfies $x_s^i \leq z_t$. In particular, $x_{s+1}^i > z_t$. Note that the set Z (see Line 3) contains all the points $x_0^i, x_1^i, \ldots, x_{T-1}^i, x_T^i$; in particular, $x_{s+1}^i \in Z$. In addition, z_t and z_{t+1} are two successive points in Z. Hence, we have $z_{t+1} \leq x_{s+1}^i$ and the interval $I_j = [z_t, z_{t+1}] \subseteq [x_s^i, x_{s+1}^i]$. By construction, $v_i([x_s^i, x_{s+1}^i]) \leq \varepsilon$ and, hence, $v_i(I_j) \leq \varepsilon$. This bound on the valuation of interval I_j implies that the computed allocation is ε-EF. The theorem stands proved. ◀

Note that the algorithm might assign some agents $i \in [n]$ an interval of value of zero; in particular, $I_i = \emptyset$. Imposing the requirement that each agent $i \in [n]$ receives an interval of nonzero value (to i) renders the problem as hard as finding an ε-EF allocation in general cake

division instances (without bounded heterogeneity). To see this, consider any cake division instance (which might not satisfy the bounded heterogeneity condition). Append to the cake another unit length interval and include $\lceil \frac{n+2}{\varepsilon} \rceil$ dummy agents that have identical valuation confined to the appended interval. This new instance satisfies bounded heterogeneity. Now, if each agent in the constructed instance receives an interval of nonzero value, then the appended interval must have been divided among the dummy agents and the underlying cake $[0, 1]$ among the original agents. This way we obtain an ε-EF allocation for the original instance. Furthermore, note that an α-mult-EF guarantee, for any $\alpha > 0$, implies that each agent receives an interval of nonzero value. Therefore, achieving multiplicative approximation bounds for envy under bounded heterogeneity is as hard as the general case.

▶ Remark 17. The discretization method used in Algorithm 2 has been utilized in prior works as well; see [10] and [19]. The relevant insight obtained here is the difference between additive and multiplicative approximations: While one can efficiently achieve an ε-additive approximation under bounded heterogeneity, establishing any multiplicative guarantee is as hard as solving the problem in complete generality.

5 Conclusion and Future Work

Algorithmically, connected envy-free cake division is a challenging and equally intriguing problem at the core of fair division. The proof of existence of envy-free cake divisions (with connected pieces) does not lend itself to efficient (approximation) algorithms and, at the same time, negative results – that rule out, say, a polynomial-time approximation scheme (PTAS) – are not known either. In this landscape, the current work improves upon the previously best-known approximation guarantee for connected envy-free cake division. We develop a computationally efficient algorithm that finds a connected cake division that is simultaneously $(1/4 + o(1))$-EF and $(1/2 - o(1))$-mult-EF. We also show that specifically for instances with bounded heterogeneity, an ε-EF division can be computed in time polynomial in n and $1/\varepsilon$.

In addition to the patent problem of efficiently finding ε-EF connected cake divisions, developing ε-EF algorithms for special valuation classes (such as single-block and single-peaked valuations) is a relevant direction of future work. Inapproximability results – similar to the ones recently obtained for ε-consensus halving [13] – are also interesting.

References

1 Reza Alijani, Majid Farhadi, Mohammad Ghodsi, Masoud Seddighin, and Ahmad Tajik. Envy-free mechanisms with minimum number of cuts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31(1), 2017.

2 Georgios Amanatidis, Georgios Birmpas, Aris Filos-Ratsikas, and Alexandros A Voudouris. Fair division of indivisible goods: A survey. *arXiv preprint*, 2022. `arXiv:2202.07551`.

3 Eshwar Ram Arunachaleswaran, Siddharth Barman, Rachitesh Kumar, and Nidhi Rathi. Fair and efficient cake division with connected pieces. In *International Conference on Web and Internet Economics*, pages 57–70. Springer, 2019.

4 Haris Aziz and Simon Mackenzie. A discrete and bounded envy-free cake cutting protocol for any number of agents. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 416–427. IEEE, 2016.

5 Dinesh Kumar Baghel, Vadim E Levit, and Erel Segal-Halevi. Fair division algorithms for electricity distribution. *arXiv preprint*, 2022. `arXiv:2205.14531`.

6 Siddharth Barman and Pooja Kulkarni. Approximation algorithms for envy-free cake division with connected pieces, 2022. `arXiv:2208.08670`.

7 Siddharth Barman and Nidhi Rathi. Fair cake division under monotone likelihood ratios. *Mathematics of Operations Research*, 2021.

8 Steven J Brams and Alan D Taylor. An envy-free cake division protocol. *The American Mathematical Monthly*, 102(1):9–18, 1995.

9 Steven J Brams and Alan D Taylor. *Fair Division: From cake-cutting to dispute resolution*. Cambridge University Press, 1996.

10 Simina Brânzei and Noam Nisan. The query complexity of cake cutting. In *Advances in Neural Information Processing Systems*, 2022.

11 Constantinos Daskalakis, Aranyak Mehta, and Christos Papadimitriou. Progress in approximate nash equilibria. In *Proceedings of the 8th ACM Conference on Electronic Commerce*, pages 355–358, 2007.

12 Xiaotie Deng, Qi Qi, and Amin Saberi. Algorithmic solutions for envy-free cake cutting. *Operations Research*, 60(6):1461–1476, 2012.

13 Aris Filos-Ratsikas, Alexandros Hollender, Katerina Sotiraki, and Manolis Zampetakis. Consensus-halving: Does it ever get easier? In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 381–399, 2020.

14 Duncan K Foley. Resource allocation and the public sector, 1967.

15 Paul Goldberg, Alexandros Hollender, and Warut Suksompong. Contiguous cake cutting: Hardness results and approximation algorithms. *Journal of Artificial Intelligence Research*, 69:109–141, 2020.

16 Hadi Hosseini, Ayumi Igarashi, and Andrew Searns. Fair division of time: Multi-layered cake cutting. In *29th International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 182–188. International Joint Conferences on Artificial Intelligence, 2020.

17 Duško Jojić, Gaiane Panina, and Rade Živaljević. Splitting necklaces, with constraints. *SIAM Journal on Discrete Mathematics*, 35(2):1268–1286, 2021.

18 Spyros C Kontogiannis, Panagiota N Panagopoulou, and Paul G Spirakis. Polynomial algorithms for approximating nash equilibria of bimatrix games. In *International Workshop on Internet and Network Economics*, pages 286–296. Springer, 2006.

19 Richard J Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM conference on Electronic commerce*, pages 125–131. ACM, 2004.

20 Gaiane Panina and Rade Živaljević. Envy-free division via configuration spaces. *arXiv preprint*, 2021. `arXiv:2102.06886`.

21 Ariel D Procaccia. Cake cutting algorithms. In *Handbook of Computational Social Choice, chapter 13*. Citeseer, 2015.

22 Jack Robertson and William Webb. *Cake-cutting algorithms: Be fair if you can*. AK Peters/CRC Press, 1998.

23 FW Simmons. Private communication to Michael Starbird, 1980.

24 Hugo Steinhaus. The Problem of Fair Division. *Econometrica*, 16:101–104, 1948.

25 Walter Stromquist. How to cut a cake fairly. *The American Mathematical Monthly*, 87(8):640–644, 1980.

26 Walter Stromquist. Envy-free cake divisions cannot be found by finite protocols. *the electronic journal of combinatorics*, 15(1):11, 2008.

27 Francis Edward Su. Rental harmony: Sperner's lemma in fair division. *The American mathematical monthly*, 106(10):930–942, 1999.

28 Haralampos Tsaknakis and Paul G Spirakis. An optimization approach for approximate nash equilibria. In *International Workshop on Web and Internet Economics*, pages 42–56. Springer, 2007.

29 Chenhao Wang and Xiaoying Wu. Cake cutting with single-peaked valuations. In *Combinatorial Optimization and Applications: 13th International Conference, COCOA 2019, Xiamen, China, December 13–15, 2019, Proceedings 13*, pages 507–516. Springer, 2019.

A Missing Proofs from Section 3

Here, we restate and prove Lemma 5.

▶ **Lemma 5.** *Given any partial allocation $\mathcal{P} = (P_1, \ldots, P_n)$, one can reassign the intervals P_is among the agents and efficiently find another partial allocation $\mathcal{Q} = (Q_1, \ldots, Q_n)$ with the properties that*

(i) *The envy-graph $G_{\mathcal{Q}}$ is acyclic.*

(ii) *The value $\widehat{v}_i(Q_i) \geq \widehat{v}_i(P_i)$, for all agents $i \in [n]$.*

Proof. If, for given partial allocation $\mathcal{P} = (P_1, \ldots, P_n)$, the envy-graph $G_{\mathcal{P}}$ is already acyclic, then we directly obtain the lemma by setting $\mathcal{Q} = \mathcal{P}$. Hence, in the remainder of the proof we consider the case wherein $G_{\mathcal{P}}$ is cyclic.

Write $C = i_1 \to i_2 \to \ldots \to i_k \to i_1$ to denote a cycle in $G_{\mathcal{P}}$. To obtain a new partial allocation $\mathcal{P}' = (P'_1, \ldots, P'_n)$, we reassign the intervals as follows: for all agents j not in the cycle (i.e., $j \notin \{i_1, i_2, \ldots, i_k\}$), set $P'_j = P_j$. Furthermore, for all the agents i_t in the cycle C, with $1 \leq t < k$, we set $P'_{i_t} = P_{i_{t+1}}$ and $P'_{i_k} = P_{i_1}$. That is, each agent in the cycle receives the interval assigned to its successor in the cycle. This reassignment ensures that, for all agents $i \in [n]$, we have $\widehat{v}_i(P'_i) \geq \widehat{v}_i(P_i)$; recall that a directed edge (i, j) is included in the graph $G_{\mathcal{P}}$ iff $\widehat{v}_i(P_i) < \widehat{v}_i(P_j)$.

We will now show that the number of edges in the envy-graph $G_{\mathcal{P}'}$ is strictly smaller than the number of edges in $G_{\mathcal{P}}$. Hence, repeated elimination of cycles leads to an allocation $\mathcal{Q}$ that satisfies the lemma. Note that the collection of intervals assigned in the allocation $\mathcal{P}'$ is the same as the collection of intervals in $\mathcal{P}$. Also, the out-degree of any vertex i in $G_{\mathcal{P}}$ (or in $G_{\mathcal{P}'}$) is equal to the number of bundles P_js (or P'_js) have value (under $\widehat{v}_i$) strictly greater than i's value (again, under $\widehat{v}_i$) for her own bundle. These observations imply that for all agents not in the cycle C, the out-degree is the same in $G_{\mathcal{P}}$ and $G_{\mathcal{P}'}$. Moreover, for all agents i_t in the cycle C, we have $\widehat{v}_{i_t}(P'_{i_t}) > \widehat{v}_{i_t}(P_{i_t})$. Hence, the out-degree of any such agent i_t in $G_{\mathcal{P}}$ is strictly smaller than its out-degree in $G_{\mathcal{P}}$. Therefore, the number of edges in $G_{\mathcal{P}'}$ is strictly smaller than the ones in $G_{\mathcal{P}}$. This strict reduction in the number of edges implies that after a polynomial number of cycle eliminations we obtain an allocation $\mathcal{Q}$ for which $G_{\mathcal{Q}}$ is acyclic and we have $\widehat{v}_i(Q_i) \geq \widehat{v}_i(P_i)$, for all agents $i \in [n]$. The lemma stands proved. ◀

A.1 Runtime Analysis of Algorithm 1

We begin by noting that, given Robertson-Webb query access to the underlying valuation v_is, we can answer cut and evaluation queries for the functions $\widehat{v}_i$ (see equation (1)) in polynomial time. That is, given points $0 \leq x \leq y \leq 1$, we can find $\widehat{v}_i([x, y])$ in polynomial time. Also, given a point $x \in [0, 1]$ and value $\nu \in [0, 1]$, we can efficiently compute the leftmost point y (if one exists) that satisfies $\widehat{v}_i([x, y]) \geq \nu$.

▷ Claim 18. For any agent $i \in [n]$, given Robertson-Webb query access to the valuation v_i, we can answer cut and evaluation queries with respect to the function $\widehat{v}_i$ in polynomial time.

Proof. We first address the evaluation query for $\widehat{v}_i$. Given interval $[x, y] \subseteq [0, 1]$, we use Eval_i to obtain the following three values: $v_i([x, y])$, $v_i([0, x])$, and $v_i([y, 1])$. These three values tell us whether $[x, y]$ is a bifurcating interval for agent i; see Definition 3. If $[x, y]$ is a bifurcating interval, then we have $\widehat{v}_i([x, y]) = 1$. Otherwise, $\widehat{v}_i([x, y]) = v_i([x, y])$.

Now, we consider the cut query for $\widehat{v}_i$. Given a point $x \in [0, 1]$, and a value $\nu \in [0, 1]$, we identify two candidate points y_1 and y_2 and set $y \coloneqq \min\{y_1, y_2\}$ as the leftmost point that satisfies $\widehat{v}_i([x, y]) \geq \nu$. The first candidate point is defined as $y_1 \coloneqq \mathsf{Cut}_i(x, \nu)$, i.e., y_1

is the leftmost point that satisfies $v_i([x, y_1]) = \nu$. The definition of $\widehat{v}_i$ (see equation (1)) implies that $\widehat{v}_i([x, y_1]) \geq v_i([x, y_1)] = \nu$. Still, there could be a point y_2 to the left of y_1 such that the interval $[x, y_2]$ is bifurcating for i and, hence, $\widehat{v}_i([x, y_2]) \geq \nu$. Therefore, the second candidate y_2 is computed by finding the smallest bifurcating interval, if one exists, starting at x. Towards this, we first use the query $\mathsf{Eval}_i(0, x)$ to ensure that $v_i([0, x]) \leq \frac{1}{2}$. If the interval $[0, x]$ is of value more than $1/2$, then we set $y_2 = 1$. In case $v_i([0, x]) \leq \frac{1}{2}$, we set $y_2 \coloneqq \max\{\mathsf{Cut}_i(x, 0.25), \mathsf{Cut}_i(0, 0.5)\}$. Finally, we return the minimum of y_1 and y_2 as the answer y to the cut query for $\widehat{v}_i$.

Overall, we get that both the cut and the evaluation queries for $\widehat{v}_i$ can be answered in polynomial time. This completes the proof. ◁

We now prove that the algorithm executes in polynomial time.

▶ **Lemma 19.** *Given a fixed constant $\delta \in \left(0, \frac{1}{4}\right)$ and any cake division instance with (Robertson-Webb) query access to the valuations of the n agents, Algorithm 1 computes an allocation in time that is polynomial in n and $\frac{1}{\delta}$.*

Proof. We will first establish the time complexity of Phase I of the algorithm. Note that, in every iteration of this phase (i.e., in every iteration of the while-loop between Lines 2 and 6), for some agent $a \in [n]$, the value $\widehat{v}_a(P_a)$ increases additively by at least $\frac{\delta}{n}$; see Lines 3 and 4. Since the functions $\widehat{v}_i$s are monotonic and upper bounded by 1, the first while-loop in the algorithm iterates at most $\frac{n^2}{\delta}$ times. We next show that each iteration of this while-loop can be implemented in polynomial time and, hence, obtain that overall Phase I executes in polynomial time. Note that the execution condition of the while-loop (Line 2) can be evaluated efficiently, since the evaluation query under $\widehat{v}_i$s can be answered in polynomial time (Claim 18). Similarly, the candidate set C in Line 3 can be computed efficiently. Finding the points r_is in Line 3 entails answering cut queries for the functions $\widehat{v}_i$s and this too can be implemented efficiently (Claim 18). Therefore, all the steps in the while-loop can be implemented efficiently, and we get that Phase I terminates in polynomial time.

For Phase II and each maintained partial allocation $\mathcal{P} = (P_1, \ldots, P_n)$, consider the potential $\varphi(\mathcal{P}) \coloneqq \sum_{i=1}^{n} \sum_{j=1}^{n} v_i(P_j)$. In each iteration of the second while-loop (Lines 7 to 12), the assigned region of the cake (i.e., $\cup_{i \in [n]} P_i$) monotonically increases. Indeed, while updating a partial allocation, the intervals might get reassigned among the agents, however, the union $\cup_{i \in [n]} P_i$ increases in each iteration of the second while-loop. Furthermore, in every iteration, for at least one agent i and the selected interval P_s (see Line 11), the value increases by $\frac{\delta}{n}$.[9] Hence, in each iteration, the potential φ increases by at least $\frac{\delta}{n}$. Also, note that the potential is upper bounded by n and, hence, the second while-loop iterates at most $\frac{n^2}{\delta}$ times. Since all the steps in each iteration of the loop can be implemented in polynomial time – including the envy cycle elimination one (Lemma 5) – we get that Phase II itself executes in polynomial time.

The final merging of the intervals takes linear time. This, overall, establishes the polynomial-time complexity of the algorithm. ◀

[9] If the unassigned interval $\widetilde{U}$, considered in Line 11, is of value less than $\frac{\delta}{n}$ for all agents $i \in [n]$, then after that update the number of unassigned intervals (i.e., $|\mathcal{U}_{\mathcal{P}}|$) strictly decreases. Hence, after such an update, the while-loop terminates.

Cumulative Memory Lower Bounds for Randomized and Quantum Computation

Paul Beame
Computer Science & Engineering, University of Washington, Seattle, WA, USA

Niels Kornerup
Computer Science, University of Texas, Austin, TX, USA

Abstract

Cumulative memory – the sum of space used per step over the duration of a computation – is a fine-grained measure of time-space complexity that was introduced to analyze cryptographic applications like password hashing. It is a more accurate cost measure for algorithms that have infrequent spikes in memory usage and are run in environments such as cloud computing that allow dynamic allocation and de-allocation of resources during execution, or when many multiple instances of an algorithm are interleaved in parallel.

We prove the first lower bounds on cumulative memory complexity for both sequential classical computation and quantum circuits. Moreover, we develop general paradigms for bounding cumulative memory complexity inspired by the standard paradigms for proving time-space tradeoff lower bounds that can only lower bound the maximum space used during an execution. The resulting lower bounds on cumulative memory that we obtain are just as strong as the best time-space tradeoff lower bounds, which are very often known to be tight.

Although previous results for pebbling and random oracle models have yielded time-space tradeoff lower bounds larger than the cumulative memory complexity, our results show that in general computational models such separations cannot follow from known lower bound techniques and are not true for many functions.

Among many possible applications of our general methods, we show that any classical sorting algorithm with success probability at least $1/\mathrm{poly}(n)$ requires cumulative memory $\tilde{\Omega}(n^2)$, any classical matrix multiplication algorithm requires cumulative memory $\Omega(n^6/T)$, any quantum sorting circuit requires cumulative memory $\Omega(n^3/T)$, and any quantum circuit that finds k disjoint collisions in a random function requires cumulative memory $\Omega(k^3n/T^2)$.

2012 ACM Subject Classification Theory of computation → Oracles and decision trees; Theory of computation → Quantum query complexity; Theory of computation → Quantum complexity theory

Keywords and phrases Cumulative memory complexity, time-space tradeoffs, branching programs, quantum lower bounds

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.17

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2301.05680` [18]

Funding *Paul Beame*: Research supported by NSF grant CCF-2006359.

Acknowledgements Many thanks to David Soloveichik for his guidance and contributions to our initial results.

1 Introduction

For some problems, algorithms can use additional memory for faster running times or additional time to reduce memory requirements. While there are different kinds of tradeoffs between time and space, the most common complexity metric for such algorithms is the maximum time-space (TS) product. This is appropriate when a machine must allocate an

algorithm's maximum space throughout its computation. However, recent technologies like AWS Lambda [15] suggest that in the context of cloud computing, space can be allocated to a program only as it is needed. When using such services, analyzing the average memory used per step leads to a more accurate picture than measuring the maximum space.

Cumulative memory (CM), the sum over time of the space used per step of an algorithm, is an alternative notion of time-space complexity that is more fair to algorithms with rare spikes in memory. Cumulative memory complexity was introduced by Alwen and Serbinenko [12] who devised it as a way to analyze time-space tradeoffs for "memory hard functions" like password hashes. Since then, lower and upper bounds on the CM of problems in structured computational models using the black pebble game have been extensively studied, beginning with the work of [12, 7, 32, 10, 9, 8]. Structured models via pebble games are natural in the context of the random oracle assumptions that are common in cryptography. By carefully interweaving their memory-intensive steps, authors of these papers devise algorithms for cracking passwords that compute many hashes in parallel using only slightly more space than is necessary to compute a single hash. While such algorithms can use parallelism to amortize costs and circumvent proven single instance TS complexity lower bounds, their cumulative memory only scales linearly with the number of computed hashes. Strong CM results have also been shown for the black-white pebble game and used to derive related bounds for resolution proof systems [11].

The ideas used for these structured models yield provable separations between CM and TS complexity in pebbling and random oracle models. The key question that we consider is whether or not the same applies to general models of computation without cryptographic or black-box assumptions: Are existing time-space tradeoff lower bounds too pessimistic for a world where cumulative memory is more representative of a computation's cost?

Our Results

The main answer we provide to this question is negative for both classical and quantum computation: We give *generic methods* that convert existing paradigms for obtaining time-space tradeoff lower bounds involving worst-case space to new lower bounds that replace the time-space product by cumulative space, immediately yielding a host of new lower bounds on cumulative memory complexity. With these methods, we show how to extend virtually all known proofs for time-space tradeoffs to equivalent lower bounds on cumulative memory complexity, implying that there cannot be cumulative memory savings for these problems. Our results, like those of existing time-space tradeoffs, apply in models in which arbitrary sequential computations may be performed between queries to a read-only input. Our lower bounds also apply to randomized and quantum algorithms that are allowed to make errors.

Classical computation. We focus on lower bound paradigms that apply to computations of multi-output functions $f : D^n \to R^m$. Borodin and Cook [22] introduced a method for proving time-space tradeoff lower bounds for such functions that takes a property such as the following: for some $K = K(R, n)$, constant γ, and distribution μ on D^n:

(*) For any partial assignment τ of $k \leq \gamma m$ output values over R and any restriction (i.e., partial assignment) π of $h = h(k, n)$ coordinates on D^n,

$$\Pr_{x \sim \mu}[f(x) \text{ is consistent with } \tau \mid x \text{ is consistent with } \pi] \leq K^{-k}.$$

and derives a lower bound of the following form:

Table 1 All CM bounds match the TS lower bound when considering RAM computation or quantum circuits. The symbol * indicates that the result requires additional assumptions.

Problem	TS Lower Bound	Source	Matching CM Bound
Ranking, Sorting	$\Omega(n^2/\log n)$	[22]	Corollary 4.4
Unique Elements, Sorting	$\Omega(n^2)$	[16]	Corollary 4.14
Matrix-Vector Product ($\mathbb{F}$)	$\Omega(n^2 \log \lvert\mathbb{F}\rvert)$	[4]	Corollary 4.5
Matrix-Multiplication ($\mathbb{F}$)	$\Omega((n^6 \log \lvert\mathbb{F}\rvert)/T)$	[4]	Corollary 4.15
Hamming Closeness	$\Omega(n^{2-o(1)})$	[19]*	Full paper [18]*
Element Distinctness	$\Omega(n^{2-o(1)})$	[19]*	Full paper [18]*
Quantum Sorting	$\Omega(n^3/T)$	[29]	Theorem 3.6
Quantum k disjoint collisions	$\Omega(k^3 n/T^2)$	[27]	Corollary 4.16
Quantum Boolean Matrix-Mult	$\Omega(n^5/T)$	[29]	Full paper [18]*

▶ **Proposition 1.1** ([22]). *Assume that Property (*) holds for $f : D^n \to R^m$ with $\gamma > 0$ constant. Then, $T\ (S + \log_2 T)$ is $\Omega(m\ h(S/\log_2 K, n) \log K)$.*

In particular, since $S \geq \log_2 n$ is essentially always required, if we have the typical case that $h(k, n) = k^{\Delta}\ h_1(n)$ for some function $h_1(n)$ then this says that $T \cdot S^{1-\Delta}$ is $\Omega(m\ h_1(n)\ \log^{1-\Delta} K)$ or, equivalently, that $\max(S, \log n)$ is $\Omega([(m\ h_1(n)/T]^{1/(1-\Delta)} \log K)$. As a simplified example of our new general paradigm, we prove the following analog for cumulative complexity:

▶ **Theorem 1.2.** *Suppose that Property (*) holds for $f : D^n \to R^m$ with $h(k, n) = k^{\Delta} h_1(n)$ and $\gamma > 0$ constant. If $T \log_2 T$ is $o(m\ h_1(n) \log K)$ then any algorithm computing f requires cumulative memory $\Omega\left(\left[(m\ h_1(n))^{1/(1-\Delta)} \log K\right]/T^{\Delta/(1-\Delta)}\right)$.*

We note that this bound corresponds exactly to the bound on the product of time and space from Borodin-Cook method. The full version of our general theorem for randomized computation (Theorem 4.8) is inspired by an extension by Abrahamson [4] of the Borodin-Cook paradigm to average case complexity.

Our full paper ([18]) also shows how the paradigms for the best time-space tradeoff lower bounds for single-output Boolean functions, which are based on the densities of *embedded rectangles* where these functions are constant, can be extended to yield cumulative memory bounds.

Quantum computation. We develop an extension of our general approach that applies to quantum computation as well. In this case Property (*) and its extensions that we use for our more general theorem must be replaced by statements about quantum circuits with a small number of queries. In this case, we first generalize the quantum time-space tradeoff for sorting proven in [29], which requires that the time order in which output values are produced must correspond to the sorted order, to a matching cumulative memory complexity bound of $\Omega(n^3/T)$ that works for any fixed time-ordering of output production, yielding a more general lower bound. (For example, an algorithm may be able to determine the median output long before it determines the other outputs.) We then show how an analog of our classical general theorem can be applied to extend to paradigms for quantum time-space tradeoffs to cumulative memory complexity bounds for other problems.

A summary of our results for both classical and quantum complexity is given in Table 1.

Previous work

Memory hard functions and cumulative memory complexity. Alwen and Serbinenko [12] introduced parallel cumulative (memory) complexity as a metric for analyzing the space footprint required to compute *memory hard functions (MHFs)*, which are functions designed to require large space to compute. Most MHFs are constructed using hashgraphs [26] of DAGs whose output is a fixed length string and their proofs of security are based on pebbling arguments on these DAGs while assuming access to truly random hash functions for their complexity bounds [12, 21, 32, 8, 10, 20]. (Also see our full paper [18] for their use in separating CM and TS complexity.) Recent constructions do not require random hash functions; however, they still rely on cryptographic assumptions [25, 14].

Classical time-space tradeoffs. While these were originally studied in restricted pebbling models similar to those considered to date for cumulative memory complexity [35, 23], the gold-standard model for time-space tradeoff analysis is that of unrestricted branching programs, which simultaneously capture time and space for general sequential computation. Following the methodology of Borodin and Cook [22], who proved lower bounds for sorting, many other problems have been analyzed (e.g., [37, 2, 3, 16, 30]), including universal hashing and many problems in linear algebra [4]. (See [34, Chapter 10] for an overview.) A separate methodology for single-output functions, introduced in the context of restricted branching programs [24, 31], was extended to general branching programs in [17], with further applications to other problems [5] including multi-precision integer multiplication [33] and error-correcting codes [28] as well as over Boolean input domains [6, 19]. Both of these methods involve breaking the program into blocks to analyze the computation under natural distributions over the inputs based on what happens at the boundaries between blocks.

Quantum time-space tradeoffs. Similar blocking strategies can be applied to quantum circuits to achieve time-space trade-offs for multi-output functions. In [29] the authors use direct product theorems to prove time-space tradeoffs for sorting and Boolean matrix multiplication. They also proved somewhat weaker lower bounds for computing matrix-vector products for fixed matrices A; those bounds were extended in [13] to systems of linear inequalities. However, both of these latter results apply to computations where the fixed matrix A defining the problem depends on the space bound and, unlike the case of sorting or Boolean matrix multiplication, do not yield a fixed problem for which the lower bound applies at all space bounds. More recently [27] extended the recording query technique of Zhandry in [38] to obtain time-space lower bounds for the k-collision problem and match the aforementioned result for sorting.

Our methods

At the highest level, we employ part of the same paradigms previously used for time-space tradeoff lower bounds. Namely breaking up the computations into blocks of time and analyzing properties of the branching programs or quantum circuits based on what happens at the boundaries between time blocks. However, for cumulative memory complexity, those boundaries cannot be at fixed locations in time and their selection needs to depend on the space used in these time steps.

Further, in many cases, the time-space tradeoff lower bound needs to set the lengths of those time blocks in a way that depends on the specific space bound. When extending the ideas to bound cumulative memory usage, there is no single space bound that can be used

throughout the computation; this sets up a tricky interplay between the choices of boundaries between time blocks and the lengths of the time blocks. Because the space usage within a block may grow and shrink radically, even with optimal selection of block boundaries, the contribution of each time block to the overall cumulative memory may be significantly lower than the time-space product lower bound one would obtain for the individual block.

We show how to bound any loss in going from time-space tradeoff lower bounds to cumulative memory lower bounds in a way that depends solely on the bound on the lengths of blocks as a function h_0 of the target space bound (cf. Lemma 4.7). For many classes of bounding functions we are able to bound the loss by a constant factor, and we are able show that it is always at most an $O(\log n)$ factor loss. If this bounding function h_0 is non-constant, we also need to bound the optimum way for the algorithm to allocate its space budget for producing the require outputs throughout its computation. This optimization again depends on the bounding function h_0. This involves minimizing a convex function based on h_0 subject to a mix of convex and concave constraints, which is not generally tractable. However, assuming that h_0 is nicely behaved, we are able to apply specialized convexity arguments (cf. Lemma 4.10) which let us derive strong lower bounds on cumulative memory complexity.

Road map. We give the overall definitions in Section 2, including a review of the standard definitions of the work space used by quantum circuits. In Section 3, we give our lower bound for quantum sorting algorithms which gives a taste of the issues involved for our general theorems. In Section 4, we give the general theorems that let us convert the Borodin-Cook-Abrahamson paradigm for multi-output functions to cumulative memory lower bounds for classical randomized algorithms; that section also contains the corresponding theorems for quantum lower bounds and statements of some sample applications for our general results. Appendix A contains the arguments that bound the optimum allocations of cumulative space budgets to time steps. Our full paper [18] contains more details, a conditional separation between CM and TS complexity, detailed applications of the general theorems we present here, and our bounds for single-output functions.

2 Preliminaries

Cumulative memory is an abstract notion of time-space complexity that can be applied to any model of computation with a natural notion of space. Here we will use branching programs and quantum circuits as concrete models, although our results generalize to any reasonable model of computation.

Branching Programs. A branching programs with input $\{x_1, \ldots, x_n\} \in D^n$ is defined using a rooted DAG in which each non-sink vertex is labeled with an $i \in [n]$ and has $|D|$ outgoing edges that correspond to possible values of x_i. Each edge is optionally labeled by some number of output statements expressed as pairs (j, o_j) where $j \in [m]$ is an output index and $o_j \in R$ (if outputs are to be ordered) or simply $o_j \in R$ (if outputs are to be unordered). Evaluation starts at the root v_0 and follows the appropriate labels of the respective x_i. We consider branching programs P that contain $T + 1$ layers where the outgoing edges from nodes in each layer t are all in layer $t + 1$. We impose no restriction on the query pattern of the branching program or when it can produce parts of the output. The *time* of the branching program is $T(P) = T$. The *space* of the branching program is $S(P) = \max_t \log_2 |L_t|$ where L_t is the set of nodes in layer t. Observe that in the absence of any limit on its space, a branching program could be a decision tree; hence the minimum time for branching programs

to compute a function f is its *decision tree complexity*. The *time-space* (product) used by the branching program is $TS(P) = T(P)S(P)$. The *cumulative memory* used by the branching program is $CM(P) = \sum_t \log_2 |L_t|$.

Branching programs are very general, and simultaneously model time and space for sequential computation. In particular they model time and space for random-access off-line multitape Turing machines and random-access machines (RAMs) when time is unit-cost, space is log-cost, and the input and output are read-only and write-only respectively. Branching programs are much more flexible than these models since they can make arbitrary changes to their storage in a single step.

Quantum Circuits. We also consider quantum circuits $\mathcal{C}$ classical read-only input $X = x_1, \ldots, x_n$ that can be queried using an XOR query oracle. As is normal in circuit models, each output wire is associated with a fixed position in the output sequence, independent of the input. As shown in Figure 1 following [29], we abstract an arbitrary quantum circuit $\mathcal{C}$ into layers $\mathcal{C} = \{L_1, \ldots, L_T\}$ where layer L_t starts with the t-th query Q to the input and ends with the start of the next layer. During each layer, an arbitrary unitary transformation V gets applied which can express an arbitrary sub-circuit involving input-independent computation. The sub-circuit/transformation V outputs S_t qubits for use in the next layer in addition to some qubits that are immediately measured in the standard basis, some of which are treated as classical write-only output. The time of $\mathcal{C}$ is lower bounded by the number of layers T and we say that the space of layer L_t is S_t. Observe that to compute a function f, T must be at least the *quantum query complexity* of f since that measure corresponds the above circuit model when the space is unbounded. Note that the cumulative memory of a circuit is lower-bounded by the sum of the S_t. For convenience we define S_0, the space of the circuit before its first query, to be zero. Thus we only consider the space after the input is queried.

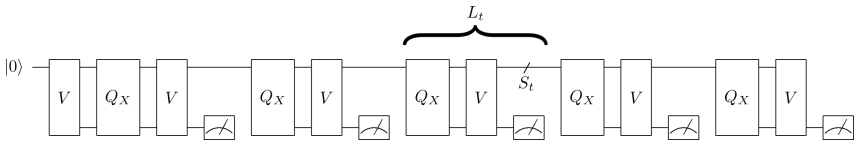


Figure 1 The abstraction of a quantum circuit into layers.

3 Quantum cumulative memory complexity of sorting

As an illustrative example, we first show that the quantum cumulative memory complexity of sorting is $\Omega(n^3/T)$, matching the TS complexity bounds given in [29, 27]. This involves the quantum circuit model which, as we have noted, produces each output position at a predetermined input-independent layer. We restrict our attention to circuits that output all elements in the input in some fixed rank order. While our proof is inspired by the time-space lower bound of [29], it can be easily adapted to follow the proof in [27] instead. We start by constructing a probabilistic reduction from the k-threshold problem to sorting.

▶ **Definition 3.1.** *In the* k-threshold problem *we receive an input* $X = x_1, \ldots, x_n$ *where* $x_i \in \{0, 1\}$. *We want to accept iff there are at least* k *distinct values for* i *where* $x_i = 1$.

▶ **Proposition 3.2** (Theorem 13 in [29]). *For every* $\gamma > 0$ *there is an* $\alpha > 0$ *such that any quantum* k*-threshold circuit with at most* $T \leq \alpha\sqrt{kn}$ *queries and with perfect soundness must have completeness* $\sigma \leq e^{-\gamma k}$ *on inputs with Hamming weight* k.

▶ **Lemma 3.3.** *Let $\gamma > 0$. Let n be sufficiently large and $\mathcal{C}(X)$ be a quantum circuit with input $X = x_1, \ldots, x_n$. There is a $\beta < 1$ depending only on γ such that for all $k \leq \beta^2 n$ and $R \subseteq \{n/2+1, \ldots, n\}$ where $|R| = k$, if $\mathcal{C}(X)$ makes at most $\beta\sqrt{kn}$ queries, then the probability that $\mathcal{C}(X)$ can correctly output all k pairs (x_i, r_j) where $r_j \in R$ and x_i is the r_j-th smallest element of X is at most $e^{(1-\gamma)k-1}$. If R is a contiguous set of integers, then the probability is at most $e^{-\gamma k}$.*

A version of this lemma was first proved in [29] with the additional assumption that the set of output ranks R is a contiguous set of integers; this was sufficient to show that any quantum circuit that produces its sorted output in sorted time order requires that T^2S is $\Omega(n^3)$. The authors stated that their proof can be generalized to any fixed rank ordering, but the generalization is not obvious. We generalize their lemma to non-contiguous R, which is sufficient to obtain an $\Omega(n^3/T)$ lower bound on the cumulative complexity of sorting independent of the time order in which the sorted output is produced.

Proof of Lemma 3.3. Choose α as the constant for γ in Proposition 3.2 and let $\beta = \sqrt{2}\alpha/6$. Let $\mathcal{C}$ be a circuit with at most $\beta\sqrt{kn}$ layers that outputs the k correct pairs (x_i, r_j) with probability p. Let $R = \{r_1, \ldots r_k\}$ where $r_1 < r_2 < \ldots < r_k$. We describe our construction of a circuit $\mathcal{C}'(X)$ solving the k-threshold problem on inputs $X = x_1, \ldots, x_{n/2}$ with exactly k ones in terms of a function $f : [n/2] \to R$. Given f, we re-interpret the input as follows: we replace each x_i with $x'_i = f(i)x_i$, add k dummy values of 0, and add one dummy value of j for each $j \in \{n/2+1, \ldots, n\} \setminus R$. Doing this gives us an input $X' = x'_1, \ldots, x'_n$ that has $n/2$ zeroes. If we assume that f is 1-1 on the k ones of X, then the image of the ones of X will be R and there will be precisely one element of X' for each $j \in \{n/2+1, \ldots, n\}$. Therefore the element of rank $j > n/2$ in X' will have value j, and hence the rank $r_1, \ldots, r_k$ elements of X' will be the images of precisely those elements of X with $x_i = 1$.

To obtain perfect soundness, we cannot rely on the output of $\mathcal{C}(X')$ and must be able to check that each of the output ranks was truly mapped to by a distinct one of X. For each element x_i of X we simply append its index i as $\log_2 n$ low order bits to its image x'_i and append an all-zero bit-vector of length $\log_2 n$ to each dummy value to obtain input X''. Doing so will not change the ranks of the elements in X', but will allow recovery of the k indices that should be the ones in X. In particular, circuit $\mathcal{C}'(X)$ will run $\mathcal{C}(X'')$ and then for each output x''_j with low order bits i, $\mathcal{C}'(X)$ will query x_i, accepting if and only if all of those $x_i = 1$. More precisely, since the mapping from each x_i to the corresponding x''_i is only a function of f, x_i, and i, as long as $\mathcal{C}'(X)$ has an explicit representation of f, it can simulate each query of $\mathcal{C}(X'')$ with two oracle queries to X. Since $\mathcal{C}'$ has at most

$$2\beta\sqrt{kn} + k \leq 3\beta\sqrt{kn} \leq \alpha\sqrt{kn/2}$$

layers, by Proposition 3.2, it can only accept with probability $\leq e^{-\gamma k}$ on inputs with k ones.

We now observe that for each fixed X with exactly k ones, for a randomly chosen function $f : [n/2] \to R$, the probability that f is 1-1 on the ones of X' is exactly $k!/k^k \geq e^{1-k}$. Therefore $\mathcal{C}'(X)$ will give the indices of the k ones in X with probability[1] at least $p \cdot e^{1-k}$. However, this probability must be at most $e^{-\gamma k}$, so we can conclude that $p \leq e^{(1-\gamma)k-1}$. In the event that R is a contiguous set of integers, observe that any choice for the function f will make X'' have the ones of X become ranks $r_1, \ldots, r_k$. So the probability of finding the ones is at least $p \leq e^{-\gamma k}$. ◀

[1] Note that though this is exponentially small in k it is still sufficiently large compared to the completeness required in the lower bound for the k-threshold problem.

By setting k and γ appropriately, Lemma 3.3 gives a useful upper bound on the number of fixed ranks successfully output by any $\beta\sqrt{Sn}$ query quantum circuit that has access to S qubits of input dependent initial state. To handle input-dependent initial state, we will need to use the following proposition.

▶ **Proposition 3.4** ([1]). *Let $\mathcal{C}$ be a quantum circuit, ρ be any S qubit (possibly mixed) state, and I be the S qubit maximally mixed state. If $\mathcal{C}$ with initial state ρ produces some output $\mathcal{O}$ with probability p, then $\mathcal{C}$ with initial state I produces $\mathcal{O}$ with probability at least $p/2^{2S}$.*

This allows us to bound the overall progress made by any short quantum circuit.

▶ **Lemma 3.5.** *There is a constant $\beta > 0$ such that, for any fixed set of $S \leq \beta^2 n$ ranks that are greater than $n/2$, the probability that any quantum circuit $\mathcal{C}$ with at most $\beta\sqrt{Sn}$ queries and S qubits of input-dependent initial state correctly produces the outputs for these S ranks is at most $1/e$.*

Proof. Choose β as the constant when γ is $1+\ln(4)$ in Lemma 3.3. Applying Proposition 3.4 to the bound in Lemma 3.3 gives us that a quantum circuit with S qubits of input-dependent state can produce a fixed set of $k \leq \beta^2 n$ outputs larger than median with a probability at most $2^{2S}e^{(1-\gamma)k-1}$. Since $\gamma = 1+\ln(4)$ setting $k = S$ gives that this probability is $\leq 1/e$. ◀

▶ **Theorem 3.6.** *When n is sufficiently large, any quantum circuit $\mathcal{C}$ for sorting a list of length n with success probability at least $1/e$ and at most T layers that produces its sorted outputs in any fixed time order requires cumulative memory that is $\Omega(n^3/T)$.*

Proof. We partition $\mathcal{C}$ into blocks with large cumulative memory that can only produce a small number of outputs. We achieve this by starting at last unpartitioned layer and finding a suitably low space layer before it so that we can apply Lemma 3.5 to upper bound the number of correct outputs that can be produced in that block with a success probability of at least $1/e$. Let β be the constant from Lemma 3.5 and $k^*(t)$ be the least non-negative integer value of k such that the interval:

$$I(k,t) = \left[t - \frac{\beta}{2}(2^{k+1}-1)\sqrt{n}, t - \frac{\beta}{2}(2^k-1)\sqrt{n}\right]$$

contains some t' such that $S_{t'} \leq 4^k - 1$. We recursively define our blocks as follows. Let ℓ be the number of blocks generated by this method. The final block $\mathcal{C}_\ell$ starts with the first layer $t_{\ell-1} \in I(k^*(T), T)$ where $S_{t_{\ell-1}} \leq 4^{k^*(T)} - 1$ and ends with layer $t_\ell = T$. Let t_i be the first layer of block $\mathcal{C}_{i+1}$. Then the block $\mathcal{C}_i$ starts with the first layer $t_{i-1} \in I(k^*(t_i), t_i)$ where $S_{t_{i-1}} \leq 4^{k^*(t_i)} - 1$ and ends with t_i. See Figure 2 for an illustration of our partitioning. Since $S_0 = 0$ we know that $k^*(t) \leq \log(T)$. Likewise since $S_t > 0$ when $t > 0$, for all $t > \frac{\beta}{2}\sqrt{n}$ we know that $0 < k^*(t) \leq \log(T)$.

Block $\mathcal{C}_i$ starts with less than $4^{k^*(t_i)}$ qubits of initial state and has length at most $\beta 2^{k^*(t_i)}\sqrt{n}$; so by Lemma 3.5, if $4^{k^*(t_i)} \leq \beta^2 n$, the block $\mathcal{C}_i$ can output at most $4^{k^*(t_i)}$ inputs with failure probability at most $1/e$. Additionally $\mathcal{C}_i$ has at least $\frac{\beta}{2}2^{k^*(t_i)-1}\sqrt{n}$ layers so

$$\sum_{i=1}^{\ell} \frac{\beta}{4} 2^{k^*(t_i)}\sqrt{n} \leq T \tag{1}$$

and each of these layers has at least $4^{k^*(t_i)-1}$ qubits[2], so the cumulative memory of $\mathcal{C}_i$ is at

[2] This may not hold for $\mathcal{C}_1$ with length less than $\frac{\beta}{2}\sqrt{N}$, but Lemma 3.3 gives us that this number of layers is insufficient to find a fixed rank input with probability at least $1/e$. Thus we can omit such a block from our analysis.

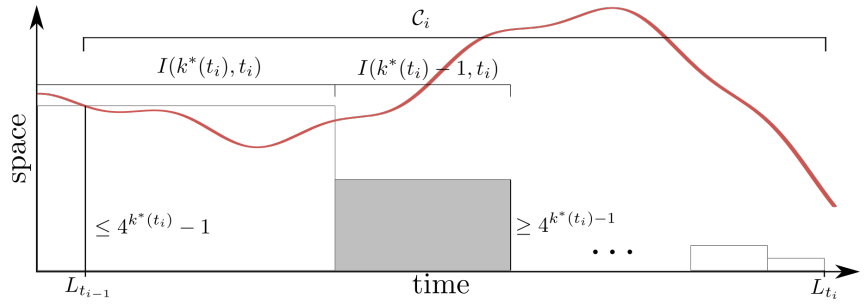


Figure 2 How we define the block $\mathcal{C}_i$ that ends at layer L_{t_i}. The red line is a plot of $\mathcal{C}$'s space over time. The grey layers are the ones used to lower bound the cumulative memory complexity of $\mathcal{C}_i$, as each of these layers uses at least $4^{k^*(t_i)-1}$ qubits and the length of this interval is $\frac{\beta}{2}2^{k^*(t_i)-1}\sqrt{n}$.

least $\frac{\beta}{2}2^{3k^*(t_i)-3}\sqrt{n}$ so

$$CM(\mathcal{C}) \geq \sum_{i=1}^{\ell} \frac{\beta}{2}2^{3k^*(t_i)-3}\sqrt{n}. \tag{2}$$

We now have two possibilities: If we have some i such that $4^{k^*(t_i)} > \beta^2 n$, the cumulative memory of $\mathcal{C}_i$ alone is at least $\beta^4 n^2/16$ which is $\Omega(n^2)$ and hence $\mathcal{C}$ has cumulatively memory $\Omega(n^3/T)$ since $T \geq n$. Otherwise, since we require that the algorithm is correct with probability at least $1/e$, each block $\mathcal{C}_i$ can produce at most $4^{k^*(t_i)}$ outputs. Since our circuit must output all $n/2$ elements larger than the median, we know $\sum_{i=1}^{\ell} 4^{k^*(t_i)} \geq n/2$. For convenience we define $w_i = 2^{k^*(t_i)}$ which allows us to express the constraints as

$$CM(\mathcal{C}) \geq \frac{\beta}{16}\sqrt{n}\sum_{i=1}^{\ell} w_i^3 \quad \text{and} \quad \frac{\beta}{4}\sqrt{n}\sum_{i=1}^{\ell} w_i \leq T \quad \text{and} \quad \sum_{i=1}^{\ell} w_i^2 \geq n/2. \tag{3}$$

Minimizing $\sum_{i=1}^{\ell} w_i^3$ is a non-convex optimization problem and can instead be solved using

$$\text{Minimize} \quad \sum_{i=1}^{\ell} x_i^3 \quad \text{subject to} \quad \sum_{i=1}^{\ell} x_i^2 \geq \xi \quad \text{and} \quad \sum_{i=1}^{\ell} x_i \leq \xi \quad \text{and} \quad \forall i, x_i \geq 0, \tag{4}$$

for $x_i = \frac{8T}{\beta n^{3/2}} w_i$ and $\xi = \frac{32T^2}{\beta^2 n^2}$. Lemma A.1 from Appendix C shows that for non-negative x_i with $\sum x_i \leq \sum x_i^2$, we have $\sum x_i^2 \leq \sum x_i^3$. Thus $\sum x_i^3 \geq \xi$ and applying the variable substitution gives us: $\sum_{i=1}^{\ell} w_i^3 \geq \frac{\beta n^{5/2}}{16T}$. Plugging this into Equation (3) gives us the bound: $CM(\mathcal{C}) \geq \frac{\beta^2 n^3}{256T}$ and hence the cumulative memory of $\mathcal{C}$ is $\Omega(n^3/T)$. ◀

4 General methods for proving cumulative memory lower bounds

Our method involves adapting techniques previously used to prove tradeoff lower bounds on worst-case time and worst-case space. We show that the same properties that yield lower bounds on the product of time and space in the worst case can also be used to produce nearly identical lower bounds on cumulative memory. To do so, we first revisit the standard approach to such time-space tradeoff lower bounds.

The standard method for time-space tradeoff lower bounds for multi-output functions

Consider a multi-output function f on D^n where the output $f(x)$ is either unordered (the output is simply a set of elements from R) or ordered (the output is a vector of elements from R). Then $|f(x)|$ is either the size of the set or the length of the vector of elements. The standard method for obtaining an ordinary time-space tradeoff lower bounds for multi-output functions on D-way branching programs is the following:

The part that depends on f. Choose a suitable probability distribution μ on D^n, often simply the uniform distribution on D^n and then:

(A) Prove that $\Pr_{x\sim\mu}[|f(x)| \geq m] \geq \alpha$.

(B) Prove that for all $k \leq m'$ and any branching program B of height $\leq h'(k,n)$, the probability for $x \sim \mu$ that B produces at least k correct output values of f on input x is at most $C \cdot K^{-k}$ for some m', h', $K = K(R,n)$, and constant C independent of n.

Observe that under any distribution μ, a branching program with ordered outputs that makes no queries can produce k outputs that are all correct with probability at least $|R|^{-k}$, so the bound in (B) shows that, roughly, up to the difference between K and $|R|$ there is not much gained by using a branching program of height h.

The generic completion. In the following outline we omit integer rounding for readability.

- Let $S' = S + \log_2 T$ and suppose that

$$S' \leq m' \log_2 K - \log_2(2C/\alpha). \tag{5}$$

- Let $k = [S' + \log_2(2C/\alpha)]/\log_2 K$, which is at most m' by hypothesis on S', and define $h(S',n) = h'(k,n)$.
- Divide time T into $\ell = T/h$ blocks of length $h = h(S',n)$.
- The original branching program can be split into at most $T \cdot 2^S = 2^{S'}$ sub-branching programs of height $\leq h$, each beginning at a boundary node between layers. By Property (B) and a union bound, for $x \sim \mu$ the probability that at least one of these $\leq 2^{S'}$ sub-branching programs of height at most h produces k correct outputs on input x is at most $2^{S'} \cdot C \cdot K^{-k} \leq \alpha/2$ by our choice of k.
- Under distribution μ, by (A), with probability at least α, an input $x \sim \mu$ has some block of time where at least $m/\ell = m \cdot h(S',n)/T$ outputs of f must be produced on input x.
- If $m \cdot h(S',n)/T \leq k$, this can occur for at most an $\alpha/2$ fraction of inputs under μ. Therefore we have $m \cdot h(S',n)/T > k = [S' + \log_2(2C/\alpha)]/\log_2 K$ and hence since $h(S',n) \geq h(S,n)$, combining with Equation (5), we have

$$T \cdot (S + \log_2 T) = T \cdot S' \geq \min\left(m\ h(S,n),\ m'\ n'\right)\ \log_2 K - \log_2(C/\alpha) \cdot T$$

where $n' \leq n$ is the decision tree complexity of f and hence a lower bound on T.

▶ Remark 4.1. Though it will not impact our argument, for many instances of the above outline, the proof of Property (B) is shown for a decision tree of the same height by proving an analog for the conditional probability along each path in the decision tree separately; this will apply to the tree as a whole since the paths are followed by disjoint inputs, so Property (B) follows from the alternative property below:

(B') For any partial assignment τ of $k \leq m'$ output values over R and any restriction (i.e., partial assignment) π of $h'(k,n)$ coordinates within D^n,

$$\Pr_{x\sim\mu}[f(x) \text{ is consistent with } \tau \mid x \text{ is consistent with } \pi] \leq C \cdot K^{-k}.$$

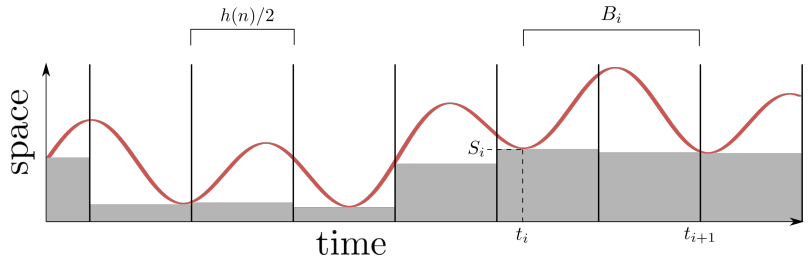


Figure 3 Our generic method for choosing blocks when $h(k,n) = h(n)$. The area marked in grey corresponds to the cumulative memory lower bound we obtain.

Observe that Property (B') is only a slightly more general version of Property (*) from the introduction where $C = 1$, m' is arbitrary, and h' is used instead of h.

▶ Remark 4.2. The above method still gives lower bounds for many multi-output functions $g : D^N \to R^M$ that have individual output values that are easy to compute or large portions of the input space on which they are easy to compute. The bounds follow by applying the method to some subfunction f of g given by $f(x) = \Pi_O(g(x,\pi))$ where π is a partial assignment to the input coordinates and Π_O is a projection onto a subset O of output coordinates. In the subsequent discussions we ignore this issue, but the idea can be applied to all of our lower bound methods.

A general extension to cumulative memory bounds

To give a feel for the basic ideas of the method, we first show this for a simple case. Observe that, other than the separate bound on time, the lower bound on cumulative memory usage we prove in this case is asymptotically identical to the bound achieved for the product of time and worst-case space using the standard outline.

▶ **Theorem 4.3.** *Let $c > 0$. Suppose that properties (A) and (B) apply for $h'(k,n) = h(n)$, $m' = m$, and $\alpha = C = 1$. If $T \log_2 T \le \frac{m\ h(n)\ \log_2 K}{6(c+1)}$ then the cumulative memory used in computing $f : D^n \to R^m$ in time T with success probability at least T^{-c} is at least $\frac{1}{6}\, m\ h(n) \log_2 K$.*

Proof. Fix a deterministic branching program P of length T computing f. Rather than choosing fixed blocks of height $h = h(n)$, layers of nodes at a fixed distance from each other, and a fixed target of k outputs per block, we choose the block boundaries depending on the properties of P and the target k depending on the property of the boundary layer chosen.

Let $H = \lfloor h(n)/2 \rfloor$. We break P into $\ell = \lceil T/H \rceil$ time segments of length H working backwards from step T so that the first segment may be shorter than the rest. We let $t_1 = 0$ and for $1 < i \le \ell$ we let $t_i = \arg\min\{\ |L_t| \ :\ T - (\ell - i + 1) \cdot H \le t < T - (\ell - i) \cdot H\ \}$ be the time step with the fewest nodes among all time steps $t \in [T - (\ell - i + 1) \cdot H, T - (\ell - i) \cdot H]$.

The i-th time block of P will be between times t_i and t_{i+1}. Observe that by construction $|t_{i+1} - t_i| \le h(n)$ so each block has length at most $h(n)$. This construction is shown in Figure 3 Set $S_i = \log_2 |L_{t_i}|$ so that L_{t_i} has at 2^{S_i} nodes. By definition of each t_i, the cumulative memory used by P,

$$CM(P) \ge \sum_{i=1}^{\ell} S_i \cdot H. \tag{6}$$

(Note that since $S_1 = 0$, it does not matter that the first segment is shorter than the rest[3].)

[3] This simplifies some calculations and is the prime reason for starting the time segment boundaries at T.

We now define the target k_i for the number of output values produced in each time block to be the smallest integer such that $K^{-k_i} \leq 2^{-S_i}/T^{c+1}$. That is,

$$k_i = \lceil (S_i + (c+1)\log_2 T)/\log_2 K \rceil.$$

For $x \sim \mu$, for each $i \in [\ell]$ and each sub-branching program B rooted at some node in L_{t_i} and extending until time t_{i+1}, by our choice of k_i and Property (B), if $k_i \leq m$, the probability that B produces at least k_i correct outputs on input x is at most $2^{-S_i}/T^{c+1}$. Therefore, by a union bound, for $x \sim \mu$ the probability that P produces at least k_i correct outputs in the i-th time block on input x is at most $|L_{t_i}| \cdot 2^{-S_i}/T^{c+1} = 1/T^{c+1}$. Therefore, if each $k_i \leq m$, the probability for $x \sim \mu$ that there is some i such that P produces at least k_i correct outputs on input x during the i-th block is at most $\ell/T^{c+1} < T^c$ and the probability for $x \sim \mu$ that P produces at most $\sum_{i=1}^{\ell}(k_i - 1)$ correct outputs in total on input x is $> 1 - 1/T^c$.

If each $k_i \leq m$, since P must produce m correct outputs on $x \in D^n$ with probability at least $1/T^c$, we must have $\sum_{i=1}^{\ell}(k_i - 1) \geq m$. On the other hand, if some $k_i > m$ we have the same bound. Using our definition of k_i we have $\sum_{i=1}^{\ell}[(S_i + (c+1)\log_2 T)]/\log_2 K] \geq m$ or $\sum_{i=1}^{\ell}(S_i + (c+1)\log_2 T) \geq m \cdot \log_2 K$. Plugging in the bound (6) on the cumulative memory and the value of ℓ, it implies that $CM(P)/H + (c+1)\lceil T/H \rceil \cdot \log_2 T \geq m \cdot \log_2 K$ or that $CM(P) + (c+1)T\log_2 T \geq \frac{1}{3}\ m \cdot h(n) \cdot \log_2 K$, where the 3 on the right rather than a 2 allows us to remove the ceiling. Therefore either

$$T\log_2 T > \frac{m \cdot h(n) \cdot \log_2 K}{6(c+1)} \qquad \text{or} \qquad CM(P) \geq \frac{1}{6}\ m\ h(n)\ \log_2 K. \qquad \blacktriangleleft$$

Simple applications. This simple case of our general theorem is sufficient to obtain many tight new lower bounds on cumulative memory complexity including the following (full proofs are in [18]):

▶ **Corollary 4.4.** *Producing the ranks (positions of each input in the sorted order) or sorting n integers from $[n^2]$ requites CMC that is $\Omega(n^2/\log_2 n)$.*

▶ **Corollary 4.5.** *For many fixed (random or explicit) $n \times n$ matrices A, computing Ax over a finite field $\mathbb{F}$ requires CMC that is $\Omega(n^2 \log |\mathbb{F}|)$.*

Corollary 4.4 uses property (B) for ranking with $m' = h'(k,n) = \Theta(n)$, $C = 1$, $K = 2^{\Theta(1/\log n)}$ proven in [22, Lemma 1]. Corollary 4.5 uses property (B') with $m' = h'(k,n) = cn$, for $0 < c \leq 1/2$, $C = 1$, and $K = |\mathbb{F}|^c$ proven in [4, Theorem 4.6].

Full general theorem. In the general version of our theorem there are a number of additional complications, most especially because the branching program height limit $h(k,n)$ in Property (B) *can depend on* k, the target for the number of outputs produced. This forces the lengths of the blocks and the space used at the boundaries between blocks to depend on each other in a quite delicate way. In order to discuss the impact of that dependence and state our general theorem, we need the following definition.

▶ **Definition 4.6.** *Given a non-decreasing function $p : \mathbb{R} \to \mathbb{R}$ with $p(1) = 1$, we define $p^{-1} : \mathbb{R} \to \mathbb{R} \cup \{\infty\}$ by $p^{-1}(R) = \min\{j \mid p(j) \geq k\}$. We also define the* loss, *$\mathcal{L}_p$, of p by*

$$\mathcal{L}_p(n) = \min_{1 \leq k \leq p(n)} \frac{\sum_{j=1}^{k} p^{-1}(j)}{k \cdot p^{-1}(k)}.$$

▶ **Lemma 4.7.** *The following hold for every non-decreasing function $p : \mathbb{R} \to \mathbb{R}$ with $p(1) = 1$:*

(a) $1/p(n) \le \mathcal{L}_p(n) \le 1$.

(b) *If p is a polynomial function $p(s) = s^{1/c}$ then $\mathcal{L}_p(n) > 1/2^{c+1}$.*

(c) *For any $c > 1$, $\mathcal{L}_p(n) \ge \min_{1 \le s \le n} \dfrac{p(s) - p(s/c)}{cp(s)}$.*

(d) *We say that p is* nice *if it is differentiable and there is an integer $c > 1$ such that for all x, $p'(cx) \ge p'(x)/c$. If p is nice then $\mathcal{L}_p(n)$ is $\Omega(1/\log_2 n)$. This is tight for p with $p(s) = 1 + \log_2 s$.*

We prove this technical lemma in the full paper [18]. Here is our full general theorem.

▶ **Theorem 4.8.** *Let $c > 0$. Suppose that function f defined on D^n has properties (A) and (B) with α that is $1/n^{O(1)}$ and m' that is $\omega(\log_2 n)$. For $s > 0$, define $h(s, n)$ to be $h'(k, n)$ for $k = s/\log_2 K$. Suppose that $h(s, n) = h_0(s)\, h_1(n)$ with $h_0(1) = 1$ and h_0 is constant or a differentiable function such that $s/h_0(s)$ is increasing and concave. Define $S^* = S^*(T, n)$ by*

$$S^*/h_0(S^*) = (m\ h_1(n)\ \log_2 K)\,/6T.$$

(a) *Either $T \log_2(2CT^{c+1}/\alpha) > \dfrac{1}{6}\, m\ h_1(n)\ \log_2 K$, which implies that T is $\Omega(\frac{m\ h_1(n)\ \log K}{\log n})$, or the cumulative memory used by a randomized branching program in computing f in time T with error $\varepsilon \le \alpha(1 - 1/(2T^c))$ is at least*

$$\frac{1}{6}\, \mathcal{L}_{h_0}(n \log_2 |D|) \cdot \min\left(m\ h(S^*(T, n), n),\ 3m'\ h'(m'/2, n)\right) \cdot \log_2 K.$$

(b) *Further any randomized random-access machine computing f in time T with error $\varepsilon \le \alpha(1 - 1/(2T^c))$ requires cumulative memory*

$$\Omega\left(\mathcal{L}_{h_0}(n \log_2 |D|) \cdot \min\left(m\ h(S^*(T, n), n),\ m'\ h'(m'/2, n)\right) \cdot \log_2 K\right).$$

Before we give the proof of the theorem, we note that by Lemma 4.7, in the case that h_0 is constant or $h_0(s) = s^{\Delta}$ for some constant $\Delta > 0$, which together account for all existing applications we are aware of, the function $\mathcal{L}_{h_0}$ is lower bounded by a constant. In the latter case, h_0 is differentiable, has $h_0(s) = 1$, and the function $s/h_0(s) = s^{1-\Delta}$ is increasing and concave so it satisfies the conditions of our theorem. By using $\alpha = 1$, $m' = m$, and $C = 1$ with h from Property (*) in place of h' in Property (B'), Theorem 4.8 yields Theorem 1.2.

More generally, the value S^* in the statement of this theorem is at least a constant factor times the value of S used in the generic time-space tradeoff lower bound methodology. Therefore, the cumulative memory lower bound in Theorem 4.8 for random-access machines is close to the lower bound on the product of time and space using standard methods.

Proof of Theorem 4.8. We prove both (a) and (b) directly for branching programs, which can model random-access machines, and will describe the small variation that occurs in the case that the branching program in question comes from a random-access machine. To prove these properties for randomized branching programs, by Yao's Lemma [36] it suffices to prove the properties for deterministic branching programs that have error at most ε under distribution μ. Fix a (deterministic) branching program P of length T computing f with error at most ε under distribution μ. Without loss of generality, P has maximum space usage at most $S^{max} = n \log_2 |D|$ space since there are at most $|D^n|$ inputs.

Let $H = \lfloor h_1(n)/2 \rfloor$. We break P into $\ell = \lceil T/H \rceil$ time segments of length H working backwards from step T so that the first segment may be shorter than the rest. We then choose a sequence of *candidates* for the time steps in which to begin new blocks, as follows: We let $\tau_1 = 0$ and for $1 < i \le \ell$ we let

$$\tau_i = \arg\min\{\ |L_t|\ :\ T - (\ell - i + 1) \cdot H \le t < T - (\ell - i) \cdot H\ \}$$

be the time step with the fewest nodes among all time steps $t \in [T-(\ell-i+1)\cdot H, T-(\ell-i)\cdot H]$. Set $\sigma_i = \log_2 |L_{\tau_i}|$ so that L_{τ_i} has at 2^{σ_i} nodes. This segment contributes at least $\sigma_i \cdot H$ to the cumulative memory bound of P.

To choose the beginning t_{i^*} of the last time block[4]. we find the smallest k such that $h_0(\sigma_{\ell-k+1}) < k$. Such a k must exist since h_0 is a non-decreasing non-negative function, $h_0(1) = 1$ and $\sigma_1 = 0 < 1$. We now observe that the length of the last block is at most $k \cdot H$ which by choice of k is less than $h(\sigma_{\ell-k+1}, n)$ and hence we have satisfied the requirements for Property (B) to apply at each starting node of the last time block.

By our choice of each τ_i, the cumulative memory used in the last k segments is at least $\sum_{j=1}^{k} \sigma_{\ell+1-j} \cdot H$. Further, since k was chosen as smallest with the above property, we know that for every $j \in [k-1]$ we have $h_0(\sigma_{\ell-j+1}) \geq j$ Hence we have $\sigma_{\ell-j+1} \geq h_0^{-1}(j)$ and we get a cumulative memory bound for the last k segments of at least

$$(\sigma_{\ell-k+1} + \sum_{j=1}^{k-1} h_0^{-1}(j)) \cdot H. \tag{7}$$

▷ Claim 4.9. $\sigma_{\ell-k+1} + \sum_{j=1}^{k-1} h_0^{-1}(j) \geq \mathcal{L}_{h_0}(S^{max}) \cdot \sigma_{\ell-k+1} \cdot k$.

Proof of Claim. Observe that it suffices to prove the claim when we replace $\sigma_{\ell-k+1}$, which appears on both sides, by a larger quantity. In particular, we show how to prove the claim with $h_0^{-1}(k)$ instead, which is larger since $h_0(\sigma_{\ell-k+1}) < k$. But this follows immediately since by definition $\mathcal{L}_{h_0}(S^{max}) \leq \frac{\sum_{j=1}^{k} h_0^{-1}(j)}{k \cdot h_0^{-1}(k)}$, which is equivalent to what we want to prove. ◁

Write $S_{i^*} = \sigma_{\ell-k+1}$. By the claim, the cumulative memory contribution associated with the last block beginning at t_{i^*} is at least $\mathcal{L}_{h_0}(S^{max}) \cdot S_{i^*} \cdot h_0(S_{i^*})H$.

We repeat this in turn to find the time step for the beginning of the next block from the end, t_{i^*-1}. One small difference now is that there is a last partial segment of height at most H from the beginning of segment containing t_{i^*} to layer t_{i^*}. However, this only adds at most $h_1(n)/2$ to the length of the segment which still remains well within the height bound of $h(S_{i^*-1}, n) = h_0(S_{i^*-1})h_1(n)$ for Property (B) to apply.

Repeating this back to the beginning of the branching program we obtain a decomposition of the branching program into some number i^* of blocks, the i-th block beginning at time step t_i with 2^{S_i} nodes, height between $h_0(S_i)H$ and $h_0(S_i)H + H \leq 2h_0(S_i)H$, and with an associated cumulative memory contribution in the i-th block of $\geq \mathcal{L}_{h_0}(S^{max}) \cdot S_i \cdot h_0(S_i)H$. (This is correct even for the partial block starting at time $t_1 = 0$ since $S_1 = 0$.) Since we know that $i^* \leq \ell$, for convenience, we also define $S_i = 0$ for $i^* + 1 \leq i \leq \ell$. Then, by definition

$$CM(P) \geq \mathcal{L}_{h_0}(S^{max}) \cdot \left(\sum_{i=1}^{i^*} S_i \cdot h_0(S_i) \right) \cdot H = \mathcal{L}_{h_0}(S^{max}) \cdot \left(\sum_{i=1}^{\ell} S_i \cdot h_0(S_i) \right) \tag{8}$$

$$\text{and} \qquad \sum_{i=1}^{\ell} h_0(S_i) \leq T/H. \tag{9}$$

As in the previous argument for the simple case, for $i \leq i^*$, we define the target k_i for the number of output values produced in each time block to be the smallest integer such that $C\dot{K}^{-k_i} \leq 2^{-S_i}\alpha/(2T^{c+1})$. That is, $k_i = \lceil (S_i + \log_2(2CT^{c+1}/\alpha))/\log_2 K \rceil$.

[4] Since we are working backwards from the end of the branching program and we do not know how many segments are included in each block, we don't actually know this index until things stop with $t_1 = 0$

If $k_i > m'$ for some i, then $S_i \geq m' \cdot \log_2 K - \log_2(2CT^{c+1}/\alpha) \geq (m' \log_2 K)/2$ since m' is $\omega(\log n)$ and $1/\alpha$ and T are $n^{O(1)}$. Therefore $h_0(S_i) \geq h'(m'/2, n)$ and hence

$$CM(P) \geq \frac{1}{2}\,\mathcal{L}_{h_0}(S^{max}) \cdot m' \cdot h'(m'/2, n) \cdot \log_2 K$$

Suppose instead that $k_i \leq m'$ for all $i \leq i^*$. Then, for $x \sim \mu$, for each $i \in [i^*]$ and each sub-branching program B rooted at some node in L_{t_i} and extending until time t_{i+1}, by our choice of k_i and Property (B), the probability that B produces at least k_i correct outputs on input x is at most $\alpha \cdot 2^{-S_i}/(2T^{c+1})$. Therefore, by a union bound, for $x \sim \mu$ the probability that P produces at least k_i correct outputs in the i-th time block on input x is at most

$$|L_{t_i}| \cdot \alpha \cdot 2^{-S_i}/(2T^{c+1}) = \alpha/(2T^{c+1})$$

and hence the probability for $x \sim \mu$ that there is some i such that P produces at least k_i correct outputs on input x during the i-th block is at most $\ell \cdot \alpha/(2T^{c+1}) < \alpha/(2T^c)$. Therefore, the probability for $x \sim \mu$ that P produces at most $\sum_{i=1}^{\ell}(k_i - 1)$ correct outputs in total on input x is $> 1 - \alpha/(2T^c)$.

Since, by Property (A) and the maximum error it allows, P must produce at least m correct outputs with probability at least $\alpha - \epsilon \geq \alpha - \alpha(1 - 1/(2T^c)) = \alpha/(2T^c)$ for $x \sim \mu$, we must have $\sum_{i=1}^{i^*}(k_i - 1) \geq m$. Using our definition of k_i we obtain

$$\sum_{i=1}^{i^*}(S_i + \log_2(2CT^{c+1}/\alpha)) \geq m \log_2 K.$$

This is the one place in the proof where there is a distinction between an arbitrary branching program and one that comes from a random access machine.

We first start with the case of arbitrary branching programs: Note that $i^* \leq \ell = \lceil T/H \rceil = \lceil T/\lfloor h_1(n)/2 \rfloor \rceil$. Suppose that $T \log_2(2CT^{c+1}/\alpha) \leq \frac{1}{6}\, m \cdot h_1(n) \cdot \log_2 K$. Then, even with rounding, we obtain $\sum_{i=1}^{i^*} S_i \geq \frac{1}{2}\, m \log_2 K$.

Unlike an arbitrary branching program that may do non-trivial computation with sub-logarithmic S_i, a random-access machine with even one register requires at least $\log_2 n$ bits of memory (just to index the input for example) and hence $S_i + \log_2(2CT^{c+1}/\alpha)$ will be $O(S_i)$, since T is at most polynomial in n and $1/\alpha$ is at most polynomial in n by assumption. Therefore we obtain that $\sum_{i=1}^{i^*} S_i$ is $\Omega(m \log_2 K)$ without the assumption on T.

In the remainder we continue the argument for the case of arbitrary branching programs and track the constants involved. The same argument obviously applies for programs coming from random-access machines with slightly different constants that we will not track. In particular, since $S_i = 0$ for $i > i^*$ we have

$$\sum_{i=1}^{\ell} S_i \geq \frac{1}{2}\, m \cdot \log_2 K. \tag{10}$$

From this point we need to do something different from the argument in the simple case because the lower bound on the total cumulative memory contribution is given by Equation (8) and is not simply $\sum_{i=1}^{\ell} S_i \cdot H$. Instead, we combine Equation (10) and Equation (9) using the following technical lemma that we prove in Appendix A.

▶ **Lemma 4.10.** *Let $p : \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$ be a differentiable function such that $q(x) = x/p(x)$ is a concave increasing function of x. For $x_1, x_2, \ldots \in \mathbb{R}^{\geq 0}$, if $\sum_i x_i \geq K$ and $\sum_i p(x_i) \leq L$ then $\sum_i x_i p(x_i) \geq q^{-1}(K/L) \cdot L$.*

In our application of Lemma 4.10, $p = h_0$, $K = \frac{1}{2}\, m \cdot \log_2 K$, and $L = T/H$. Let S^* be the solution to $\frac{S^*}{h_0(S^*)} = K/L = \frac{m \cdot H \cdot \log_2 K}{2T} \geq \frac{m \cdot h_1(n) \log_2 K}{6T}$. Then Lemma 4.10 implies that $\sum_{i=1}^{\ell} S_i \cdot h_0(S_i) \geq S^* \cdot T/H = \frac{1}{2}, m \cdot h_0(S^*) \cdot \log_2 K$. and hence

$$CM(P) \geq \mathcal{L}_{h_0}(S^{max}) \cdot \frac{1}{2}\, m \cdot h_0(S^*) \cdot H \cdot \log_2 K \geq \frac{1}{6}\, \mathcal{L}_{h_0}(S^{max}) \cdot m \cdot h(S^*, n) \cdot \log_2 K$$

since $H = \lfloor h_1(n)/2 \rfloor$ and $h(S^*, n) = h_0(S^*) \cdot h_1(n)$. ◀

In the special case that $h_0(s) = s^{\Delta}$ (and indeed for any nice function h_0), there is an alternative variant of the above in which one breaks up time into exponentially growing segments starting with time step T. We used that alternative approach in Section 3.

▶ Remark 4.11. If we restrict our attention to $o(m' \log K)$-space bounded computation, then each $k_i \leq m'$ and the cumulative memory bound for a branching program in Theorem 4.8 becomes $\frac{1}{6}\, \mathcal{L}_{h_0}(n \log_2 |D|) \cdot m \cdot h(S^*(T, n), n) \cdot \log_2 K$. And the bound for RAM cumulative memory becomes $\Omega\left(\mathcal{L}_{h_0}(n \log_2 |D|) \cdot m \cdot h(S^*(T, n), n) \cdot \log_2 K\right)$.

Generic method for quantum time-space tradeoffs

Quantum circuit time-space lower bounds have the same general structure as their classical branching program counterparts. They require a lemma similar to (B) that gives an exponentially small probability of producing k outputs with a small number of queries.

▶ **Lemma 4.12** (Quantum generic property). *For all $k \leq m'$ and any quantum circuit $\mathcal{C}$ with at most $h'(k, n)$ layers, there exists a distribution μ such that when $x \sim \mu$, the probability that $\mathcal{C}$ produces at least k correct output values of $f(x)$ is at most $C \cdot K^{-k}$.*

Such lemmas have historically been proving using direct product theorems [29, 13] or the recording query technique [27]. Quantum time-space tradeoffs use the same blocking strategy as branching programs; however, they cannot use union bounds to account for input-dependent state at the start of a block. Instead, Proposition 3.4 lets us apply Lemma 4.12 to blocks in the middle of a quantum circuit. The 2^{2S} factor in Proposition 3.4 means that a quantum time-space or cumulative memory lower bound is half of what you would expect from a classical bound. Since a quantum circuit requires $\log_2 n$ qubits to make a query, we know that the space between layers is always at least $\log_2 n$ and

$$T \cdot S \text{ is } \Omega\left(\min\{m\ h'(S, n), m'\ Q(f)\} \cdot \log_2 K\right)$$

where $Q(f)$ is the bounded-error quantum query complexity of f.

Generic method for quantum cumulative complexity bounds

Our generic argument can just as easily be applied to quantum lower bounds for problems where we have an instance of Lemma 4.12 using Proposition 3.4 to bound the number of outputs produced even with initial input-dependent state. Quantum circuits require at least $\log_2 n$ qubits to hold the query index so the bounds derived are like those from Theorem 4.8(b).

▶ **Corollary 4.13.** *Let $c > 0$. Suppose that function f defined on D^n satisfies generic Lemma 4.12 with m' that is $\omega(\log_2 n)$. For $s > 0$, let $h(s, n) = h'(s/\log_2 K, n)$. Let $h(s, n) = h_0(s)h_1(n)$ where $h_0(1) = 1$ and h_0 is constant or a differentiable function where $s/h_0(s)$ is increasing and concave. Let S^* be defined by:*

$$S^*/h_0(S^*) = (m\ h_1(n)\ \log_2 K)\,/6T.$$

The CM used by a quantum circuit that computes f in time T with error $\varepsilon \leq (1 - 1/(2T^c))$ is

$$\geq \frac{1}{6}\,\mathcal{L}_{h_0}(n \log_2 |D|) \cdot \min\{m\ h(S^*, n),\ 3m'\ h'(m'/2, n)\} \cdot \log_2 K.$$

If the circuit uses $o(m' \log K)$ qubits, the CMC instead is $\frac{1}{6}\,\mathcal{L}_{h_0}(n \log_2 |D|) \cdot m \cdot h(S^*, n) \cdot \log_2 K$.

Full general applications. Some applications of our full general theorem generalizing classical and quantum time-space product lower bounds are the following (full proofs are in [18]):

▶ **Corollary 4.14.** *The CMC for listing all uniquely occurring elements or sorting n integers from $[n]$ is $\Omega(n^2)$.*

▶ **Corollary 4.15.** *Matrix multiplication for $n \times n$ matrices over finite field $\mathbb{F}$ requires classical CMC that is $\Omega((n^6 \log |\mathbb{F}|)/T)$.*

▶ **Corollary 4.16.** *For every $m \geq n$, finding k disjoint collisions in a random function from $[m]$ to $[n]$ requires quantum CMC that is $\Omega(k^3 n/T^2)$.*

Corollary 4.14 uses properties (A) and (B) for unique elements with $h'(k, n) = n/4$, $m' = n/4$, $m = n/(2e)$, $\alpha = 1/(2e - 1)$, $K = 1/(2 \ln N)$ and $C = 1$ that follow from [16, Lemmas 2, 3]. Corollary 4.15 uses properties (A) and (B') with $h'(k, n) = \Theta(n\sqrt{k})$, $m' = m = n^2$, $\alpha = 1$, $K = |\mathbb{F}|^{\Theta(1)}$ and $C = d^2$. as proven in [4, Theorem 7.1]. Corollary 4.16 uses Lemma 4.12 with $h'(k, n) = \Theta(k^{2/3} n^{1/3}), m' = m = k$, and $C = K = 2$ which follow from [27, Theorem 9].

References

1. Scott Aaronson. Limitations of quantum advice and one-way communication. *Theory of Computing*, 1(1):1–28, 2005. doi:10.4086/toc.2005.v001a001.
2. Karl R. Abrahamson. Generalized string matching. *SIAM J. Comput.*, 16(6):1039–1051, 1987. doi:10.1137/0216067.
3. Karl R. Abrahamson. A time-space tradeoff for Boolean matrix multiplication. In *31st Annual IEEE Symposium on Foundations of Computer Science, Volume I*, pages 412–419, 1990. doi:10.1109/FSCS.1990.89561.
4. Karl R. Abrahamson. Time-space tradeoffs for algebraic problems on general sequential machines. *J. Comput. Syst. Sci.*, 43(2):269–289, 1991. doi:10.1016/0022-0000(91)90014-v.
5. Miklós Ajtai. Determinism versus nondeterminism for linear time RAMs with memory restrictions. *J. Comput. Syst. Sci.*, 65(1):2–37, 2002. doi:10.1006/jcss.2002.1821.
6. Miklós Ajtai. A non-linear time lower bound for Boolean branching programs. *Theory Comput.*, 1(1):149–176, 2005. doi:10.4086/toc.2005.v001a008.
7. Joël Alwen and Jeremiah Blocki. Efficiently computing data-independent memory-hard functions. In *Advances in Cryptology – CRYPTO 2016*, pages 241–271, 2016.
8. Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. Depth-robust graphs and their cumulative memory complexity. In *Advances in Cryptology – EUROCRYPT 2017*, pages 3–32, 2017.
9. Joël Alwen, Binyi Chen, Chethan Kamath, Vladimir Kolmogorov, Krzysztof Pietrzak, and Stefano Tessaro. On the complexity of Scrypt and proofs of space in the parallel random oracle model. In *Advances in Cryptology - EUROCRYPT 2016, Proceedings, Part II*, volume 9666 of *LNCS*, pages 358–387, 2016. doi:10.1007/978-3-662-49896-5_13.
10. Joël Alwen, Binyi Chen, Krzysztof Pietrzak, Leonid Reyzin, and Stefano Tessaro. Scrypt is maximally memory-hard. In *Advances in Cryptology - EUROCRYPT 2017, Proceedings, Part III*, volume 10212 of *Lecture Notes in Computer Science*, pages 33–62, 2017. doi:10.1007/978-3-319-56617-7_2.

11 Joël Alwen, Susanna F. de Rezende, Jakob Nordström, and Marc Vinyals. Cumulative space in black-white pebbling and resolution. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 38:1–38:21, 2017. doi:10.4230/LIPIcs.ITCS.2017.38.

12 Joël Alwen and Vladimir Serbinenko. High parallel complexity graphs and memory-hard functions. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, pages 595–603, 2015. doi:10.1145/2746539.2746622.

13 Andris Ambainis, Robert Špalek, and Ronald de Wolf. A new quantum lower bound method, with applications to direct product theorems and time-space tradeoffs. *Algorithmica*, 55(3):422–461, 2009. doi:10.1007/s00453-007-9022-9.

14 Mohammad Hassan Ameri, Alexander R. Block, and Jeremiah Blocki. Memory-hard puzzles in the standard model with applications to memory-hard functions and resource-bounded locally decodable codes. Cryptology ePrint Archive, Paper 2021/801, 2021. URL: https://eprint.iacr.org/2021/801.

15 Andrew Baird, Bryant Bost, Stefano Buliani, Vyom Nagrani, Ajay Nair, Rahul Popat, and Brajendra Singh. AWS serverless multi-tier architectures with Amazon API Gateway and AWS Lambda, 2021. URL: https://docs.aws.amazon.com/whitepapers/latest/serverless-multi-tier-architectures-api-gateway-lambda/welcome.html.

16 Paul Beame. A general sequential time-space tradeoff for finding unique elements. *SIAM J. Comput.*, 20(2):270–277, 1991. doi:10.1137/0220017.

17 Paul Beame, T. S. Jayram, and Michael E. Saks. Time-space tradeoffs for branching programs. *J. Comput. Syst. Sci.*, 63(4):542–572, 2001. doi:10.1006/jcss.2001.1778.

18 Paul Beame and Niels Kornerup. Cumulative memory lower bounds for randomized and quantum computation. *CoRR*, abs/2301.05680, 2023. doi:10.48550/arXiv.2301.05680.

19 Paul Beame, Michael E. Saks, Xiaodong Sun, and Erik Vee. Time-space trade-off lower bounds for randomized computation of decision problems. *J. ACM*, 50(2):154–195, 2003. doi:10.1145/636865.636867.

20 Jeremiah Blocki and Samson Zhou. On the depth-robustness and cumulative pebbling cost of Argon2i. In *Theory of Cryptography*, pages 445–465, 2017.

21 Dan Boneh, Henry Corrigan-Gibbs, and Stuart Schechter. Balloon hashing: A memory-hard function providing provable protection against sequential attacks. In *Advances in Cryptology – ASIACRYPT 2016*, pages 220–248, 2016.

22 Allan Borodin and Stephen A. Cook. A time-space tradeoff for sorting on a general sequential model of computation. *SIAM J. Comput.*, 11(2):287–297, 1982. doi:10.1137/0211022.

23 Allan Borodin, Michael J. Fischer, David G. Kirkpatrick, Nancy A. Lynch, and Martin Tompa. A time-space tradeoff for sorting on non-oblivious machines. *J. Comput. Syst. Sci.*, 22(3):351–364, 1981. doi:10.1016/0022-0000(81)90037-4.

24 Allan Borodin, Alexander A. Razborov, and Roman Smolensky. On lower bounds for read-k-times branching programs. *Comput. Complex.*, 3:1–18, 1993. doi:10.1007/BF01200404.

25 Binyi Chen and Stefano Tessaro. Memory-hard functions from cryptographic primitives. In *Advances in Cryptology – CRYPTO 2019*, pages 543–572, 2019.

26 Cynthia Dwork, Moni Naor, and Hoeteck Wee. Pebbling and proofs of work. In *Advances in Cryptology – CRYPTO 2005*, pages 37–54, 2005.

27 Yassine Hamoudi and Frédéric Magniez. Quantum time-space tradeoff for finding multiple collision pairs. In *16th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2021)*, volume 197 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1:1–1:21, 2021. doi:10.4230/LIPIcs.TQC.2021.1.

28 Stasys Jukna. A nondeterministic space-time tradeoff for linear codes. *Inf. Process. Lett.*, 109(5):286–289, 2009. doi:10.1016/j.ipl.2008.11.001.

29 Hartmut Klauck, Robert Špalek, and Ronald de Wolf. Quantum and classical strong direct product theorems and optimal time-space tradeoffs. *SIAM Journal on Computing*, 36(5):1472–1493, 2007. doi:10.1137/05063235x.

30 Yishay Mansour, Noam Nisan, and Prasoon Tiwari. The computational complexity of universal hashing. *Theor. Comput. Sci.*, 107(1):121–133, 1993. doi:10.1016/0304-3975(93)90257-T.

31 E. Okol'nishnikova. On lower bounds for branching programs. *Siberian Advances in Mathematics*, 3(1):152–166, 1993.

32 Ling Ren and Srinivas Devadas. Proof of space from stacked expanders. In *Proceedings, Part I, of the 14th International Conference on Theory of Cryptography - Volume 9985*, pages 262–285. Springer-Verlag, 2016. doi:10.1007/978-3-662-53641-4_11.

33 Martin Sauerhoff and Philipp Woelfel. Time-space tradeoff lower bounds for integer multiplication and graphs of arithmetic functions. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 186–195, 2003. doi:10.1145/780542.780571.

34 John E. Savage. *Models of Computation: Exploring the Power of Computing*. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition, 1997.

35 Martin Tompa. Time-space tradeoffs for computing functions, using connectivity properties of their circuits. *J. Comput. Syst. Sci.*, 20(2):118–132, 1980. doi:10.1016/0022-0000(80)90056-2.

36 Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *18th Annual IEEE Symposium on Foundations of Computer Science*, pages 222–227, 1977. doi:10.1109/sfcs.1977.24.

37 Yaacov Yesha. Time-space tradeoffs for matrix multiplication and the discrete Fourier transform on any general sequential random-access computer. *Journal of Computer and System Sciences*, 29(2):183–197, 1984. doi:10.1016/0022-0000(84)90029-1.

38 Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In *Advances in Cryptology – CRYPTO 2019*, pages 239–268, 2019.

A Optimizations

In this section we prove general optimization lemmas that allow us to derive worst-case properties of the allocation of branching program layers into blocks.

▶ **Lemma A.1.** *For non-negative reals $x_1, x_2, \ldots$ if $\sum_i x_i \le \sum_i x_i^2$ then $\sum_i x_i^3 \ge \sum_i x_i^2$.*

Proof. Without loss generality we remove all x_i that are 0 or 1 since they contribute the same amount to each of $\sum_i x_i$, $\sum_i x_i^2$, and $\sum_i x_i^3$. Therefore every x_i satisfies $0 < x_i < 1$ or it satisfies $x_i > 1$. We rename those x_i with $0 < x_i < 1$ by y_i and those x_i with $x_i > 1$ by z_j.

Then $\sum_i x_i \le \sum_i x_i^2$ can be rewritten as $\sum_i y_i(1-y_i) \le \sum_j z_j(z_j-1)$, and both quantities are positive. Let y^* be the largest value < 1 and z^* be the smallest value > 1. Thus:

$$\sum_i (y_i^2 - y_i^3) = \sum_i y_i^2(1-y_i) \le \sum_i y^* y_i(1-y_i) = y^* \sum_i y_i(1-y_i) \le y^* \sum_j z_j(z_j-1)$$
$$< z^* \sum_j z_j(z_j-1) = \sum_j z^* z_j(z_j-1) \le \sum_j z_j^2(z_j-1) = \sum_j (z_j^3 - z_j^2).$$

Rewriting gives $\sum_i y_i^2 + \sum_j z_j^2 < \sum_i y_i^3 + \sum_j z_j^3$, or $\sum_i x_i^3 > \sum_i x_i^2$, as required. ◀

The following is a generalization of the above to all differentiable functions $p : \mathbb{R}^{\ge 0} \to \mathbb{R}^{\ge 0}$ such that $s/p(s)$ is a concave increasing function of s.

▶ **Lemma 4.10.** *Let $p : \mathbb{R}^{\ge 0} \to \mathbb{R}^{\ge 0}$ be a differentiable function such that $q(x) = x/p(x)$ is a concave increasing function of x. For $x_1, x_2, \ldots \in \mathbb{R}^{\ge 0}$, if $\sum_i x_i \ge K$ and $\sum_i p(x_i) \le L$ then $\sum_i x_i p(x_i) \ge q^{-1}(K/L) \cdot L$.*

Proof. By hypothesis, $\sum_i (x_i - Kp(x_i)/L) \geq 0$. Observe that $s - Kp(s)/L$ is an increasing function of s since $s/p(s)$ is an increasing function of s that is 0 precisely when $s = q^{-1}(K/L)$. Since all x_i with $x_i = q^{-1}(K/L)$ evaluate to 0 in the sum, we can rewrite it as

$$\sum_{x_i > q^{-1}(K/L)} (x_i - Kp(x_i)/L) \geq \sum_{x_i < q^{-1}(K/L)} (Kp(x_i)/L - x_i)\,, \tag{11}$$

where each of the summed terms is positive. For $x_i \neq q^{-1}(K/L)$, define

$$f(x_i) = x_i \cdot \frac{p(x_i) - q^{-1}(K/L) \cdot L/K}{x_i - Kp(x_i)/L}.$$

Observe that for $x_i = q^{-1}(K/L)$ the denominator is 0 and the numerator equals $p(x_i) - x_i \cdot L/K$ which is also 0. For $x_i > q^{-1}(K/L)$ both the numerator and denominator are positive and for $x_i < q^{-1}(K/L)$ both the numerator and denominator are negative. Hence $f(x_i)$ is non-negative for every $x_i \neq q^{-1}(K/L)$. The following claim holds because of the concavity of q; its proof is in the full paper [18].

▷ Claim A.2. If q is a convex differentiable function, we can complete f to a (non-decreasing) continuous function of x with $f'(x) \geq 0$ for all x with $0 < x \neq q^{-1}(K/L)$

We now have the tools we need. Let x^*_- be the largest $x_i < q^{-1}(K/L)$ and x^*_+ be the smallest $x_i > q^{-1}(K/L)$. Then we have $f(x^*_+) \geq f(x^*_-)$ and

$$\begin{aligned}
&\sum_{x_i > q^{-1}(K/L)} \left(x_i\ p(x_i) - q^{-1}(K/L) \cdot L/K \cdot x_i\right) \\
&\qquad = \sum_{x_i > q^{-1}(K/L)} f(x_i) \cdot (x_i - Kp(x_i)/L) \\
&\qquad \geq \sum_{x_i > q^{-1}(K/L)} f(x^*_+) \cdot (x_i - Kp(x_i)/L) \\
&\qquad \geq f(x^*_-) \sum_{x_i > q^{-1}(K/L)} (x_i - Kp(x_i)/L) \\
&\qquad \geq f(x^*_-) \sum_{x_i < q^{-1}(K/L)} (Kp(x_i)/L - x_i) \qquad \text{by Equation (11)} \\
&\qquad \geq \sum_{x_i < q^{-1}(K/L)} f(x_i) \cdot (Kp(x_i)/L - x_i) \\
&\qquad = \sum_{x_i < q^{-1}(K/L)} \left(q^{-1}(K/L) \cdot L/K \cdot x_i - x_i\ p(x_i)\right).
\end{aligned}$$

Adding back the terms where $x_i = q^{-1}(K/L)$, which have value 0, and rewriting we obtain

$$\sum_i \left(x_i\ p(x_i) - q^{-1}(K/L) \cdot L/K \cdot x_i\right) \geq 0.$$

Therefore we have

$$\sum_i x_i\ p(x_i) \geq q^{-1}(K/L) \cdot L/K \cdot \sum_i x_i \geq q^{-1}(K/L) \cdot (L/K) \cdot K = q^{-1}(K/L) \cdot L. \qquad \blacktriangleleft$$

Dynamic Averaging Load Balancing on Arbitrary Graphs

Petra Berenbrink
Universität Hamburg, Germany

Lukas Hintze
Universität Hamburg, Germany

Hamed Hosseinpour
Universität Hamburg, Germany

Dominik Kaaser
TU Hamburg, Germany

Malin Rau
Universität Hamburg, Germany

Abstract

In this paper we study dynamic averaging load balancing on general graphs. We consider infinite time and dynamic processes, where in every step new load items are assigned to randomly chosen nodes. A matching is chosen, and the load is averaged over the edges of that matching. We analyze the discrete case where load items are indivisible, moreover our results also carry over to the continuous case where load items can be split arbitrarily. For the choice of the matchings we consider three different models, random matchings of linear size, random matchings containing only single edges, and deterministic sequences of matchings covering the whole graph. We bound the discrepancy, which is defined as the difference between the maximum and the minimum load. Our results cover a broad range of graph classes and, to the best of our knowledge, our analysis is the first result for discrete and dynamic averaging load balancing processes. As our main technical contribution we develop a drift result that allows us to apply techniques based on the effective resistance in an electrical network to the setting of dynamic load balancing.

2012 ACM Subject Classification Theory of computation → Random walks and Markov chains; Mathematics of computing → Stochastic processes; Theory of computation → Distributed algorithms

Keywords and phrases Dynamic Load Balancing, Distributed Computing, Randomized Algorithms, Drift Analysis

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.18

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2302.12201`

Funding *Petra Berenbrink*: Supported by DFG Research Group ADYN (FOR 2975) under grant 411362735, and by DFG grant 427756233.
Hamed Hosseinpour: Supported by DFG grant 427756233.
Malin Rau: Supported by DFG Research Group ADYN (FOR 2975) under grant DFG 411362735.

Acknowledgements We thank the anonymous reviewers for their comments.

1 Introduction

Parallel and distributed computing is ubiquitous in science, technology, and beyond. Key to the performance of a distributed system is the efficient utilization of resources: in order to obtain a substantial speed-up it is of utmost importance that all processors have to handle the same amount of work. Unfortunately, many practical applications such as finite element

simulations are highly "irregular", and the amount of load generated on some processors is much larger than the amount of load generated on others. We therefore investigate *load balancing* to redistribute the load. Efficient load balancing schemes have a plenitude of applications, including high performance computing [34], cloud computing [27], numerical simulations [26], and finite element simulations [29].

In this paper we consider *neighborhood* load balancing on arbitrary graphs with n nodes, where the nodes balance their load in each step only with their direct neighbors. We assume *discrete* load items as opposed to *continuous* (or *idealized*) load items which can be broken into arbitrarily small pieces. We study *infinite* and *dynamic* processes where new load items are generated in every step.

We consider two different settings. In the *synchronous* setting m load items are generated on randomly chosen nodes. Then a matching is chosen and the load of the nodes is balanced (via weighted averaging) over the edges of that matching. Here we further distinguish between two matching models. We consider the *random matching model* where linear-size matchings are randomly chosen, and the *balancing circuit model* where the graph is divided deterministically into $d_{\max}$ many matchings. Here $d_{\max}$ is the maximum degree of any node. In the *asynchronous model* exactly one load item is generated on a randomly chosen node. In turn, the node chooses one of its edges at random and balances its load with the corresponding neighbor. This model can be regarded as a variant of the synchronous model where the randomly chosen matching has size one. It was introduced by [2] where the authors show results for cycles assuming continuous load. Our goal is to bound the so-called *discrepancy*, which is defined as the maximal load of any node minus the minimal load of any node.

The assumption that load items initially arrive at uniformly random nodes is a limitation of the model. However, we believe this to be a good starting point for further investigations into the behavior of load balancing methods in dynamic settings.

Results in a Nutshell. In this paper we present bounds on the expected discrepancy and bounds that hold with high probability for the three models introduced above. Our bounds for the synchronous model with balancing circuits hold for arbitrary graphs G, the bounds for the asynchronous model and the synchronous model with random matchings hold for regular graphs G only. For the asynchronous model and the model with random matchings our bounds on the discrepancy are expressed in terms of hitting times of a standard random walk on G, as well as in terms of the spectral gap of the Laplacian of G. For the synchronous model with balancing circuits we express our bounds in terms of the *global divergence*. This can be thought of as a measure of the convergence speed of the Markov chains modeling a random walk on G. However, it does not directly measure the speed of convergence of the chain. It accounts for the time period in which the chain keeps a given distance from the stationary (and uniform) distribution. In physics terminology, it is a measure of total *absement*, which is the time-integral of displacement.

For all three infinite processes our bounds on the discrepancy hold at an arbitrary point of time as long as the system is initially empty. Otherwise, the bounds hold after an initial time period, its length is a function of the initial discrepancy. In the following we give some exemplary results assuming that the system is initially empty and $m = n$. For the synchronous model with random matchings and the asynchronous model we can bound the discrepancy by $\mathcal{O}(\sqrt{n}\log(n))$ for *any* regular graph G. Our results show a polylogarithmic bound on the discrepancy for all regular graphs with a hitting time at most $\mathcal{O}(n \operatorname{poly}\log(n))$ (e.g., the two-dimensional torus or the hypercube). In all models we can bound the discrepancy

Table 1 Asymptotic upper bounds on the discrepancy in specific graph classes.

Graph	$\text{SBAL}(\mathcal{D}_{\text{RM}}(G), 1, m)$	$\text{SBAL}(\mathcal{D}_{\text{BC}}(G), 1, m)$	$\text{ABAL}(\mathcal{D}_{\text{A}}(G), 1)$
d-regular graph (const. d)	$\log(n) + \sqrt{m \cdot \log(n)}$	$\log(n) + \sqrt{m \cdot \log(n)}$	$\sqrt{n \cdot \log(n)}$
cycle C_n	$\log(n) + \sqrt{m \cdot \log(n)}$	$\log(n) + \sqrt{m \cdot \log(n)}$	$\sqrt{n \cdot \log(n)}$
2-D torus	$\log(n) + \sqrt{m/n} \cdot \log^{3/2}(n)$	$(1 + \sqrt{m/n}) \cdot \log(n)$	$\log^{3/2}(n)$
r-D torus (const. $r \geq 3$)	$(1 + \sqrt{m/n}) \cdot \log(n)$	$\log(n) + \sqrt{m/n \cdot \log(n)}$	$\log(n)$
hypercube	$(1 + \sqrt{m/n}) \cdot \log(n)$	$(1 + \sqrt{m/n}) \cdot \log(n)$	$\log(n)$
expander	$\log n + \sqrt{m/n \cdot \log(n)}$	$\log n + \sqrt{\zeta / \lambda(\mathbf{R})} \cdot \sqrt{m/n \cdot \log n}$	$\log(n)$

by $\mathcal{O}(\sqrt{n \log(n)})$ for arbitrary constant-degree regular graphs. For the full results we refer the reader to Theorem 3.1, Theorem 4.1, and Theorem 5.1. We show an overview of our bounds on the discrepancy for specific graph classes in Table 1. The corresponding results are formally derived and can be found in the full version.

All bounds presented in this paper also hold for the corresponding continuous processes without rounding. The authors of [2] consider the asynchronous process on cycles in the continuous setting where the load items can be divided into arbitrary small pieces. They bound the expected discrepancy, showing that $\mathbb{E}[\text{disc}(G)] = \mathcal{O}(\sqrt{n} \log(n))$ for a cycle G with n nodes. In contrast, we improve that bound for the cycle to $\mathcal{O}(\sqrt{n \log(n)})$, both in expectation and with high probability.

Our main analytical vehicle is a drift theorem that bounds the tail of the sum of a non-increasing sequence of random variables. Our drift theorem adapts known drift results from the literature, similarly to the Variable Drift Theorem in [23].

1.1 Related Work

There is a vast body of literature on iterative load balancing schemes on graphs where nodes are allowed to balance (or average) their load with neighbors only. One distinguishes between *diffusion* load balancing where the nodes balance their load with all neighbors at the same time and the *matching model* (or *dimension exchange*) model where the edges which are used for the balancing form a matching. In the latter model every resource is only involved in one balancing action per step, which greatly facilitates the analysis.

In this overview we only consider theoretical results and, as it is beyond the scope of this work to provide a complete survey, we focus on results for discrete load balancing. For results about continuous load balancing see, for example, [14, 22]. There are also many results in the context of balancing schemes where not the resources try to balance their load but the tokens (acting as selfish players) try to find a resource with minimum load. See [16] for a comprehensive survey about selfish load balancing and [1, 21, 11] for some recent results. Another related topic is token distribution where nodes do not balance their entire load with neighbors but send only single tokens over to neighboring nodes with a smaller load. See [18, 5, 30] for the static setting and [4] for the dynamic setting.

Discrete Models. The authors of [28] give the first rigorous result for discrete load balancing in the diffusion model. They assume that the number of tokens sent along each edge is obtained by rounding down the amount of load that would be sent in the continuous case. Using this approach they establish that the discrepancy is at most $O(n^2)$ after $O(\log(Kn))$ steps, where K is the initial discrepancy. Similar results for the matching model are shown in [19]. While always rounding down may lead to quick stabilization, the discrepancy tends to be quite large, a function of the diameter of the graph. Therefore, the authors of [32] suggest to use randomized rounding in order to get a better approximation of the continuous case. They show results for a wide class of diffusion and matching load balancing protocols and introduce the so-called *local divergence*, which aggregates the sum of load differences over all edges in all rounds. The authors prove that the local divergence gives an upper bound on the maximum deviation between the continuous and discrete case of a protocol. In [17] the authors show several results for a randomized protocol with rounding in the matching model. For complete graphs their results show a discrepancy of $\mathcal{O}(n\sqrt{\log n})$ after $\Theta(\log(Kn))$ steps. Later, [7] extended some of these results to the diffusion model. In [33] the authors show that the number of rounds needed to reach constant discrepancy is w.h.p. bounded by a function of the spectral gap of the relevant mixing matrix and the initial discrepancy. In [8] the authors propose a very simple potential function technique to analyze discrete diffusion load balancing schemes, both for discrete and continuous settings. In [9] the authors investigate a load balancing process on complete graphs. In each round a pair of nodes is selected uniformly at random and completely balance their loads up to a rounding error of ± 1.

The authors of [12] study load balancing via matchings assuming random placement of the load items. The initial load distribution is sampled from exponentially concentrated distributions (including the uniform, binomial, geometric, and Poisson distributions). The authors show that in this setting the convergence time is smaller than in the worst case setting. Regardless of the graph's topology, the discrepancy decreases by a factor of $\sqrt[4]{t}$ within t synchronous rounds. Their approach of using concentration inequalities to bound the discrepancy (in terms of the squared 2-norm of the columns of the matrices underlying the mixing process) strongly influenced our approach.

Dynamic Models. There are far fewer results for the *dynamic* diffusion models where new loads enter the system over time. In [2] the authors study a model similar to our asynchronous model. In each step one load item is allocated to a chosen node. In the same step, the chosen node picks a random neighbor, and the two nodes balance their loads by averaging them (continuous model). The authors show that the expected discrepancy is bounded by $\mathcal{O}(\sqrt{n}\log n)$, as well as a lower bound on the square of the discrepancy of $\Omega(n)$. The authors of [3] consider load balancing via matchings in a dynamic model where the load is, in every step, distributed by an adversary. They show the system is stable for sufficiently limited adversaries. They also give some upper bounds on the maximum load for the somewhat more restricted adversary. The authors of [10] consider discrete dynamic diffusion load balancing on arbitrary graphs. In each step up to n load items are generated on arbitrary nodes (the allocation is determined by an adversary). Then the nodes balance their load with each neighbor and finally one load item is deleted from every non-empty node. The authors show that the system is stable, which means that the total load remains bounded over time (as a function of n alone and independently of the time t).

In the *graphical balanced allocations* setting, the initial allocation of a load item is constrained to a randomly chosen edge of a graph, but load items cannot be moved after allocation (in contrast to our setting). For d-regular graphs, Peres, Talwar, and Wieder [31]

show that for the greedy algorithm which allocates the load item to the lower-loaded node at the edge, with the edge distribution being uniform, the discrepancy is in $\mathcal{O}(\log(n)d/\varepsilon)$ with high probability, where ε is the edge expansion of the graph. In fact, they show a more general result in terms of distributions over arbitrary subsets of nodes. Furthermore, Bansal and Feldheim [6] give a non-greedy algorithm using some non-local information that achieves a discrepancy in $\mathcal{O}((d/k)\log^4(n)\log(\log(n)))$ for k-edge-connected d-regular graphs, as well as a lower bound for the graphical balanced allocation setting stating that the discrepancy is in $\Omega(d/k + \log(n))$ with constant probability at any given time for any allocation strategy.

2 Balancing Models and Notation

We consider the following class of dynamic load balancing processes on d-regular graphs G with n nodes $V(G) = [n]$. Each process is modeled by a Markov chain $(\vec{X}(t))_{t\in\mathbb{N}_0}$, where the *load vector* $\vec{X}(t) = (X_i(t))_{i\in[n]} \in \mathbb{R}^n$ is the *state* of the process at the end of step t, and $X_i(t)$ is the load of node i at time t. We measure a load vector's imbalance by the discrepancy $\operatorname{disc}(\vec{x})$, which is the difference between the maximum load and the minimum load $\operatorname{disc}(\vec{x}) := \max_{i\in[n]} x_i - \min_{j\in[n]} x_j$.

We consider two balancing processes, the synchronous process SBal and the asynchronous process ABal. Both processes are parameterized by a *balancing parameter* β determining the balancing speed and a matching distribution $\mathcal{D}(G)$. For SBal, $\mathcal{D}(G)$ is a distribution over linear-sized matchings of G. For ABal, $\mathcal{D}(G)$ is a distribution over edges of G. SBal is additionally parameterized by the number of load items $m \in \mathbb{N}^+$ allocated in each round. ABal allocates only one new load item per step.

Synchronous Processes. The synchronous process $\text{SBal}(\mathcal{D}(G), \beta, m)$ works as follows. The process first allocates m items to randomly chosen nodes. Then it uses the matching distribution $\mathcal{D}(G)$ to determine the matching which is applied. Finally it balances the load over the edges of the matching (see Process $\text{Bal}(\mathbf{m}, \beta)$ described below). The parameter $\beta \in (0, 1]$ controls the fraction of the load difference that is sent over an edge in a step.

For the synchronous process SBal we consider two families of matching distributions, random matchings $(\mathcal{D}_{\text{RM}}(G))$ and balancing circuits $(\mathcal{D}_{\text{BC}}(G))$. $\mathcal{D}_{\text{RM}}(G)$ is generated according to the following method described in [19]. In essence, in a first step, nodes mark edges independently with probability $1/(8d)$, so that each edge is marked independently with probability $1/(4d) - 1/(16d^2) = \Theta(1/d)$ (as it can be marked from either end); in a second step, marked edges which are not incident to any other marked edge are selected for the matching. In expectation, the resulting matching has a size which is linear in the number of nodes; we refer to [19] for a more detailed description.

We will use capital $\mathbf{M}$ for randomly chosen matchings. The analysis for the random matching model can be found in Section 3. In the *balancing circuit model* we assume G is covered by ζ fixed matchings $\mathbf{m}(1), \ldots, \mathbf{m}(\zeta)$. $\mathcal{D}_{\text{BC}}(G)$ deterministically chooses matchings in periodic manner such that in step t the matching $\mathbf{m}(t) = \mathbf{m}(t \bmod \zeta)$ is chosen. We will use small $\mathbf{m}$ for deterministically chosen matchings. The analysis for the balancing circuit model can be found in Section 4.

Asynchronous Process. The asynchronous process $\text{ABal}(\mathcal{D}(G), \beta)$ works as follows. The process first uses $\mathcal{D}(G)$ to generate a matching, this time containing one edge only. The distribution we consider, $\mathcal{D}_{\text{A}}(G)$, first chooses a node i uniformly at random and then it chooses one of the nodes' edges (i, j) uniformly at random. Finally one new token is assigned

to either node i or j and then the edge (i, j) is used for balancing (see $\textsc{Bal}(\mathbf{m}, \beta)$). Note that for $\textsc{ABal}(\mathcal{D}_{\mathrm{A}}(G), \beta)$ the load allocation heavily depends on the edges which are used for balancing. This makes the analysis for this model quite challenging. In contrast, in $\textsc{SBal}(\mathcal{D}_{\mathrm{A}}(G), \beta, m)$ the load allocation and the balancing are independent. Note that in the case of d-regular graphs $\mathcal{D}_{\mathrm{A}}(G)$ is equivalent to the uniform distribution over all edges or to choosing a random matching of size one. We analyze the asynchronous model in Section 5.

$\textsc{SBal}(\mathcal{D}(G), \beta, m)$: In each round $t \in \mathbb{N}^+$:

1. Allocate m discrete, unit-sized load items to the nodes uniformly and independently at random. Define $\ell_i(t)$ as the number of tokens assigned to node i.
2. Sample a matching $\mathbf{M}(t)$ according to $\mathcal{D}(G)$.
3. Balance with $\textsc{Bal}(\mathbf{M}(t), \beta)$ applied to $X_i(t) := X_i(t) + \ell_i(t)$, $i \in \{1, \dots n\}$.

$\textsc{ABal}(\mathcal{D}(G), \beta)$: In each round $t \in \mathbb{N}^+$:

1. Select an edge $\{i, j\}$ according to $\mathcal{D}(G)$.
2. Allocate a single unit-size load item to either node i or j with a probability of $1/2$. I.e., with prob. $1/2$ set $\ell_i(t) = 1$ and $\ell_k = 0$ for all $k \neq i$, otherwise set $\ell_j(t) = 1$ and $\ell_k = 0$ for all $k \neq j$.
3. Balance with $\textsc{Bal}(\mathbf{M}(t), \beta)$ applied to $X_i(t) := X_i(t) + \ell_i(t)$, where $\mathbf{M}(t)$ includes just the edge $\{i, j\}$.

$\textsc{Bal}(\mathbf{m}, \beta)$: For each edge $\{i, j\}$ in the matching $\mathbf{m}$ balance loads of i and j:

1. Assume w.l.o.g. that $X_i(t) \geq X_j(t)$.
2. Let $p = \frac{\beta \cdot (X_i(t) - X_j(t))}{2} - \left\lfloor \frac{\beta \cdot (X_i(t) - X_j(t))}{2} \right\rfloor$.
3. Then, node i sends $L_{i,j}$ load items to node j where

$$L_{i,j} := \begin{cases} \left\lceil \frac{\beta \cdot (X_i(t) - X_j(t))}{2} \right\rceil, & \text{with probability } p, \\ \left\lfloor \frac{\beta \cdot (X_i(t) - X_j(t))}{2} \right\rfloor, & \text{with probability } 1 - p. \end{cases}$$

In the idealized setting, where the load is continuously divisible, a load of $\beta(X_i(t) - X_j(t))/2$ is sent from node i to node j.

2.1 Notation

We are given an arbitrary graph $G = (V, E)$ with n nodes. We mainly assume that G is regular and write d for the node degree. Recall that the process is modeled by a Markov chain $(\vec{X}(t))_{t \in \mathbb{N}}$, where $\vec{X}(t) = (X_i(t))_{i \in [n]} \in \mathbb{R}^n$ is the *load vector* at the end of step t, and $X_i(t)$ is the load of node i at time t. We write $\ell_i(t)$ for the number of load items allocated to node i in step t and define $\vec{\ell}(t) = (\ell_i(t))_{i \in [n]}$. We will use upper case letters such as $X_i(t)$ and $\mathbf{M}(t)$ to denote random variables and random matrices and lower case letters (like $x_i(t)$, $\mathbf{m}(t)$) for fixed outcomes. If clear from the context we will omit t from a random variable.

We model the idealized balancing step in round t by multiplication with a matrix $\mathbf{M}^{\beta}(t) \in \mathbb{R}^{n \times n}$ given by

$$\mathbf{M}^{\beta}_{i,j}(t) := \begin{cases} 1, & \text{if } i = j \text{ and } i \text{ is not matched at time } t, \\ 1 - \beta/2, & \text{if } i = j \text{ and } i \text{ is matched at time } t, \\ \beta/2, & \text{if } i \text{ and } j \text{ are matched at time } t, \\ 0, & \text{otherwise.} \end{cases}$$

We will omit the parameter β if it is clear from context. With slight abuse of notation we use the same symbol $\mathbf{M}(t)$ for the matching itself and the associated balancing matrix and refer to both as just "matchings". Furthermore, we write $E(\mathbf{M}(t))$ for their edges. For the product of all matching matrices from time t_1 to time t_2 we write

$$\mathbf{M}^{[t_1,t_2]} \coloneqq \mathbf{M}(t_2) \cdot \mathbf{M}(t_2-1) \cdot \dots \cdot \mathbf{M}(t_1+1) \cdot \mathbf{M}(t_1),$$

where for $t_1 > t_2$ we consider this to be the identity matrix. We generally refer to these matrices as *mixing matrices*. Moreover, we write $\mathbf{M}^{[t]}$ for the sequence of matching matrices $(\mathbf{M}(\tau))_{\tau\in[t]}$ and analogously $\mathbf{m}^{[t]}$ for a fixed sequence of matching matrices $(\mathbf{m}(\tau))_{\tau\in[t]}$. We will write $\mathbf{M}_{k,\cdot}$ for the vector forming the kth row of the matrix $\mathbf{M}$ (which we often treat as a column vector despite it being a row).

In the balancing circuit model we define the *round matrix* $\mathbf{R} \coloneqq \mathbf{m}^{[1,\zeta]}$ as the product of the matching matrices forming a complete period of the balancing circuit. Note that ζ has no relation to the minimum or maximum degree, although we may assume w.l.o.g. that each edge is covered by at least one of the matchings. We write $\lambda(\mathbf{R})$ for the spectral gap of the round matrix $\mathbf{R}$, i.e., for the difference between the largest two eigenvalues of $\mathbf{R}$.

We write $\vec{\varepsilon}(t) \in \mathbb{R}^n$ for the vector of additive rounding errors in round t. Then $\varepsilon_k(t)$ is the difference between the load at node k after step t and the load at node k after step t in an idealized scheme where loads are arbitrarily divisible.

Putting all of this together we can express the load vector at the end of step $t \in \mathbb{N}^+$ as

$$\vec{X}(t) = \mathbf{M}(t) \cdot \left(\vec{X}(t-1) + \vec{\ell}(t)\right) + \vec{\varepsilon}(t). \tag{1}$$

We write $\mathrm{t_{hit}}(\mathrm{G})$ for the *hitting time* of G, which is the maximum expected time it takes for a standard random walk on G (i.e., the walk moves to a neighbor chosen uniformly at random in each step) to reach a given node i from a given node j, with the maximum taken over all such pairs of nodes. We write $\mathrm{t^*_{hit}}(\mathrm{G})$ for the *edge hitting time* of G, which is defined like the hitting time, except that the maximum is taken over adjacent nodes only. We write $\mathbf{L}(G)$ for the normalized Laplacian matrix of a graph G. For regular graphs it may be defined as $\mathbf{L}(G) \coloneqq \mathbf{I} - \mathbf{A}(G)/d$, where $\mathbf{A}(G)$ is the adjacency matrix of G. Writing $\lambda_0 \le \lambda_1 \le \ldots \le \lambda_{n-1}$ for the real eigenvalues of $\mathbf{L}(G)$, we let $\lambda(\mathbf{L}(G)) \coloneqq \lambda_1 - \lambda_0$ be the spectral gap of the Laplacian of G.

3 Random Matching Model

In this section we analyze the process $\textsc{SBal}(\mathcal{D}_{\mathrm{RM}}(G), \beta, m)$ for d-regular graphs G, where the matching distribution $\mathcal{D}_{\mathrm{RM}}(G)$ is generated by the algorithm given in [19]. Note that the result (as well as the results for the two other models) holds at any point of time t if the system is initially empty. Furthermore, we can show the same results in the idealized setting where load items can be divided into arbitrarily small pieces (see [2]). For more details we refer the reader to the paragraph directly after Equation (3).

▶ **Theorem 3.1.** *Let G be a d-regular graph and define $T(G) \coloneqq \min\left\{\frac{\mathrm{t_{hit}}(\mathrm{G})}{n}\cdot\log(n), \sqrt{\frac{d}{\lambda(\mathbf{L}(G))}}, \frac{1}{\lambda(\mathbf{L}(G))}\right\}$. Let $\vec{X}(t)$ be the state of process $\mathrm{SBAL}(\mathcal{D}_{RM}(G), \beta, m)$ at time t with $\mathrm{disc}(\vec{X}(0)) =: K \ge 1$. There exists a constant $c > 0$ such that for all $t \ge c \cdot \log(K \cdot n)/(\lambda(\mathbf{L}(G)) \cdot \beta)$ it*

holds w.h.p.[1] *and in expectation*

$$\operatorname{disc}(\vec{X}(t)) = \mathcal{O}\left(\log(n)\cdot\left(1+\sqrt{\frac{m}{n}\cdot\frac{\mathrm{t}^*_{\mathrm{hit}}(\mathrm{G})}{n}}\right)+\sqrt{\frac{\log(n)}{\beta}\cdot\frac{m}{n}\cdot T(G)}\right).$$

Proof. We first expand the recurrence of Equation (1) (cf. [32]). After one step we get

$$\begin{aligned}\vec{X}(t) &= \mathbf{M}(t)\cdot\left(\vec{X}(t-1)+\vec{\ell}(t)\right)+\vec{\varepsilon}(t)\\ &= \mathbf{M}(t)\cdot\Big(\underbrace{\left(\mathbf{M}(t-1)\cdot\left(\vec{X}(t-2)+\vec{\ell}(t-1)\right)+\vec{\varepsilon}(t-1)\right)}_{\vec{X}(t-1)}+\vec{\ell}(t)\Big)+\vec{\varepsilon}(t)\\ &= \mathbf{M}^{[t-1,t]}\cdot\vec{X}(t-2)+\sum_{\tau=t-1}^{t}\mathbf{M}^{[\tau,t]}\cdot\vec{\ell}(\tau)+\sum_{\tau=t-1}^{t}\mathbf{M}^{[\tau+1,t]}\cdot\vec{\varepsilon}(\tau)\end{aligned}$$

We repeatedly expand this form up to the beginning of the process and get

$$\vec{X}(t) = \underbrace{\mathbf{M}^{[1,t]}\cdot\vec{X}(0)}_{\vec{I}(t)}+\underbrace{\sum_{\tau=1}^{t}\mathbf{M}^{[\tau,t]}\cdot\vec{\ell}(\tau)}_{\vec{D}(t)}+\underbrace{\sum_{\tau=1}^{t}\mathbf{M}^{[\tau+1,t]}\cdot\vec{\varepsilon}(\tau)}_{\vec{R}(t)}. \tag{2}$$

We write $\vec{I}(t)$, $\vec{D}(t)$, and $\vec{R}(t)$ for the three terms as indicated. Note that in general these terms are vectors of real numbers. The sum $\vec{I}(t)+\vec{D}(t)$ can be regarded as the contribution of an idealized process, where $\vec{I}(t)$ is the contribution of the initial load and $\vec{D}(t)$ is the contribution of the dynamically allocated load. Thus, $\vec{R}(t)$ is the deviation between the idealized process without rounding and the discrete process described in Section 2.

To bound the discrepancy $\operatorname{disc}(\vec{X}(t))$ of the load vector $\vec{X}(t)$ at time t, we use the fact that the discrepancy is sub-additive, i.e., that $\operatorname{disc}(\vec{x}+\vec{y})\le\operatorname{disc}(\vec{x})+\operatorname{disc}(\vec{y})$. Hence, to bound $\operatorname{disc}(\vec{X}(t))$, we individually bound the discrepancies of the three terms in Equation (2) and get

$$\operatorname{disc}(\vec{X}(t)) \le \operatorname{disc}(\vec{I}(t))+\operatorname{disc}(\vec{D}(t))+\operatorname{disc}(\vec{R}(t)). \tag{3}$$

If the system is initially empty, then $\operatorname{disc}(\vec{I}(t))=0$. Moreover, in the idealized setting without rounding $\operatorname{disc}(\vec{R}(t))=0$. Techniques to bound the first term $\operatorname{disc}(\vec{I}(t))$ and the last term $\operatorname{disc}(\vec{R}(t))$ are well-established. We state the corresponding results in Lemma 3.2 and Lemma 3.3 directly below the proof of our theorem. The main part of the proof is to bound $\operatorname{disc}(\vec{D}(t))$, which will be done in Section 3.1.

Let now $\gamma>1$. First, it follows from Lemma 3.2 that for all $t\ge c\cdot\log(K\cdot n)/(\lambda(\mathbf{L}(G))\cdot\beta)$ we have $\operatorname{disc}(\vec{I}(t))\le 1$ with probability at least $1-n^{-\gamma}$. Second, it follows from Lemma 3.4 that $\operatorname{disc}(\vec{R}(t))\le 2\sqrt{\gamma\log(n)/\beta}$ with probability at least $1-3\cdot n^{-\gamma+1}$. Third, it follows from Lemma 3.3 that

$$\operatorname{disc}(\vec{D}(t)) = \mathcal{O}\left(\gamma\log(n)\cdot\left(1+\sqrt{\frac{m}{n}\cdot\frac{\mathrm{t}^*_{\mathrm{hit}}(\mathrm{G})}{n}}\right)+\sqrt{\frac{\gamma\log(n)}{\beta}\cdot\frac{m}{n}\cdot T(G)}\right)$$

with probability at least $1-2\cdot n^{-\gamma+1}$. The statement of the theorem therefore follows from a union bound over the statements of Lemma 3.2, Lemma 3.3, and Lemma 3.4. The bound on expectation follows analogously from the linearity of expectation and the bounds on the expected discrepancies in the aforementioned lemmas. ◀

[1] The expression *with high probability (w.h.p.)* denotes a probability of at least $1-n^{-\Omega(1)}$.

Intuitively, Lemma 3.2 states that the contribution of the initial load to the discrepancy is insignificant if t is large enough. We generalize the analysis of Theorem 1 [32] (or Theorem 2.9 in [33]) to establish a bound on the discrepancy of the initial load as a function of β. We prove it in the full version.

▶ **Lemma 3.2** (Memorylessness Property). *Let G be a d-regular graph. Let $K = \operatorname{disc}(\vec{X}(0))$. Then there exists a constant $c > 0$ such that for all $\gamma > 0$ and $t \in \mathbb{N}$ with $t \geq t_0(\gamma) := c \cdot \max\{\gamma \log(n), \log(K \cdot n)\} \cdot \frac{1}{\lambda(\mathbf{L}(G)) \cdot \beta}$ we get with probability at least $1 - n^{-\gamma}$ and in expectation*

$$\operatorname{disc}(\vec{I}(t)) \leq 1.$$

The next lemma bounds $\operatorname{disc}(\vec{R}(t))$, the discrepancy contribution of cumulative rounding errors. Note that this result does not just hold for the random matching model, but for all the three models that we consider in this paper. In the proof of the lemma we extend then results of Theorem 3.6 in [33] (which is based on work in [7]) to establish a bound as a function of β. We prove it in the full version.

▶ **Lemma 3.3** (Insignificance of Rounding Errors). *Let G be an arbitrary graph. Then for all $\gamma > 1$, $t \in \mathbb{N}$, and $k \in [n]$ we get with probability at least $1 - 2n^{-\gamma+1}$ and in expectation*

$$\operatorname{disc}(\vec{R}(t)) \leq 2 \cdot \sqrt{\gamma \log(n)/\beta}.$$

To bound $\operatorname{disc}(\vec{D}(t))$, the discrepancy contribution of dynamically allocated load items we apply the next lemma. It is in fact the core of our work. We prove it in Section 3.1.

▶ **Lemma 3.4** (Contribution of Dynamically Allocated Load). *Let G be a d-regular graph. Define $T(G) := \min\left\{\mathrm{t}_{\mathrm{hit}}(\mathrm{G}) \cdot \log n/n, \sqrt{d/\lambda(\mathbf{L}(G))}, 1/\lambda(\mathbf{L}(G))\right\}$. Then for all $\gamma > 1$ and $t \in \mathbb{N}$ we get with probability at least $1 - 3n^{-\gamma+1}$ and in expectation*

$$\operatorname{disc}(\vec{D}(t)) = \mathcal{O}\left(\gamma \log(n) \cdot \left(1 + \sqrt{\frac{m}{n} \cdot \frac{\mathrm{t}^*_{\mathrm{hit}}(\mathrm{G})}{n}}\right) + \sqrt{\frac{\gamma \log(n)}{\beta} \cdot \frac{m}{n} \cdot T(G)}\right).$$

3.1 Bounding the Contribution of Dynamically Allocated Load

In this section we prove Lemma 3.4. Some of the proofs are omitted and can be found in full version. As a first step, we bound $\operatorname{disc}(\vec{D}(t))$ using the *global divergence* $\Upsilon(\mathbf{M}^{[t]})$, which is defined over a sequence of matching matrices $\mathbf{M}^{[t]}$ as

$$\Upsilon(\mathbf{M}^{[t]}) := \max_{k \in [n]} \Upsilon_k(\mathbf{M}^{[t]}), \quad \text{where} \quad \Upsilon_k(\mathbf{M}^{[t]}) := \sqrt{\sum_{\tau=1}^{t} \left\| \mathbf{M}^{[\tau,t]}_{k,\cdot} - \frac{\vec{1}}{n} \right\|_2^2}.$$

The global divergence can be regarded as a measure of the convergence speed of a random walk that uses the matching matrices as transition probabilities. In [17, 33, 7] the authors use a related notion which they call the *local p-divergence*, also defined on a sequence of matchings $\mathbf{m}^{[t]}$. The difference lies in the fact that the global divergence, essentially, measures differences between nodes' values and a global average, while the local divergence measures differences between neighboring nodes. To show Lemma 3.4 we first observe the following.

▶ **Observation 3.5.** *It holds that* $\operatorname{disc}(\vec{D}(t)) \leq 2 \cdot \max_{k \in [n]} |D_k(t) - t \cdot m/n|$.

Next we consider a fixed node k and show a concentration inequality on $D_k(t)$ in terms of $\Upsilon_k(\mathbf{m}^{[t]})$, where $\mathbf{m}^{[t]}$ is the sequence of matchings applied by our process (Lemma 3.6). Note that in the lemma we assume the matchings are fixed and the randomness is due to the random load placement only. Hence, the lemma directly applies to $\mathcal{D}_{\mathrm{BC}}(G)$. Afterwards, we bound the global divergence of the random sequence of matchings, $\Upsilon_k(\mathbf{M}^{[t]})$ in terms of a notion of "goodness" of the used matching distribution $\mathcal{D}$, for the random sequence of matchings (Lemma 3.9), and then bound the "goodness" of the distribution $\mathcal{D}_{\mathrm{RM}}(G)$ used in the random matching model (Lemma 3.10). We start with a bound on the deviation of $D_k(t)$ from the average load $t \cdot m/n$ in terms of $\Upsilon(\mathbf{m}^{[t]})$.

▶ **Lemma 3.6** (Load Concentration). *Let $\mathbf{m}^{[t]}$ be an arbitrary sequence of matchings. Then for all $\gamma > 0$, $t \in \mathbb{N}$, and $k \in [n]$ we get with probability at most $2 \cdot n^{-\gamma}$*

$$\left| D_k(t) - t \cdot \frac{m}{n} \right| \geq \frac{4}{3} \cdot \gamma \log(n) + \sqrt{8\gamma \log(n) \cdot \frac{m}{n}} \cdot \Upsilon_k(\mathbf{m}^{[t]}).$$

Proof. Our goal is to decompose $D_k(t)$ into a sum of independent random variables. Recall that we assume that the matching matrices are fixed and all randomness is due to the random choices of the load items. This will enable us to apply a concentration inequality to this sum. For the decomposition observe that $\vec{D}(t) = \sum_{\tau=1}^{t} \mathbf{m}^{[\tau,t]} \cdot \vec{\ell}(\tau)$, where $\vec{\ell}(\tau)$ is the random load vector corresponding to the m load items allocated at time τ. So the kth coordinate of $\vec{D}(t)$ is $D_k(t) = \sum_{\tau=1}^{t} \sum_{w \in [n]} \mathbf{m}_{k,w}^{[\tau,t]} \cdot \ell_w(\tau)$. We define the indicator random variable $B(\tau, j, w)$ for $\tau \in [t], j \in [m]$ and $w \in [n]$ as

$$B(\tau, j, w) \coloneqq \begin{cases} 1, & \text{if the } j\text{-th load item of step } \tau \text{ is allocated to node } w, \\ 0, & \text{otherwise.} \end{cases}$$

Note that for fixed τ and j we have $\sum_{w \in [n]} B(\tau, j, w) = 1$, $\mathbb{P}[B(\tau, j, w) = 1] = 1/n$ and $\mathbb{E}[B(\tau, j, w)] = 1/n$. Observe that $\ell_w(\tau)$, the load allocated to node w at step τ, can be expressed as $\sum_{j \in [m]} B(\tau, j, w)$. Merging this with the value of $D_k(t)$ gives

$$D_k(t) = \sum_{\tau=1}^{t} \sum_{w \in [n]} \mathbf{m}_{k,w}^{[\tau,t]} \cdot \left(\sum_{j \in [m]} B(\tau, j, w) \right) = \sum_{\tau=1}^{t} \sum_{j \in [m]} \underbrace{\left(\sum_{w \in [n]} \left(\mathbf{m}_{k,w}^{[\tau,t]} \cdot B(\tau, j, w) \right) \right)}_{=: C_k(\tau, j)}.$$

For a fixed $\tau \in [t]$ and $j \in [m]$ we define $C_k(\tau, j) \coloneqq \sum_{w \in [n]} \mathbf{m}_{k,w}^{[\tau,t]} \cdot B(\tau, j, w)$. This random variable measures the contribution of j-th load item of round τ to $D_k(t)$. Note that the load items are allocated independently from each other. Since $\mathbf{m}^{[\tau,t]}$ are fixed matrices, then $C_k(\tau, j)$ and $C_k(\tau', j')$ are independent for all τ and τ' and $j \neq j'$. To apply the concentration inequality Theorem 3.4 in [13], we need to show that $C_k(\tau, j) \leq 1$ and compute an upper bound on $\mathrm{Var}[C_k(\tau, j)]$. Showing the first condition is easy since exactly one of the indicator random variables $B(\tau, j, w)$ is one and $\mathbf{m}_{k,w}^{[\tau,t]}$ has a value between zero and one.

It remains to consider the variance of $C_k(\tau, j)$. First note that by linearity of expectation

$$\mathbb{E}[C_k(\tau, j)] = \mathbb{E}\left[\sum_{w \in [n]} \left(\mathbf{m}_{k,w}^{[\tau,t]} \cdot B(\tau, j, w) \right) \right] = \sum_{w \in [n]} \mathbf{m}_{k,w}^{[\tau,t]} \cdot \mathbb{E}[B(\tau, j, w)] = \sum_{w \in [n]} \mathbf{m}_{k,w}^{[\tau,t]} \cdot \frac{1}{n} = \frac{1}{n},$$

where the last equality follows form the fact that $\mathbf{m}^{[\tau,k]}$ is doubly stochastic. Now we get

$$\begin{aligned}\mathrm{Var}[C_k(\tau,j)] &= \mathbb{E}\Big[(C_k(\tau,j)-\mathbb{E}[C_k(\tau,j)])^2\Big] = \mathbb{E}\left[\Big(\Big(\sum_{w\in[n]} \mathbf{m}^{[\tau,t]}_{k,w}\cdot B(\tau,j,w)\Big)-\frac{1}{n}\Big)^2\right]\\ &= \sum_{w'\in[n]}\frac{1}{n}\cdot\left(\mathbf{m}^{[\tau,t]}_{k,w'}-\frac{1}{n}\right)^2 = \frac{1}{n}\cdot\left\|\mathbf{m}^{[\tau,t]}_{k,\cdot}-\frac{\vec{1}}{n}\right\|_2^2,\end{aligned}$$

where we used that for each τ and each j exactly one of the $B(\tau,j,w)$ is one and all others are zero, and each of the n possible cases has uniform probability.

Recall that $C_k(\tau,j)$ and $C_k(\tau',j')$ are independent for all τ,τ' and $j\neq j'$. Hence we get

$$\begin{aligned}\mathrm{Var}\left[\sum_{\tau=1}^{t}\sum_{j\in[m]}C_k(\tau,j)\right] &= \sum_{\tau=1}^{t}\sum_{j\in[m]}\mathrm{Var}[C_k(\tau,j)] = \frac{1}{n}\cdot\sum_{\tau=1}^{t}\sum_{j\in[m]}\left\|\mathbf{m}^{[\tau,t]}_{k,\cdot}-\frac{\vec{1}}{n}\right\|_2^2\\ &= \frac{m}{n}\cdot\Big(\Upsilon_k(\mathbf{m}^{[t]})\Big)^2,\end{aligned}$$

where the final equality uses the definition of the global divergence $\Upsilon_k(\mathbf{m}^{[t]})$. Applying Theorem 3.4 in [13] with $M=1$ and $X=D_k(t)=\sum_{\tau=1}^{t}\sum_{j\in[m]}C_k(\tau,j)$ with $\lambda = 2\gamma\log(n)/3+\Upsilon_k(\mathbf{m}^{[t]})\cdot\sqrt{2\gamma m/n}$ results in

$$\mathbb{P}\left[D_k(t)-t\cdot\frac{m}{n}\geq\frac{2}{3}\cdot\gamma\log(n)+\sqrt{2\gamma\log(n)\cdot\frac{m}{n}}\cdot\Upsilon_k(\mathbf{m}^{[t]})\right]\leq n^{-\gamma}.$$

The lower bound can be established using Theorem 4.1 in [13] (with $a_i=0$ and $M=1$). Via a union bound we get

$$\mathbb{P}\left[\left|D_k(t)-t\cdot\frac{m}{n}\right|\geq\frac{4}{3}\cdot\gamma\log(n)+\sqrt{8\gamma\log(n)\cdot\frac{m}{n}}\cdot\Upsilon_k(\mathbf{m}^{[t]})\right]\leq 2\cdot n^{-\gamma}.$$ ◀

To bound the global divergence of the matching sequence used by the process we use two potential functions. The *quadratic node potential* $\Phi(\vec{x})$ is given by

$$\Phi(\vec{x}) \coloneqq \sum_{i\in[n]}(x_i-\overline{x})^2,\quad \text{where}\quad \overline{x}\coloneqq\frac{1}{n}\cdot\sum_{j\in[n]}x_j.$$

For a set of edges S on the nodes $[n]$ and a vector $\vec{x}\in\mathbb{R}^n$, the *quadratic edge potential* is

$$\Psi_S(\vec{x}) \coloneqq \sum_{\{i,j\}\in S}(x_i-x_j)^2.$$

We may also write $\Psi_G\coloneqq\Psi_{E(G)}$ whenever G is a graph, and $\Psi_{\mathbf{M}}\coloneqq\Psi_{E(\mathbf{M})}$ whenever $\mathbf{M}$ is a matching matrix. The following observation relates the drop of node potential to the edge potential in terms of β.

▶ **Observation 3.7.** *Let $\mathbf{M}^\beta$ be a matching matrix with parameter $\beta\in(0,1]$. Then for any $\vec{x}\in\mathbb{R}^n$ we have $\Phi(\vec{x})-\Phi(\mathbf{M}^\beta\cdot\vec{x}) = \frac{1-(1-\beta)^2}{2}\cdot\Psi_{E(\mathbf{M}^\beta)}(\vec{x})$.*

We now define a notion of a matching distribution being *good*. In Lemma 3.9 below we show that the notion is sufficient for showing that matching sequences generated from such distributions have bounded global divergence. Note that the "goodness" of a distribution does not depend on β but on graph properties and the random choices with which the matchings are chosen. Hence, we assume $\beta=1$.

▶ **Definition 3.8.** *Assume G is an arbitrary d-regular graph. Let $g\colon \mathbb{R}_0^+ \to \mathbb{R}^+$ be an increasing function and let $\sigma^2 > 1$. Then a matching distribution $\mathcal{D}(G)$ is (g, σ^2)-*good *if the following conditions hold for $\mathbf{M}^1 \sim \mathcal{D}(G)$ and all stochastic vectors $\vec{x} \in \mathbb{R}^n$.*

1. $\Phi(\vec{x}) - \mathbb{E}[\Phi(\mathbf{M}^1 \cdot \vec{x})] \geq g(\Phi(\vec{x}))$.
2. $\mathrm{Var}[\Phi(\mathbf{M}^1 \cdot \vec{x})] \leq (\sigma^2 - 1) \cdot \big(\Phi(\vec{x}) - \mathbb{E}[\Phi(\mathbf{M}^1 \cdot \vec{x})]\big)^2$.

It remains to show two results. First, assuming a matching distribution is (g, σ^2)-good, the global divergence of a matching sequence generated by that distribution can be bounded in terms of g and σ (Lemma 3.9). Second, we have to calculate a function g_G and the values of σ_G for which the matching distribution $\mathcal{D}_{\mathrm{RM}}(G)$ is (g_G, σ_G^2)-good (see Lemma 3.10).

▶ **Lemma 3.9** (Global Divergence)**.** *Assume G is an arbitrary graph. Let $g\colon \mathbb{R}_0^+ \to \mathbb{R}^+$ be an increasing function, $\sigma^2 > 1$, and $\beta \in (0, 1]$. Let $\mathbf{M}^{[t]} = (\mathbf{M}^\beta(\tau))_{\tau=1}^t$ be an i.i.d. sequence of matching matrices generated by $\mathcal{D}(G)$ and assume $\mathcal{D}(G)$ is a (g, σ^2)-good matching distribution. Then for all $\gamma > 0$ and $k \in [n]$ we get with probability at least $1 - n^{-\gamma}$*

$$\Big(\Upsilon_k(\mathbf{M}^{[t]})\Big)^2 \leq 8\sigma^2(\gamma \log(n) + \log(8\sigma^2)) + \frac{2}{\beta} \cdot \int_0^1 \frac{x}{g(x)}\,\mathrm{d}x.$$

▶ **Lemma 3.10.** *Assume G is an arbitrary d-regular graph. Let*

$$g_G(x) \coloneqq \frac{1}{16d} \cdot \max\left\{d \cdot \lambda(\mathbf{L}(G)) \cdot x, \frac{x^2}{\mathrm{Res}(G)}, \frac{4}{27} \cdot x^3\right\} \text{ and } \sigma_G^2 = 32 \cdot (\mathrm{t}^*_{\mathrm{hit}}(\mathrm{G})\,/n) + 5.$$

Then $\mathcal{D}_{RM}(G)$ is (g_G, σ_G^2)-good.

Proof. First, note that the function $g_G(x)$ is increasing in x. Applying the first part of Lemma 3.11 (see below) we get that for any vector $\vec{x} \in \mathbb{R}^n$ it holds that

$$\Phi(\vec{x}) - \mathbb{E}\big[\Phi(\mathbf{M}^1 \cdot \vec{x})\big] \geq \frac{1}{16d} \cdot \Psi_G(\vec{x}).$$

From the first two statements of Lemma 3.12 (stated behind Lemma 3.12) we see that for $\mathbf{M}^1 \sim \mathcal{D}_{\mathrm{RM}}(G)$ and all stochastic vectors $\vec{x} \in \mathbb{R}^n$

$$\Psi_G(\vec{x}) \geq \max\left\{d \cdot \lambda(\mathbf{L}(G)) \cdot \Phi(\vec{x}), \frac{\Phi(\vec{x})^2}{\mathrm{Res}(G)}, \frac{4}{27} \cdot \Phi(\vec{x})^3\right\}.$$

Hence,

$$\Phi(\vec{x}) - \mathbb{E}\big[\Phi(\mathbf{M}^1 \cdot \vec{x})\big] \geq \frac{1}{16d} \cdot \max\left\{d \cdot \lambda(\mathbf{L}(G)) \cdot \Phi(\vec{x}), \frac{\Phi(\vec{x})^2}{\mathrm{Res}(G)}, \frac{4}{27} \cdot \Phi(\vec{x})^3\right\},$$

and as a consequence, $\Phi(\vec{x}) - \mathbb{E}[\Phi(\mathbf{M}^1 \cdot \vec{x})] \geq g_G(\Phi(\vec{x}))$ by the definition of g_G.

It remains to check the second condition of Definition 3.8 with our claimed value σ_G^2. Inserting its value as stated in the lemma, the condition requires that

$$\mathrm{Var}[\Phi(\mathbf{M}^1 \cdot \vec{x})] \leq (32(\mathrm{t}^*_{\mathrm{hit}}(\mathrm{G})\,/n) + 5 - 1) \cdot \big(\Phi(\vec{x}) - \mathbb{E}[\Phi(\mathbf{M}^1 \cdot \vec{x})]\big)^2,$$

which is given in the second part of Lemma 3.11 (see below). ◀

In Lemma 3.11 we first relate the drop of Φ to the quadratic edge potential Ψ. In the second part we bound the variance of the potential drop as a function of the edge hitting time.

▶ **Lemma 3.11.** *Let G be a d-regular graph, let $\mathbf{M}^1 \sim \mathcal{D}_{RM}(G)$, and let $\vec{x} \in \mathbb{R}^n$, then*

1. $\Phi(\vec{x}) - \mathbb{E}\big[\Phi(\mathbf{M}^1 \cdot \vec{x})\big] \geq \frac{1}{16d} \cdot \Psi_G(\vec{x})$.
2. $\mathrm{Var}\big[\Phi(\mathbf{M}^1 \cdot \vec{x})\big] \leq (32 \cdot (\mathrm{t}^*_{\mathrm{hit}}(\mathrm{G})/n) + 4) \cdot \big(\Phi(\vec{x}) - \mathbb{E}\big[\Phi(\mathbf{M}^1 \cdot \vec{x})\big]\big)^2$.

In Lemma 3.12 we relate the size of the quadratic edge potential Ψ_G to the second-largest eigenvalue of $\mathbf{L}(G)$, effective resistances of G and node potential. To state it, we need some additional definitions. For any two nodes i and j of the graph G, $\mathrm{Res}(i,j)$ is the *effective resistance* (or *resistive distance*) between i and j in G (see Chapter 9 in [24] for a definition, and refer to further details and properties can also be found in [15] and [25, Section 4]; note that in our case, all edges have unit weight). Furthermore, we write $\mathrm{Res}(G)$ for the *resistive diameter* of G, i.e., the largest resistive distance between any pair of nodes in G, and write $\mathrm{Res}^*(G)$ for the maximum effective resistance between any pair of nodes adjacent in G. I.e., $\mathrm{Res}(G) \coloneqq \max_{i,j\in[n]} \mathrm{Res}(i,j)$ and $\mathrm{Res}^*(G) \coloneqq \max_{\{i,j\}\in E(G)} \mathrm{Res}(i,j)$. The first part of the following lemma was previously shown in [19, 33].

▶ **Lemma 3.12.** *Let $\vec{x} \in \mathbb{R}^n$, and let G be a connected d-regular graph.*

1. $\Psi_G(\vec{x}) \geq d \cdot \lambda(\mathbf{L}(G)) \cdot \Phi(\vec{x})$.
2. *If $\vec{x}$ is stochastic, then* $\Psi_G(\vec{x}) \geq \max\left\{\frac{1}{\mathrm{Res}(G)} \cdot \Phi(\vec{x})^2, \frac{4}{27} \cdot \Phi(\vec{x})^3\right\}$
3. $\max_{\{i,j\}\in E(G)} (x_i - x_j)^2 \leq \mathrm{Res}^*(G) \cdot \Psi_G(\vec{x})$.

Proof of Lemma 3.4

Proof. Define $g_G(x) = \frac{1}{16d} \cdot \max\{d \cdot \lambda(\mathbf{L}(G)) \cdot x, x^2/\mathrm{Res}(G), 4x^3/27\}$ and let $\sigma_G^2 \coloneqq 32 \cdot (\mathrm{t}^*_{\mathrm{hit}}(\mathrm{G})/n) + 5$. Then by Lemma 3.10 the matching distribution $\mathcal{D}_{\mathrm{RM}}(G)$ is (g_G, σ_G^2)-good. By Lemma 3.9 we have for all $t \in \mathbb{N}$, $k \in [n]$

$$\mathbb{P}\left[\left(\Upsilon_k(\mathbf{M}^{[t]})\right)^2 \leq 8\sigma_G^2((\gamma+1)\log(n) + \log(8\sigma_G^2)) + \frac{1}{\beta} \cdot \int_0^1 \frac{x}{g_G(x)}\,\mathrm{d}x\right] \geq 1 - n^{-(\gamma+1)}.$$

To bound $\Upsilon_k(\mathbf{M}^{[t]})$ we use the following two claims, which we prove in the full version.

▷ Claim 3.13. It holds that $\int_0^1 x/g_G(x)\,\mathrm{d}x = \mathcal{O}(T(G))$.

▷ Claim 3.14. For any d-regular graph G it holds that $\mathrm{t}^*_{\mathrm{hit}}(\mathrm{G})/n \geq 1/2$.

Together we get from Claim 3.13 and Claim 3.14 that with probability at least $1 - n^{-(\gamma+1)}$

$$\left(\Upsilon_k(\mathbf{M}^{[t]})\right)^2 = \mathcal{O}\left(\frac{\mathrm{t}^*_{\mathrm{hit}}(\mathrm{G})}{n} \cdot \left(\gamma\log(n) + \log\left(\frac{\mathrm{t}^*_{\mathrm{hit}}(\mathrm{G})}{n}\right)\right) + \frac{T(G)}{\beta}\right). \tag{4}$$

Since $\mathrm{t}^*_{\mathrm{hit}}(\mathrm{G}) = \mathcal{O}(n^3)$ (Proposition 10.16 in [24]), $\log(\mathrm{t}^*_{\mathrm{hit}}(\mathrm{G})/n) = \mathcal{O}(\log n)$, and $\gamma > 1$,

$$\Upsilon_k(\mathbf{M}^{[t]}) = \mathcal{O}\left(\sqrt{\gamma\log(n) \cdot \frac{\mathrm{t}^*_{\mathrm{hit}}(\mathrm{G})}{n} + \frac{T(G)}{\beta}}\right) = \mathcal{O}\left(\sqrt{\gamma\log(n) \cdot \frac{\mathrm{t}^*_{\mathrm{hit}}(\mathrm{G})}{n}} + \sqrt{\frac{T(G)}{\beta}}\right).$$

Now Lemma 3.6 states that for any fixed sequence of matching matrices $\mathbf{m}^{[t]}$, with probability at least $1 - 2n^{-(\gamma+1)}$ it holds that

$$\left|D_k(t) - t \cdot \frac{m}{n}\right| = \mathcal{O}\left(\gamma\log(n) + \sqrt{\gamma\log(n) \cdot \frac{m}{n}} \cdot \Upsilon_k(\mathbf{m}^{[t]})\right). \tag{5}$$

Applying a union bound over all $k \in [n]$, Equation (4) and Equation (5) hold for all k with probability at least $1 - 3n^{-\gamma}$. Hence, for all $k \in [n]$

$$\left|D_k(t) - t \cdot \frac{m}{n}\right| = \mathcal{O}\left(\gamma \log(n) + \sqrt{\gamma \log(n) \cdot \frac{m}{n}} \cdot \left(\sqrt{\gamma \log(n) \cdot \frac{\mathrm{t}^*_{\mathrm{hit}}(\mathrm{G})}{n}} + \sqrt{\frac{T(G)}{\beta}}\right)\right)$$
$$= \mathcal{O}\left(\gamma \log(n) \cdot \left(1 + \sqrt{\frac{m}{n} \cdot \frac{\mathrm{t}^*_{\mathrm{hit}}(\mathrm{G})}{n}}\right) + \sqrt{\frac{(\gamma+1)\log(n)}{\beta} \cdot \frac{m}{n} \cdot T(G)}\right).$$

The high-probability bound now follows from Observation 3.5. The corresponding bound on $\mathbb{E}[\mathrm{disc}(\vec{D}(t)]$ follows readily; see the full version for details. ◀

4 Balancing Circuit Model

Here we assume $\beta = 1$. Recall that we assume G is covered by ζ fixed matchings $\mathbf{m}(1), \ldots, \mathbf{m}(\zeta)$. The matching distribution $\mathcal{D}_{\mathrm{BC}}(G)$ then deterministically chooses the matching $\mathbf{m}(t) = \mathbf{m}(t \bmod \zeta)$ in step t. The round matrix is defined as $\mathbf{R} := \mathbf{m}^{[1,\zeta]}$. Thus, for a sequence of matchings $\mathbf{m}^{[t]}$ the global divergence is $\Upsilon(\mathbf{m}^{[t]}) := \max_{k\in[n]} \sqrt{\sum_{\tau=1}^{t} \left\|\mathbf{m}^{[\tau,t]}_{k,\cdot} - 1/n\right\|_2^2}$. The next theorem provides an upper bound on the discrepancy for this model. Note that the following theorem holds for arbitrary graphs, while Theorem 3.1 only holds for d-regular graphs.

▶ **Theorem 4.1.** *Let G be an arbitrary graph and let $\vec{X}(t)$ be the state of process $SB_{AL}(\mathcal{D}_{BC}(G), 1, m)$ at time t with $\mathrm{disc}(\vec{X}(0)) =: K$. For all $t \in \mathbb{N}$ with $t \geq \frac{\zeta}{\lambda(\mathbf{R})} \cdot (\ln(K \cdot n))$ it holds w.h.p. and in expectation*

$$\mathrm{disc}(\vec{X}(t)) = \mathcal{O}\Big(\log(n) + \sqrt{m/n} \cdot \Upsilon(\mathbf{m}^{[t]}) \cdot \sqrt{\log(n)}\Big).$$

Proof. The proof follows the same line as the proof Theorem 3.1, which is proved via Lemma 3.2, Lemma 3.4, and Lemma 3.3 bounding $\vec{I}(t), \vec{D}(t)$, and $\vec{R}(t)$, respectively. Lemma 3.2 is replaced by Lemma 4.2 below. Lemma 3.2 can also be applied to the balancing circuit model since it only requires that the subgraph used for balancing is a matching.

It remains to replace Lemma 3.3. Since the matching matrices are fixed this time the proof is much simpler. The proof of Lemma 3.6 carries to over to this model giving us a bound on $|D_k(t) - tm/n|$ for $k \in [n]$ with probability at least $1 - 2 \cdot n^{-\gamma}$. Applying the union bound over all nodes $k \in [n]$, together with Observation 3.5 (stating that $\mathrm{disc}(\vec{D}(t)) \leq 2 \cdot \max_{k\in[n]} |D_k(t) - t \cdot m/n|$), gives a bound on $\mathrm{disc}(\vec{D}(t))$ which holds with probability at least $1 - 2 \cdot n^{\gamma+1}$. ◀

▶ **Lemma 4.2** (Memorylessness Property). *For all $t \in \mathbb{N}$ with $t \geq \zeta/\lambda(\mathbf{R}) \cdot (\ln(K \cdot n))$ it holds that $\mathrm{disc}(\vec{I}(t)) \leq 2$.*

Proof. Since $\Phi(\vec{x}) \leq K^2 \cdot n$ it follows from Lemma 2 in [20] that

$$\Phi\left(\mathbf{m}^{[1,t]} \cdot \vec{x}\right) \leq (1 - \lambda(\mathbf{R}))^{2\lfloor t \rfloor/\zeta} \cdot \Phi(\vec{x}) \leq (1 - \lambda(\mathbf{R}))^{2\lfloor t \rfloor/\zeta} \cdot K^2 \cdot n \leq e^{-2\lfloor t \rfloor \cdot \lambda(\mathbf{R})/\zeta + 2\ln(Kn)}.$$

Setting $t \geq (\zeta/\lambda(\mathbf{R})) \cdot (\ln(Kn))$ gives $\Phi\left(\mathbf{m}^{[1,t]} \cdot \vec{x}\right) \leq 1$ which implies that $\mathrm{disc}(\vec{I}(t) \leq 2$. ◀

Note that a similar statement was shown in [32, 33, 7].

The next theorem provides a lower bound on the discrepancy for this model. The proof can be found in the full version.

▶ **Theorem 4.3.** *Let G be an arbitrary graph and let $\vec{X}(t)$ be the state of process $SBAL(\mathcal{D}_{BC}(G), 1, m)$ at time t. Then for all $t \in \mathbb{N}$ and $m \geq 4n \cdot \log(n)/\Upsilon(\mathbf{m}^{[t]})$ it holds with constant probability*

$$\operatorname{disc}(\vec{X}(t)) = \Omega\Big(\sqrt{m/n} \cdot \Upsilon(\mathbf{m}^{[t]})\Big).$$

5 Asynchronous Model

The following is our main theorem for the asynchronous model. The bounds provided by Theorem 5.1 for the asynchronous model differ from those in Theorem 3.1 for the random matching model in two details. First, the lower bound on the balancing time is larger by a factor of n. This is due to the fact that the asynchronous model balances across just one edge per round in contrast to $\Theta(n)$ edges in the random matching model. Second, the upper bound on $\operatorname{disc}(\vec{X}(t))$ is much simpler. Note, however that setting $m = n$ in Theorem 3.1 and further simplifying the result by using $\mathrm{t}^*_{\mathrm{hit}}(\mathrm{G})/n = \Omega(1)$ (see also Claim 3.14 in the proof of Lemma 3.4) results in the same asymptotic bound as in Theorem 5.1.

▶ **Theorem 5.1.** *Let G be a d-regular graph and define $T(G) := \min\left\{\frac{\mathrm{t}_{\mathrm{hit}}(\mathrm{G})}{n} \cdot \log(n), \sqrt{\frac{d}{\lambda(\mathbf{L}(G))}}, \frac{1}{\lambda(\mathbf{L}(G))}\right\}$. Let $\vec{X}(t)$ be the state of process $ABAL(\mathcal{D}_A(G), \beta)$ at time t with $\operatorname{disc}(\vec{X}(0)) =: K \geq 1$. There exists a constant $c > 0$ such that for all $t \geq c \cdot n \cdot \log(K \cdot n)/(\lambda(\mathbf{L}(G)) \cdot \beta)$ it holds w.h.p. and in expectation*

$$\operatorname{disc}(\vec{X}(t)) = \mathcal{O}\left(\log(n)\sqrt{\frac{\mathrm{t}^*_{\mathrm{hit}}(\mathrm{G})}{n}} + \sqrt{\frac{\log(n)}{\beta} \cdot T(G)}\right).$$

Proof Sketch of Theorem 5.1. The proof of the theorem follows along the same lines at the proof of Theorem 3.1. However, there are some major differences. Most importantly, the proof of Lemma 3.6 (giving a concentration bound on $D_k(t)$ in terms of the global divergence of the sequence of matching matrices) can not be applied for ABAL. The proof heavily relies on the fact that the load allocation and the matching edges are chosen independently from each other, which is certainly not the case for ABAL. In the full version, we carefully analyze the dependency using a stronger concentration inequality. In addition, we also have to re-calculate the function g_G and σ_G to show that the matching distribution used by $\mathcal{D}_{\mathrm{A}}$ is (g_G, σ_G^2)-good. ◀

6 Drift Result

In our analysis we use the following tail bound for the sum of a non-increasing sequence of random variables with variable negative drift. The proof uses established methods from drift analysis. In particular, it relies one techniques found in the proof of the Variable Drift Theorem in [23]. We prove it in the full version.

▶ **Theorem 6.1.** *Let $(X(t))_{t\geq 0}$ be a non-increasing sequence of discrete random variables with $X(t) \in \mathbb{R}_0^+$ for all t with fixed $X(0) = x_0$. Assume there exists an increasing function $h\colon \mathbb{R}_0^+ \to \mathbb{R}^+$ and a constant $\sigma > 0$ such that the following holds. For all $t \in \mathbb{N}$ and all $x > 0$ with $\mathbb{P}[X(t) = x] > 0$*

1. $\mathbb{E}[X(t+1) \mid X(t) = x] \leq x - h(x)$,
2. $\operatorname{Var}[X(t+1) \mid X(t) = x] \leq \sigma \cdot (\mathbb{E}[X(t+1) \mid X(t) = x] - x)^2$.

Then the following statements hold.

1. *For all* $\delta \in (0,1)$ *and any arbitrary but fixed* t

$$\mathbb{P}\left[\int_{X(t)}^{x_0} \frac{1}{h(\varphi)}\,\mathrm{d}\varphi \leq (1-\delta)t\right] \leq \exp\left(-\frac{\delta^2 t}{2(\sigma+1)}\right).$$

2. *For all* $\delta \in (0,1)$ *and* $p \in (0,1)$ *we define* $t_0 \coloneqq \frac{2(\sigma+1)}{\delta^2}\left(-\log(p) + \log\left(\frac{2(\sigma+1)}{\delta^2}\right)\right)$. *Then*

$$\mathbb{P}\left[\sum_{t=t_0+1}^{\infty} X(t) \leq \frac{1}{1-\delta} \cdot \int_0^{x_0} \frac{\varphi}{h(\varphi)}\mathrm{d}\varphi\right] \geq 1-p.$$

7 Conclusions and Open Problems

In this paper we analyze discrete load balancing processes on graphs. As our main contribution we bound the discrepancy that arises in dynamic load balancing in three models, the random matching model, the balancing circuit model, and the asynchronous model. Our results for the random matching model and the asynchronous model hold for d-regular graphs, while our analysis for the balancing circuit model applies to arbitrary graphs.

To the best of our knowledge our results constitute the first discrepancy bounds for discrete, dynamic balancing processes on general graphs. Furthermore, our results improve the work by Alistarh et al. [2] who prove that the expected discrepancy is bounded by $\sqrt{n}\log(n)$ in the (arguably simpler) continuous asynchronous process $\textsc{ABal}^{(\mathrm{cont})}(\mathcal{D}_{\mathrm{A}}(G),1)$. We improve their bound to $\sqrt{n\log(n)}$ and additionally show that it holds with high probability. We conjecture that our results are tight, up to polylogarithmic factors. However, showing tight upper and lower bounds remains an open problem.

One interesting feature of our bound on the discrepancy is the scaling with the parameter β: decreasing it linearly only increases the bound on the discrepancy by a square root factor. This means for sufficiently small β, the expected amount of load transferred per edge and round is constant.

Open Problems. We are confident that our results carry over to arbitrary graphs (as opposed to regular graphs), provided that there exists a lower bound on the probability p_{min} with which an edge is used for balancing. However, to show bounds on the discrepancy one has to overcome fundamental problems such as the bias introduced by high-degree nodes. Analyzing the behavior for more general load arrival distributions is also an interesting but likely challenging open problem. More avenues for generalization are the deletion of load over time as well as varying the amount of load generated in each round dynamically.

Another interesting open question is whether the results carry over to a model where the amount of load that may be transmitted over an edge in each step is bounded by a constant. If only a single load item can be transferred per edge and step the problem is similar to the token distribution problem (see, for example, [5]).

Finally, we believe that one can also adapt our analysis to variant of a graphical balls-into-bins process. The process works as follows. In each step an edge (i,j) is sampled uniformly at random. W.l.o.g. assume that the load of i is smaller than the load of j by an additive term Δ. Then a biased coin is tossed showing heads with probability $p \coloneqq \min\{1,(1+\beta\cdot\Delta)/2\}$ and tails otherwise, where β is a suitably chosen and non-constant parameter. If the coin hits heads one item is allocated to i and otherwise to j. A formal analysis of this allocation process (as well as of other, related balls-into-bins processes) is beyond the scope of our paper and remains an open problem.

References

1 Heiner Ackermann, Simon Fischer, Martin Hoefer, and Marcel Schöngens. Distributed algorithms for QoS load balancing. *Distributed Comput.*, 23(5-6):321–330, 2011. doi:10.1007/s00446-010-0125-1.

2 Dan Alistarh, Giorgi Nadiradze, and Amirmojtaba Sabour. Dynamic averaging load balancing on cycles. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020*, volume 168 of *LIPIcs*, pages 7:1–7:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.7.

3 Aris Anagnostopoulos, Adam Kirsch, and Eli Upfal. Load balancing in arbitrary network topologies with stochastic adversarial input. *SIAM Journal on Computing*, 34(3):616–639, 2005. doi:10.1137/S0097539703437831.

4 Elliot Anshelevich, David Kempe, and Jon M. Kleinberg. Stability of load balancing algorithms in dynamic adversarial systems. *SIAM J. Comput.*, 37(5):1656–1673, 2008. doi:10.1137/050639272.

5 Friedhelm Meyer auf der Heide, Brigitte Oesterdiekhoff, and Rolf Wanka. Strongly adaptive token distribution. *Algorithmica*, 15(5):413–427, 1996. doi:10.1007/BF01955042.

6 Nikhil Bansal and Ohad N. Feldheim. The power of two choices in graphical allocation. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20–24, 2022*, pages 52–63. ACM, 2022. doi:10.1145/3519935.3519995.

7 Petra Berenbrink, Colin Cooper, Tom Friedetzky, Tobias Friedrich, and Thomas Sauerwald. Randomized diffusion for indivisible loads. *J. Comput. Syst. Sci.*, 81(1):159–185, 2015. doi:10.1016/j.jcss.2014.04.027.

8 Petra Berenbrink, Tom Friedetzky, and Zengjian Hu. A new analytical method for parallel, diffusion-type load balancing. *J. Parallel Distributed Comput.*, 69(1):54–61, 2009. doi:10.1016/j.jpdc.2008.05.005.

9 Petra Berenbrink, Tom Friedetzky, Dominik Kaaser, and Peter Kling. Tight & simple load balancing. In *2019 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2019*, pages 718–726. IEEE, 2019. doi:10.1109/IPDPS.2019.00080.

10 Petra Berenbrink, Tom Friedetzky, and Russell A. Martin. On the stability of dynamic diffusion load balancing. *Algorithmica*, 50(3):329–350, 2008. doi:10.1007/s00453-007-9081-y.

11 Petra Berenbrink, Peter Kling, Christopher Liaw, and Abbas Mehrabian. Tight load balancing via randomized local search. In *2017 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2017*, pages 192–201. IEEE Computer Society, 2017. doi:10.1109/IPDPS.2017.52.

12 Leran Cai and Thomas Sauerwald. Randomized load balancing on networks with stochastic inputs. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017*, volume 80 of *LIPIcs*, pages 139:1–139:14, 2017. doi:10.4230/LIPIcs.ICALP.2017.139.

13 Fan R. K. Chung and Lincoln Lu. Survey: Concentration inequalities and martingale inequalities: A survey. *Internet Math.*, 3(1):79–127, 2006. doi:10.1080/15427951.2006.10129115.

14 Ralf Diekmann, Andreas Frommer, and Burkhard Monien. Efficient schemes for nearest neighbor load balancing. *Parallel Comput.*, 25(7):789–812, 1999. doi:10.1016/S0167-8191(99)00018-6.

15 Peter G. Doyle and J. Laurie Snell. *Random Walks and Electric Networks*. Number Book 22 in Carus Mathematical Monographs. Mathematical Association of America, Washington, DC, 1984.

16 Simon Fischer, Harald Räcke, and Berthold Vöcking. Fast convergence to wardrop equilibria by adaptive sampling methods. *SIAM J. Comput.*, 39(8):3700–3735, 2010. doi:10.1137/090746720.

17 Tobias Friedrich and Thomas Sauerwald. Near-perfect load balancing by randomized rounding. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*, pages 121–130. ACM, 2009. doi:10.1145/1536414.1536433.

18 Bhaskar Ghosh, Frank Thomson Leighton, Bruce M. Maggs, S. Muthukrishnan, C. Greg Plaxton, Rajmohan Rajaraman, Andréa W. Richa, Robert Endre Tarjan, and David Zuckerman. Tight analyses of two local load balancing algorithms. *SIAM J. Comput.*, 29(1):29–64, 1999. doi:10.1137/S0097539795292208.

19 Bhaskar Ghosh and S. Muthukrishnan. Dynamic load balancing by random matchings. *J. Comput. Syst. Sci.*, 53(3):357–370, 1996. doi:10.1006/jcss.1996.0075.

20 Bhaskar Ghosh, S. Muthukrishnan, and Martin H. Schultz. First and second order diffusive methods for rapid, coarse, distributed load balancing (extended abstract). In *Proceedings of the 8th Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA '96*, pages 72–81. ACM, 1996. doi:10.1145/237502.237509.

21 Martin Hoefer and Thomas Sauerwald. Threshold load balancing in networks. *CoRR*, abs/1306.1402, 2013. arXiv:1306.1402.

22 David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *44th Symposium on Foundations of Computer Science (FOCS 2003)*, pages 482–491. IEEE Computer Society, 2003. doi:10.1109/SFCS.2003.1238221.

23 Johannes Lengler. Drift analysis. In Benjamin Doerr and Frank Neumann, editors, *Theory of Evolutionary Computation – Recent Developments in Discrete Optimization*, Natural Computing Series, pages 89–131. Springer, 2020. doi:10.1007/978-3-030-29414-4_2.

24 David Levin and Yuval Peres. *Markov Chains and Mixing Times*. AMS, 2017. doi:10.1090/mbk/107.

25 László Lovász. Random walks on graphs. *Combinatorics, Paul Erdős is Eighty*, 2:1–46, 1993.

26 Henning Meyerhenke. Shape optimizing load balancing for mpi-parallel adaptive numerical simulations. In *Graph Partitioning and Graph Clustering, 10th DIMACS Implementation Challenge Workshop*, volume 588 of *Contemporary Mathematics*, pages 67–82. American Mathematical Society, 2012. URL: http://www.ams.org/books/conm/588/11699.

27 Vahid Mohammadian, Nima Jafari Navimipour, Mehdi Hosseinzadeh, and Aso Mohammad Darwesh. Fault-tolerant load balancing in cloud computing: A systematic literature review. *IEEE Access*, 10:12714–12731, 2022. doi:10.1109/ACCESS.2021.3139730.

28 S. Muthukrishnan, Bhaskar Ghosh, and Martin H. Schultz. First- and second-order diffusive methods for rapid, coarse, distributed load balancing. *Theory Comput. Syst.*, 31(4):331–354, 1998. doi:10.1007/s002240000092.

29 Borek Patzák and Daniel Rypl. Object-oriented, parallel finite element framework with dynamic load balancing. *Adv. Eng. Softw.*, 47(1):35–50, 2012. doi:10.1016/j.advengsoft.2011.12.008.

30 David Peleg and Eli Upfal. The token distribution problem. *SIAM J. Comput.*, 18(2):229–243, 1989. doi:10.1137/0218015.

31 Yuval Peres, Kunal Talwar, and Udi Wieder. Graphical balanced allocations and the $(1 + \beta)$-choice process. *Random Struct. Algorithms*, 47(4):760–775, 2015. doi:10.1002/rsa.20558.

32 Yuval Rabani, Alistair Sinclair, and Rolf Wanka. Local divergence of markov chains and the analysis of iterative load balancing schemes. In *39th Annual Symposium on Foundations of Computer Science, FOCS '98*, pages 694–705. IEEE Computer Society, 1998. doi:10.1109/SFCS.1998.743520.

33 Thomas Sauerwald and He Sun. Tight bounds for randomized load balancing on arbitrary network topologies. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012*, pages 341–350. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.86.

34 Gengbin Zheng, Abhinav Bhatele, Esteban Meneses, and Laxmikant V. Kalé. Periodic hierarchical load balancing for large supercomputers. *Int. J. High Perform. Comput. Appl.*, 25(4):371–385, 2011. doi:10.1177/1094342010394383.

Fast Approximation of Search Trees on Trees with Centroid Trees

Benjamin Aram Berendsohn
Institut für Informatik, Freie Universität Berlin, Germany

Ishay Golinsky
Blavatnik School of Computer Science, Tel Aviv University, Israel

Haim Kaplan
Blavatnik School of Computer Science, Tel Aviv University, Israel

László Kozma
Institut für Informatik, Freie Universität Berlin, Germany

Abstract

Search trees on trees (STTs) generalize the fundamental binary search tree (BST) data structure: in STTs the underlying search space is an arbitrary tree, whereas in BSTs it is *a path*. An optimal BST of size n can be computed for a given distribution of queries in $\mathcal{O}(n^2)$ time [Knuth, Acta Inf. 1971] and *centroid* BSTs provide a nearly-optimal alternative, computable in $\mathcal{O}(n)$ time [Mehlhorn, SICOMP 1977].

By contrast, optimal STTs are not known to be computable in polynomial time, and the fastest constant-approximation algorithm runs in $\mathcal{O}(n^3)$ time [Berendsohn, Kozma, SODA 2022]. Centroid trees can be defined for STTs analogously to BSTs, and they have been used in a wide range of algorithmic applications. In the unweighted case (i.e., for a uniform distribution of queries), the centroid tree can be computed in $\mathcal{O}(n)$ time [Brodal, Fagerberg, Pedersen, Östlin, ICALP 2001; Della Giustina, Prezza, Venturini, SPIRE 2019]. These algorithms, however, do not readily extend to the weighted case. Moreover, no approximation guarantees were previously known for centroid trees in either the unweighted or weighted cases.

In this paper we revisit centroid trees in a general, weighted setting, and we settle both the algorithmic complexity of constructing them, and the quality of their approximation. For constructing a weighted centroid tree, we give an *output-sensitive* $\mathcal{O}(n \log h) \subseteq \mathcal{O}(n \log n)$ time algorithm, where h is the height of the resulting centroid tree. If the weights are of polynomial complexity, the running time is $\mathcal{O}(n \log \log n)$. We show these bounds to be optimal, in a general decision tree model of computation. For approximation, we prove that the cost of a centroid tree is at most *twice* the optimum, and this guarantee is best possible, both in the weighted and unweighted cases. We also give tight, fine-grained bounds on the approximation-ratio for bounded-degree trees and on the approximation-ratio of more general α-centroid trees.

2012 ACM Subject Classification Theory of computation → Data structures design and analysis

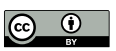
Keywords and phrases centroid tree, search trees on trees, approximation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.19

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: arxiv.org/abs/2209.08024 [6]

Funding *Benjamin Aram Berendsohn*: DFG grant KO 6140/1-1.
Ishay Golinsky: ISF grant no. 1595/19 and the Blavatnik Research Foundation.
Haim Kaplan: ISF grant no. 1595/19 and the Blavatnik Research Foundation.
László Kozma: DFG grant KO 6140/1-1.



1 Introduction

Search trees on trees (STTs) are a far-reaching generalization of binary search trees (BSTs), modeling the exploration of tree-shaped search spaces. Given an undirected tree $\mathcal{T}$, an STT on $\mathcal{T}$ is a tree rooted at an arbitrary vertex r of $\mathcal{T}$, with subtrees built recursively on the components resulting after removing r from $\mathcal{T}$, see Figure 1 for an example. BSTs correspond to the special case where the underlying tree $\mathcal{T}$ is a *path*.

STTs and, more generally, search trees on graphs arise in several different contexts and have been studied under different names: *tubings* [14], *vertex rankings* [20, 8, 23], *ordered colorings* [35], *elimination trees* [43, 49, 2, 9]. STTs have been crucial in many algorithmic applications, e.g., in pattern matching and counting [24, 37, 27], cache-oblivious data structures [4, 25], tree clustering [26], geometric visibility [30], planar point location [29], distance oracles [16]. They arise in matrix factorization (e.g., see [22, § 12]), and have also been related to the competitive ratio in certain online hitting set problems [23].

Similarly to the setting of BSTs, a natural goal is to find an STT in which the expected depth of a vertex is as small as possible; we refer to such a tree as an *optimal tree*, noting that it is not necessarily unique. This optimization task can be studied both for the uniform probability distribution over the vertices, and for the more general case of an arbitrary distribution given as input. We refer to the first as the *unweighted* and the second as the *weighted* problem.

For BSTs, both the unweighted and the weighted problems are well-understood. In the unweighted case, a simple balanced binary tree achieves the optimum. In the weighted case, an optimal tree on n vertices can be found in time $\mathcal{O}(n^2)$ by Knuth's algorithm [36], a textbook example of dynamic programming. No faster algorithm is known in general, although Larmore's algorithm [40] achieves better bounds under certain regularity assumptions on the weights; for example, if the probability assigned to each vertex is $\Omega(1/n)$, then the optimum can be found in time $\mathcal{O}(n^{1.591})$.

By contrast, the complexity of computing an optimal STT is far less understood. Even in the unweighted case, no polynomial-time algorithm is known, and the problem is not known to be NP-hard even with arbitrary weights. Recently, a PTAS was given for the weighted problem [7], but its running time for obtaining a $(1+\varepsilon)$-approximation of the optimal STT is $\mathcal{O}(n^{1+2/\varepsilon})$, which is prohibitive for reasonably small values of ε. Note that the apparently easier problem of minimizing the *maximum depth* of a vertex, i.e., computing the *treedepth* of a tree, can be solved in linear time by Schäffer's algorithm [50], and treedepth itself has many algorithmic applications, e.g., see [47, § 6,7].

Centroid trees. Given the relatively high cost of computing optimal binary search trees, research has turned already half a century ago to efficient approximations. Mehlhorn has shown [44, 45] that a simple BST that can be computed in $\mathcal{O}(n)$ time closely approximates the optimum. More precisely, both the optimum cost and the cost of the obtained tree are in $[H/\log(3), H+1]$, where H is the binary entropy of the input distribution.[1] Alternatively, the cost can be upper bounded by $\texttt{OPT} + \log(\texttt{OPT}) + \log e$, where $\texttt{OPT}$ is the cost of the optimal tree. Observe that this means that the approximation ratio gets arbitrarily close to 1 as $\texttt{OPT}$ goes to infinity.[2]

[1] All logarithms in this paper are base 2.

[2] Results for BSTs are sometimes presented in a more general form, where the input distribution also accounts for *unsuccessful searches*, i.e., it may assign non-zero probabilities to the *gaps* between neighboring vertices and outside the two extremes. Extending such a model to STTs is straightforward, but perhaps less natural in the case of general trees, we therefore omit it for the sake of simplicity, and consider only successful searches.

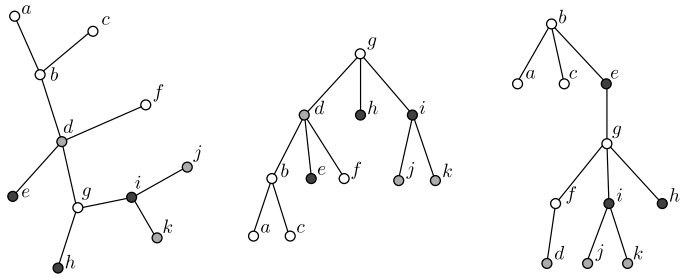


Figure 1 (*Left.*) Tree $\mathcal{T}$. (*Middle.*) Centroid tree of $\mathcal{T}$. (*Right.*) A different STT on $\mathcal{T}$. Colors indicate weights (probabilities), $w(e) = w(h) = w(i) = 0.15$, $w(d) = w(j) = w(k) = 0.10$, and all other vertices have weight 0.05. Observe that the centroid tree is (in this example) unique.

The BST that achieves the above guarantees is built by recursively picking roots such as to make the weights of the left and right subtrees "as equal as possible". This is a special case of a *centroid tree*, defined as follows. Given a tree $\mathcal{T}$, a *centroid* of $\mathcal{T}$ is a vertex whose removal from $\mathcal{T}$ results in components with weight *at most half* of the total weight of $\mathcal{T}$. A centroid tree is built by iteratively finding a centroid and recursing on the components resulting after its removal. See Figure 1 for an example.

The fact that an (unweighted) centroid always exists was already shown in the 19-th century by C. Jordan [34]. We sketch the easy, constructive argument that also shows the existence of a weighted centroid: start at an arbitrary vertex of $\mathcal{T}$ and, as long as the current vertex is not a centroid, move one edge in the direction of the component with largest weight. It is not hard to see that the procedure succeeds, visiting each vertex at most once.

A straightforward implementation of the above procedure finds an unweighted centroid tree in $\mathcal{O}(n \log n)$ time. This running time has been improved to $\mathcal{O}(n)$ by carefully using data structures [11, 28]. The run-time guarantees however, do not readily generalize from the unweighted to the weighted setting. Intuitively, the difficulty lies in the fact that in the weighted case, the removal of a centroid vertex may split the tree in a very unbalanced way, leaving up to $n - 1$ vertices in one component. Thus, a naive recursive approach will take $\Theta(n^2)$ time in the worst case.

Most algorithmic applications of STTs, including those mentioned before, rely on centroid trees. It is therefore surprising that nothing appears to be known about how well the centroid tree approximates the optimal STT in either the unweighted or weighted cases. In this paper we prove that the centroid tree is a 2-approximation of the optimal STT, and that the factor 2 is, in general, best possible, both in the unweighted and weighted settings. As our main result, we also show a more precise bound on the approximation ratio of centroid trees, in terms of the maximum degree of the underlying tree $\mathcal{T}$.[3]

Before stating our results, we need a few definitions. Consider an undirected, unrooted tree $\mathcal{T}$ given as input, together with a *weight function* $w : V(\mathcal{T}) \to \mathbb{R}_{\geq 0}$. For convenience, for any subgraph $\mathcal{H}$ of $\mathcal{T}$, we denote $w(\mathcal{H}) = \sum_{x \in V(\mathcal{H})} w(x)$. (To interpret the weights as probabilities, we need the condition $w(\mathcal{T}) = 1$. It is, however, often convenient to relax this requirement and allow arbitrary non-negative weights, which is the approach we will take.)

[3] In their recent paper on dynamic STTs, Bose, Cardinal, Iacono, Koumoutsos, and Langerman [10] remark that the ratio between the costs of the centroid- and optimal trees may be unbounded. In light of our results, this observation is erroneous. It is true, however, that a centroid tree built using the uniform distribution may be far from the optimum w.r.t. a different distribution.

A *search tree* on $\mathcal{T}$ is a rooted tree T with vertex set $V(\mathcal{T})$ whose root is an arbitrary vertex $r \in V(\mathcal{T})$. The children of r in T are the roots of search trees built on the connected components of the forest $\mathcal{T} - r$. A tree consisting of a single vertex admits only itself as a search tree. It follows from the definition that for all x, the subtree T_x of T rooted at x induces a connected subgraph $\mathcal{T}[V(T_x)]$ of $\mathcal{T}$, and moreover, T_x is a search tree on $\mathcal{T}[V(T_x)]$.

The *cost* of a search tree T on $\mathcal{T}$ is $\texttt{cost}_w(T) = \sum_{x \in V(T)} w(x) \cdot \texttt{depth}_T(x)$, where the depth of the root is taken to be 1. The *optimum cost* $\texttt{OPT}(\mathcal{T}, w)$ is the minimum of $\texttt{cost}_w(T)$ over all search trees T of $\mathcal{T}$.

A vertex $v \in V(\mathcal{T})$ is a *centroid* if for all components $\mathcal{H}$ of $\mathcal{T} - v$, we have $w(\mathcal{H}) \leq w(\mathcal{T})/2$. A search tree T of $\mathcal{T}$ is a *centroid tree* if vertex x is a centroid of $\mathcal{T}[V(T_x)]$ for all $x \in V(\mathcal{T})$. In general, the centroid tree is not unique, and centroid trees of the same tree can have different costs.[4] We denote by $\texttt{cent}(\mathcal{T}, w)$ the *maximum* cost of a centroid tree of $(\mathcal{T}, w)$, with weight function w.

We can now state our approximation guarantee for centroid trees.

▶ **Theorem 1.** *Let $\mathcal{T}$ be a tree, $w : V(\mathcal{T}) \to \mathbb{R}_{\geq 0}$, and $m = w(\mathcal{T})$. Then*

$$\texttt{cent}(\mathcal{T}, w) \leq 2 \cdot \texttt{OPT}(\mathcal{T}, w) - m.$$

We show that this result is optimal, including in the additive term. Moreover, the constant factor 2 cannot be improved even for unweighted instances.

▶ **Theorem 2.**

(i) *For every $\varepsilon > 0$ there is a sequence of instances $(\mathcal{T}_n, w_n)$ with $w_n(\mathcal{T}_n) = 1$, and for every centroid tree C_n of $(\mathcal{T}_n, w_n)$*

$$\texttt{cost}_{w_n}(C_n) \geq 2 \cdot \texttt{OPT}(\mathcal{T}_n, w_n) - 1 - \varepsilon.$$

(ii) *There is a sequence of instances $(\mathcal{T}_n, w_n)$, where w_n is the uniform distribution on $V(\mathcal{T}_n)$, and for every centroid tree C_n of $(\mathcal{T}_n, w_n)$*

$$\lim_{n \to \infty} \frac{\texttt{cost}_{w_n}(C_n)}{\texttt{OPT}(\mathcal{T}_n, w_n)} = 2.$$

In both cases $\lim_{n \to \infty} \texttt{OPT}(\mathcal{T}_n, w_n) = \infty$.

Note that the fact that $\lim_{n \to \infty} \texttt{OPT}(\mathcal{T}_n, w_n) = \infty$ in Theorem 2 establishes that the *asymptotic* approximation ratio is 2. By this we mean that every bound of the form $\texttt{cent} \leq c \cdot \texttt{OPT} + o(\texttt{OPT})$ must have $c \geq 2$.

We next show a stronger guarantee when the underlying tree has bounded degree.

▶ **Theorem 3.** *Let $\mathcal{T}$ be a tree, $w : V(\mathcal{T}) \to \mathbb{R}_{\geq 0}$, and let Δ be the* maximum degree *of $\mathcal{T}$. Then*

$$\texttt{cent}(\mathcal{T}, w) \leq \left(2 - \frac{1}{2^\Delta}\right) \cdot \texttt{OPT}(\mathcal{T}, w).$$

We complement this result by two lower bounds. The first establishes the tightness of the approximation ratio. The second shows a (slightly smaller) lower bound on the approximation ratio for instances where $\texttt{OPT}$ is unbounded.

[4] Consider, for instance the two different centroid trees of a path on four vertices, with weights $(0.2, 0.3, 0.2, 0.3)$.

▶ **Theorem 4.** *Let $\Delta \geq 3$ be integer.*

(i) *There is a sequence of instances $(\mathcal{T}_n, w_n)$ such that $\mathcal{T}_n$ has maximum degree at most Δ, and for every centroid tree C_n of $(\mathcal{T}_n, w_n)$*

$$\lim_{n\to\infty} \frac{\texttt{cost}_{w_n}(C_n)}{\texttt{OPT}(\mathcal{T}_n, w_n)} = 2 - \frac{1}{2^\Delta}.$$

(ii) *There is a sequence of instances $(\mathcal{T}_n, w_n)$ such that $\mathcal{T}_n$ has maximum degree at most Δ, $\lim_{n\to\infty} \texttt{OPT}(\mathcal{T}_n, w_n) = \infty$, $w_n(\mathcal{T}_n) = 1$, and for every centroid tree C_n of $(\mathcal{T}_n, w_n)$*

$$\texttt{cost}_{w_n}(C_n) \geq \left(2 - \frac{4}{2^\Delta}\right) \cdot \texttt{OPT}(\mathcal{T}_n, w_n) - 1.$$

We remark that Theorem 4(i) does not exclude the possiblity of a bound of the form $\texttt{cent} \leq c \cdot \texttt{OPT} + o(\texttt{OPT})$, where $c < 2 - \frac{1}{2^\Delta}$, as here $\texttt{OPT}(\mathcal{T}_n, w_n)$ is bounded. Part (ii), however, establishes that a bound of the form $\texttt{cent} \leq c \cdot \texttt{OPT} + o(\texttt{OPT})$ must have $c \geq 2 - \frac{4}{2^\Delta}$. We leave open the problem of closing the gap in the asymptotic approximation ratio in terms of Δ.

Computing centroid trees. On the algorithmic side, we show that the weighted centroid tree can be computed in $\mathcal{O}(n \log n)$ time. Previously, the fastest known constant-approximation algorithm [7] took $\mathcal{O}(n^3)$ time (similarly achieving an approximation ratio of 2). The main step of our algorithm, finding the weighted centroid of a tree, is achievable in $\mathcal{O}(\log n)$ time, assuming that the underlying tree is stored in a *top tree* data structure [1]. Iterating this procedure in combination with known algorithms for *constructing* and *splitting* top trees yields the algorithm that runs in $\mathcal{O}(n \log n)$ time. As our main algorithmic result, we also develop an improved, *output-sensitive* algorithm, with running time $\mathcal{O}(n \log h)$, where h is the height of the resulting centroid tree, yielding a running time $\mathcal{O}(n \log \log n)$ in the typical case when the height is $\mathcal{O}(\log n)$.

▶ **Theorem 5.** *Let $\mathcal{T}$ be a tree on n vertices and w be a weight function. We can compute a centroid tree of $(\mathcal{T}, w)$ in time $\mathcal{O}(n \log h)$, where h is the height of the computed centroid tree.*

One may ask whether the weighted centroid tree can be computed in linear time, similarly to the unweighted centroid tree, or to the weighted centroid BST. We show that, assuming a general decision tree model of computation, this is not possible, and the algorithm of Theorem 5 is optimal for all n and h (up to a constant factor). Our lower bound on the running time applies, informally, to any deterministic algorithm in which the input weights affect program flow in the form of binary decisions, involving arbitrary computable functions.

More precisely, consider a tree $\mathcal{T}$ on n vertices. We say that a *binary decision tree $D_\mathcal{T}$ solves $\mathcal{T}$* for a class of weight functions $\mathcal{W}$ mapping $V(\mathcal{T})$ to $\mathbb{R}_{\geq 0}$, if the leaves of $D_\mathcal{T}$ are search trees on $\mathcal{T}$, every branching of $D_\mathcal{T}$ is of the form "$f(w)>0$?" for some computable function $f : \mathcal{W} \to \{-1, +1\}$, and for every weight function $w \in \mathcal{W}$, starting from the root of $D_\mathcal{T}$ and following branchings down the tree, we reach a leaf T of $D_\mathcal{T}$ that is a valid centroid tree for $(\mathcal{T}, w)$. The height of $D_\mathcal{T}$ is then a lower bound on the worst-case running time.

▶ **Theorem 6.** *Let $h \geq 3$ and $n \geq h + 1$ be integers. Then there is a tree $\mathcal{T}$ on at most n vertices and a class $\mathcal{W}$ of weight functions on $V(\mathcal{T})$ such that for every $w \in \mathcal{W}$, every centroid tree of $(\mathcal{T}, w)$ has height h, and every binary decision tree that solves $\mathcal{T}$ for $\mathcal{W}$ has height $\Omega(n \log h)$.*

We can nonetheless improve the running time, when the weights are restricted in certain (natural) ways. We define the *spread* σ of a weight function w as the ratio between the total weight $w(\mathcal{T})$, and the smallest *non-zero* weight of a vertex. As we show, $\mathcal{O}(n \log h) \subseteq \mathcal{O}(n \log\log(\sigma + n))$ and therefore, when $\sigma \in n^{\mathcal{O}(1)}$ (for instance, if the weights are integers stored in RAM words), we obtain a running time of $\mathcal{O}(n \log\log n)$.

When many vertices have zero weight, we obtain further improvements, e.g., if only $\mathcal{O}(n/\log n)$ of the weights are non-zero, we can compute a centroid tree in $\mathcal{O}(n)$ time, even if the height h is large. The precise statement of these refined bounds and the discussion of their optimality are available in the full version of this paper [6].

Approximate centroid trees. Finally, we consider the approximation guarantees of a generalized form of centroid trees. Let us call a vertex v of a tree $\mathcal{T}$ an α-centroid, for $0 \leq \alpha \leq 1$, if $w(\mathcal{H}) \leq \alpha \cdot w(\mathcal{T})$, for all components $\mathcal{H}$ of $\mathcal{T} - v$. An α-centroid tree is an STT in which every vertex x is an α-centroid of its subtree $\mathcal{T}[V(T_x)]$.

Observe that the standard centroid tree is a $\frac{1}{2}$-centroid tree, and all STTs are 1-centroid trees. Also note that an α-centroid is a β-centroid for all $\beta \geq \alpha$ and that the existence of an α-centroid is not guaranteed for $\alpha < \frac{1}{2}$ (consider a single edge with the two endpoints having the same weight). On the other hand, an α-centroid for $\alpha < \frac{1}{2}$, if it exists, is unique, and therefore the α-centroid tree is also unique. To see this, consider an α-centroid c that splits $\mathcal{T}$ into components $\mathcal{T}_1, \ldots, \mathcal{T}_k$. If an alternative α-centroid c' were in component $\mathcal{T}_i$, then its removal would yield a component containing all vertices in $\mathcal{T} - V(\mathcal{T}_i)$, of weight at least $(1-\alpha) \cdot w(\mathcal{T}) > \alpha \cdot w(\mathcal{T})$.

Denote by $\texttt{cent}^\alpha(\mathcal{T}, w)$ the maximum cost of an α-centroid tree of $(\mathcal{T}, w)$, or 0 if no α-centroid tree exists. We refine our guarantee from Theorem 1 to approximate centroid trees:

▶ **Theorem 7.** *Let $\mathcal{T}$ be a tree, $w : V(\mathcal{T}) \to \mathbb{R}_{\geq 0}$, $m = w(\mathcal{T})$. We have*

$$(i)\quad \texttt{cent}^\alpha(\mathcal{T}, w) \leq \frac{1}{1-\alpha} \cdot \texttt{OPT}(\mathcal{T}, w) - \frac{\alpha}{1-\alpha} m, \qquad \text{for } \alpha \in (0, 1),$$

$$(ii)\quad \texttt{cent}^\alpha(\mathcal{T}, w) \leq \frac{1}{2-3\alpha} \cdot \texttt{OPT}(\mathcal{T}, w) - \frac{3\alpha - 1}{2-3\alpha} m, \qquad \text{for } \alpha \in \left[\frac{1}{3}, \frac{1}{2}\right].$$

Note that the second bound is a strengthening of the first when $\alpha < \frac{1}{2}$. In particular, for $\alpha \leq \frac{1}{3}$, it implies that an α-centroid tree is *optimal*, if it exists.

We show that the result is tight when $\alpha \geq \frac{1}{2}$ by proving a matching lower bound.

▶ **Theorem 8.** *For every $\alpha \in [\frac{1}{2}, 1)$ there is a sequence of instances $(\mathcal{T}_n, w_n)$ with $\lim_{n\to\infty} \texttt{OPT}(\mathcal{T}_n, w_n) = \infty$, $w_n(\mathcal{T}_n) = 1$ and*

$$\texttt{cent}^\alpha(\mathcal{T}_n, w_n) \geq \frac{1}{1-\alpha} \cdot \texttt{OPT}(\mathcal{T}_n, w_n) - \frac{\alpha}{1-\alpha}.$$

Note that if $\alpha > \frac{1}{2}$, we cannot prove such a lower bound for *all* α-centroid trees of $(\mathcal{T}_n, w_n)$ (as in Theorem 2), since a $\frac{1}{2}$-centroid tree exists and has stronger approximation guarantees according to Theorem 1.

Finally, we argue that every optimal STT is a $\frac{2}{3}$-centroid tree. A special case of this result (for BSTs) was shown by Hirschberg, Larmore, and Molodowitch [32], who also showed that the ratio $\frac{2}{3}$ is tight (in the special case of BSTs, and thus, also for STTs).

▶ **Theorem 9.** *Let T be an optimal STT of $(\mathcal{T}, w)$. Then, T is a $\frac{2}{3}$-centroid tree of $(\mathcal{T}, w)$.*

Structure of the paper. In this extended abstract we omit some of the proofs, discussions, and technical details which are included in the full paper [6]. In Section 2 we state a number of results needed in the proofs. The general upper and lower bounds on the approximation ratio of centroid trees (Theorems 1 and 2) are proved in Section 3. These results can be seen as a warm-up towards the fine-grained bounds on the approximation ratio of centroid trees (Theorems 3 and 4), which we prove in Section 4. The algorithmic results (Theorem 5) are discussed in Section 5, with the details of the output-sensitive algorithm, the lower bounds (Theorem 6), and further extensions available in the full paper. Results on α-centroids (Theorems 7, 8, and 9) are proved in the full paper. In Section 6 we conclude with open questions.

Related work. Different models of searching in trees have also been considered, e.g., the one where we query edges instead of vertices [3, 39, 46, 48], with connections to searching in posets [42, 31]. In the edge-query setting, Cicalese, Jacobs, Laber and Molinaro [17, 18] study the problem of minimizing the average search time of a vertex, and show this to be an NP-hard problem [17]. They also show that an "edge-centroid" tree (in their terminology, a greedy algorithm) gives a 1.62-approximation of the optimum [18].

STTs generalize BSTs, therefore it is natural to ask to what extent the theory developed for BSTs can be extended to STTs. Defining a natural *rotation* operation on STTs, Bose, Cardinal, Iacono, Koumoutsos, and Langerman [10] develop an $O(\log\log n)$ competitive dynamic STT, analogously to Tango BSTs [19]. In a similar spirit, Berendsohn and Kozma [7] generalize Splay trees [51] to STTs. The rotation operation on STTs naturally leads to the definition of *tree associahedra*, a combinatorial structure that extends the classical associahedron defined over BSTs or other Catalan-structures. Properties of tree- and more general graph associahedra have been studied in [14, 21, 15, 12, 13, 5].

Searching in trees and graphs has also been motivated with applications, including file system synchronisation [3, 46], software testing [3, 46], asymmetric communication protocols [38], VLSI layout [41], and assembly planning [33].

2 Preliminaries

Given a graph G, we denote by $V(G)$ its set of vertices, by $E(G)$ its set of edges, and by $\mathbb{C}(G)$ its set of connected components. If $v \in V(G)$, denote by $N_G(v)$ the set of neighbors of v in G, and $\deg_G(v) = |N_G(v)|$. For $S \subseteq V(G)$, denote by $G[S]$ the subgraph of G induced by S, and for brevity, $G - v = G[V(G) - \{v\}]$, and $G - S = G[V(G) - S]$.

The following observation is straightforward.

▶ **Observation 10.** *Let T be a search tree on $\mathcal{T}$, $w : V(\mathcal{T}) \to \mathbb{R}_{\geq 0}$, $m = w(\mathcal{T})$ and $r = \mathtt{root}(T)$. For each component $\mathcal{H} \in \mathbb{C}(\mathcal{T} - r)$, denote by $T_{\mathcal{H}}$ the subtree of T rooted at the unique child of r in $\mathcal{H}$. Then*

$$\mathtt{cost}_w(T) = m + \sum_{\mathcal{H} \in \mathbb{C}(\mathcal{T}-r)} \mathtt{cost}_w(T_{\mathcal{H}}) \geq m + \sum_{\mathcal{H} \in \mathbb{C}(\mathcal{T}-r)} \mathtt{OPT}(\mathcal{H}, w).$$

Projection of a search tree. For a rooted tree T and a vertex $v \in V(T)$, we denote by $\mathtt{Path}_T(v)$ the set of vertices on the path in T from $\mathtt{root}(T)$ to v, including both endpoints. Our upper bounds require the following notion of *projection* of a search tree.

▶ **Theorem 11.** *Let T be a search tree on $\mathcal{T}$ and $\mathcal{H}$ a connected subgraph of $\mathcal{T}$. There is a unique search tree $T|_{\mathcal{H}}$ on $\mathcal{H}$ such that for every $v \in V(\mathcal{H})$,*

$$\texttt{Path}_{T|_{\mathcal{H}}}(v) = \texttt{Path}_T(v) \cap V(\mathcal{H}).$$

▶ **Definition 12** (Projection)**.** *Let T be a search tree on $\mathcal{T}$ and $\mathcal{H}$ a connected subgraph of $\mathcal{T}$. We call $T|_{\mathcal{H}}$, whose existence is established by Theorem 11, the projection of T to $\mathcal{H}$.*

Tie-breaking. Our lower bounds require the following tie-breaking procedure.

▶ **Lemma 13.** *Let $\mathcal{T}$ be a tree, $w : V(\mathcal{T}) \to \mathbb{R}_{\geq 0}$ a weight function and let C be a centroid tree of $(\mathcal{T}, w)$. For every $\varepsilon > 0$ there exists a weight function $w' : V(\mathcal{T}) \to \mathbb{R}_{\geq 0}$ such that C is the unique centroid tree of $(\mathcal{T}, w')$ and $\|w' - w\|_\infty < \varepsilon$.*

Centroid and median. A certain concept of a *median vertex* of a tree has been used previously in the literature. If $\mathcal{T}$ is a tree with positive vertex weights and positive edge weights, then the median of $\mathcal{T}$ is the vertex v minimizing the quantity $\sum_{u \neq v} w(u) \cdot d(u, v)$. Here $w(u)$ is the weight of the vertex u, and $d(u, v)$ is the distance from u to v, i.e., the sum of the edge weights on the path from u to v. We show that if all edge-weights are 1, then medians are precisely centroids.

▶ **Lemma 14.** *Let $\mathcal{T}$ be a graph and w be a weight function on $V(\mathcal{T})$. For each $u \in V(\mathcal{T})$, define $W(u) = \sum_{v \in V(\mathcal{T})} d_{\mathcal{T}}(u, v) \cdot w(v)$, where $d_{\mathcal{T}}(u, v)$ denotes the number of edges on the path from u to v in $\mathcal{T}$. Then $c \in V(\mathcal{T})$ is a centroid of $(\mathcal{T}, w)$ if and only if $W(c)$ is minimal.*

Proofs to Theorem 11 and Lemmas 13 and 14 are available in the full paper [6].

3 Approximation guarantees for general trees

In this section we prove the general upper bound and lower bounds the approximation quality of centroid trees. We start with the upper bound.

▶ **Theorem 1.** *Let $\mathcal{T}$ be a tree, $w : V(\mathcal{T}) \to \mathbb{R}_{\geq 0}$, and $m = w(\mathcal{T})$. Then*

$$\texttt{cent}(\mathcal{T}, w) \leq 2 \cdot \texttt{OPT}(\mathcal{T}, w) - m.$$

We prove the following lemma.

▶ **Lemma 15.** *Let c be a centroid of $(\mathcal{T}, w)$ and $m = w(\mathcal{T})$. Then*

$$\texttt{OPT}(\mathcal{T}, w) \geq \frac{m}{2} + \frac{w(c)}{2} + \sum_{\mathcal{H} \in \mathbb{C}(\mathcal{T} - c)} \texttt{OPT}(\mathcal{H}, w). \tag{1}$$

Proof. Let T be an arbitrary search tree on $\mathcal{T}$. We will show that $\texttt{cost}_w(T)$ is at least the right hand side of Equation (1).

Denote $r = \texttt{root}(T)$. If $r = c$, using Observation 10, we have

$$\texttt{cost}_w(T) \geq m + \sum_{\mathcal{H} \in \mathbb{C}(\mathcal{T} - c)} \texttt{OPT}(\mathcal{H}, w),$$

which implies the claim. Assume therefore that $r \neq c$. Denote by $\mathcal{H}^*$ the connected component of $\mathcal{T} - c$ where r is. The contribution of vertices of $\mathcal{H}^*$ to $\texttt{cost}_w(T)$ is at least $\texttt{cost}_w(T|_{\mathcal{H}^*})$. For $\mathcal{H} \in \mathbb{C}(\mathcal{T} - c)$, $\mathcal{H} \neq \mathcal{H}^*$ and $v \in V(\mathcal{H})$, we have $\texttt{Path}_T(v) \supseteq \{r\} \cup \texttt{Path}_{T|_{\mathcal{H}}}(v)$, therefore

the contribution of vertices of every $\mathcal{H} \neq \mathcal{H}^*$ is at least $w(\mathcal{H}) + \texttt{cost}_w(T|_{\mathcal{H}})$. Finally, the contribution of c is at least $2w(c)$, since both $c, r \in \texttt{Path}_T(c)$. Summing the contributions of all the vertices, we get

$$\begin{aligned}\texttt{cost}_w(T) &\geq 2w(c) + \texttt{cost}_w(T|_{\mathcal{H}^*}) + \sum_{\mathcal{H}\neq\mathcal{H}^*} (w(\mathcal{H}) + \texttt{cost}_w(T|_{\mathcal{H}})) \\ &\geq m - w(\mathcal{H}^*) + w(c) + \sum_{\mathcal{H}} \texttt{OPT}(\mathcal{H}, w) \\ &\geq \frac{m}{2} + w(c) + \sum_{\mathcal{H}} \texttt{OPT}(\mathcal{H}, w),\end{aligned}$$

where the last inequality follows from c being a centroid. ◀

Proof of Theorem 1. The proof is by induction on the number of vertices. When $|V(\mathcal{T})| = 1$ we have

$$2 \cdot \texttt{OPT}(\mathcal{T}, w) - m = 2m - m = m = \texttt{cent}(\mathcal{T}, w), \quad \text{as required.}$$

Assume $|V(\mathcal{T})| > 1$. Let C be a centroid tree on $\mathcal{T}$ and $c = \texttt{root}(C)$. Using Observation 10 and the induction hypothesis we have:

$$\begin{aligned}\texttt{cost}_w(C) &\leq m + \sum_{\mathcal{H}\in\mathbb{C}(\mathcal{T}-c)} \texttt{cent}(\mathcal{H}, w) \\ &\leq m + \sum_{\mathcal{H}\in\mathbb{C}(\mathcal{T}-c)} (2 \cdot \texttt{OPT}(\mathcal{H}, w) - w(\mathcal{H})) \\ &= w(c) + 2 \cdot \sum_{\mathcal{H}\in\mathbb{C}(\mathcal{T}-c)} \texttt{OPT}(\mathcal{H}, w),\end{aligned}$$

therefore it is enough to show that

$$w(c) + 2 \cdot \sum_{\mathcal{H}\in\mathbb{C}(\mathcal{T}-c)} \texttt{OPT}(\mathcal{H}, w) \leq 2 \cdot \texttt{OPT}(\mathcal{T}, w) - m,$$

which is just a re-arrangement of Lemma 15. This concludes the proof. ◀

Next, we prove the lower bounds on the approximation quality of centroid trees, showing the tightness of Theorem 1. We note that in the *edge-query model* of search trees a 2-approximation was shown in [18] using techniques similar to those in the proof of Theorem 1. In contrast to that result, however, our approximation guarantee is best possible.

▶ **Theorem 2.**

(i) *For every $\varepsilon > 0$ there is a sequence of instances $(\mathcal{T}_n, w_n)$ with $w_n(\mathcal{T}_n) = 1$, and for every centroid tree C_n of $(\mathcal{T}_n, w_n)$*

$$\texttt{cost}_{w_n}(C_n) \geq 2 \cdot \texttt{OPT}(\mathcal{T}_n, w_n) - 1 - \varepsilon.$$

(ii) *There is a sequence of instances $(\mathcal{T}_n, w_n)$, where w_n is the uniform distribution on $V(\mathcal{T}_n)$, and for every centroid tree C_n of $(\mathcal{T}_n, w_n)$*

$$\lim_{n\to\infty} \frac{\texttt{cost}_{w_n}(C_n)}{\texttt{OPT}(\mathcal{T}_n, w_n)} = 2.$$

In both cases $\lim_{n\to\infty} \texttt{OPT}(\mathcal{T}_n, w_n) = \infty$.

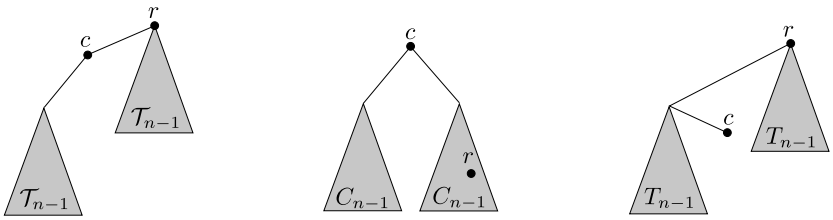


Figure 2 Illustration of the proof of Theorem 2(i). Vertex r is the root of $\mathcal{T}_n$. (*Left.*) The underlying tree $\mathcal{T}_n$. (*Middle.*) The entroid tree C_n of $\mathcal{T}_n$. (*Right.*) The search tree T_n.

As the proofs of both parts of Theorem 2 use the same construction with only slightly different analyses, we prove here only part (i). The proof of part (ii) appears in the full paper [6].

We proceed by constructing a sequence $(\mathcal{T}_n, w_n)$ such that for *some* centroid tree C_n,

$$\texttt{cost}_{w_n}(C_n) \geq 2 \cdot \texttt{OPT}(\mathcal{T}_n, w_n) - 1.$$

Using Lemma 13, we can then add an arbitrarily small perturbation to w_n to make C_n the *unique* centroid tree. (Observe that for every search tree T, $\texttt{cost}_w(T)$ is continuous in w, therefore so is $\texttt{OPT}(\mathcal{T}, w)$.)

The sequence $(\mathcal{T}_n, w_n)$ is constructed recursively as follows. For the sake of the construction we view $\mathcal{T}_n$ as a rooted tree. The base case $\mathcal{T}_0$ is a tree with a single vertex v and $w_0(v) = 1$. For $n > 0$, take two copies $(\mathcal{A}, w_\mathcal{A})$ and $(\mathcal{B}, w_\mathcal{B})$ of $(\mathcal{T}_{n-1}, w_{n-1})$. Connect the roots of $\mathcal{A}$ and $\mathcal{B}$ to a new vertex c. Finally, set $\texttt{root}(\mathcal{T}_n) = \texttt{root}(\mathcal{A})$ (see Figure 2). We define w_n as follows. (observe that $w_n(\mathcal{T}_n) = 1$, by induction on n.)

$$w_n(v) = \begin{cases} 0, & v = c \\ \frac{1}{2} w_\mathcal{A}(v), & v \in V(\mathcal{A}) \\ \frac{1}{2} w_\mathcal{B}(v), & v \in V(\mathcal{B}). \end{cases}$$

Let C_n denote the search tree on $\mathcal{T}_n$ obtained by setting c as the root and recursing. Observe that C_n is a centroid tree of $(\mathcal{T}_n, w_n)$.

▶ **Lemma 16.** *The following hold*

(a) $\texttt{cost}_{w_n}(C_n) = n + 1$,

(b) $\lim_{n\to\infty} \texttt{OPT}(\mathcal{T}_n, w_n) = \infty$.

Proof. Let $c_n = \texttt{cost}_{w_n}(C_n)$. Clearly, $c_0 = 1$. Assume $n > 0$. Let $C_\mathcal{A}$ and $C_\mathcal{B}$ be search trees on $\mathcal{A}$ and $\mathcal{B}$ respectively, each a copy of C_{n-1}. By construction of C_n we have

$$c_n = 1 + \frac{1}{2}\texttt{cost}_{w_\mathcal{A}}(C_\mathcal{A}) + \frac{1}{2}\texttt{cost}_{w_\mathcal{B}}(C_\mathcal{B}) = 1 + c_{n-1},$$

and (a) follows by induction.

Using (a) and Theorem 1, part (b) follows:

$$\texttt{OPT}(\mathcal{T}_n, w_n) \geq \frac{c_n + 1}{2} = \frac{n}{2} + 1 \to \infty. \qquad \blacktriangleleft$$

Next, in order to bound $\texttt{OPT}(\mathcal{T}_n, w_n)$ from above, we construct a sequence of search trees T_n on $\mathcal{T}_n$. For $n = 0$, tree T_0 is a single vertex. Assume $n > 0$. Let $\mathcal{A}$, $\mathcal{B}$, and c be as in the definition of $\mathcal{T}_n$. Let $T_\mathcal{A}$ and $T_\mathcal{B}$ be search trees over $\mathcal{A}$ and $\mathcal{B}$ respectively, each a copy of T_{n-1}. Denote $r_\mathcal{A} = \texttt{root}(\mathcal{A})$ and $r_\mathcal{B} = \texttt{root}(\mathcal{B})$. Tree T_n is obtained by adding an edge from $r_\mathcal{A}$ to $r_\mathcal{B}$ and an edge from $r_\mathcal{B}$ to c, and setting $\texttt{root}(T_n) = r_\mathcal{A}$.

▶ **Lemma 17.** $\texttt{cost}_{w_n}(T_n) = \frac{n}{2} + 1$.

Proof. Denote $t_n = \texttt{cost}_{w_n}(T_n)$. Clearly $t_0 = 1$. Assume $n > 0$. The contribution of vertices of $\mathcal{A}$ to t_n is exactly $\frac{1}{2}\texttt{cost}_{w_\mathcal{A}}(T_\mathcal{A}) = \frac{t_{n-1}}{2}$. Since r is an ancestor of all vertices in $\mathcal{B}$, the contribution of these vertices to t_n is exactly $\frac{1}{2}(1 + \texttt{cost}_{w_\mathcal{B}}(T_\mathcal{B})) = \frac{1+t_{n-1}}{2}$. Summing the contribution of all vertices, we get $t_n = t_{n-1} + \frac{1}{2}$ and the claim follows. ◀

Proof of Theorem 2(i). By Lemma 17, $\texttt{OPT}(\mathcal{T}_n, w_n) \leq \frac{n}{2} + 1$. Together with Lemma 16, the claim follows. ◀

4 Approximation guarantees for trees with bounded degrees

In this section we show the upper and lower bounds on the approximation quality of centroid trees when the underlying tree $\mathcal{T}$ has bounded degree. We start with the upper bound.

▶ **Theorem 3.** *Let $\mathcal{T}$ be a tree, $w : V(\mathcal{T}) \to \mathbb{R}_{\geq 0}$, and let Δ be the* maximum degree *of $\mathcal{T}$. Then*

$$\texttt{cent}(\mathcal{T}, w) \leq \left(2 - \frac{1}{2^\Delta}\right) \cdot \texttt{OPT}(\mathcal{T}, w).$$

For simplicity, in what follows we omit the weight function w from notations.

▶ **Lemma 18.** *Let C be a centroid tree of $\mathcal{T}$ such that $\texttt{cost}(C) = \texttt{cent}(\mathcal{T})$. Let $P = (v_0, v_1, \ldots, v_p)$ be any path in C. Then*

$$\texttt{cent}(\mathcal{T}) \leq \left(2 - \frac{1}{2^p}\right) m + \sum_{\mathcal{H} \in \mathbb{C}(\mathcal{T} - P)} \texttt{cent}(\mathcal{H}).$$

Proof. By induction on p. For $p = 0$ we have

$$\texttt{cent}(\mathcal{T}) = m + \sum_{\mathcal{H} \in \mathbb{C}(\mathcal{T} - v_0)} \texttt{cent}(\mathcal{H}), \quad \text{as required.}$$

Assume now $p > 0$. Denote by $\tilde{\mathcal{T}}$ the connected component of $\mathcal{T} - v_0$ where v_1 is. Denote $\tilde{P} = (v_1, \ldots, v_p)$, and $\tilde{m} = w(\tilde{\mathcal{T}})$. Observe that $\tilde{m} \leq m/2$ and that $\mathbb{C}(\mathcal{T} - P) = \mathbb{C}(\tilde{\mathcal{T}} - \tilde{P}) \uplus (\mathbb{C}(\mathcal{T} - v_0) - \{\tilde{\mathcal{T}}\})$. By the induction hypothesis we have

$$\texttt{cent}(\tilde{\mathcal{T}}) \leq \left(2 - \frac{1}{2^{p-1}}\right) \tilde{m} + \sum_{\mathcal{H} \in \mathbb{C}(\tilde{\mathcal{T}} - \tilde{P})} \texttt{cent}(\mathcal{H}), \quad \text{therefore}$$

$$\begin{aligned}
\texttt{cent}(\mathcal{T}) &= m + \texttt{cent}(\tilde{\mathcal{T}}) + \sum_{\substack{\mathcal{H} \in \mathbb{C}(\mathcal{T} - v_0) \\ \mathcal{H} \neq \tilde{\mathcal{T}}}} \texttt{cent}(\mathcal{H}) \\
&\leq m + \left(2 - \frac{1}{2^{p-1}}\right) \tilde{m} + \sum_{\mathcal{H} \in \mathbb{C}(\tilde{\mathcal{T}} - \tilde{P})} \texttt{cent}(\mathcal{H}) + \sum_{\substack{\mathcal{H} \in \mathbb{C}(\mathcal{T} - v_0) \\ H \neq \tilde{\mathcal{T}}}} \texttt{cent}(\mathcal{H}) \\
&\leq m + \left(2 - \frac{1}{2^{p-1}}\right) \frac{m}{2} + \sum_{\mathcal{H} \in \mathbb{C}(\mathcal{T} - P)} \texttt{cent}(\mathcal{H}) \\
&= \left(2 - \frac{1}{2^p}\right) m + \sum_{\mathcal{H} \in \mathbb{C}(\mathcal{T} - P)} \texttt{cent}(\mathcal{H}), \quad \text{as required.}
\end{aligned}$$

◀

Proof of Theorem 3. The proof is by induction on $|V(\mathcal{T})|$. Let T be any search tree on $\mathcal{T}$. We will show that $\texttt{cent}(\mathcal{T}) \leq \left(2 - \frac{1}{2^\Delta}\right)\texttt{cost}(T)$.

Denote $r = \texttt{root}(T)$. Let C be a centroid tree on $\mathcal{T}$ with $\texttt{cost}(C) = \texttt{cent}(\mathcal{T})$. Denote by $v_0, v_1, \ldots, v_d = r$ the vertices along the path to r in C. Denote $\mathcal{T}_i = \mathcal{T}[V(C_{v_i})]$. Observe that $r \in V(\mathcal{T}_d) \subseteq \cdots \subseteq V(\mathcal{T}_0) = V(\mathcal{T})$. For $i < d$, denote by $\mathcal{K}_i$ the connected component of $\mathcal{T} - r$ where v_i is. Denote by s_i the unique child of r in T such that $s_i \in V(\mathcal{K}_i)$, i.e., $V(T_{s_i}) = V(\mathcal{K}_i)$. Finally, denote by p the minimal i for which one of the following holds:

1. $v_i = r$, i.e., $i = d$,
2. $s_i \in V(\mathcal{T}_{i+1})$, or
3. there exists $j < i$ such that $\mathcal{K}_j = \mathcal{K}_i$, i.e., $s_j = s_i$.

Note that from the third condition above it follows that $p \leq \Delta$. Denote $P = (v_0, \ldots, v_p)$. We will prove the following.

▷ Claim 19.

$$\texttt{cost}(T) \geq m + \sum_{\mathcal{H} \in \mathbb{C}(\mathcal{T} - P)} \texttt{cost}(T|_{\mathcal{H}}).$$

Assume for now that Claim 19 holds. Using Lemma 18, the fact that $p \leq \Delta$, the induction hypothesis and Claim 19, we have

$$\begin{aligned}
\texttt{cent}(\mathcal{T}) &\leq \left(2 - \frac{1}{2^p}\right) m + \sum_{\mathcal{H} \in \mathbb{C}(\mathcal{T} - P)} \texttt{cent}(\mathcal{H}) \\
&\leq \left(2 - \frac{1}{2^\Delta}\right) m + \sum_{\mathcal{H} \in \mathbb{C}(\mathcal{T} - P)} \left(2 - \frac{1}{2^\Delta}\right) \texttt{cost}(T|_{\mathcal{H}}) \\
&\leq \left(2 - \frac{1}{2^\Delta}\right) \texttt{cost}(T).
\end{aligned}$$

◀

Proof of Claim 19. The proof breaks into cases according to the defining condition of p.

Case 1. Assume $v_p = r$. For every $\mathcal{H} \in \mathbb{C}(\mathcal{T} - P)$ and $v \in V(\mathcal{H})$ we have $\texttt{Path}_T(v) \supseteq \{r\} \uplus \texttt{Path}_{T|_{\mathcal{H}}}(v)$. The contribution of such v to $\texttt{cost}(T)$ is therefore at least $w(v)(1 + |\texttt{Path}_{T|_{\mathcal{H}}}(v)|)$. The contribution of each v_i to $\texttt{cost}(T)$ is at least $w(v_i)$. Summing the contribution of all vertices yields the required result. See Figure 3.

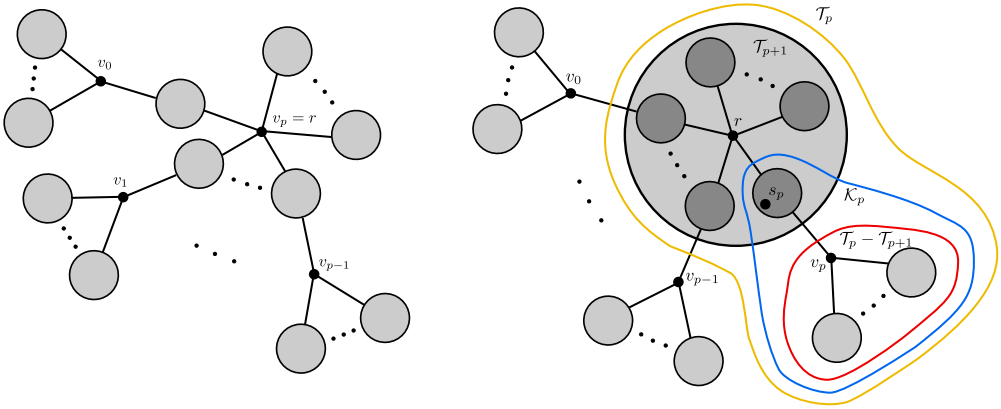


Figure 3 Illustration of the proof of Claim 19. Connected components of $\mathcal{T} - P$ are represented by light gray circles. (*Left.*) Case 1. (*Right.*) Case 2. Vertices in $\mathcal{T} - \mathcal{T}_p$ have r as ancestor. Vertices in $\mathcal{T}_p - \mathcal{T}_{p+1}$ have both r and s_p as ancestors.

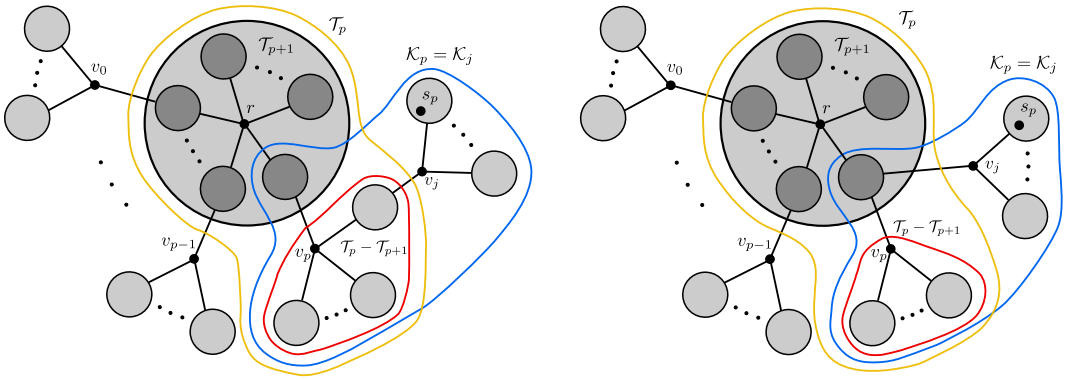


Figure 4 Case 3 in the proof of Claim 19. (*Left.*) Sub-case where v_p is in the path in $\mathcal{T}$ between r and v_j. (*Right.*) Complementary sub-case. Connected components of $\mathcal{T} - P$ are represented by light gray circles. In both sub-cases, vertices in $\mathcal{T}_p - \mathcal{T}_{p+1}$ have both r and s_p as ancestors.

Case 2. Assume $s_p \in V(\mathcal{T}_{p+1})$. Denote by $\mathbb{C}_1$ the set of connected components of $\mathcal{T} - P$ that are not contained in $\mathcal{T}_p$. Denote $\mathbb{C}_2 = \mathbb{C}(\mathcal{T} - P) - \mathbb{C}_1$ (see Figure 3). For every $\mathcal{H} \in \mathbb{C}_1$, if $v \in V(\mathcal{H})$, then $\texttt{Path}_T(v) \supseteq \{r\} \cup \texttt{Path}_{T|_\mathcal{H}}(v)$. Therefore the contribution of vertices in $V(\mathcal{T} - \mathcal{T}_p)$ to $\texttt{cost}(T)$ is at least

$$m - w(\mathcal{T}_p) + \sum_{\mathcal{H} \in \mathbb{C}_1} \texttt{cost}(T|_\mathcal{H}). \tag{2}$$

For vertices $v \in V(\mathcal{T}_p - \mathcal{T}_{p+1})$ we have $\{r, s_p\} \subseteq \texttt{Path}_T(v)$. Therefore, using the fact that $w(\mathcal{T}_p - \mathcal{T}_{p+1}) \geq \frac{w(\mathcal{T}_p)}{2}$, the contribution of vertices in $V(\mathcal{T}_p)$ to $\texttt{cost}(T)$ is at least

$$2 \cdot w(\mathcal{T}_p - \mathcal{T}_{p+1}) + \sum_{\mathcal{H} \in \mathbb{C}_2} \texttt{cost}(T|_\mathcal{H}) \geq w(\mathcal{T}_p) + \sum_{\mathcal{H} \in \mathbb{C}_2} \texttt{cost}(T|_\mathcal{H}). \tag{3}$$

Summing Equation (2) and Equation (3) yields the required result.

Case 3. Assume that there is a $j < p$ such that $\mathcal{K}_p = \mathcal{K}_j$. Since p is minimal, we further assume that Case 2 did not occur for indices smaller that p. In particular, $s_p = s_j \notin V(\mathcal{T}_p)$. As in Case 2, the contribution of vertices in $\mathcal{T} - \mathcal{T}_p$ to $\texttt{cost}(T)$ is at least as in Equation (2). We have $r \notin V(\mathcal{T}_p - \mathcal{T}_{p+1})$ and $v_p \in V(\mathcal{T}_p - \mathcal{T}_{p+1}) \cap V(\mathcal{K}_p) \neq \emptyset$, therefore, since $\mathcal{T}_p - \mathcal{T}_{p+1}$ is connected, $V(\mathcal{T}_p - \mathcal{T}_{p+1}) \subseteq V(\mathcal{K}_p)$ (see Figure 4). It follows that vertices in $\mathcal{T}_p - \mathcal{T}_{p+1}$ have both r and s_p as ancestors. Since $w(\mathcal{T}_p - \mathcal{T}_{p+1}) \geq \frac{w(\mathcal{T}_p)}{2}$, the contribution of vertices in $\mathcal{T}_p$ is at least as in Equation (3). As in Case 2, the result follows by summing Equation (2) and Equation (3). ◁

We now proceed to the lower bounds.

▶ **Theorem 4.** *Let $\Delta \geq 3$ be integer.*

(i) *There is a sequence of instances $(\mathcal{T}_n, w_n)$ such that $\mathcal{T}_n$ has maximum degree at most Δ, and for every centroid tree C_n of $(\mathcal{T}_n, w_n)$*

$$\lim_{n \to \infty} \frac{\texttt{cost}_{w_n}(C_n)}{\texttt{OPT}(\mathcal{T}_n, w_n)} = 2 - \frac{1}{2^\Delta}.$$

(ii) *There is a sequence of instances $(\mathcal{T}_n, w_n)$ such that $\mathcal{T}_n$ has maximum degree at most Δ, $\lim_{n \to \infty} \texttt{OPT}(\mathcal{T}_n, w_n) = \infty$, $w_n(\mathcal{T}_n) = 1$, and for every centroid tree C_n of $(\mathcal{T}_n, w_n)$*

$$\texttt{cost}_{w_n}(C_n) \geq \left(2 - \frac{4}{2^\Delta}\right) \cdot \texttt{OPT}(\mathcal{T}_n, w_n) - 1.$$

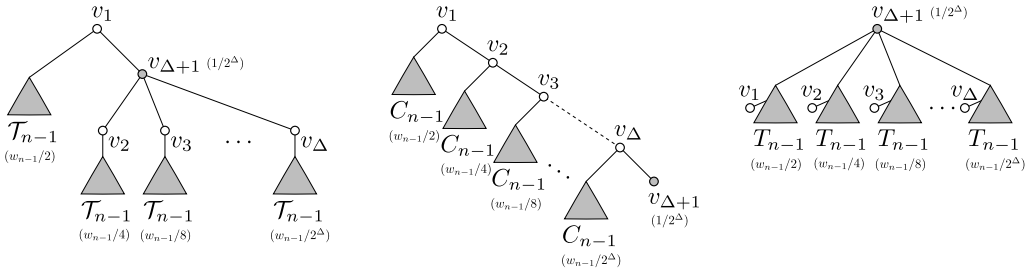


Figure 5 Illustration of Theorem 4(i). (*Left.*) The underlying tree $\mathcal{T}_n$. (*Middle.*) The centroid tree C_n. (*Right.*) The search tree T_n.

Part (i). As in the proof of Theorem 2, it will suffice to prove Theorem 4(i) for *some* centroid tree C_n. Using Lemma 13, we can then add arbitrarily small perturbation to w_n, making C_n the unique centroid tree.

The sequence $(\mathcal{T}_n, w_n)$ of Theorem 4(i) is constructed recursively as follows. For the sake of the construction we regard $\mathcal{T}_n$ as a rooted tree. $\mathcal{T}_0$ is simply a single vertex v (which is the root) and $w_0(v) = 1$. For $n > 0$, $\mathcal{T}_n$ is constructed from Δ copies of $\mathcal{T}_{n-1}$ and $\Delta + 1$ additional vertices, $v_1, \ldots, v_{\Delta+1}$, as shown in Figure 5 (left). The i'th copy of $\mathcal{T}_{n-1}$ gets the weight function $w_{n-1}/2^i$. We set $w_n(v_i) = 0$ for $1 \le i \le \Delta$ and $w_n(v_{\Delta+1}) = 1/2^\Delta$. Finally, we set $\mathrm{root}(\mathcal{T}_n) = v_1$. By induction, $\mathcal{T}_n$ has maximal degree Δ and w_n is a distribution on $V(\mathcal{T}_n)$.

Let C_n be a search tree on $\mathcal{T}_n$ defined recursively as follows. Connect the vertices $v_1, \ldots, v_{\Delta+1}$ to form a path and set v_1 as the root of C_n. Continue recursively on each connected component of $\mathcal{T} - \{v_1, \ldots, v_{\Delta+1}\}$. (See Figure 5 (middle).) Observe that C_n is a centroid tree of $(\mathcal{T}_n, w_n)$.

▶ **Lemma 20.** *For all n,*

$$\mathtt{cost}_{w_n}(C_n) = 2^{\Delta+1} - 1 - (2^{\Delta+1} - 2)\left(1 - \frac{1}{2^\Delta}\right)^n. \tag{4}$$

Proof. Denote $c_n = \mathtt{cost}_{w_n}(C_n)$. Clearly $c_0 = 1$ as required. Let $n > 0$. For each i, the subtree of all the descendants of v_i in C_n has weight $1/2^{i-1}$. Therefore

$$c_n = \sum_{i=1}^{\Delta+1} \frac{1}{2^{i-1}} + \sum_{i=1}^{\Delta} \frac{1}{2^i} c_{n-1} = 2 - \frac{1}{2^\Delta} + \left(1 - \frac{1}{2^\Delta}\right) c_{n-1}.$$

It is straightforward to verify that the right hand side of Equation (4) is the solution to the recursive formula above. ◀

In order to upper bound $\mathtt{OPT}(\mathcal{T}_n, w_n)$ we construct recursively a search tree T_n on $\mathcal{T}_n$. For $n > 0$, T_n is constructed by setting $v_{\Delta+1}$ as root and attaching to it Δ copies of T_{n-1}. The vertices $v_1, \ldots, v_\Delta$ are finally attached as leaves of T_n, each at its unique valid place. See Figure 5 (right).

▶ **Lemma 21.** *For all n,*

$$\mathtt{cost}_{w_n}(T_n) = 2^\Delta - 2^\Delta \left(1 - \frac{1}{2^\Delta}\right)^{n+1}. \tag{5}$$

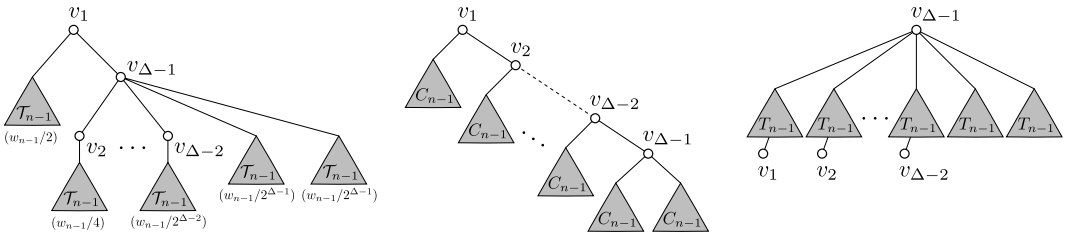


Figure 6 An illustration of Theorem 4(ii). (*Left.*) The underlying tree $\mathcal{T}_n$. (*Middle.*) The centroid tree C_n. (*Right.*) The search tree T_n.

Proof. Denote $t_n = \texttt{cost}_{w_n}(T_n)$. We have $t_0 = 1$. For $n > 0$, t_n obeys the recursive relation

$$t_n = 1 + \sum_{i=1}^{\Delta} \frac{1}{2^i} t_{n-1} = 1 + \left(1 - \frac{1}{2^\Delta}\right) t_{n-1},$$

of which the right hand side of Equation (5) is the solution. ◀

Proof of Theorem 4(i). Using Lemma 20 and Lemma 21,

$$\frac{\texttt{cost}_{w_n}(C_n)}{\texttt{OPT}(\mathcal{T}_n, w_n)} \geq \frac{\texttt{cost}_{w_n}(C_n)}{\texttt{cost}_{w_n}(T_n)} \to 2 - \frac{1}{2^\Delta}.$$ ◀

Observe that, as discussed in Section 1, $\texttt{OPT}(\mathcal{T}_n, w_n)/w_n(\mathcal{T}_n)$ is bounded.

Part (ii). To prove Theorem 4(ii), we repeat the recursive construction of Theorem 4(i) with a slight modification. As before, $\mathcal{T}_0$ is a tree with a single vertex. For $n > 0$, $(\mathcal{T}_n, w_n)$ is constructed from Δ weighted copies of $(\mathcal{T}_{n-1}, w_{n-1})$ and $\Delta - 1$ additional vertices, $v_1, \ldots, v_{\Delta-1}$, each with weight 0, as shown in Figure 6 (left). We set $\texttt{root}(\mathcal{T}_n) = v_1$.

As before, the search tree C_n is defined by connecting the vertices $v_1, \ldots, v_{\Delta-1}$ to a path, setting v_1 as root and recursing on the remaining connected component. Observe that C_n is a centroid tree of $(\mathcal{T}_n, w_n)$. (See Figure 6 (middle).) The search tree T_n is defined by setting $v_{\Delta-1}$ as root, attaching to it Δ copies of T_{n-1}, then adding the vertices $v_1, \ldots, v_{\Delta-2}$ as leaves, each at its unique valid place. See Figure 6 (right).

▶ **Lemma 22.** *For all n,*
(a) $\texttt{cost}_{w_n}(C_n) = \left(2 - \frac{4}{2^\Delta}\right) \cdot n + 1$,
(b) $\texttt{cost}_{w_n}(T_n) = n + 1$.

The proof follows an analysis similar to that of Lemma 20 and Lemma 21.

Proof. Denote $c_n = \texttt{cost}_{w_n}(C_n)$ and $t_n = \texttt{cost}_{w_n}(T_n)$. Clearly $c_0 = t_0 = 1$. For $n > 0$ we have

$$c_n = \sum_{i=1}^{\Delta-1} \frac{1}{2^{i-1}} + \sum_{i=1}^{\Delta-2} \frac{1}{2^i} c_{n-1} + 2\frac{1}{2^{\Delta-1}} c_{n-1} = 2 - \frac{4}{2^\Delta} + c_{n-1}, \quad \text{and}$$

$$t_n = \sum_{i=1}^{\Delta-2} \frac{1}{2^i}(t_{n-1} + 1) + 2\frac{1}{2^{\Delta-1}}(t_{n-1} + 1) = 1 + t_{n-1},$$

and the lemma follows by induction. ◀

Proof of Theorem 4(ii). The fact that $\lim_{n\to\infty} \texttt{OPT}(\mathcal{T}_n, w_n) = \infty$ follows from Lemma 22 and Theorem 3 (or Theorem 1). Using Lemma 22 again, we have

$$\texttt{cost}_{w_n}(C_n) \geq \left(2 - \frac{4}{2^\Delta}\right) \texttt{OPT}(\mathcal{T}_n, w_n) - 1 + \frac{4}{2^\Delta}.$$

Using Lemma 13, for each n we can add small enough perturbation to w_n such that C_n is the unique centroid tree and the claimed bound holds. ◀

5 Computing centroid trees

In this section, we show how to compute centroid trees using the *top tree* framework of Alstrup, Holm, de Lichtenberg, and Thorup [1]. *Top trees* are a data structure used to maintain dynamic forests under insertion and deletion of edges. Most importantly, they expose a simple interface that allows the user to maintain information in the trees of the forest. For this, the user only needs to implement a small number of internal operations.

Alstrup et al. in particular show how to maintain the *median* of trees in $\mathcal{O}(\log n)$ per operation, see Section 2 for the definition of the median. As mentioned before, if all edge-weights are 1, then medians are precisely centroids (see Lemma 14).

▶ **Theorem 23** ([1, Theorem 3.6]). *We can maintain a forest with positive vertex weights on n vertices under the following operations:*

- *Add an edge between two given vertices u, v that are not in the same connected component;*
- *Remove an existing edge;*
- *Change the weight of a vertex;*
- *Retrieve a pointer to the tree containing a given vertex;*
- *Find the centroid of a given tree in the forest.*

Each operation requires $\mathcal{O}(\log n)$ time. A forest without edges and with n arbitrarily weighted vertices can be initialized in $\mathcal{O}(n)$ time.

Note that Theorem 23 only admits *positive* vertex weights, whereas we allowed zero-weight vertices. We show how to handle this problem in the full paper [6].

We now show how to use Theorem 23 to construct a centroid tree in $\mathcal{O}(n \log n)$ time.

▶ **Theorem 24.** *Given a tree $\mathcal{T}$ on n vertices and a positive weight function w, we can compute a centroid tree of $(\mathcal{T}, w)$ in $\mathcal{O}(n \log n)$ time.*

Proof. First build a top tree on $\mathcal{T}$ by adding the edges one-by-one, in $\mathcal{O}(n \log n)$ time. Then, find the centroid c, and remove each incident edge. Then, recurse on each newly created tree (except for the one containing only c). The algorithm finds each vertex precisely once and removes each edge precisely once, for a total running time of $\mathcal{O}(n \log n)$. ◀

Output-sensitive algorithm. We improve the algorithm given above to run in time $\mathcal{O}(n \log h)$, where n is the number of vertices in $\mathcal{T}$ and h is the height of the computed centroid tree.

The main idea of the algorithm is inspired by the linear-time algorithm for *unweighted* centroids by Della Giustina, Prezza, and Venturini [28], with a number of further technical challenges. Instead of building a top tree on the whole tree $\mathcal{T}$, we first split $\mathcal{T}$ into connected subgraphs of size roughly h, and build a top tree on each component. Contracting each component into a single vertex yields *super-vertices* in a *super-tree*. Each search for a centroid consists of a global search and a local search: We first find the super-vertex containing the

centroid, then we find the centroid within that super-vertex. After finding the centroid, we remove it, which may split up the super-vertex into multiple super-vertices with a top tree each, and also may split the super-tree into a super-forest. Finally, we recurse on each component of the super-forest.

It can be seen that the total number of top tree operations needed is $\mathcal{O}(n)$. Since the top trees each contain only h vertices, a top tree operation takes $\mathcal{O}(\log h)$ time, for a total of $\mathcal{O}(n \log h)$. Detailed description and analysis of the algorithm, as well as the matching lower bound, appear in the full paper [6].

6 Conclusions

We showed that the average search time in a centroid tree is larger by at most a factor of 2 than the smallest possible average search time in an STT and that this bound is tight. We also showed that centroid trees can be computed in $\mathcal{O}(n \log h)$ time where h is the height of the centroid tree. Perhaps the most intriguing question is to determine whether the problem of computing an optimal STT is in P. A secondary goal would be to achieve an approximation ratio better than 2 in near linear time. (The running time of the STT's of Berendsohn and Kozma [7] degrade as $\mathcal{O}(n^{2k+1})$ for a $\left(1 + \frac{1}{k}\right)$-approximation.) As for centroid trees, a remaining question is whether they can be computed in $\mathcal{O}(n)$ time whenever the spread of the weight function is $\sigma \in \mathcal{O}(n)$.

A special case in which high quality approximation can be efficiently found is when an α-centroid tree exists for $\alpha < \frac{1}{2}$. This case can be recognized and handled in near linear time using our algorithm. (Observe that an α-centroid tree for $\alpha < \frac{1}{2}$ is also the unique $\frac{1}{2}$-centroid tree.) Theorem 7(ii) gives strong approximation guarantees for this case, yielding the *optimum* when $\alpha \leq \frac{1}{3}$. It is an interesting question whether the bounds can be improved for α in the range $\left(\frac{1}{3}, \frac{1}{2}\right)$, i.e., whether Theorem 7(ii) is tight.

A small gap remains in the exact approximation ratio of centroid trees when $\mathcal{T}$ has maximum degree Δ and `OPT` is unbounded, i.e., between the upper bound $(2 - \frac{1}{2^\Delta})$ of Theorem 3 and the lower bound $(2 - \frac{4}{2^\Delta})$ of Theorem 4(ii).

References

1 Stephen Alstrup, Jacob Holm, Kristian De Lichtenberg, and Mikkel Thorup. Maintaining information in fully dynamic trees with top trees. *ACM Trans. Algorithms*, 1(2):243–264, October 2005. doi:10.1145/1103963.1103966.

2 Bengt Aspvall and Pinar Heggernes. Finding minimum height elimination trees for interval graphs in polynomial time. *BIT*, 34:484–509, 1994.

3 Yosi Ben-Asher, Eitan Farchi, and Ilan Newman. Optimal search in trees. *SIAM J. Comput.*, 28(6):2090–2102, 1999. doi:10.1137/S009753979731858X.

4 Michael A. Bender, Martin Farach-Colton, and Bradley C. Kuszmaul. Cache-oblivious string b-trees. In *ACM SIGMOD-SIGACT-SIGART*, pages 233–242, 2006.

5 Benjamin Aram Berendsohn. The diameter of caterpillar associahedra. In Artur Czumaj and Qin Xin, editors, *18th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2022, June 27-29, 2022, Tórshavn, Faroe Islands*, volume 227 of *LIPIcs*, pages 14:1–14:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SWAT.2022.14.

6 Benjamin Aram Berendsohn, Ishay Golinsky, Haim Kaplan, and László Kozma. Fast approximation of search trees on trees with centroid trees, 2022. arXiv:2209.08024.

7 Benjamin Aram Berendsohn and László Kozma. Splay trees on trees. In *SODA*, pages 1875–1900, 2022.

8 Hans L. Bodlaender, Jitender S. Deogun, Klaus Jansen, Ton Kloks, Dieter Kratsch, Haiko Müller, and Zsolt Tuza. Rankings of graphs. *SIAM Journal on Discrete Mathematics*, 11(1):168–181, 1998.

9 H.L. Bodlaender, J.R. Gilbert, H. Hafsteinsson, and T. Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *Journal of Algorithms*, 18(2):238–255, 1995. doi:10.1006/jagm.1995.1009.

10 Prosenjit Bose, Jean Cardinal, John Iacono, Grigorios Koumoutsos, and Stefan Langerman. Competitive online search trees on trees. In *SODA*, pages 1878–1891, 2020.

11 Gerth Stølting Brodal, Rolf Fagerberg, Christian N. S. Pedersen, and Anna Östlin. The complexity of constructing evolutionary trees using experiments. In *ICALP*, pages 140–151. Springer, 2001.

12 Jean Cardinal, Stefan Langerman, and Pablo Pérez-Lantero. On the diameter of tree associahedra. *Electron. J. Comb.*, 25(4):P4.18, 2018. URL: http://www.combinatorics.org/ojs/index.php/eljc/article/view/v25i4p18, doi:10.37236/7762.

13 Jean Cardinal, Lionel Pournin, and Mario Valencia-Pabon. Bounds on the diameter of graph associahedra. In *Proceedings of the XI Latin and American Algorithms, Graphs and Optimization Symposium (LAGOS)*, volume 195 of *Procedia Computer Science*, pages 239–247. Elsevier, 2021.

14 Michael Carr and Satyan L. Devadoss. Coxeter complexes and graph-associahedra. *Topology and its Applications*, 153(12):2155–2168, 2006.

15 Cesar Ceballos, Thibault Manneville, Vincent Pilaud, and Lionel Pournin. Diameters and geodesic properties of generalizations of the associahedron. In *Proceedings of the 27th International Conference on Formal Power Series and Algebraic Combinatorics (FPSAC)*, pages 345–356, 2015.

16 Panagiotis Charalampopoulos, Pawel Gawrychowski, Shay Mozes, and Oren Weimann. An almost optimal edit distance oracle. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 48:1–48:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.48.

17 Ferdinando Cicalese, Tobias Jacobs, Eduardo Laber, and Marco Molinaro. On the complexity of searching in trees and partially ordered structures. *Theor. Comput. Sci.*, 412(50):6879–6896, 2011. doi:10.1016/j.tcs.2011.08.042.

18 Ferdinando Cicalese, Tobias Jacobs, Eduardo Laber, and Marco Molinaro. Improved approximation algorithms for the average-case tree searching problem. *Algorithmica*, 68(4):1045–1074, 2014. doi:10.1007/s00453-012-9715-6.

19 Erik D. Demaine, Dion Harmon, John Iacono, and Mihai Pătraşcu. Dynamic optimality - almost. *SIAM J. Comput.*, 37(1):240–251, 2007. doi:10.1137/S0097539705447347.

20 Jitender S Deogun, Ton Kloks, Dieter Kratsch, and Haiko Müller. On vertex ranking for permutation and other graphs. In *STACS 1994*, pages 747–758. Springer, 1994.

21 Satyan L. Devadoss. A realization of graph associahedra. *Discrete Mathematics*, 309(1):271–276, 2009.

22 Iain S Duff, Albert Maurice Erisman, and John Ker Reid. *Direct methods for sparse matrices*. Oxford University Press, 2017.

23 Guy Even and Shakhar Smorodinsky. Hitting sets online and unique-max coloring. *Discret. Appl. Math.*, 178:71–82, 2014. doi:10.1016/j.dam.2014.06.019.

24 Paolo Ferragina. On the weak prefix-search problem. *Theor. Comput. Sci.*, 483:75–84, 2013. doi:10.1016/j.tcs.2012.06.011.

25 Paolo Ferragina and Rossano Venturini. Compressed cache-oblivious string b-tree. *ACM Trans. Algorithms*, 12(4):52:1–52:17, 2016. doi:10.1145/2903141.

26 Greg N. Frederickson and Donald B Johnson. Finding kth paths and p-centers by generating and searching good data structures. *Journal of Algorithms*, 4(1):61–80, 1983.

27 Travis Gagie, Danny Hermelin, Gad M Landau, and Oren Weimann. Binary jumbled pattern matching on trees and tree-like structures. *Algorithmica*, 73(3):571–588, 2015.

28 Davide Della Giustina, Nicola Prezza, and Rossano Venturini. A new linear-time algorithm for centroid decomposition. In *Proceedings of the 26th International Symposium on String Processing and Information Retrieval (SPIRE)*, volume 11811 of *Lecture Notes in Computer Science*, pages 274–282. Springer, 2019.

29 Michael T. Goodrich and Roberto Tamassia. Dynamic trees and dynamic point location. *SIAM J. Comput.*, 28(2):612–636, 1998. doi:10.1137/S0097539793254376.

30 Leonidas J. Guibas, John Hershberger, Daniel Leven, Micha Sharir, and Robert Endre Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987. doi:10.1007/BF01840360.

31 Brent Heeringa, Marius Catalin Iordan, and Louis Theran. Searching in dynamic tree-like partial orders. In *WADS 2011*, volume 6844 of *Lecture Notes in Computer Science*, pages 512–523. Springer, 2011. doi:10.1007/978-3-642-22300-6_43.

32 D.S. Hirschberg, L.L. Larmore, and M. Molodowitch. Subtree weight ratios for optimal binary search trees. Technical Report TR 86-02, ICS Department, University of California, Irvine, 1986.

33 Ananth V. Iyer, H. Donald Ratliff, and Gopalakrishnan Vijayan. Optimal node ranking of trees. *Inf. Process. Lett.*, 28(5):225–229, 1988. doi:10.1016/0020-0190(88)90194-9.

34 Camille Jordan. Sur les assemblages de lignes. *Journal für die reine und angewandte Mathematik*, 70:185–190, 1869.

35 Meir Katchalski, William McCuaig, and Suzanne Seager. Ordered colourings. *Discrete Mathematics*, 142(1-3):141–154, 1995.

36 Donald E. Knuth. Optimum binary search trees. *Acta Informatica*, 1(1):14–25, 1971. doi:10.1007/BF00264289.

37 Tomasz Kociumaka, Jakub Pachocki, Jakub Radoszewski, Wojciech Rytter, and Tomasz Waleń. Efficient counting of square substrings in a tree. *Theoretical Computer Science*, 544:60–73, 2014.

38 Eduardo Laber and Marco Molinaro. An approximation algorithm for binary searching in trees. *Algorithmica*, 59(4):601–620, 2011. doi:10.1007/s00453-009-9325-0.

39 Eduardo Laber and Loana Nogueira. Fast searching in trees. *Electronic Notes in Discrete Mathematics*, 7:90–93, 2001. doi:10.1016/S1571-0653(04)00232-X.

40 Lawrence L. Larmore. A subquadratic algorithm for constructing approximately optimal binary search trees. *J. Algorithms*, 8(4):579–591, 1987. doi:10.1016/0196-6774(87)90052-6.

41 Charles E. Leiserson. Area-efficient graph layouts (for VLSI). In *STOC 1980*, pages 270–281. IEEE Computer Society, 1980. doi:10.1109/SFCS.1980.13.

42 Nathan Linial and Michael E. Saks. Every poset has a central element. *J. Comb. Theory, Ser. A*, 40(2):195–210, 1985. doi:10.1016/0097-3165(85)90087-1.

43 Joseph W.H. Liu. The role of elimination trees in sparse factorization. *SIAM journal on matrix analysis and applications*, 11(1):134–172, 1990.

44 Kurt Mehlhorn. Nearly optimal binary search trees. *Acta Informatica*, 5(4):287–295, 1975.

45 Kurt Mehlhorn. A best possible bound for the weighted path length of binary search trees. *SIAM Journal on Computing*, pages 235–239, 1977.

46 Shay Mozes, Krzysztof Onak, and Oren Weimann. Finding an optimal tree searching strategy in linear time. In *SODA 2008*, pages 1096–1105. SIAM, 2008. URL: http://dl.acm.org/citation.cfm?id=1347082.1347202.

47 Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-27875-4.

48 Krzysztof Onak and Pawel Parys. Generalization of binary search: Searching in trees and forest-like partial orders. In *FOCS 2006*, pages 379–388, 2006. doi:10.1109/FOCS.2006.32.

49 Alex Pothen, Horst D. Simon, and Kang-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM journal on matrix analysis and applications*, 11(3):430–452, 1990.

50 Alejandro A. Schäffer. Optimal node ranking of trees in linear time. *Information Processing Letters*, 33(2):91–96, 1989.

51 Daniel Dominic Sleator and Robert Endre Tarjan. Self-adjusting binary search trees. *J. ACM*, 32(3):652–686, July 1985. doi:10.1145/3828.3835.

Improved Product-State Approximation Algorithms for Quantum Local Hamiltonians

Thiago Bergamaschi
Department of Computer Science, University of California, Berkeley, CA, USA

Abstract

The ground state energy and the free energy of Quantum Local Hamiltonians are fundamental quantities in quantum many-body physics, however, it is QMA-Hard to estimate them in general. In this paper, we develop new techniques to find classical, additive error product-state approximations for these quantities on certain families of Quantum k-Local Hamiltonians. Namely, those which are either dense, have low threshold rank, or are defined on a sparse graph that excludes a fixed minor, building on the methods and the systems studied by Brandão and Harrow, Gharibian and Kempe, and Bansal, Bravyi and Terhal.

We present two main technical contributions. First, we discuss a connection between product-state approximations of local Hamiltonians and combinatorial graph property testing. We develop a series of *weak Szemerédi regularity* lemmas for k-local Hamiltonians, built on those of Frieze and Kannan and others. We use them to develop *constant time* sampling algorithms, and to characterize the "vertex sample complexity" of the Local Hamiltonian problem, in an analog to a classical result by Alon, de la Vega, Kannan and Karpinski. Second, we build on the information-theoretic product-state approximation techniques by Brandão and Harrow, extending their results to the free energy and to an asymmetric graph setting. We leverage this structure to define families of algorithms for the free energy at low temperatures, and new algorithms for certain sparse graph families.

2012 ACM Subject Classification Theory of computation → Quantum information theory; Theory of computation → Approximation algorithms analysis

Keywords and phrases Approximation Algorithms, Quantum Information

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.20

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2210.08680`

Funding *Thiago Bergamaschi*: NSF GRFP No. DGE 2146752.

1 Introduction

The mean-field approximation is a popular heuristic in quantum many-body physics, in which product-states are used as an ansatz for generic quantum states. The low-energy states of quantum systems may be highly entangled objects, and possibly exponentially more complex than simple (unentangled) product states. This often makes computing properties of these low-energy states classically intractable. From a complexity-theoretic point of view, the mean-field approach casts these quantum problems that are in the complexity class QMA [34], into problems in NP, since product-states have a polynomial-size description and can act as classical, efficiently verifiable certificates. However, in the absence of a hardness-of-approximation result for QMA [1, 6, 2, 29] and assuming QMA≠NP, it is generally unknown if the ground states of quantum systems can even have "good" approximations with succinct classical descriptions, let alone if we can compute or approximate them efficiently.

In this work, we develop a series of classical algorithms to efficiently find mean-field approximations for quantum systems described by *local Hamiltonians*, and we develop new techniques to show that good mean-field approximations exist for fairly general classes of

these systems. A local Hamiltonian corresponds to a sparse matrix $H \in \mathbb{C}^{d^n \times d^n}$ which is exponentially large in the number n of quantum particles (or qu$d$its), and can be described as a sum over "local" terms $H = \sum_{e \in E} h_e$ defined by some hypergraph $G = ([n], E)$. H is said to be k-local if each hyperedge $e \in E$ is a k-tuple of vertices in $[n]$, and is said to have "bounded" interaction strengths if the operator norm $\|h_e\|_\infty$ is at most a constant independent of n for each hyperedge in E[1].

It is well known that the existence of product-state approximations to H is very sensitive to the structure of the underlying interaction graph G. In a seminal result, Brandão and Harrow [15] proved that so long as H has bounded interaction strengths, and is defined on a graph G of high degree or small expansion, then there exists a product state which approximates the ground state energy of H up to an additive error $\epsilon \cdot m$ (scaling with the number of edges or "interactions" m of H). Their results can be interpreted as rigorous proofs of accuracy of the mean-field approximation to the ground state energy of certain systems, and they opened the door to classical approximation schemes to find these "good" mean-field solutions efficiently. One of the main focuses of this work is to relax certain assumptions on the structure of the interaction graphs G, to extend the scope of their algorithms and existence statements.

The second main focus of this work is to study the structure and classical computation of properties of quantum systems in thermal equilibrium. The Helmholtz Free Energy $F(\beta)$ of a Quantum Local Hamiltonian H at a given temperature β^{-1} arises as an approximate counting analog to the ground state energy, as it reveals the degeneracy of the ground state (the number of QMA witnesses), the density of states of the Hamiltonian, as well as the existence of phase transitions. Quantitatively, $F(\beta)$ can be described as the optimum of a maximum entropy program:

$$F(\beta) \equiv \min_{\rho \geq 0,\ \mathrm{Tr}\rho = 1} f(\rho) = \min_{\rho \geq 0,\ \mathrm{Tr}\rho = 1} \mathrm{Tr}[H\rho] - S(\rho)/\beta \tag{1}$$

where the optimizer $\rho \propto e^{-\beta H}$ of the program above is called the Gibbs state of H. The computational complexity, and in particular the hardness of approximation of $F(\beta)$ is similarly not comprehensively understood. While QMA-Hard to estimate in general due to a reduction to the "low temperature" limit, and exactly computable in polynomial time using a #P oracle [19], it would seem there is much to uncover regarding the computational tradeoffs between error and temperature [16].

1.1 Our Main Contributions

In this section we overview our main contributions, which we present formally and in more detail in section 2.2.

Rigorous Mean-Field Approximations and Guarantees in NP

Our first contributions concern improvements and extensions to the existence statements by Brandão and Harrow [15]. Their methods had roots in the information-theoretic techniques by [39] and [10], developed in the context of approximating CSPs using the Lasserre Heirarchy. Informally, we show how to use their self-decoupling arguments to construct mixed states which are tensor products of single-particle mixed states, which approximate the Free Energy up to an additive error. We view these results as rigorous proofs of accuracy for the mean-field

[1] Please refer to section 2.1 for more background on local Hamiltonians and Schatten norms.

approximation to the Free Energy of Quantum Local Hamiltonians, and they imply that approximating the Free Energy of dense Hamiltonians up to an extensive error (scaling with the number of edges) is in NP.

▶ **Theorem 1.** *Fix $k, d = O(1)$, and an inverse temperature β. Let $H = \sum_{e \in E} h_e$ be a k-Local Hamiltonian on n qudits of local dimension d, and m interactions each of strength $\|h_e\|_\infty \leq 1$. Then, there exists a product state σ_β such that*

$$F \leq f(\sigma_\beta) = Tr[H\sigma_\beta] - S(\sigma_\beta)/\beta \leq F + O(n^{\frac{k-1}{3}} m^{2/3}) \tag{2}$$

That is, σ_β is an $O(n^{\frac{k-1}{3}} m^{2/3})$ additive error approximation to the Free Energy of H.

Note that when $k = 2$ (Hamiltonians on Graphs), the error becomes $O(n^{1/3}m^{2/3}) = O(m/D^{1/3})$, which recovers Brandão and Harrow's [15] result in terms of the *average* degree $D = m/n$ of the graph.

We emphasize two important points about the result above. First and foremost, the existence of approximations to $F(\beta)$ in NP implies that we can now use classical approximation schemes to search for optimal mean-field approximations to the free energy, and they will also be good additive approximations to the "entangled value" of $F(\beta)$. As we later discuss, this enables us to import practically all the previous machinery of approximation schemes for the ground state energy, to the Free Energy, developing novel algorithms for many quantum systems and improving on recent results by Bravyi et al. [16].

The second point of emphasis is that the result above holds at all temperatures β^{-1}. In this fashion, we are able to bypass the "low temperature bottleneck" of many approximation schemes for the Free Energy which constrain approaches in previous work, such as the polynomial interpolation method [11, 27] or Markov Chain Monte Carlo methods. We present a comprehensive comparison with previous work and the scope of our techniques for thermal systems in section 2.3.

Hamiltonian Regularity Lemmas and Approximation Algorithms

From an algorithmic point of view, our main contribution is a connection between product state approximations and graph property testing. We discuss quantum analogs of the *weak Szemerédi regularity lemmas* for dense graphs, hyper-graphs and low-threshold rank graphs [22, 3, 23], developed in the context of additive approximation schemes for Max-Cut and Max-kCSPs. At their heart lies a powerful combinatorial characterization of these systems, Szemerédi's celebrated regularity lemma [42], which states that dense graphs can be approximately decomposed into unions of complete bipartite graphs. We develop natural, constructive generalizations of these results for Quantum Local Hamiltonians, by combining our new product state approximations with multi-coloured versions of known weak regularity results, leading to improved approximation algorithms and novel structural characterizations of local Hamiltonians. Our central result in this vein is an additive error approximation scheme for dense k-Local Hamiltonians, which runs in *constant time*:

▶ **Theorem 2.** *Fix $d, k = O(1)$, $\epsilon > 0$, and let $H = \sum_e h_e$ be a k-Local Hamiltonian on n qudits of local dimension d and bounded strength interactions $\|h_e\|_\infty \leq 1$. Then, there exists a randomized algorithm which runs in time $2^{poly(1/\epsilon)}$, and with probability .99 returns an estimate for the ground state energy of H accurate up to an additive error of $\epsilon \cdot n^k$.*

We report our sampling algorithms, including that in theorem 2, in the *probe model of computation* introduced by Goldreich et al. [26]. In a nutshell, the time complexity measured above corresponds to the number of queries to a description of H, see section 2.1 for more details.

Our simplest algorithm is remarkably clean to describe, and is based on the "Vertex Sample Complexity' result for Max-kCSPs by Alon et al. [3]: Given a Hamiltonian H, sample a uniformly random subset of qudits $Q \subset [n]$ of certain constant size $|Q| = q = \text{poly}(1/\epsilon)$, and let H_Q be the $d^q \times d^q$ matrix corresponding to the restriction to the interactions of H contained entirely in Q. Then, exactly diagonalize the (constant-sized) $d^q \times d^q$ matrix H_Q, and output its lowest eigenvalue multiplied by $(n/q)^k$. For small constant $d, k, \epsilon > 0$, it is clear that this approach requires just a constant number of queries to H. The challenge, of course, lies in proving that this estimate corresponds to a $\epsilon \cdot n^k$ additive error estimate to the true ground state energy of the original Hamiltonian H.

In the body, we show how these ideas can be used to develop improvements in runtime from $n^{\text{poly}(1/\epsilon)}$ to $\text{poly}(n, 1/\epsilon) + 2^{\text{poly}(1/\epsilon)}$ or just $2^{\text{poly}(1/\epsilon)}$ for a wide range of problems on Quantum Local Hamiltonians, such as approximation schemes for the ground state energy, the Free Energy, and for Hamiltonians defined on low threshold rank graphs.

2 Technical Overview

2.1 Background and Notation

Linear Algebra and Matrix Norms. Given an $w \times w$ matrix A we refer to $\|A\|_p$ as the Schatten p-norm of A (the L_p norm of the singular values of A), and we refer to $|A|_p$ as the L_p norm of the w^2-dimensional vectorization of A. The graph decompositions are phrased in terms of the *cut norm* $\|A\|_C$ introduced by [22], defined by

$$A^+ = \max_{S_1, S_2 \subseteq [w]} \sum_{i \in S_1, j \in S_2} A_{ij} \text{ and } \|A\|_C = \max(A^+, (-A)^+) \tag{3}$$

where we have $\|A\|_C \leq \|A\|_{\infty \to 1} = \sup_{x \neq 0} \frac{|Ax|_1}{|x|_\infty} \leq 4 \cdot \|A\|_C$.

Asymptotic Notation. For any function $f(n)$ we refer to the asymptotic notation $\tilde{O}(f(n)) = O(f(n)\text{polylog}(f(n))) \leq c_1 \cdot f(n) \log^{c_2} f(n)$ for a choice of real positive constants c_1, c_2.

Local Hamiltonians. We denote a k-Local Hamiltonian on n qudits of local dimension d via a $d^n \times d^n$ Hermitian matrix, which can be expressed as a sum of local interactions $H = \sum_{e \in E} h_e$. By "local", we simply mean that each summand $h_e = H_e \otimes \mathbb{I}_{V \setminus e}$ acts non-trivially only on k particles at a time, as indicated by each k-tuple $e = (u_1 \cdots u_k)$ in a set of hyper-edges E. In this manner, we can specify any Local Hamiltonian "instance" simply by specifying the $d^k \times d^k$ submatrices of each local term. If $d, k = O(1)$, then the input has a polynomial-sized description in n. For notational convenience, we often omit the trivial support $\mathbb{I}_{V \setminus e}$. The ground state energy and the ground state of H are its minimum eigenvalue and corresponding eigenvector, and the *variational* minimum energy of H is the minimum energy of H among all product states $\min_{\rho = \otimes \rho_u} \text{Tr}[H \otimes_u \rho_u]$ with $\rho_u \in \mathbb{C}^{d \times d}$ and $\rho_u \geq 0, \text{Tr}_u[\rho_u] = 1$.

Interaction Graphs. We refer to the "Interaction Graph" $G = ([n], E)$ of a 2-Local Hamiltonian H as the graph with undirected edges $e = (u, v) \in E$ whenever the particles u, v interact non-trivially in H. That is, whenever the spectral norm is non-zero $\|H_e\|_\infty \neq 0$. By expressing each $d^2 \times d^2$ Hermitian matrix $H_{u,v} = \sum_{i,j \in [d^2]} H^{i,j}_{u,v} \cdot \sigma^i_u \otimes \sigma^j_v$ in an orthogonal basis decomposition, and grouping all the interactions with the same basis i, j, we refer to the i, j "Pauli Graph" as the subgraph of G induced on all the *directed* edges $e = (u, v)$ with

non-zero $H^{i,j}_{u,v} = d^{-2}\mathrm{Tr}[H_{u,v}\sigma^i_u \otimes \sigma^j_v]$, with weighted adjacency matrix $J^{ij} = \{H^{i,j}_{u,v}\}_{u,v\in[n]}$. We note that the matrices J^{ij} are degenerate, since $J^{ij} = (J^{ji})^T$, but we often brush over this issue via a handshaking argument. If we are given a density matrix $\rho = \otimes\rho_u$ which is a product of single qudit density matrices with a basis decomposition $\rho_u = d^{-1}\sum_i \alpha^i_u \cdot \sigma^i$, then the energy of ρ, $\mathrm{Tr}[H\rho]$ is a polynomial over the real variables α:

$$\sum_{(u,v)\in E} \mathrm{Tr}[H_{u,v}\rho_u\otimes\rho_v] = d^{-2}\sum_{(u,v)\in E}\sum_{i,j\in[d^2]} H^{i,j}_{u,v}\alpha^i_u\cdot\alpha^j_v = (2d^2)^{-1}\sum_{i,j\in[d^2]}\sum_{u\neq v\in[n]} J^{ij}_{uv}\alpha^i_u\cdot\alpha^j_v \quad (4)$$

Model of Computation. We report our sampling algorithms in the *probe model of computation* introduced by [26] in the context of graph property testing. That is, we assume we can sample a uniformly random vertex or hyper-edge in $O(1)$ time (or "probes"). Formally, fixed a k-Local Hamiltonian "instance" $H = \sum_{e\in E} H_e \otimes \mathbb{I}_{V\setminus e}$, for any k-tuple of vertices/hyper-edge $e = (u_1 \cdots u_k), u_i \in [n]$, we assume we can query the (constant-sized) $d^k \times d^k$ sub-matrix H_e in $O(1)$ time. We emphasize that since our goal is often a sublinear time algorithm, we always enforce that our algorithms output estimates for the energy (or free energy), and *implicit* descriptions of product states. If requested, these implicit descriptions can always be expanded into n-qudit product states in an additional $\mathrm{poly}(n, 1/\epsilon)$ time.

2.2 Our Results

Approximation Guarantees in NP

The first of our results are rigorous proofs of accuracy of the mean-field approximation on Quantum k-Local Hamiltonians. We argue the existence of product states, or products of single-particle mixed states, which provide additive error approximations to the ground state energy and the free energy of these systems. We build on the information-theoretic techniques by Brandão and Harrow [15], presenting an extension to the free energy and modestly refining their techniques on generic (hyper-) graphs.

▶ **Theorem 3.** *Fix $k, d = O(1)$. Let $H = \sum_{e\in E} h_e$ be a k-Local Hamiltonian on n qudits of local dimension d, and m interactions each of strength $\|h_e\|_\infty \leq 1$. Then, there exists a product state $|\psi\rangle = \otimes_{u\in[n]}|\psi_u\rangle, |\psi_u\rangle \in \mathbb{C}^d$ such that*

$$\langle\psi|H|\psi\rangle \leq \min_\phi \langle\phi|H|\phi\rangle + O(n^{\frac{k-1}{3}}m^{2/3}) \quad (5)$$

In the body, we prove more general versions of the theorem above sensitive to the matrix of interaction strengths of H. Theorem 3 matches the previous results in [15] whenever the Hamiltonian is defined on D-regular or dense graphs $m = \Omega(n^k)$, and generalizes their statements to just depend on the number of edges m. In the setting of Theorem 3, whenever $m = \Omega(n^{k-1}/\epsilon^3)$, approximating the ground state energy of H up to additive error $\epsilon \cdot m$ is in the complexity class NP, as the product state has a polynomial size description and acts as a classical witness. While these optimal product states may still be NP-Hard to find in the worst case, there are many examples where one can approximate these solutions efficiently.

To extend both these information-theoretic ideas and algorithmic applications to the free energy, we need further insights on the structure of these product state approximations. We discuss[2] how the "entanglement-breaking" procedure of [15], not only approximately

[2] In section B of the full version.

preserves the energy, but in fact also increases the entropy as well. When applied to the Gibbs state, we show one can carefully extract a tensor product of single particle mixed-states which is a good approximation to the free energy. We formalize this statement in Theorem 4,

▶ **Theorem 4.** *Fix $k, d = O(1)$, and an inverse temperature β. Let $H = \sum_{e\in E} h_e$ be a k-Local Hamiltonian on n qudits of local dimension d, and m interactions each of strength $\|h_e\|_\infty \leq 1$. Then, there exists a product state $\sigma_\beta = \otimes_{u\in[n]}\sigma_u, \sigma_u \in \mathbb{C}^{d\times d}$ such that*

$$f(\sigma_\beta) = Tr[H\sigma_\beta] - S(\sigma_\beta)/\beta \leq F + O(n^{\frac{k-1}{3}} m^{2/3}) \tag{6}$$

We emphasize that the statement above implies a product state approximation exists at all temperatures β^{-1} (and recovers the ground state approximation at $T = 0$), and moreover uses very little of the underlying graph structure apart from the *average* dense condition.

Hamiltonian Weak Regularity Lemmas

We develop an approach to designing approximations algorithms for Local Hamiltonians based on *weak Szemerédi regularity lemmas*, which are approximate decompositions to graphs, matrices, and tensors [42, 22, 3, 23].

The idea behind this construction lies in a powerful tool in extremal combinatorics. In his celebrated regularity lemma, Szemerédi [42] proved that any dense graph can be approximated by a union of a constant number of complete bipartite graphs. However, the number of partitions grew very fast with the intended quality of approximation. Frieze and Kannan [22] developed a constructive decomposition under a weaker notion of approximation, what they refered to as a "weak" regularity lemma. Concretely, they prove that any real $n \times n$ matrix with bounded entries can be decomposed into a sum of $O(1/\epsilon^2)$ cut matrices (complete bipartite graphs), up to an error $\epsilon \cdot n^2$ in the cut norm. Moreover, [22] proved that one can in fact construct such a "cut decomposition" implicitly in time polynomial in $1/\epsilon$, which enabled them to devise constant time sampling-based approximation schemes for many problems on dense graphs.

We define a natural adaptation of their results to a quantum setting, by constructing an approximate decomposition H_D of a Local Hamiltonian H which is a sum over complete, bipartite, sub-Hamiltonians. The structure of H_D can be understood as a "multi-colored" matrix cut decomposition, as essentially we apply the cut decomposition by [22] to each term in a basis decomposition of H. For concreteness, let $H = \sum_{u,v} h_{u,v}$ be a 2-Local Hamiltonian on qubits, and let us consider re-writing its Pauli basis decomposition below. We suppress the identity terms $\otimes \mathbb{I}_{V\setminus\{u,v\}}$ on the qubits that each interaction acts trivially on.

$$H = \sum_{(u,v)\in E} h_{u,v} = \sum_{(u,v)\in E} \sum_{i,j\in\{I,X,Y,Z\}} h^{i,j}_{u,v}\sigma^i_u \otimes \sigma^j_v = \sum_{i,j\in\{I,X,Y,Z\}} \sum_{u<v} h^{i,j}_{u,v}\sigma^i_u \otimes \sigma^j_v \tag{7}$$

We associate each pair of indices $i, j \in \{I, X, Y, Z\}$ to a color, and consider the $n \times n$ real valued weighted adjacency matrix $J^{ij} = \{h^{i,j}_{u,v}\}_{u,v\in[n]}$ of the i, j "Pauli Graph". By applying the cut decomposition by [22] to each of these 16 matrices J^{ij}, we construct an approximate decomposition of H into roughly $16 \cdot O(1/\epsilon^2)$ complete bipartite sub-Hamiltonians. In this context, a "complete bipartite sub-Hamiltonian" is defined by two Pauli matrices, (say, X, Y), two subsets $S, T \subset [n]$ (which, for now, we assume to be disjoint), and an interaction strength $\alpha \in \mathbb{R}$, and can be expressed as $\alpha \sum_{u\in S, v\in T} X_u \otimes Y_v$.

In the body we argue that the approximation guarantees in the cut norm are precisely what we need to ensure that for any product state $\sigma = \otimes_u \sigma_u$, the energy of σ under H or H_D are close: $\mathrm{Tr}[H\sigma] \approx \mathrm{Tr}[H_D\sigma]$. By further combining this product state regularity with our asymmetric product state approximations, we prove a stronger property on the spectra of H_D:

▶ **Lemma 5** (Informal). *Fix $d, k = O(1)$ and a constant $\epsilon > 0$, and let $H = \sum_e h_e$ be a k-Local Hamiltonian on n qudits of local dimension d and m interactions of strength bounded by $\|h_e\|_\infty \leq 1$. Then, there exists a decomposition $H_D = \sum_i^s D^{(i)}$ of H into $s = O(1/\epsilon^2)$ complete bipartite sub-Hamiltonians such that*

$$\|H - H_D\|_\infty \leq \epsilon \cdot n^{k/2} m^{1/2} \tag{8}$$

Additive Error Approximation Schemes

Leveraging the structure of the Hamiltonian regularity (lemma 5) in combination with the product state approximation toolkit enables us to devise a series of approximation schemes for Quantum Local Hamiltonians. We follow the ideas of [22, 3, 23] in establishing LP relaxations to Max Cut and other Max CSPs, and we develop an SDP relaxation scheme for finding the minimal energy product state of a Local Hamiltonian. These ideas enable us to devise an efficient additive error approximation scheme for dense Hamiltonians,

▶ **Theorem 6** (Theorem 2, restatement). *Fix $d, k = O(1)$ and $\epsilon > 0$. Let $H = \sum_e h_e$ be a k-Local Hamiltonian on n qudits of local dimension d, and m interactions of bounded strength $\|h_e\|_\infty \leq 1$. There exists a randomized algorithm which runs in time $2^{\tilde{O}(1/\epsilon^{2k-2})}$ in the probe model of computation, and with probability .99 computes an estimate for the ground state energy of H accurate up to an additive error of $\epsilon \cdot n^{k/2}\sqrt{m}$.*

We note that $n^{k/2}\sqrt{m} \geq m$, and thus in polynomial or sublinear time this approximation scheme only provides a non-trivial guarantee when the hyper-graph is dense, $m = \Omega(n^k/\log^c n)$ for some small positive constant c. However, it provides an improvement over the $n^{O(1/\epsilon^2)}$ time algorithms by [24] and [15] in this additive error regime. On the other hand, a simple explicit variant of this result provides a sub-exponential time approximation algorithm whenever $m = \omega(n^{k-1}\log n)$:

▶ **Theorem 7.** *In the context of Theorem 2, there exists a randomized algorithm which runs in time $\tilde{O}(n^k) \cdot 2^{\tilde{O}(n^k/\epsilon^2 m)}$ and with high probability computes an estimate for the ground state energy of H accurate up to an additive error of $\epsilon \cdot m$.*

Concretely, the key idea behind these relaxations is that for any product state $\sigma = \otimes_u \sigma_u$, the energy of σ on the cut decomposition H_D is a simple function of the average magnetization of a small number of subsets of the n qudits. To illustrate how this enables a relaxation scheme, consider a single complete bipartite sub-Hamiltonian, such as $H_{S,T} = \sum_{u\in S, v\in T} X_u \otimes Y_v$. The energy of σ on $H_{S,T}$ is

$$\mathrm{Tr}[H_{S,T}\sigma] = \sum_{u\in S, v\in T} \mathrm{Tr}_{u,v}[X_u\sigma_u \otimes Y_v\sigma_v] = \left(\sum_{u\in S} \mathrm{Tr}[X_u\sigma_u]\right) \cdot \left(\sum_{v\in T} \mathrm{Tr}[Y_v\sigma_v]\right), \tag{9}$$

simply the product of the average X direction magnetization of $S \subset [n]$ with the average Y magnetization of T. If we fix a "guess" $r, c \in [-n, n]$, one can introduce affine constraints on the single particle density matrices σ_u, constraining their average magnetizations to lie within a $\pm\gamma \cdot n$ range of the guess r, c:

$$r - \gamma \cdot n \leq \sum_{u \in S} \mathrm{Tr}[X_u \sigma_u] \leq r + \gamma \cdot n, \tag{10}$$

$$c - \gamma \cdot n \leq \sum_{v \in T} \mathrm{Tr}[Y_v \sigma_v] \leq c + \gamma \cdot n. \tag{11}$$

Then, we are guaranteed that any product state σ which is feasible for the constraints above must have energy in a range around the guess: $|\mathrm{Tr}[H_{S,T}\sigma] - r \cdot c| \leq (2 \cdot \gamma + \gamma^2) \cdot n^2$. In this manner, one can discretize over the space of "guesses" (r, c) and define an overlapping set of convex constraints on the description of the product states σ, such that every product state is feasible for at least one set of constraints. Approximating the ground state energy among product states ultimately reduces to checking the feasibility of a constant number of SDPs, one for each guess of r, c, and outputting whichever gives us the smallest energy estimate.

Using the techniques by [23], we can extend these insights to the setting of symmetric 2-Local Hamiltonians defined on graphs of low threshold rank. They proved that the weak regularity results of [22] could be extended to low-threshold rank graphs, by constructing a cut decomposition of a low rank approximation to the normalized adjacency matrix of these graphs. While in the appendix we formalize approximation algorithms for generic symmetric Hamiltonians (on low threshold rank graphs), perhaps the most faithful extension of this result to the quantum setting would be its application to approximating the Quantum Max Cut [25, 37, 36, 38]. Given an undirected graph $G = (V, E)$, the "Quantum Max-Cut" corresponds to the maximum eigenvalue of the Hamiltonian

$$H = \frac{1}{2} \sum_{e \in E} \left(\mathbb{I}_u \otimes \mathbb{I}_v - X_u \otimes X_v - Y_u \otimes Y_v - Z_u \otimes Z_v \right) \otimes \mathbb{I}_{V \setminus \{u,v\}} \tag{12}$$

If A is the adjacency matrix of G and D the diagonal matrix of degrees, the δ-SOS threshold rank $t_\delta(A)$ of A is the number of eigenvalues of the normalized adjacency matrix $D^{-1/2}AD^{-1/2}$ which are outside of the range $[-\delta, \delta]$. We prove

▶ **Theorem 8.** *Fix $\epsilon, \delta > 0$. Let $G = (V, E)$ be a graph on n vertices and m edges with adjacency matrix A and threshold rank $t \equiv t_{\epsilon/2}(A)$. Then, there exists an algorithm which finds an $\epsilon \cdot m + O(n^{1/3}m^{2/3})$ additive error approximation to the Quantum Max Cut of G in time poly$(n, 1/\epsilon, t) + 2^{\tilde{O}(t/\epsilon^2)}$.*

For instance, sparse D-regular random graphs have $\Theta(D^{-1/2})$-SOS threshold rank 1. In this manner, for any constant ϵ and if $D = \Omega(1/\epsilon^3)$, then one can compute an $\epsilon \cdot m$ approximation to the Quantum Max Cut of a D-regular random graph in polynomial time.

A series of works [30, 31, 32] showed that the matrix weak regularity lemma [22] could be used to approximate the free energy of Ising Models, and to give interesting structural results on the quality of the mean-field approximation and the "vertex sample complexity" of these systems. They observed that the maximum entropy program subject to the linear relaxation constraints described above, reveals properties of the Gibbs distribution and enables an additive error approximation to the free energy at all temperatures. By combining these ideas with the Hamiltonian regularity Lemma 5 and Theorem 4 on product state approximations to the free energy, we develop a series of additive error approximation schemes for the free energy of Quantum Local Hamiltonians. The first of which is a constant time approximation scheme, which provides an additive error guarantee in a low temperature regime.

▶ **Theorem 9.** *Fix $k, d = O(1)$, and $\epsilon, \delta > \omega(n^{-1/(2k-2)})$ and an inverse temperature $\beta > 0$, and let H be a k-Local Hamiltonian on n qudits of local dimension d and m bounded strength interactions. Then, there exists an algorithm that runs in time $2^{\tilde{O}(\epsilon^{2-2k})} \cdot O(\delta^{-2})$ in the probe model of computation, that returns an estimate to the free energy accurate up to an additive error of $\epsilon n^{k/2} m^{1/2} + \delta n/\beta$ and is correct with probability .99.*

We emphasize that the free energy is a convex program regularized by temperature, and thereby our approximation schemes often incur a tradeoff between combinatorial errors and thermal (temperature dependent) errors. In the low temperature regime, whenever $\beta = \Omega(n^{1-k/2} m^{-1/2})$, the algorithm above recovers the behavior of the ground state energy approximation scheme, and is largely temperature independent. However, as the temperature increases and surpasses the threshold, the leading source of error becomes the thermal error $\delta n/\beta$. In our second algorithm, we show that an explicit approach significantly improves this thermal error dependence, at the cost of a polynomial runtime.

▶ **Theorem 10.** *Fix $k, d = O(1)$, and $\epsilon, \delta > 0$ and an inverse temperature $\beta > 0$, and let H be a k-Local Hamiltonian on n qudits of local dimension d and m bounded strength interactions. Then, there exists an algorithm that runs in time $2^{\tilde{O}(\epsilon^{-2})} \cdot \tilde{O}(n^k \log 1/\delta)$, that returns an estimate to the free energy accurate up to an additive error of $\epsilon n^{k/2} m^{1/2} + \delta n/\beta$ and is correct with high probability.*

The Vertex Sample Complexity

The Regularity Lemma Lemma 20 enables us to derive an insightful structural statement for Local Hamiltonians. Namely, the definition of a "vertex sample complexity" for Local Hamiltonians of bounded interaction strengths, in an analogy to the vertex sample complexity of Max-kCSPs of [3] and [4]. They showed that the restriction of any Max-kCSP to a uniformly random sample of $\text{poly}(1/\epsilon)$ variables, sufficed to estimate the maximum number of satisfiable clauses up to an additive error of $\epsilon \cdot n^k$. We develop a generalization of this result to Quantum Local Hamiltonians, by combining the Hamiltonian regularity lemma with some extensions to the proof techniques by [3] to SDPs.

▶ **Theorem 11.** *Fix $d, k = O(1)$ and $\epsilon > 0$, and let H be a k-local Hamiltonian on n qudits of local dimension d and m bounded interaction strengths. Let $Q \subset [n]$ be a uniformly random sample of $q = \Omega(\epsilon^{-6} \log 1/\epsilon)$ of those qudits, and let H_Q be the sum of interactions with support contained entirely in Q. Then, with probability 0.99,*

$$\left| \min_{\rho} Tr[H\rho] - \frac{n^k}{q^k} \min_{\rho_Q} Tr[H_Q \rho_Q] \right| \leq \epsilon \cdot n^k \tag{13}$$

We rely crucially on the guarantee of product state approximations to Quantum Local Hamiltonians in this regime of additive error. Indeed, one of the directions of the statement above is quite intuitive for both classical and quantum systems: If the ground state energy of H is low, then the ground state energy of the restriction H_Q can't be much higher than the estimate. This is since the reduced density matrix $\rho_Q = \text{Tr}_{V \setminus Q}[\psi]$ of the ground state ψ of H, probably also has low energy $\text{Tr}[H_Q \rho_Q] \approx \frac{q^2}{n^2} \cdot \text{Tr}[H\psi]$, and the true ground state energy of H_Q can only be lower than that.

In the converse, however, lies an interesting "semi-classical" characterization of this additive error regime. Note that if the ground state energy of H is "high", then in particular there doesn't exist any product states with low energy on H. Using the proof techniques in [3], we show this implies the existence of a certain succinctly describable classical certificate

to this product-state "infeasibility", which we sample from to prove the absence of product states with low energy on H_Q. Here is where we require the product state approximations of Theorem 3: for sufficiently large Q, the absence of low energy product states for H_Q must imply a high ground state energy for H_Q. In this sense, the ground state energy of H_Q can't be much lower than its estimate either.

As a straightforward corollary to this structural result, now we can easily devise an algorithm which provides an additive error guarantee by exactly diagonalizing the Hamiltonian H_Q on $q = \tilde{O}(\epsilon^{-6})$ vertices in time $2^{\tilde{O}(1/\epsilon^6)}$. However, we can in fact do slightly better, simply by applying the additive error, product state approximation algorithm by [24] to the subsample:

▶ **Corollary 12.** *Fix $d, k = O(1)$ and $\epsilon > 0$, and let H be a k-Local Hamiltonian on n qudits of local dimension d and m bounded interaction strengths. There exists a randomized algorithm which runs in time $2^{\tilde{O}(\epsilon^{-2})}$, and with probability .99 outputs an estimate to the ground state energy accurate up to an additive error of $\epsilon \cdot n^k$.*

Aside from the improved dependence on k in the exponent, this result may seem to only subtly differ from that in Theorem 2. However, we emphasize that Theorem 2 requires an exponential number in $1/\epsilon$ of samples of vertices, whereas Theorem 11 guarantees a polynomial number suffices.

Approximation Schemes on Graphs that exclude a Fixed Minor

Finally, we develop novel singly-exponential time algorithms for sparse, 2-Local Hamiltonians defined on graphs that exclude a fixed minor. Formally, the family of h-minor free graphs are all the graphs G that can not produce another (smaller) graph h, by deleting edges and vertices and by contracting edges [41]. Planar graphs, and bounded genus graphs (such as toriods) are among the interesting special cases of these classes. Our approach builds on previous work by [9] and [15] on planar graphs, using more general combinatorial decompositions [21] and improving on their "quantum-to-classical" mappings. We show how such 2-Local Hamiltonians can be approximately understood as classical Max k-CSPs defined on the high degree vertices in the graph, and develop a dynamic programming algorithm to solve it using a simple hyper-dimensional version of a tree decomposition. Our first result for these systems is a classical algorithm to approximate the ground state energy in time singly exponential in $\text{poly}(1/\epsilon)$,

▶ **Theorem 13.** *Fix $\epsilon > 0$. Let H be a 2-Local Hamiltonian defined on n qubits and $m = \Theta(n)$ bounded strength interactions of norm < 1, configured on an h-minor free graph $G = (V, E)$ where the minor is constant size $|h| = O(1)$. Then, we can approximate the ground state energy of H up to additive error $\epsilon \cdot n$, in time poly$(n) + n \cdot 2^{poly(1/\epsilon)}$.*

We build on these ideas by combining them with our information-theoretic techniques for the free energy of quantum systems, to construct novel algorithms for the free energy of these classes of sparse graphs at low temperatures as well.

▶ **Theorem 14.** *Fix $\epsilon > 0$ and an inverse temperature β. Let H be a 2-Local Hamiltonian on n qubits and $m = \Theta(n)$ bounded strength interactions of norm < 1, configured on an h-minor free graph $G = (V, E)$ where the minor is constant size $|h| = O(1)$. Then, we can approximate the the free energy $F(\beta)$ of H up to additive error $\epsilon \cdot n$, in time poly$(n) + n \cdot \max(2, \beta^{-1})^{poly(1/\epsilon)}$, respectively.*

2.3 Related Work

Classical Approximation Schemes for QMA Complete Problems

While the systematic study of approximation algorithms to QMA-Complete problems is still emerging, there are a number of works we would like to highlight on the topic. [9] developed classical approximation schemes for ground state energies of classical and Quantum 2-Local Hamiltonians configured on planar graphs (of bounded degree, in the quantum case). They leveraged Baker's technique [8] and structural properties of planar graphs to approximately decompose the Hamiltonian into non-interacting partitions, which then could be analyzed by exact diagonalization, or dynamic programming. [24] were among the first to construct an approximation algorithm for the k-Local Hamiltonian Problem. They argued that product states can provide a d^{-k+1}-relative factor approximations to the ground state energy of k-Local Hamiltonians defined on qu$d$its, similarly to how Max Cut admits a $1/2$ multiplicative approximation. They then developed an approximation algorithm for the variational problem of finding the minimal energy product state of a given Local Hamiltonian H. It constructs a product state that provides an (extensive) $\epsilon \cdot n^k$ additive approximation to the ground state energy, in runtime $n^{O(\epsilon^{-2} \log 1/\epsilon)}$. Their approach was based on an adaptation of a classical technique, the "exhaustive sampling method" by [7] to the quantum setting, developed in the context of approximating Max Cut on dense graphs.

Later, [15] developed information-theoretic techniques to argue the existence of product state approximations to the the ground state energy. More precisely, they show that so long as H is *everywhere dense* ($\Omega(n^{k-1})$ minimum degree), has bounded expansion, or is clustered into regions of sub-volume law entanglement entropy, there exist product states that provide *additive error* approximations to the minimum energy. Leveraging their information-theoretic statements, they turned the algorithm of [24] into a PTAS for the ground state energy, albeit only meaningful when the number of interactions $m = \Omega(n^k)$. Additionally, they devise approximation schemes for Quantum Hamiltonians defined on generic planar graphs (not just those of bounded degree), solving an open problem posed by [9]. Their key insight was what we refer to as a "high-low degree" technique, in which one could consider a product state over all vertices of degree larger than some tunable cutoff Δ, and a generic (entangled) quantum state over the hilbert space of the low-degree particles, while incurring only a small error to the ground state energy. It is worthwhile to raise however, that the runtime of the resulting algorithm is triply-exponential in $1/\epsilon$, where the algorithm returns an $\epsilon \cdot n$ additive approximation.

More recently, in the context of *relative error* approximation schemes, [28] showed that one can find a product state within a relative error of l of the ground state of a traceless k-Local Hamiltonian of bounded norm, where l is the maximum degree of the underlying hyper-graph. [18] devised a $O(\log n)$ multiplicative approximation scheme to the ground state energy of 2-Local traceless Hamiltonians by rounding the solutions of SDPs to product states.

Classical Approximation Schemes for the Free Energy of Quantum Systems

Our results also contribute to a rich literature of classical techniques for thermal quantum systems. Perhaps the most well known of these techniques are the Quantum Monte Carlo methods, which approximate the quantum partition function of a quantum system to that of a classical spin system, which in turn is approximated via Markov chain Monte Carlo methods. Despite the enormous practical success of these techniques, rigorous proofs of convergence have only been presented in certain restricted systems [17, 13, 20], and they generically

are efficient only in the high temperature limit. Another high-temperature technique is the polynomial interpolation method [11, 27], based on a Taylor expansion of the partition function in the high temperature limit. Although both of these approaches are only provably efficient either on restricted classes of systems (such as substochastic Hamiltonians) and/or in the high temperature limit (typically β is a constant, or at most $O(\log n)$), they provide quite strong notions of approximation. In fact, they generally provide $(1+\epsilon)$ multiplicative approximations to the partition function (which translates to an ϵ additive approximation to the free energy), while in this paper we only attempt *extensive*, additive, $\epsilon \cdot m$ error approximations to the free energy.

By approaching the problem via this weaker notion of error, it is possible to devise approximation schemes in a much wider range of temperatures. A recent result by [16] presented an algorithm that estimates the free energy of dense Local Hamiltonians, also building on the information-theoretic techniques by [15]. Their approach is based on a quantum generalization to a classical correlation rounding approach by [40], and their algorithm finds a $\epsilon \cdot n^2$ additive approximation to the free energy of 2-Local Hamiltonians, in runtime $n^{O(\epsilon^{-2})}$.

Comparison to Previous Work

To conclude our introduction we summarize our algorithmic improvements in constrast to previous known constructions for the quantum systems studied. In table 1 below we label the Hamiltonians, and runtime and accuracy guarantees of the additive error approximation schemes in previous work for the systems we consider. In table 2, we present our results for these same systems.

For simplicity, unless otherwise stated we concern ourselves with Quantum Local Hamiltonians of bounded interaction strengths $\|H_e\|_\infty \leq 1$ on n qubits and m interactions. In both tables, we refer to a "low threshold rank" Hamiltonian as having constant ϵ-SOS threshold rank of its interaction graph. With the exception of the recent work by [16], all the results in table 1 concern ground state energy approximation schemes.

Table 1 A summary of previous algorithms for related Quantum Systems.

Result	System/Context	Accuracy	Runtime
[24]	k-local Hamiltonians	$\epsilon \cdot n^k$	$n^{\tilde{O}(\epsilon^{-2})}$
[15]	Low Threshold Rank Hamiltonians	$\epsilon \cdot \sum_{e\in E} \|H_e\|_\infty$	$n^{O(\epsilon^{-1})}$
[16]	Free Energy of 2-local Hamiltonians	$\epsilon \cdot n^2 + \delta \cdot n/\beta$	$n^{\tilde{O}(\epsilon^{-2})} \cdot O(\log 1/\delta)$
[9]	Planar Graphs of bounded degree Δ	$\epsilon \cdot \sum_{e\in E} \|H_e\|_\infty$	$n^{O(1)} \cdot 2^{2^{\mathrm{poly}(\Delta,\epsilon^{-1})}}$
[15]	Planar Graphs	$\epsilon \cdot \sum_{e\in E} \|H_e\|_\infty$	$n^{O(1)} \cdot 2^{2^{2^{\mathrm{poly}(\epsilon^{-1})}}}$

We remark* that the runtime results for k-local Hamiltonians are reported in the probe model [26], and thus may seem apriori incomparable to more standard model runtimes. However, we emphasize that we can easily convert between models by suitably pre-processing the input Hamiltonian and underlying Graph. For instance, if we are allowed query access to the input Hamiltonian in time $O(1)$, and arithmetic operations on entries of H take time $O(1)$, *but* sampling a random element of $[n]$ takes time $O(\log n)$, then the algorithm of theorem 11 outputs an estimate to the ground state energy in total time $O(\mathrm{poly}(1/\epsilon) \cdot \log n + 2^{\mathrm{poly}(1/\epsilon)})$ -

Table 2 The main algorithms in this work.

System	Context	Accuracy	Runtime*
k-local Hamiltonians	G.S. Energy	$\epsilon \cdot n^k$	$2^{\text{poly}(\epsilon^{-1})}$
	Free Energy	$\epsilon \cdot n^k + \delta \cdot n/\beta$	$2^{\text{poly}(\epsilon^{-1})} \cdot O(\delta^{-2})$
Low Threshold Rank Quantum Max Cut	Maximum Eigenvalue	$\epsilon \cdot m + O(n^{1/3}m^{2/3})$	$(n/\epsilon)^{O(1)} + 2^{\tilde{O}(1/\epsilon^2)}$
h-Minor Free Graphs of bounded degree Δ	G.S. Energy	$\epsilon \cdot n$	$n^{O(1)} + n \cdot 2^{\text{poly}(\Delta, \epsilon^{-1})}$
	Free Energy		
h-Minor Free Graphs	G.S. Energy	$\epsilon \cdot n$	$n^{O(1)} + n \cdot 2^{\text{poly}(\epsilon^{-1})}$
	Free Energy		$n^{O(1)} + n \cdot \max(2, \beta^{-1})^{\text{poly}(\epsilon^{-1})}$

as we only require a $\text{poly}(1/\epsilon)$ number of sampled vertices. On the other hand, if don't have query access to the description of H, simply spending initial $O(d^{2k}(n+m)) = O(n+m)$ preprocessing time to read out the description of H is sufficient to reduce the setting to the previous one, assuming $d, k = O(1)$.

3 Discussion

We conclude this work by raising some open problems. The first of which is a curious gap between the quality of the mean field approximation to classical and Quantum Local Hamiltonians. To contrast our results to those in the classical setting, [14, 12, 30, 32] studied the quality of the mean-field approximation to classical spin glass models with generic interaction matrices. The work of [32] culminated in the result that the mean-field approximation is within an additive error of $O(n^{2/3}m^{1/3})$ of the free energy, a strictly better dependence on the number of interactions than our upper bound, $O(n^{1/3}m^{2/3})$. As both these results have roots in the information-theoretic techniques by [39], it seems intriguing to ask whether there is some deeper structure. A possible direction would be to combine the regularity insights with the correlation rounding techniques, as in [32]. However, there remain certain technical obstacles to approaching the free energy of quantum systems with the regularity lemma, namely analyzing the matrix exponential of the cut decomposition H_D.

Another interesting problem is to improve the weak regularity results for "low threshold rank" Hamiltonians (Such as theorem 8 and section G of the full version). While we are able to devise approximation schemes based on graph regularity for a range of Hamiltonians whose interaction graphs have low threshold rank, we are unable to provide an actual construction of an approximate Hamiltonian H'. It would also be interesting to see whether the coarsest partition technique could be lifted to be applied to more general low threshold rank Hamiltonians, as opposed to relying on the high degree of symmetry of the Quantum Max Cut.

Finally, while the focus of this paper is on product-state approximations, the author considers it to be an outstanding open problem whether one can devise entangled ansatz's for classical approximations schemes to quantum problems. For examples, see [33, 5, 35], who devised low-depth quantum circuits which perform slightly better than the best product state on certain Hamiltonians.

4 Organization

In the appendix, we present a proof of the Hamiltonian regularity lemma 20, and, for readability, defer to the full version (`https://arxiv.org/abs/2210.08680`) our information-theoretic statements and algorithms.

References

1 Scott Aaronson. The quantum pcp manifesto, October 2006. URL: `http://www.scottaaronson.com/blog/?p=139`.

2 Dorit Aharonov, Itai Arad, Zeph Landau, and Umesh V. Vazirani. The detectability lemma and quantum gap amplification. In *STOC '09*, 2009.

3 Noga Alon, Wenceslas Fernandez de la Vega, Ravi Kannan, and Marek Karpinski. Random sampling and approximation of max-csp problems. *Electron. Colloquium Comput. Complex.*, 2002.

4 Gunnar Andersson and Lars Engebretsen. Property testers for dense constraint satisfaction programs on finite domains. *Random Struct. Algorithms*, 21:14–32, 2002.

5 Anurag Anshu, David Gosset, and Karen J. Morenz. Beyond product state approximations for a quantum analogue of max cut. In *TQC*, 2020.

6 Itai Arad. A note about a partial no-go theorem for quantum pcp. *Quantum Inf. Comput.*, 11:1019–1027, 2011.

7 Sanjeev Arora, David R. Karger, and Marek Karpinski. Polynomial time approximation schemes for dense instances of np-hard problems. *Journal of Computer and System Sciences*, 58:193–210, 1999.

8 S. Baker. Approximation algorithms for np-complete problems on planar graphs. *Journal of the ACM*, 1994.

9 Nikhil Bansal, Sergey Bravyi, and Barbara M. Terhal. Classical approximation schemes for the ground-state energy of quantum and classical ising spin hamiltonians on planar graphs. *Quantum Inf. Comput.*, 9:701–720, 2009.

10 Boaz Barak, Prasad Raghavendra, and David Steurer. Rounding semidefinite programming hierarchies via global correlation. *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 472–481, 2011.

11 Alexander I. Barvinok. Combinatorics and complexity of partition functions. In *Algorithms and combinatorics*, 2016.

12 Anirban Basak and Sumit Mukherjee. Universality of the mean-field for the potts model. *Probability Theory and Related Fields*, 168:557–600, 2015.

13 Thiago Bergamaschi. Simulated quantum annealing is efficient on the spike hamiltonian. *arXiv:Quantum Physics*, 2020. `arXiv:2011.15094`.

14 Christian Borgs, Jennifer T. Chayes, László Miklós Lovász, Vera T. Sós, and Katalin Vesztergombi. Convergent sequences of dense graphs ii. multiway cuts and statistical physics. *Annals of Mathematics*, 176:151–219, 2012.

15 Fernando G. S. L. Brandão and Aram Wettroth Harrow. Product-state approximations to quantum ground states. In *STOC '13*, 2013.

16 Sergey Bravyi, Anirban Narayan Chowdhury, David Gosset, and Pawel Wocjan. On the complexity of quantum partition functions. *arXiv*, abs/2110.15466, 2021. `arXiv:2110.15466`.

17 Sergey Bravyi and David Gosset. Polynomial-time classical simulation of quantum ferromagnets. *Physical review letters*, 119 10:100503, 2017.

18 Sergey Bravyi, David Gosset, Robert Koenig, and Kristan Temme. Approximation algorithms for quantum many-body problems. *Journal of Mathematical Physics*, 2019.

19 Brielin Brown, Steven T. Flammia, and Norbert Schuch. Computational difficulty of computing the density of states. *Physical review letters*, 107 4:040501, 2011.

20 Elizabeth Crosson and Aram Wettroth Harrow. Rapid mixing of path integral monte carlo for 1d stoquastic hamiltonians. *Quantum*, 5:395, 2021.

21 Erik D. Demaine, Mohammad Taghi Hajiaghayi, and Ken ichi Kawarabayashi. Algorithmic graph minor theory: Decomposition, approximation, and coloring. *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 637–646, 2005.

22 Alan M. Frieze and Ravi Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19:175–220, 1999.

23 Shayan Oveis Gharan and Luca Trevisan. A new regularity lemma and faster approximation algorithms for low threshold rank graphs. In *APPROX-RANDOM*, 2013.

24 Sevag Gharibian and Julia Kempe. Approximation algorithms for qma-complete problems. *2011 IEEE 26th Annual Conference on Computational Complexity*, pages 178–188, 2011.

25 Sevag Gharibian and Ojas Parekh. Almost optimal classical approximation algorithms for a quantum generalization of max-cut. *arXiv*, abs/1909.08846, 2019. `arXiv:1909.08846`.

26 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Electron. Colloquium Comput. Complex.*, 3, 1998.

27 Aram Wettroth Harrow, Saeed Adel Mehraban, and Mehdi Soleimanifar. Classical algorithms, correlation decay, and complex zeros of partition functions of quantum many-body systems. *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, 2020.

28 Aram Wettroth Harrow and Ashley Montanaro. Extremal eigenvalues of local hamiltonians. *arXiv*, abs/1507.00739, 2015. `arXiv:1507.00739`.

29 Matthew B. Hastings. Trivial low energy states for commuting hamiltonians, and the quantum pcp conjecture. *Quantum Inf. Comput.*, 13:393–429, 2013.

30 Vishesh Jain, Frederic Koehler, and Elchanan Mossel. The mean-field approximation: Information inequalities, algorithms, and complexity. In *COLT*, 2018.

31 Vishesh Jain, Frederic Koehler, and Elchanan Mossel. The vertex sample complexity of free energy is polynomial. In *COLT*, 2018.

32 Vishesh Jain, Frederic Koehler, and Andrej Risteski. Mean-field approximation, convex hierarchies, and the optimality of correlation rounding: a unified perspective. *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, 2019.

33 Robbie King. An improved approximation algorithm for quantum max-cut. *arXiv*, abs/2209.02589, 2022. `arXiv:2209.02589`.

34 Alexei Y. Kitaev, Alexander Shen, and Mikhail N. Vyalyi. Classical and quantum computation. In *Graduate studies in mathematics*, 2002.

35 Eun Jee Lee. Optimizing quantum circuit parameters via sdp. In *International Symposium on Algorithms and Computation*, 2022.

36 Ojas Parekh and Kevin Thompson. Application of the level-2 quantum lasserre hierarchy in quantum approximation algorithms. In *ICALP*, 2021.

37 Ojas Parekh and Kevin Thompson. Beating random assignment for approximating quantum 2-local hamiltonian problems. *arXiv*, abs/2012.12347, 2021. `arXiv:2012.12347`.

38 Ojas Parekh and Kevin Thompson. An optimal product-state approximation for 2-local quantum hamiltonians with positive terms. *arXiv*, abs/2206.08342, 2022. `arXiv:2206.08342`.

39 Prasad Raghavendra and Ning Tan. Approximating csps with global cardinality constraints using sdp hierarchies. In *SODA*, 2012.

40 Andrej Risteski. How to calculate partition functions using convex programming hierarchies: provable bounds for variational methods. *arXiv*, abs/1607.03183, 2016. `arXiv:1607.03183`.

41 Neil Robertson and Paul D. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *J. Algorithms*, 7:309–322, 1986.

42 Endre Szemerédi. Regular partitions of graphs. *Colloq. Internat. CNRS*, 260, 1975.

A The Hamiltonian Regularity Lemma

Let us begin by reviewing the cut decomposition of [22]. The key intuition behind their result is the notion that dense graphs can be roughly viewed as a sum of complete bipartite sub-graphs between subsets of vertices in the graph. Each of these bipartite sub-graphs is essentially a "cut" in the graph, hence the name.

▶ **Definition 15.** *Given two sets $S, T \subset [n]$ and a number $d \in \mathbb{R}$, the $n \times n$ cut matrix $D = CUT(S, T, d)$ is defined by $D_{u,v} = d \cdot \delta_{u \in S} \delta_{v \in T}$.*

▶ **Definition 16.** *A "cut decomposition" expresses a real matrix J as the sum*

$$J = \sum_{k=0}^{s} D^{(k)} + W \tag{14}$$

where each $D^{(k)}$ is a cut matrix defined on sets $R_k, L_k \subset [n]$, and of weight d_k. Such a decomposition is said to have width s, coefficient length $(\sum d_k^2)^{1/2}$, and error $\|W\|_{\infty \to 1}$.

The main result of [22] is precisely an algorithm to efficiently find such a decomposition:

▶ **Theorem 17** ([22])**.** *Let J be an arbitrary real matrix and fix a constant $\epsilon > 0$. Then there exists a cut decomposition of width $O(\epsilon^{-2})$, coefficient length $O(\|J\|_F/n)$, error at most $\epsilon n \|J\|_F$, and such that $\|W\|_F \leq \|J\|_F$. Moreover, with probability $1 - \delta$ said decomposition can be found implicitly in time $2^{\tilde{O}(\epsilon^{-2})}/\delta^2$, and explicitly in time $\tilde{O}(n^2/\epsilon^4) + 2^{\tilde{O}(\epsilon^{-2})}/\delta^2$.*

▶ Remark 18. The key point of the cut decomposition is that the number of cuts only depends on the quality of the approximation, not the size of the graph.

Perhaps the main tool we introduce in this work is a generalization of this result to the quantum setting. We exploit the fact that quantum density matrices and quantum Hamiltonians can be expressed in a Pauli basis, to reduce the problem of decomposing Hamiltonians into that of a "multi-colored" cut decomposition. For simplicity, here we discuss the case of 2-Local Hamiltonians, on qudits of local dimension $d = 2^{d'}$ which is a power of 2, and defer further generalizations to the appendix.

Let $H = \sum H_e$ be 2-local Hamiltonian defined on n qudits, and define $\mathbb{P}_{\log d} = \{\mathbb{I}, X, Y, Z\}^{\otimes \log d}$ be the set of Pauli operators acting on a single qudit. Any operator $h_{u,v}$ acting on the Hilbert space of 2 qudits can be decomposed into basis of $\mathbb{P}_{\log d} \otimes \mathbb{P}_{\log d}$:

$$H_{uv} = \sum_{i,j \in [d^2]} h_{uv}^{ij} \sigma_i^u \otimes \sigma_j^v \tag{15}$$

Where the h_{uv}^{ij} are all real coefficients. Group the coefficients of the interactions defined on the same Pauli matrices i, j into an interaction matrix $J^{ij} = \{h_{uv}^{ij}\}_{u,v}$, i.e., a matrix for each of d^4 "colors". We note that this essentially defines $O(d^4)$ different weighted adjacency matrices. Now, let us apply the regularity lemma of [22] on each of the colored interaction/adjacency matrices J^{ij} above. By construction, for each pair (i, j) one can express

$$J^{ij} = \sum_{k=1}^{s} D^{ijk} + W^{ij} \equiv D^{ij} + W^{ij} \tag{16}$$

Where $D^{ijk} = CUT(R^{ijk}, L^{ijk}, d^{ijk})$ are the s cut matrices of the interaction $i, j \in [d^2]$, defined on partitions $\{R^{ijk}, L^{ijk}\}$ of the vertex set of the graph, and real constants d^{ijk}, for $k \in [s]$. We can thereby define the cut decomposition H_D of the Hamiltonian H to be the edges of the D^{ijk} *crossing* any such cut:

$$\text{The Hamiltonian Cut Decomposition: } H_D = \frac{1}{2}\sum_{\substack{i,j\in[d^2]\\k\in[s]}}\sum_{\substack{u\in R^{ijk}\\v\neq u, v\in L^{ijk}}} D^{ijk}_{uv}\sigma^u_i\otimes\sigma^v_j\otimes\mathbb{I}_{V\setminus\{u,v\}} \quad (17)$$

where we appropriately order the tensor product such that $u < v$ and add a factor of $1/2$ via a handshaking argument. More importantly, we filter out the diagonal entries D^{ijk}_{uu}, since the cuts S, T returned by the cut decomposition in Theorem 17 need not be disjoint, and Local Hamiltonians can't have "self-edges" in a basis decomposition. While unfortunately we no longer can interpret the interaction graph of H_D as an exact sum of complete bipartite sub-Hamiltonians, fortunately, we will later recover this interpretation in an approximate sense.

We dedicate the rest of this section to proving two interesting properties of H_D. First, we argue that the energy of any product state $\rho = \otimes_{u\in V}\rho_u$ is close, whether in H or H_D, arising from the combinatorial structure of the decomposition. Then, we leverage our product state approximation toolkit, to argue that H_D is in fact close to H in the spectral norm $\|H - H_D\|_\infty$.

▶ **Theorem 19.** *Let $H = \sum_{u,v} H_{u,v}$ be a 2-Local Hamiltonian defined on qudits of local dimension $d = 2^{d'} = O(1)$, let $J_{uv} = \|H_{uv}\|_\infty$ be the matrix of interaction strengths, and let H_D be the Hamiltonian cut decomposition of H of width $s = O(\epsilon^{-2})$. Then, for all product states $\rho = \otimes_{u\in V}\rho_u$,*

$$|Tr[(H - H_D)\rho]| \leq \epsilon n\|J\|_F \quad (18)$$

Moreover, with probability $1 - \delta$ said decomposition can be found implicitly in time $2^{\tilde{O}(\epsilon^{-2})}/\delta^2$, and explicitly in time $\tilde{O}(n^2/\epsilon^4) + 2^{\tilde{O}(\epsilon^{-2})}/\delta^2$

Proof. By restricting our attention to product states, we are able to essentially decouple the "colors" (different Pauli terms) in the Cut Decomposition.

$$\left|\mathrm{Tr}[(H - H_D)\rho]\right| = \left|\sum_{u<v}\sum_{i,j}(h^{ij}_{uv} - D^{ij}_{uv})\mathrm{Tr}[\sigma^i_u\otimes\sigma^j_v\rho]\right| = \quad (19)$$

$$= \left|\frac{1}{2}\sum_{i,j}\sum_{u\neq v} W^{ij}_{uv}\mathrm{Tr}[\sigma^i_u\rho_u]\mathrm{Tr}[\sigma^j_v\rho_v]\right| \leq \sum_{i,j}\left|\sum_{u\neq v} W^{ij}_{uv}\mathrm{Tr}[\sigma^i_u\rho_u]\mathrm{Tr}[\sigma^j_v\rho_v]\right| = \quad (20)$$

$$= \sum_{i,j}\left|\sum_v\left(\sum_{u:u\neq v} W^{ij}_{uv}\mathrm{Tr}[\sigma^i_u\rho_u]\right)\mathrm{Tr}[\sigma^j_v\rho_v]\right| \leq \sum_{i,j}\sum_v\left|\sum_u W^{ij}_{uv}\mathrm{Tr}[\sigma^i_u\rho_u]\right| + \sum_{i,j}\sum_v|W^{ij}_{vv}| \leq \quad (21)$$

$$\leq \sum_{ij}\left(\|W^{ij}\|_{\infty\to 1} + n\cdot\max_v|W^{ij}_{vv}|\right), \quad (22)$$

where we re-introduced the diagonal terms to obtain the $\infty \to 1$ norm. From Theorem 17 we can pick a width $s = O(d^8\epsilon^{-2}) = O(\epsilon^{-2})$ s.t. $\|W^{ij}\|_{\infty\to 1} \leq \epsilon n\|J^{ij}\|_F/d^4$. Finally, the original interation graph has no diagonal elements ($J^{ij}_{vv} = 0$), and thus the Cauchy-Schwartz inequality tells us the diagonal entries of D^{ij} are bounded: $|W^{ij}_{vv}| = |J^{ij}_{vv} - D^{ij}_{vv}| \leq \sum_k|d^{ijk}| \leq s^{1/2}\cdot(\sum(d^{ijk})^2)^{1/2} \leq s^{1/2}\cdot\|J^{ij}\|_F/n$. The observation $\|J^{ij}\|_F \leq \|J\|_F$ and assuming $\epsilon^{-2} = o(n)$ concludes the proof. ◀

By combining the product state cut decomposition above with our results on product state approximations in theorem 3 and in section B of the full version, we can extend our results to entangled states as well.

▶ **Lemma 20** (The Hamiltonian Weak Regularity Lemma)**.** *In the context of Theorem 19,* $\|H - H_D\| \leq \epsilon \cdot n \|J\|_F$.

Proof. By Schatten norm duality, there exists a normalized state ψ^* s.t.

$$\|H - H_D\|_\infty = \max_\psi |\mathrm{Tr}[(H - H_D)\psi]| = |\mathrm{Tr}[(H - H_D)\psi^*]| \tag{23}$$

We now apply the product state approximation Theorem 3 on the state ψ^* and Hamiltonian $H' = H - H_D$, to argue there exists a separable state σ s.t.

$$|\mathrm{Tr}[(H - H_D)(\psi^* - \sigma)]| \leq \epsilon n \|J\|_F / 2 \tag{24}$$

where we observe that if J' is the matrix of interaction strengths of $H' = H - H_D$, then $\|J'\|_1 \leq n\|J'\|_F$ (Cauchy-Schwartz) and $\|J'\|_F \leq \sum_{i,j \in [d^2]} \|W^{ij}\|_F \leq O(d^4 \|J\|_F)$ by means of a triangle inequality and the guarantees on W in Theorem 17. Since σ is separable, we can appropriately pick the width $s = O(\epsilon^{-2})$ in Theorem 19 to guarantee

$$|\mathrm{Tr}[(H - H_D)\sigma]| \leq \epsilon n \|J\|_F / 2 \tag{25}$$

and thereby via the triangle inequality:

$$\|H - H_D\|_\infty \leq |\mathrm{Tr}[(H - H_D)(\psi^* - \sigma)]| + |\mathrm{Tr}[(H - H_D)\sigma]| \leq \epsilon n \|J\|_F \tag{26}$$

◀

Using the existing technology of matrix regularity lemmas, in the full version we present extensions to the result above for Local Hamiltonians defined on hyper-graphs and for graphs of low threshold rank.

Sublinear Time Eigenvalue Approximation via Random Sampling

Rajarshi Bhattacharjee
Manning College of Information and Computer Sciences,
University of Massachusetts, Amherst, MA, USA

Gregory Dexter
Department of Computer Science, Purdue University, West Lafayette, IN, USA

Petros Drineas
Department of Computer Science, Purdue University, West Lafayette, IN, USA

Cameron Musco
Manning College of Information and Computer Sciences,
University of Massachusetts, Amherst, MA, USA

Archan Ray
Manning College of Information and Computer Sciences,
University of Massachusetts, Amherst, MA, USA

Abstract

We study the problem of approximating the eigenspectrum of a symmetric matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ with bounded entries (i.e., $\|\mathbf{A}\|_\infty \leq 1$). We present a simple sublinear time algorithm that approximates all eigenvalues of $\mathbf{A}$ up to additive error $\pm\epsilon n$ using those of a randomly sampled $\tilde{O}\left(\frac{\log^3 n}{\epsilon^3}\right) \times \tilde{O}\left(\frac{\log^3 n}{\epsilon^3}\right)$ principal submatrix. Our result can be viewed as a concentration bound on the complete eigenspectrum of a random submatrix, significantly extending known bounds on just the singular values (the magnitudes of the eigenvalues). We give improved error bounds of $\pm\epsilon\sqrt{\text{nnz}(\mathbf{A})}$ and $\pm\epsilon\|\mathbf{A}\|_F$ when the rows of $\mathbf{A}$ can be sampled with probabilities proportional to their sparsities or their squared ℓ_2 norms respectively. Here $\text{nnz}(\mathbf{A})$ is the number of non-zero entries in $\mathbf{A}$ and $\|\mathbf{A}\|_F$ is its Frobenius norm. Even for the strictly easier problems of approximating the singular values or testing the existence of large negative eigenvalues (Bakshi, Chepurko, and Jayaram, FOCS '20), our results are the first that take advantage of non-uniform sampling to give improved error bounds. From a technical perspective, our results require several new eigenvalue concentration and perturbation bounds for matrices with bounded entries. Our non-uniform sampling bounds require a new algorithmic approach, which judiciously zeroes out entries of a randomly sampled submatrix to reduce variance, before computing the eigenvalues of that submatrix as estimates for those of $\mathbf{A}$. We complement our theoretical results with numerical simulations, which demonstrate the effectiveness of our algorithms in practice.

2012 ACM Subject Classification Theory of computation → Sketching and sampling; Mathematics of computing → Computations on matrices

Keywords and phrases sublinear algorithms, eigenvalue approximation, randomized linear algebra

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.21

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2109.07647 [10]

Funding RB, CM and AR were partially supported by an Adobe Research grant, along with NSF Grants 2046235 and 1763618. PD and GD were partially supported by NSF AF 1814041, NSF FRG 1760353, and DOE-SC0022085.

Acknowledgements We thank Ainesh Bakshi, Rajesh Jayaram, Anil Damle, Nicholas Monath and Christopher Musco for helpful conversations about this work.

1 Introduction

Approximating the eigenvalues of a symmetric matrix is a fundamental problem – with applications in engineering, optimization, data analysis, spectral graph theory, and beyond. For an $n \times n$ matrix, all eigenvalues can be computed to high accuracy using direct eigendecomposition in $O(n^\omega)$ time, where $\omega \approx 2.37$ is the exponent of matrix multiplication [14, 2]. When just a few of the largest magnitude eigenvalues are of interest, the power method and other iterative Krylov methods can be applied [39]. These methods repeatedly multiply the matrix of interest by query vectors, requiring $O(n^2)$ time per multiplication when the matrix is dense and unstructured.

For large n, it is desirable to have even faster eigenvalue approximation algorithms, running in $o(n^2)$ time – i.e., sublinear in the size of the input matrix. Unfortunately, for general matrices, no non-trivial approximation can be computed in $o(n^2)$ time: without reading $\Omega(n^2)$ entries, it is impossible to distinguish with reasonable probability if all entries (and hence all eigenvalues) are equal to zero, or if there is a single pair of arbitrarily large entries at positions (i, j) and (j, i), leading to a pair of arbitrarily large eigenvalues. Given this, we seek to address the following question:

Under what assumptions on a symmetric $n \times n$ input matrix, can we compute non-trivial approximations to its eigenvalues in $o(n^2)$ time?

It is well known that $o(n^2)$ time eigenvalue computation is possible for highly structured inputs, like tridiagonal or Toeplitz matrices [26]. For sparse or structured matrices that admit fast matrix vector multiplication, one can compute a small number of the largest in magnitude eigenvalues in $o(n^2)$ time using iterative methods. Through the use of robust iterative methods, fast top eigenvalue estimation is also possible for matrices that admit fast *approximate* matrix-vector multiplication, such as kernel similarity matrices [25, 27, 4]. Our goal is to study simple, sampling-based sublinear time algorithms that work under much weaker assumptions on the input matrix.

1.1 Our Contributions

Our main contribution is to show that a very simple algorithm can be used to approximate *all eigenvalues* of any symmetric matrix with *bounded entries*. In particular, for any $\mathbf{A} \in \mathbb{R}^{n \times n}$ with maximum entry magnitude $\|\mathbf{A}\|_\infty \leq 1$, sampling an $s \times s$ principal submatrix $\mathbf{A}_S$ of $\mathbf{A}$ with $s = \tilde{O}\left(\frac{\log^3 n}{\epsilon^3}\right)$ and scaling its eigenvalues by n/s yields a $\pm\epsilon n$ additive error approximation to all eigenvalues of $\mathbf{A}$ with good probability.[1] This result is formally stated below, where $[n] \stackrel{\text{def}}{=} \{1, \ldots, n\}$.

▶ **Theorem 1** (Sublinear Time Eigenvalue Approximation). *Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be symmetric with $\|\mathbf{A}\|_\infty \leq 1$ and eigenvalues $\lambda_1(\mathbf{A}) \geq \ldots \geq \lambda_n(\mathbf{A})$. Let $S \subseteq [n]$ be formed by including each index independently with probability s/n as in Algorithm 1. Let $\mathbf{A}_S$ be the corresponding principal submatrix of $\mathbf{A}$, with eigenvalues $\lambda_1(\mathbf{A}_S) \geq \ldots \geq \lambda_{|S|}(\mathbf{A}_S)$.*

For all $i \in [|S|]$ with $\lambda_i(\mathbf{A}_S) \geq 0$, let $\tilde{\lambda}_i(\mathbf{A}) = \frac{n}{s} \cdot \lambda_i(\mathbf{A}_S)$. For all $i \in [|S|]$ with $\lambda_i(\mathbf{A}_S) < 0$, let $\tilde{\lambda}_{n-(|S|-i)}(\mathbf{A}) = \frac{n}{s} \cdot \lambda_i(\mathbf{A}_S)$. For all other $i \in [n]$, let $\tilde{\lambda}_i(\mathbf{A}) = 0$. If $s \geq \frac{c \log(1/(\epsilon\delta)) \cdot \log^3 n}{\epsilon^3 \delta}$, for large enough constant c, then with probability $\geq 1 - \delta$, for all $i \in [n]$, $\lambda_i(\mathbf{A}) - \epsilon n \leq \tilde{\lambda}_i(\mathbf{A}) \leq \lambda_i(\mathbf{A}) + \epsilon n$.

[1] Here and throughout, $\tilde{O}(\cdot)$ hides logarithmic factors in the argument. Note that by scaling, our algorithm gives a $\pm\epsilon n \cdot \|\mathbf{A}\|_\infty$ approximation for any $\mathbf{A}$.

See Figure 1 for an illustration of how the $|S|$ eigenvalues of $\mathbf{A}_S$ are mapped to estimates for all n eigenvalues of $\mathbf{A}$. Note that the principal submatrix $\mathbf{A}_S$ sampled in Theorem 1 will have $O(s) = \tilde{O}\left(\frac{\log^3 n}{\epsilon^3 \delta}\right)$ rows/columns with high probability. Thus, with high probability, the algorithm reads just $\tilde{O}\left(\frac{\log^6 n}{\epsilon^6 \delta^2}\right)$ entries of $\mathbf{A}$ and runs in $\text{poly}(\log n, 1/\epsilon, 1/\delta)$ time. Standard matrix concentration bounds imply that one can sample $O\left(\frac{s \log(1/\delta)}{\epsilon^2}\right)$ random entries from the $O(s) \times O(s)$ random submatrix $\mathbf{A}_S$ and preserve its eigenvalues to error $\pm \epsilon s$ with probability $1 - \delta$ [1]. See Appendix F of [10] for a proof. This can be directly combined with Theorem 1 to give improved sample complexity:

▶ **Corollary 2** (Improved Sample Complexity via Entrywise Sampling). *Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be symmetric with $\|\mathbf{A}\|_\infty \leq 1$ and eigenvalues $\lambda_1(\mathbf{A}) \geq \ldots \geq \lambda_n(\mathbf{A})$. For any $\epsilon, \delta \in (0,1)$, there is an algorithm that reads $\tilde{O}\left(\frac{\log^3 n}{\epsilon^5 \delta}\right)$ entries of $\mathbf{A}$ and returns, with probability at least $1 - \delta$, $\tilde{\lambda}_i(\mathbf{A})$ for each $i \in [n]$ satisfying $|\tilde{\lambda}_i(\mathbf{A}) - \lambda_i(\mathbf{A})| \leq \epsilon n$.*

Observe that the dependence on δ in Theorem 1 and Corollary 2 can be improved via standard arguments: running the algorithm with failure probability $\delta' = 2/3$, repeating $O(\log(1/\delta))$ times, and taking the median estimate for each $\lambda_i(\mathbf{A})$. This guarantees that the algorithm will succeed with probability at most $1 - \delta$ at the expense of a $\log(1/\delta)$ dependence in the complexity.

We note that our $\pm \epsilon n$ error guarantee is particularly useful in applications where the matrix $\mathbf{A}$ has low stable rank and the top eigenvalues have magnitude scaling roughly with n. Low stable rank is a common feature of real-life data matrices [47], including classes of bounded entry matrices, such as kernel similarity matrices [18] and adjacency matrices of power law graphs [36].

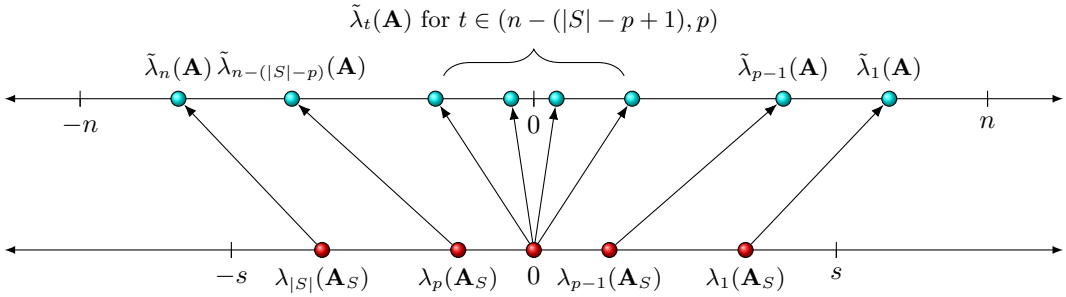


Figure 1 Alignment of eigenvalues in Thm. 1 and Algo. 1. We illustrate how the eigenvalues of $\mathbf{A}_S$, scaled by $\frac{n}{s}$, are used to approximate all eigenvalues of $\mathbf{A}$. If $\mathbf{A}_S$ has $p-1$ positive eigenvalues, they are set to the top $p-1$ eigenvalue estimates. Its $|S| - p + 1$ negative eigenvalues are set to the bottom eigenvalue estimates. All remaining eigenvalues are simply approximated as zero.

Comparison to known bounds. Theorem 1 can be viewed as a concentration inequality on the full eigenspectrum of a random principal submatrix $\mathbf{A}_S$ of $\mathbf{A}$. This significantly extends prior work, which was able to bound just the spectral norm (i.e., the magnitude of the top eigenvalue) of a random principal submatrix [38, 44]. Bakshi, Chepurko, and Jayaram [5] recently identified developing such full eigenspectrum concentration inequalities as an important step in expanding our knowledge of sublinear time property testing algorithms for bounded entry matrices.

Standard matrix concentration bounds [22] can be used to show that the *singular values* of $\mathbf{A}$ (i.e., the magnitudes of its eigenvalues) are approximated by those of a $O\left(\frac{\log n}{\epsilon^2}\right) \times O\left(\frac{\log n}{\epsilon^2}\right)$ random submatrix (see Appendix G of [10]) with independently sampled rows and columns. However, such a random matrix will not be symmetric or even have real eigenvalues in general, and thus no analogous bounds were previously known for the eigenvalues themselves.

Lower Bounds. Recently, Bakshi, Chepurko, and Jayaram [5] studied the closely related problem of testing positive semidefiniteness in the bounded entry model. They show how to test whether the minimum eigenvalue of $\mathbf{A}$ is either greater than 0 or smaller than $-\epsilon n$ by reading just $\tilde{O}(\frac{1}{\epsilon^2})$ entries. They show that this result is optimal in terms of query complexity, up to logarithmic factors. Like our approach, their algorithm is based on random principal submatrix sampling. Our eigenvalue approximation guarantee strictly strengthens the testing guarantee – given $\pm\epsilon n$ approximations to all eigenvalues, we immediately solve the testing problem. Thus, our query complexity is tight up to a $\text{poly}(\log n, 1/\epsilon)$ factor. It is open if our higher sample complexity is necessary to solve the harder full eigenspectrum estimation problem. See Section 1.4 for further discussion.

Improved bounds for non-uniform sampling. Our second main contribution is to show that, when it is possible to efficiently sample rows/columns of $\mathbf{A}$ with probabilities proportional to their sparsities or their squared ℓ_2 norms, significantly stronger eigenvalue estimates can be obtained. In particular, letting $\text{nnz}(\mathbf{A})$ denote the number of nonzero entries in $\mathbf{A}$ and $\|\mathbf{A}\|_F$ denote its Frobenius norm, we show that sparsity-based sampling yields eigenvalue estimates with error $\pm\epsilon\sqrt{\text{nnz}(\mathbf{A})}$ and norm-based sampling gives error $\pm\epsilon\|\mathbf{A}\|_F$. See Theorems 3 and 4 for formal statements. Observe that when $\|\mathbf{A}\|_\infty \leq 1$, its eigenvalues are bounded in magnitude by $\|\mathbf{A}\|_2 \leq \|\mathbf{A}\|_F \leq \sqrt{\text{nnz}(\mathbf{A})} \leq n$. Thus, Theorems 3 and 4 are natural strengthenings of Theorem 1. Row norm-based sampling (Theorem 4) additionally removes the bounded entry requirement of Theorems 1 and 3.

As discussed in Section 1.3.1, sparsity-based sampling can be performed in sublinear time when $\mathbf{A}$ is stored in a slightly augmented sparse matrix format, or when $\mathbf{A}$ is the adjacency matrix of a graph accessed in the standard graph query model of the sublinear algorithms literature [23]. Norm-based sampling can also be performed efficiently with an augmented matrix format, and is commonly studied in randomized and "quantum-inspired" algorithms for linear algebra [19, 43].

▶ **Theorem 3** (Sparse Matrix Eigenvalue Approximation). *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be symmetric with $\|\mathbf{A}\|_\infty \leq 1$ and eigenvalues $\lambda_1(\mathbf{A}) \geq \ldots \geq \lambda_n(\mathbf{A})$. Let $S \subseteq [n]$ be formed by including the i^{th} index independently with probability $p_i = \min\left(1, \frac{s\,\text{nnz}(\mathbf{A}_i)}{\text{nnz}(\mathbf{A})}\right)$ as in Algorithm 2 of [10]. Here $\text{nnz}(\mathbf{A}_i)$ is the number of non-zero entries in the i^{th} row of $\mathbf{A}$. Let $\mathbf{A}_S$ be the corresponding principal submatrix of $\mathbf{A}$, and let $\tilde{\lambda}_i(\mathbf{A})$ be the estimate of $\lambda_i(\mathbf{A})$ computed from $\mathbf{A}_S$ as in Algorithm 2 of citebhattacharjee2021sublinear. If $s \geq \frac{c\log^8 n}{\epsilon^8\delta^4}$, for large enough constant c, then with probability $\geq 1-\delta$, for all $i \in [n]$, $|\tilde{\lambda}_i(\mathbf{A}) - \lambda_i(\mathbf{A})| \leq \epsilon\sqrt{\text{nnz}(\mathbf{A})}$.*

▶ **Theorem 4** (Row Norm Based Matrix Eigenvalue Approximation). *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be symmetric and eigenvalues $\lambda_1(\mathbf{A}) \geq \ldots \geq \lambda_n(\mathbf{A})$. Let $S \subseteq [n]$ be formed by including the i^{th} index independently with probability $p_i = \min\left(1, \frac{s\|\mathbf{A}_i\|_2^2}{\|\mathbf{A}\|_F^2} + \frac{1}{n^2}\right)$ as in Algorithm 3 of [10]. Here $\|\mathbf{A}_i\|_2$ is the ℓ_2 norm of the i^{th} row of $\mathbf{A}$. Let $\mathbf{A}_S$ be the corresponding principal submatrix of $\mathbf{A}$, and let $\tilde{\lambda}_i(\mathbf{A})$ be the estimate of $\lambda_i(\mathbf{A})$ computed from $\mathbf{A}_S$ as in Algorithm 3 of [10]. If $s \geq \frac{c\log^{10} n}{\epsilon^8\delta^4}$, for large enough constant c, then with probability $\geq 1-\delta$, for all $i \in [n]$, $|\tilde{\lambda}_i(\mathbf{A}) - \lambda_i(\mathbf{A})| \leq \epsilon\|\mathbf{A}\|_F$.*

The above non-uniform sampling theorems immediately yield algorithms for testing the presence of a negative eigenvalue with magnitude at least $\epsilon\sqrt{\text{nnz}(\mathbf{A})}$ or $\epsilon\|\mathbf{A}\|_F$ respectively, strengthening the results of [5], which require eigenvalue magnitude at least ϵn. In the graph property testing literature, there is a rich line of work exploring the testing of bounded degree or sparse graphs [23, 7]. Theorem 3 can be thought of as first step in establishing a related theory of sublinear time approximation algorithms and property testers for sparse matrices. Due to lack of space, we defer the proofs of Theorems 3 and 4 to Section 4 and Appendix E of [10] respectively.

Surprisingly, in the non-uniform sampling case, the eigenvalue estimates derived from $\mathbf{A}_S$ cannot simply be its scaled eigenvalues, as in Theorem 1. E.g., when $\mathbf{A}$ is the identity, our row sampling probabilities are uniform in all cases. However, the scaled submatrix $\frac{n}{s} \cdot \mathbf{A}_S$ will be a scaled identity, and have eigenvalues equal to n/s – failing to give a $\pm\epsilon\sqrt{\text{nnz}(\mathbf{A})} = \pm\epsilon\|\mathbf{A}\|_F = \pm\epsilon\sqrt{n}$ approximation to the true eigenvalues (all of which are 1) unless $s \gtrsim \frac{\sqrt{n}}{\epsilon}$. To handle this, and related cases, we must argue that selectively zeroing out entries in sufficiently low probability rows/columns of $\mathbf{A}$ (see Algorithms 2 and 3 of [10]) does not significantly change the spectrum, and ensures concentration of the submatrix eigenvalues. It is not hard to see that simple random submatrix sampling fails even for the easier problem of singular value estimation. Theorems 3 and 4 give the first results of their kinds for this problem as well.

1.2 Related Work

Eigenspectrum estimation is a key primitive in numerical linear algebra, typically known as *spectral density estimation*. The eigenspectrum is viewed as a distribution with mass $1/n$ at each of the n eigenvalues, and the goal is to approximate this distribution [49, 35]. Applications include identifying motifs in social networks [15], studying Hessian and weight matrix spectra in deep learning [40, 51, 21], "spectrum splitting" in parallel eigensolvers [31], and the study of many systems in experimental physics and chemistry [48, 41, 28].

Recent work has studied sublinear time spectral density estimation for graph structured matrices – Braverman, Krishnan, and Musco [11] show that the spectral density of a normalized graph adjacency or Laplacian matrix can be estimated to ϵ error in the Wasserstein distance in $\tilde{O}(n/\operatorname{poly}(\epsilon))$ time. Cohen-Steiner, Kong, Sohler, and Valiant study a similar setting, giving runtime $2^{O(1/\epsilon)}$ [13]. We note that the additive error eigenvalue approximation result of Theorem 1 (analogously Theorems 3 and 4) directly gives an ϵn approximation to the spectral density in the Wasserstein distance – extending the above results to a much broader class of matrices. When $\|\mathbf{A}\|_\infty \leq 1$, $\mathbf{A}$ can have eigenvalues as large as n, while the normalized adjacency matrices studied in [13, 11] have eigenvalues in $[-1, 1]$. So, while the results are not directly comparable, our Wasserstein error can be thought as on order of their error of ϵ after scaling.

Our work is also closely related to a line of work on sublinear time property testing for bounded entry matrices, initiated by Balcan et al. [6]. In that work, they study testing of rank, Schatten-p norms, and several other global spectral properties. Sublinear time testing algorithms for the rank and other properties have also been studied under low-rank and bounded row norm assumptions on the input matrix [30, 33]. Recent work studies positive semidefiniteness testing and eigenvalue estimation in the matrix-vector query model, where each query computes $\mathbf{Ax}$ for some $\mathbf{x} \in \mathbb{R}^{n\times n}$. As in Theorem 4, $\pm\epsilon\|\mathbf{A}\|_F$ eigenvalue estimation can be achieved with $\operatorname{poly}(\log n, 1/\epsilon)$ queries in this model [37]. Finally, several works study streaming algorithms for eigenspectrum approximation [3, 32, 34]. These algorithms are not sublinear time – they require at least linear time to process the input matrix. However, they

use sublinear working memory. Note that Theorem 1 immediately gives a sublinear space streaming algorithm for eigenvalue estimation. We can simply store the sampled submatrix $\mathbf{A}_S$ as its entries are updated.

1.3 Technical Overview

In this section, we overview the main techniques used to prove Theorems 1, and then how these techniques are extended to prove Theorems 3 and 4. We start by defining a decomposition of any symmetric $\mathbf{A}$ into the sum of two matrices containing its large and small magnitude eigendirections.

▶ **Definition 5** (Eigenvalue Split). *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be symmetric. For any $\epsilon, \delta \in (0,1)$, let $\mathbf{A}_o = \mathbf{V}_o\mathbf{\Lambda}_o\mathbf{V}_o^T$ where $\mathbf{\Lambda}_o$ is diagonal, with the eigenvalues of $\mathbf{A}$ with magnitude $\geq \epsilon\sqrt{\delta}n$ on its diagonal, and $\mathbf{V}_o$ has the corresponding eigenvectors as columns. Similarly, let $\mathbf{A}_m = \mathbf{V}_m\mathbf{\Lambda}_m\mathbf{V}_m^T$ where $\mathbf{\Lambda}_m$ has the eigenvalues of $\mathbf{A}$ with magnitude $< \epsilon\sqrt{\delta}n$ on its diagonal and $\mathbf{V}_m$ has the corresponding eigenvectors as columns. Then, $\mathbf{A}$ can be decomposed as*

$$\mathbf{A} = \mathbf{A}_o + \mathbf{A}_m = \mathbf{V}_o\mathbf{\Lambda}_o\mathbf{V}_o^T + \mathbf{V}_m\mathbf{\Lambda}_m\mathbf{V}_m^T.$$

Any principal submatrix of $\mathbf{A}$, $\mathbf{A}_S$, can be similarly written as

$$\mathbf{A}_S = \mathbf{A}_{o,S} + \mathbf{A}_{m,S} = \mathbf{V}_{o,S}\mathbf{\Lambda}_o\mathbf{V}_{o,S}^T + \mathbf{V}_{m,S}\mathbf{\Lambda}_m\mathbf{V}_{m,S}^T,$$

where $\mathbf{V}_{o,S}, \mathbf{V}_{m,S}$ are the corresponding submatrices obtained by sampling rows of $\mathbf{V}_o, \mathbf{V}_m$.

Since $\mathbf{A}_S$, $\mathbf{A}_{m,S}$ and $\mathbf{A}_{o,S}$ are all symmetric, we can use Weyl's eigenvalue perturbation theorem [50] to show that for all eigenvalues of $\mathbf{A}_S$,

$$|\lambda_i(\mathbf{A}_S) - \lambda_i(\mathbf{A}_{o,S})| \leq \|\mathbf{A}_{m,S}\|_2. \tag{1}$$

We will argue that the eigenvalues of $\mathbf{A}_{o,S}$ approximate those of $\mathbf{A}_o$ – i.e. all eigenvalues of $\mathbf{A}$ with magnitude $\geq \epsilon\sqrt{\delta}n$. Further, we will show that $\|\mathbf{A}_{m,S}\|_2$ is small with good probability. Thus, via (1), the eigenvalues of $\mathbf{A}_S$ approximate those of $\mathbf{A}_o$. In the estimation procedure of Theorem 1, all other small magnitude eigenvalues of $\mathbf{A}$ are estimated to be 0, which will immediately give our $\pm\epsilon n$ approximation bound when the original eigenvalue has magnitude $\leq \epsilon n$.

Bounding the eigenvalues of $\mathbf{A}_{o,S}$. The first step is to show that the eigenvalues of $\mathbf{A}_{o,S}$ well-approximate those of $\mathbf{A}_o$. As in [5], we critically use that the eigenvectors corresponding to large eigenvalues are *incoherent* – intuitively, since $\|\mathbf{A}\|_\infty$ is bounded, their mass must be spread out in order to witness a large eigenvalue. Specifically, [5] shows that for any eigenvector $\mathbf{v}$ of $\mathbf{A}$ with corresponding eigenvalue $\geq \epsilon\sqrt{\delta}n$, $\|\mathbf{v}\|_\infty \leq \frac{1}{\epsilon\sqrt{\delta}\cdot\sqrt{n}}$. We give related bounds on the Euclidean norms of the rows of $\mathbf{V}_o$ (the *leverage scores* of $\mathbf{A}_o$), and on these rows after weighting by $\mathbf{\Lambda}_o$.

Using these incoherence bounds, we argue that the eigenvalues of $\mathbf{A}_{o,S}$ approximate those of $\mathbf{A}_o$ up to $\pm\epsilon n$ error. A key idea is to bound the eigenvalues of $\mathbf{\Lambda}_o^{1/2}\mathbf{V}_{o,S}^T\mathbf{V}_{o,S}\mathbf{\Lambda}_o^{1/2}$, which are identical to the non-zero eigenvalues of $\mathbf{A}_{o,S} = \mathbf{V}_{o,S}\mathbf{\Lambda}_o\mathbf{V}_{o,S}^T$. Via a matrix Bernstein bound and our incoherence bounds on $\mathbf{V}_o$, we show that this matrix is close to $\mathbf{\Lambda}_o$ with high probability. However, since $\mathbf{\Lambda}_o^{1/2}$ may be complex, the matrix is *not necessarily Hermitian* and standard perturbation bounds [42, 29] do not apply. Thus, to derive an eigenvalue

bound, we apply a perturbation bound of Bhatia [9], which generalizes Weyl's inequality to the non-Hermitian case, with a $\log n$ factor loss. To the best of our knowledge, this is the first time that perturbation theory bounds for non-Hermitian matrices have been used to prove improved algorithmic results in the theoretical computer science literature.

We note that in Appendix B of [10], we give an alternate bound, which instead analyzes the Hermitian matrix $(\mathbf{V}_{o,S}^T\mathbf{V}_{o,S})^{1/2}\mathbf{\Lambda}_o(\mathbf{V}_{o,S}^T\mathbf{V}_{o,S})^{1/2}$, whose eigenvalues are again identical to those of $\mathbf{A}_{o,S}$. This approach only requires Weyl's inequality, and yields an overall bound of $s = O\left(\frac{\log n}{\epsilon^4\delta}\right)$, improving the $\log n$ factors of Theorem 1 at the cost of worse ϵ dependence.

Bounding the spectral norm of $\mathbf{A}_{m,S}$. The next step is to show that all eigenvalues of $\mathbf{A}_{m,S}$ are small provided a sufficiently large submatrix is sampled. This means that the "middle" eigenvalues of $\mathbf{A}$, *i.e.* those with magnitude $\leq \epsilon\sqrt{\delta}n$ do not contribute much to any eigenvalue $\lambda_i(\mathbf{A}_S)$. To do so, we apply a theorem of [38, 44] which shows concentration of the spectral norm of a uniformly random submatrix of an entrywise bounded matrix. Observe that while $\|\mathbf{A}\|_\infty \leq 1$, such a bound will not in general hold for $\|\mathbf{A}_m\|_\infty$. Nevertheless, we can use the incoherence of $\mathbf{V}_o$ to show that $\|\mathbf{A}_o\|_\infty$ is bounded, which via triangle inequality, yields a bound on $\|\mathbf{A}_m\|_\infty \leq \|\mathbf{A}\|_\infty + \|\mathbf{A}_o\|_\infty$. In the end, we show that if $s \geq O(\frac{\log n}{\epsilon^2\delta})$, with probability at least $1-\delta$, $\|\mathbf{A}_{m,S}\|_2 \leq \epsilon s$. After the n/s scaling in the estimation procedure of Theorem 1, this spectral norm bound translates into an additive ϵn error in approximating the eigenvalues of $\mathbf{A}$.

Completing the argument. Once we establish the above bounds on $\mathbf{A}_{o,S}$ and $\mathbf{A}_{m,S}$, Theorem 1 is essentially complete. Any eigenvalue in $\mathbf{A}$ with magnitude $\geq \epsilon n$ will correspond to a nearby eigenvalue in $\frac{n}{s}\cdot\mathbf{A}_{o,S}$ and in turn, $\frac{n}{s}\cdot\mathbf{A}_S$ given our spectral norm bound on $\mathbf{A}_{m,S}$. An eigenvalue in $\mathbf{A}$ with magnitude $\leq \epsilon n$ may or may not correspond to a nearby by eigenvalue in $\mathbf{A}_{o,S}$ (it will only if it lies in the range $[\epsilon\sqrt{\delta}n, \epsilon n]$). In any case, in the estimation procedure of Theorem 1, such an eigenvalue will either be estimated using a small eigenvalue of $\mathbf{A}_S$, or be estimated as 0. In both instances, the estimate will give $\pm\epsilon n$ error.

Can we beat additive error? It is natural to ask if our approach can be improved to yield sublinear time algorithms with stronger relative error approximation guarantees for $\mathbf{A}$'s eigenvalues. Unfortunately, this is not possible – consider a matrix with just a single pair of entries $\mathbf{A}_{i,j}, \mathbf{A}_{j,i}$ set to 1. To obtain relative error approximations to the two non-zero eigenvalues, we must find the pair (i,j), as otherwise we cannot distinguish $\mathbf{A}$ from the all zeros matrix. This requires reading a $\Omega(n^2)$ of $\mathbf{A}$'s entries. More generally, consider $\mathbf{A}$ with a random $n/t \times n/t$ principal submatrix populated by all 1s, and with all other entries equal to 0. $\mathbf{A}$ has largest eigenvalue n/t. However, if we read $s \ll t^2$ entries of $\mathbf{A}$, with good probability, we will not see even a single one, and thus we will not be able to distinguish $\mathbf{A}$ from the all zeros matrix. This example establishes that any sublinear time algorithm with query complexity s must incur additive error at least $\Omega(n/\sqrt{s})$.

1.3.1 Improved Bounds via Non-Uniform Sampling

We now discuss how to give improved approximation bounds via non-uniform sampling. We focus on the $\pm\epsilon\sqrt{\text{nnz}(\mathbf{A})}$ bound of Theorem 3 using sparsity-based sampling. Theorem 4's proof (for row norm sampling) follows the same general ideas, but with some additional complications.

Theorem 3 requires sampling a submatrix $\mathbf{A}_S$, where each index i is included in S with probability $p_i = \min(1, \frac{s\,\mathrm{nnz}(\mathbf{A}_i)}{\mathrm{nnz}(\mathbf{A})})$. We reweight each sampled row by $\frac{1}{\sqrt{p_i}}$. Thus, if entry $\mathbf{A}_{ij}$ is sampled, it is scaled by $\frac{1}{\sqrt{p_i \cdot p_j}}$. When the rows have uniform sparsity (so all $p_i = s/n$), this ensures that the full submatrix is scaled by n/s, as in Theorem 1.

The proof of Theorem 3 follows the same outline as that of Theorem 1: we first argue that the outlying eigenvectors in $\mathbf{V}_o$ are incoherent, giving a bound on the norm of each row of $\mathbf{V}_o$ in terms of $\mathrm{nnz}(\mathbf{A}_i)$. We then apply a matrix Bernstein bound and Bhatia's non-Hermitian eigenvalue perturbation bound to show that the eigenvalues of $\mathbf{A}_{o,S}$ approximate those of $\mathbf{A}_o$ up to $\pm\epsilon\sqrt{\mathrm{nnz}(\mathbf{A})}$.

Bounding the spectral norm of $\mathbf{A}_{m,S}$. The major challenge is showing that the subsampled middle eigendirections do not significantly increase the approximation error by bounding the $\|\mathbf{A}_{m,S}\|_2$ by $\epsilon\sqrt{\mathrm{nnz}(\mathbf{A})}$. This is difficult since the indices in $\mathbf{A}_{m,S}$ are sampled nonuniformly, so existing bounds [44] on the spectral norm of uniformly random submatrices do not apply. We extend these bounds to the non-uniform sampling case, but still face an issue due to the rescaling of entries by $\frac{1}{\sqrt{p_i p_j}}$. In fact, without additional algorithmic modifications, $\|\mathbf{A}_{m,S}\|_2$ is simply not bounded by $\epsilon\sqrt{\mathrm{nnz}(\mathbf{A})}$! For example, as already discussed, if $\mathbf{A} = \mathbf{I}$ is the identity matrix, we get $\mathbf{A}_{m,S} = \frac{n}{s} \cdot \mathbf{I}$ and so $\|\mathbf{A}_{m,S}\|_2 = \frac{n}{s} > \epsilon\sqrt{\mathrm{nnz}(\mathbf{A})}$, assuming $s < \frac{\sqrt{n}}{\epsilon}$. Relatedly, suppose that $\mathbf{A}$ is tridiagonal, with zeros on the diagonal and ones on the first diagonals above and below the main diagonal. Then, if $s \geq \sqrt{n}$, with constant probability, one of the ones will be sampled and scaled by $\frac{n}{s}$. Thus, we will again have $\|\mathbf{A}_{m,S}\|_2 \geq \frac{n}{s} \geq \epsilon\sqrt{\mathrm{nnz}(\mathbf{A})}$, assuming $s < \frac{\sqrt{n}}{2\epsilon}$. Observe that this issue arrises even when trying to approximate just the singular values (the eigenvalue magnitudes). Thus, while an analogous bound to the uniform sampling result of Theorem 1 can easily be given for singular value estimation via matrix concentration inequalities (see Appendix G of [10]), to the best of our knowledge, Theorems 3 and 4 are the first of their kind even for singular value estimation.

Zeroing out entries in sparse rows/columns. To handle the above cases, we prove a novel perturbation bound, arguing that the eigenvalues of $\mathbf{A}$ are not perturbed by more than $\epsilon\sqrt{\mathrm{nnz}(\mathbf{A})}$ if we zero out any entry $\mathbf{A}_{ij}$ of $\mathbf{A}$ where $\sqrt{\mathrm{nnz}(\mathbf{A}_i) \cdot \mathrm{nnz}(\mathbf{A}_j)} \leq \frac{\epsilon\sqrt{\mathrm{nnz}(\mathbf{A})}}{c \log n}$. This can be thought of as a strengthening of Girshgorin's circle theorem, which would ensure that zeroing out entries in rows/columns with $\mathrm{nnz}(\mathbf{A}_i) \leq \epsilon\sqrt{\mathrm{nnz}(\mathbf{A})}$ does not perturb the eigenvalues by more than $\epsilon\sqrt{\mathrm{nnz}(\mathbf{A})}$. Armed with this perturbation bound, we argue that if we zero out the appropriate entries of $\mathbf{A}_S$ before computing its eigenvalues, then since we have removed entries in very sparse rows and columns which would be scaled by a large $\frac{1}{\sqrt{p_i p_j}}$ factor in $\mathbf{A}_S$, we can bound $\|\mathbf{A}_{m,S}\|_2$. This requires relating the magnitudes of the entries in $\mathbf{A}_{m,S}$ to those in $\mathbf{A}_S$ using the incoherence of the top eigenvectors, which gives bounds on the entries of $\mathbf{A}_{o,S} = \mathbf{A}_S - \mathbf{A}_{m,S}$.

Sampling model. We note that the sparsity-based sampling of Theorem 3 can be efficiently implemented in several natural settings. Given a matrix stored in sparse format, i.e., as a list of nonzero entries, we can easily sample a row with probability $\frac{\mathrm{nnz}(\mathbf{A}_i)}{\mathrm{nnz}(\mathbf{A})}$ by sampling a uniformly random non-zero entry and looking at its corresponding row. Via standard techniques, we can convert several such samples into a sampled set S close in distribution to having each $i \in [n]$ included independently with probability $\min\left(1, \frac{s\,\mathrm{nnz}(\mathbf{A}_i)}{\mathrm{nnz}(\mathbf{A})}\right)$. If we store the values of $\mathrm{nnz}(\mathbf{A}), \mathrm{nnz}(\mathbf{A}_1), \ldots, \mathrm{nnz}(\mathbf{A}_n)$, we can also efficiently access each p_i, which is

needed for rescaling and zeroing out entries. Also observe that if $\mathbf{A}$ is the adjacency matrix of a graph, in the standard graph query model [23], it is well known how to approximately count edges and sample them uniformly at random, i.e., compute $\text{nnz}(\mathbf{A})$ and sample its nonzero entries, in sublinear time [24, 17]. Further, it is typically assumed that one has access to the node degrees, i.e., $\text{nnz}(\mathbf{A}_1), \ldots, \text{nnz}(\mathbf{A}_n)$. Thus, our algorithm can naturally be used to estimate spectral graph properties in sublinear time.

The ℓ_2 norm-based sampling of Theorem 4 can also be performed efficiently using an augmented data structure for storing $\mathbf{A}$. Such data structures have been used extensively in the literature on quantum-inspired algorithms, and require just $O(\text{nnz}(\mathbf{A}))$ time to construct, $O(\text{nnz}(\mathbf{A}))$ space, and $O(\log n)$ time to update give an update to an entry of $\mathbf{A}$ [43, 12].

1.4 Towards Optimal Query Complexity

As discussed, Bakshi et al. [5] show that any algorithm which can test with good probability whether $\mathbf{A}$ has an eigenvalue $\leq -\epsilon n$ or else has all non-negative eigenvalues must read $\tilde{\Omega}\left(\frac{1}{\epsilon^2}\right)$ entries of $\mathbf{A}$. This testing problem is strictly easier than outputting $\pm\epsilon n$ error estimates of all eigenvalues, so gives a lower bound for our setting. If the queried entries are restricted to fall in a submatrix, [5] shows that this submatrix must have dimensions $\Omega\left(\frac{1}{\epsilon^2}\right) \times \Omega\left(\frac{1}{\epsilon^2}\right)$, giving total query complexity $\Omega\left(\frac{1}{\epsilon^4}\right)$. Closing the gap between our upper bound of $\tilde{O}\left(\frac{\log^3 n}{\epsilon^3}\right) \times \tilde{O}\left(\frac{\log^3 n}{\epsilon^3}\right)$ and the lower bound of $\Omega\left(\frac{1}{\epsilon^2}\right) \times \Omega\left(\frac{1}{\epsilon^2}\right)$ for submatrix queries is an intriguing open question.

Closing the gap. We show in Appendix A of [10] that this gap can be easily closed via a surprisingly simple argument if $\mathbf{A}$ is positive semidefinite (PSD). In that case, $\mathbf{A} = \mathbf{B}\mathbf{B}^T$ with $\mathbf{B} \in \mathbb{R}^{n \times n}$. Writing $\mathbf{A}_S = \mathbf{S}^T\mathbf{A}\mathbf{S}$ for a sampling matrix $\mathbf{S} \in \mathbb{R}^{n \times |S|}$, the non-zero eigenvalues of $\mathbf{A}_S$ are identical to those of $\mathbf{B}\mathbf{S}\mathbf{S}^T\mathbf{B}^T$. Via a standard approximate matrix multiplication analysis [16], one can then show that, for $s \geq \frac{1}{\epsilon^2\delta}$, with probability at least $1-\delta$, $\|\mathbf{B}\mathbf{B}^T - \mathbf{B}\mathbf{S}\mathbf{S}^T\mathbf{B}\|_F \leq \epsilon n$. Via Weyl's inequality, this shows that the eigenvalues of $\mathbf{B}\mathbf{S}\mathbf{S}^T\mathbf{B}$, and hence $\mathbf{A}_S$, approximate those of $\mathbf{A}$ up to $\pm\epsilon n$ error.[2]

Unfortunately, this approach breaks down when $\mathbf{A}$ has negative eigenvalues, and so cannot be factored as $\mathbf{B}\mathbf{B}^T$ for real $\mathbf{B} \in \mathbb{R}^{n \times n}$. This is more than a technical issue: observe that when $\mathbf{A}$ is PSD and has $\|\mathbf{A}\|_\infty \leq 1$, it can have at most $1/\epsilon$ eigenvalues larger than ϵn – since its trace, which is equal to the sum of its eigenvalues, is bounded by n, and since all eigenvalues are non-negative. When $\mathbf{A}$ is not PSD, it can have $\Omega(1/\epsilon^2)$ eigenvalues with magnitude larger than ϵn. In particular, if $\mathbf{A}$ is the tensor product of a $1/\epsilon^2 \times 1/\epsilon^2$ random ± 1 matrix and the $\epsilon^2 n \times \epsilon^2 n$ all ones matrix, the bulk of its eigenvalues (of which there are $1/\epsilon^2$) will concentrate around $1/\epsilon \cdot \epsilon^2 n = \epsilon n$. As a result it remains unclear whether we can match the $1/\epsilon^2$ dependence of the PSD case, or if a stronger lower bound can be shown for indefinite matrices.

Outside the ϵ dependence, it is unknown if full eigenspectrum approximation can be performed with sample complexity independent of the matrix size n. [5] achieve this for the easier positive semidefiniteness testing problem, giving sample complexity $\tilde{O}(1/\epsilon^2)$. However our bounds have additional $\log n$ factors. As discussed, in Appendix B of [10] we give an alternate analysis for Theorem 1, which shows that sampling a $O\left(\frac{\log n}{\epsilon^4\delta}\right) \times O\left(\frac{\log n}{\epsilon^4\delta}\right)$ submatrix suffices for $\pm\epsilon n$ eigenvalue approximation, saving a $\log^2 n$ factor at the cost of

[2] In fact, via more refined eigenvalue perturbation bounds [9] one can show an ℓ_2 norm bound on the eigenvalue approximation errors, which can be much stronger than the ℓ_∞ norm bound of Theorem 1.

worse ϵ dependence. However, removing the final $\log n$ seems difficult – it arises when bounding $\|\mathbf{A}_{m,S}\|_2$ via bounds on the spectral norms of random principal submatrices [38]. Removing it seems as though it would require either improving such bounds, or taking a different algorithmic approach, as simple modifications such as using bounds depending on the intrinsic dimension do not seem to help.

Also note that our $\log n$ and ϵ dependencies for non-uniform sampling (Theorems 3 and 4) are likely not tight. It is not hard to check that the lower bounds of [5] still hold in these settings. For example, in the sparsity-based sampling setting, by simply having the matrix entirely supported on a $\sqrt{\text{nnz}(\mathbf{A})} \times \sqrt{\text{nnz}(\mathbf{A})}$ submatrix, the lower bounds of [5] directly carry over. Giving tight query complexity bounds here would also be interesting. Finally, it would be interesting to go beyond principal submatrix based algorithms, to achieve improved query complexity, as in Corollary 2. Finding an algorithm matching the $\tilde{O}\left(\frac{1}{\epsilon^2}\right)$ overall query complexity lower bound of [5] is open even in the much simpler PSD setting.

2 Notation and Preliminaries

We now define notation and foundational results that we use throughout our work. For any integer n, let $[n]$ denote the set $\{1, 2, \ldots, n\}$. We write matrices and vectors in bold literals – e.g., $\mathbf{A}$ or $\mathbf{x}$. For a vector $\mathbf{x}$, we let $\|\mathbf{x}\|_2$ denote its Euclidean norm. We denote the eigenvalues of a symmetric matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ by $\lambda_1(\mathbf{A}) \geq \ldots \geq \lambda_n(\mathbf{A})$, in decreasing order. A symmetric matrix is positive semidefinite if all its eigenvalues are non-negative. For two matrices $\mathbf{A}, \mathbf{B}$, we let $\mathbf{A} \succeq \mathbf{B}$ denote that $\mathbf{A} - \mathbf{B}$ is positive semidefinite. For any matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $i \in [n]$, we let $\mathbf{A}_i$ denote the i^{th} row of $\mathbf{A}$. We let $\text{nnz}(\mathbf{A})$ denote the total number of non-zero elements in $\mathbf{A}$, $\|\mathbf{A}\|_\infty$ denote the largest magnitude of an entry, and $\|\mathbf{A}\|_2 = \max_{\mathbf{x}} \frac{\|\mathbf{A}\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$ denote the spectral norm. We let $\|\mathbf{A}\|_F = (\sum_{i,j} \mathbf{A}_{ij}^2)^{1/2}$ denote the Frobenius norm, and $\|\mathbf{A}\|_{1\to 2}$ denote the maximum Euclidean norm of a column. For $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $S \subseteq [n]$ we let $\mathbf{A}_S$ denote the principal submatrix corresponding to S. We let $\mathbb{E}_2$ denote the L_2 norm of a random variable, $\mathbb{E}_2[X] = (\mathbb{E}[X^2])^{1/2}$, where $\mathbb{E}[\cdot]$ denotes expectation.

We use the following basic facts and identities on eigenvalues throughout our proofs.

▶ **Fact 1** (Eigenvalue of Matrix Product). *For any two matrices $\mathbf{A} \in \mathbb{C}^{n\times m}, \mathbf{B} \in \mathbb{C}^{m\times n}$, the non-zero eigenvalues of $\mathbf{AB}$ are identical to those of $\mathbf{BA}$.*

▶ **Fact 2** (Gershgorin's circle theorem [20]). *Let $\mathbf{A} \in \mathbb{C}^{n\times n}$ with entries $\mathbf{A}_{ij}$. For $i \in [n]$, let $\mathbf{R}_i$ be the sum of absolute values of non-diagonal entries in the i^{th} row. Let $D(\mathbf{A}_{ii}, \mathbf{R}_i)$ be the closed disc centered at $\mathbf{A}_{ii}$ with radius $\mathbf{R}_i$. Then every eigenvalue of $\mathbf{A}$ lies within one of the discs $D(\mathbf{A}_{ii}, \mathbf{R}_i)$.*

▶ **Fact 3** (Weyl's Inequality [50]). *For any two Hermitian matrices $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{n\times n}$ with $\mathbf{A} - \mathbf{B} = \mathbf{E}$, $\max_i |\lambda_i(\mathbf{A}) - \lambda_i(\mathbf{B})| \leq \|\mathbf{E}\|_2$.*

Weyl's inequality ensures that a small Hermitian perturbation of a Hermitian matrix will not significantly change its eigenvalues. The bound can be extended to the case when the perturbation is not Hermitian, with a loss of an $O(\log n)$ factor; to the best of our knowledge this loss is necessary:

▶ **Fact 4** (Non-Hermitian perturbation bound [9]). *Let $\mathbf{A} \in \mathbb{C}^{n\times n}$ be Hermitian and $\mathbf{B} \in \mathbb{C}^{n\times n}$ be any matrix whose eigenvalues are $\lambda_1(\mathbf{B}), \ldots, \lambda_n(\mathbf{B})$ such that $Re(\lambda_1(\mathbf{B})) \geq \ldots \geq Re(\lambda_n(\mathbf{B}))$ (where $Re(\lambda_i(\mathbf{B}))$ denotes the real part of $\lambda_i(\mathbf{B})$). Let $\mathbf{A} - \mathbf{B} = \mathbf{E}$. For some universal constant C, $\max_i |\lambda_i(\mathbf{A}) - \lambda_i(\mathbf{B})| \leq C \log n \|\mathbf{E}\|_2$*

Beyond the above facts, we use several theorems to obtain eigenvalue concentration bounds. We first state a theorem from [44], which bounds the spectral norm of a principal submatrix sampled uniformly at random from a bounded entry matrix. We build on this to prove the full eigenspectrum concentration result of Theorem 1.

▶ **Theorem 6** (Random principal submatrix spectral norm bound [38, 44]). *Let $\mathbf{A} \in \mathbb{C}^{n\times n}$ be Hermitian, decomposed into diagonal and off-diagonal parts: $\mathbf{A} = \mathbf{D} + \mathbf{H}$. Let $\mathbf{S} \in \mathbb{R}^{n\times n}$ be a diagonal sampling matrix with the j^{th} diagonal entry set to 1 independently with probability s/n and 0 otherwise. Then, for some universal constant C,*

$$\mathbb{E}_2\|\mathbf{SAS}\|_2 \leq C\left[\log n \cdot \mathbb{E}_2\|\mathbf{SHS}\|_\infty + \sqrt{\frac{s\log n}{n}} \cdot \mathbb{E}_2\|\mathbf{HS}\|_{1\to 2} + \frac{s}{n}\cdot\|\mathbf{H}\|_2\right] + \mathbb{E}_2\|\mathbf{SDS}\|_2.$$

For Theorems 3 and 4, we need an extension of Theorem 6 to the setting where rows are sampled non-uniformly. We will use two bounds here. The first is a decoupling and recoupling result for matrix norms. One can prove this lemma following an analogous result in [44] for sampling rows/columns uniformly. The proof is almost identical so we omit it.

▶ **Lemma 7** (Decoupling and recoupling). *Let $\mathbf{H}$ be a Hermitian matrix with zero diagonal. Let δ_j be a sequence of independent random variables such that $\delta_j = \frac{1}{\sqrt{p_j}}$ with probability p_j and 0 otherwise. Let $\mathbf{S}$ be a square diagonal sampling matrix with j^{th} diagonal entry set to δ_j. Then:*

$$\mathbb{E}_2\|\mathbf{SHS}\|_2 \leq 2\mathbb{E}_2\|\mathbf{SH\hat{S}}\|_2 \quad \textit{and} \quad \mathbb{E}_2\|\mathbf{SH\hat{S}}\|_\infty \leq 4\mathbb{E}_2\|\mathbf{SHS}\|_\infty,$$

where $\hat{\mathbf{S}}$ is an independent diagonal sampling matrix drawn from the same distribution as $\mathbf{S}$.

The second theorem bounds the spectral norm of a non-uniform random column sample of a matrix. We give a proof for uniform sampling in Appendix D of [10], following the results of [45].

▶ **Theorem 8** (Non-uniform column sampling – spectral norm bound). *Let $\mathbf{A}$ be an $m \times n$ matrix with rank r. Let δ_j be a sequence of independent random variables such that $\delta_j = \frac{1}{\sqrt{p_j}}$ with probability p_j and 0 otherwise. Let $\mathbf{S}$ be a square diagonal sampling matrix with j^{th} diagonal entry set to δ_j.*

$$\mathbb{E}_2\|\mathbf{AS}\|_2 \leq 5\sqrt{\log r}\cdot\mathbb{E}_2\|\mathbf{AS}\|_{1\to 2} + \|\mathbf{A}\|_2$$

We use a standard Matrix Bernstein inequality to bound the spectral norm of random submatrices.

▶ **Theorem 9** (Matrix Bernstein [46]). *Consider a finite sequence $\{\mathbf{S}_k\}$ of random matrices in $\mathbb{R}^{d\times d}$. Assume that for all k, $\mathbb{E}[\mathbf{S}_k] = \mathbf{0}$ and $\|\mathbf{S}_k\|_2 \leq L$. Let $\mathbf{Z} = \sum_k \mathbf{S}_k$ and let $\mathbf{V}_1, \mathbf{V}_2$ be semidefinite upper-bounds for the matrix valued variances $\mathbf{Var}_1(\mathbf{Z})$ and $\mathbf{Var}_2(\mathbf{Z})$:*

$$\mathbf{V}_1 \succeq \mathbf{Var}_1(\mathbf{Z}) \stackrel{\text{def}}{=} \mathbb{E}\left(\mathbf{ZZ}^T\right) = \sum_k \mathbb{E}\left(\mathbf{S}_k\mathbf{S}_k^T\right), \quad \textit{and}$$

$$\mathbf{V}_2 \succeq \mathbf{Var}_2(\mathbf{Z}) \stackrel{\text{def}}{=} \mathbb{E}\left(\mathbf{Z}^T\mathbf{Z}\right) = \sum_k \mathbb{E}\left(\mathbf{S}_k^T\mathbf{S}_k\right).$$

Then, letting $v = \max(\|\mathbf{V}_1\|_2, \|\mathbf{V}_2\|_2)$, for any $t \geq 0$,

$$\mathbb{P}(\|\mathbf{Z}\|_2 \geq t) \leq 2d\cdot\exp\left(\frac{-t^2/2}{v + Lt/3}\right).$$

For real valued random variables, we use the standard Bernstein inequality.

▶ **Theorem 10** (Bernstein inequality [8]). *Let $\{z_j\}$ for $j \in [n]$ be independent random variables with zero mean such that $|z_j| \leq M$ for all j. Then for all positive t,*

$$\mathbb{P}\left(\left|\sum_{j=1}^{n} z_j\right| \geq t\right) \leq \exp\left(\frac{-t^2/2}{\sum_{i=1}^{n} \mathbb{E}[z_i^2] + Mt/3}\right).$$

3 Sublinear Time Eigenvalue Estimation using Uniform Sampling

We now prove our main eigenvalue estimation result – Theorem 1. We give the pseudocode for our principal submatrix based estimation procedure in Algorithm 1. We will show that any positive or negative eigenvalue of $\mathbf{A}$ with magnitude $\geq \epsilon n$ will appear as an approximate eigenvalue in $\mathbf{A}_S$ with good probability. Thus, in step 5 of Algorithm 1, the positive and negative eigenvvalues of $\mathbf{A}_S$ are used to estimate the outlying largest and smallest eigenvalues of $\mathbf{A}$. All other interior eigenvalues of $\mathbf{A}$ are estimated to be 0, which will immediately give our $\pm\epsilon n$ approximation bound when the original eigenvalue has magnitude $\leq \epsilon n$.

Algorithm 1 Eigenvalue estimator using uniform sampling.

1: **Input:** Symmetric $\mathbf{A} \in \mathbb{R}^{n\times n}$ with $\|\mathbf{A}\|_\infty \leq 1$, Accuracy $\epsilon \in (0,1)$, failure prob. $\delta \in (0,1)$.
2: Fix $s = \frac{c\log(1/(\epsilon\delta))\cdot\log^3 n}{\epsilon^3\delta}$ where c is a sufficiently large constant.
3: Add each index $i \in [n]$ to the sample set S independently with probability $\frac{s}{n}$. Let the principal submatrix of $\mathbf{A}$ corresponding S be $\mathbf{A}_S$.
4: Compute the eigenvalues of $\mathbf{A}_S$: $\lambda_1(\mathbf{A}_S) \geq \ldots \geq \lambda_{|S|}(\mathbf{A}_S)$.
5: For all $i \in [|S|]$ with $\lambda_i(\mathbf{A}_S) \geq 0$, let $\tilde{\lambda}_i(\mathbf{A}) = \frac{n}{s}\cdot\lambda_i(\mathbf{A}_S)$. For all $i \in [|S|]$ with $\lambda_i(\mathbf{A}_S) < 0$, let $\tilde{\lambda}_{n-(|S|-i)}(\mathbf{A}) = \frac{n}{s}\cdot\lambda_i(\mathbf{A}_S)$. For all remaining $i \in [n]$, let $\tilde{\lambda}_i(\mathbf{A}) = 0$.
6: **Return:** Eigenvalue estimates $\tilde{\lambda}_1(\mathbf{A}) \geq \ldots \geq \tilde{\lambda}_n(\mathbf{A})$.

Running time. Observe that the expected number of indices chosen by Algorithm 1 is $s = \frac{c\log(1/(\epsilon\delta))\cdot\log^3 n}{\epsilon^3\delta}$. A standard concentration bound can be used to show that with high probability $(1 - 1/\operatorname{poly}(n))$, the number of sampled entries is $O(s)$. Thus, the algorithm reads a total of $O(s^2)$ entries of $\mathbf{A}$ and runs in $O(s^\omega)$ time – the time to compute a full eigendecomposition of $\mathbf{A}_S$.

3.1 Outer and Middle Eigenvalue Bounds

Recall that we will split $\mathbf{A}$ into two symmetric matrices (Definition 5): $\mathbf{A}_o = \mathbf{V}_o\mathbf{\Lambda}_o\mathbf{V}_o^T$ which contains its large magnitude (outlying) eigendirections with eigenvalue magnitudes $\geq \epsilon\sqrt{\delta}n$ and $\mathbf{A}_m = \mathbf{V}_m\mathbf{\Lambda}_m\mathbf{V}_m^T$ which contains its small magnitude (middle) eigendirections.

We first show that the eigenvectors in $\mathbf{V}_o$ are *incoherent*. I.e., that their (eigenvalue weighted) squared row norms are bounded. This ensures that the outlying eigenspace of $\mathbf{A}$ is well-approximated via uniform sampling.

▶ **Lemma 11** (Incoherence of outlying eigenvectors). *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be symmetric with $\|\mathbf{A}\|_\infty \leq 1$. Let $\mathbf{V}_o$ be as in Definition 5. Let $\mathbf{V}_{o,i}$ denote the i^{th} row of $\mathbf{V}_o$. Then,*

$$\|\mathbf{\Lambda}_o^{1/2}\mathbf{V}_{o,i}\|_2^2 \leq \frac{1}{\epsilon\sqrt{\delta}} \qquad \text{and} \qquad \|\mathbf{V}_{o,i}\|_2^2 \leq \frac{1}{\epsilon^2\delta n}.$$

Proof. Observe that $\mathbf{AV}_o = \mathbf{V}_o\mathbf{\Lambda}_o$. Let $[\mathbf{AV}_o]_i$ denote the i^{th} row of the $\mathbf{AV}_o$. Then we have

$$\|[\mathbf{AV}_o]_i\|_2^2 = \|[\mathbf{V}_o\mathbf{\Lambda}_o]_i\|_2^2 = \sum_{j=1}^{r} \lambda_j^2 \cdot \mathbf{V}_{o,i,j}^2, \tag{2}$$

where $r = \text{rank}(\mathbf{A}_o)$, $\mathbf{V}_{o,i,j}$ is the $(i,j)^{\text{th}}$ element of $\mathbf{V}_o$ and $\lambda_j = \mathbf{\Lambda}_o(j,j)$. $\|\mathbf{A}\|_\infty \leq 1$ by assumption and since $\mathbf{V}_o$ has orthonormal columns, its spectral norm is bounded by 1, thus we have $\|[\mathbf{AV}_o]_i\|_2^2 = \|[\mathbf{A}]_i\mathbf{V}_o\|_2^2 \leq \|[\mathbf{A}]_i\|_2^2 \cdot \|\mathbf{V}_o\|_2^2 \leq n$. Therefore, by (2), we have:

$$\sum_{j=1}^{r} \lambda_j^2 \cdot \mathbf{V}_{o,i,j}^2 \leq n. \tag{3}$$

Since by definition of $\mathbf{\Lambda}_o$, $|\lambda_j| \geq \epsilon\sqrt{\delta}n$ for all j, we finally have $\|\mathbf{\Lambda}_o^{1/2}\mathbf{V}_{o,i}\|_2^2 = \sum_{j=1}^{r} \lambda_j \cdot \mathbf{V}_{o,i,j}^2 \leq \frac{n}{\epsilon\sqrt{\delta}n} = \frac{1}{\epsilon\sqrt{\delta}}$ and $\|\mathbf{V}_{o,i}\|_2^2 = \sum_{j=1}^{r} \mathbf{V}_{o,i,j}^2 \leq \frac{n}{\epsilon^2\delta n^2} = \frac{1}{\epsilon^2\delta n}$. ◀

Let $\bar{\mathbf{S}} \in \mathbb{R}^{n\times|S|}$ be the scaled sampling matrix satisfying $\bar{\mathbf{S}}^T\mathbf{A}\bar{\mathbf{S}} = \frac{n}{s} \cdot \mathbf{A}_S$. We next apply Lemma 11 in conjunction with a matrix Bernstein bound to show that $\mathbf{\Lambda}_o^{1/2}\mathbf{V}_o^T\bar{\mathbf{S}}\bar{\mathbf{S}}^T\mathbf{V}_o\mathbf{\Lambda}_o^{1/2}$ concentrates around its expectation, $\mathbf{\Lambda}_o$. Since by Fact 1, this matrix has identical eigenvalues to $\frac{n}{s}\cdot\mathbf{A}_{o,S} = \bar{\mathbf{S}}^T\mathbf{V}_o\mathbf{\Lambda}_o\mathbf{V}_o^T\bar{\mathbf{S}}$, this allows us to argue that the eigenvalues of $\frac{n}{s}\cdot\mathbf{A}_{o,S}$ approximate those of $\mathbf{\Lambda}_o$.

▶ **Lemma 12** (Concentration of outlying eigenvalues). *Let $S \subseteq [n]$ be sampled as in Algorithm 1 for $s \geq \frac{c\log(1/(\epsilon\delta))}{\epsilon^3\sqrt{\delta}}$ where c is a sufficiently large constant. Let $\bar{\mathbf{S}} \in \mathbb{R}^{n\times|S|}$ be the scaled sampling matrix satisfying $\bar{\mathbf{S}}^T\mathbf{A}\bar{\mathbf{S}} = \frac{n}{s} \cdot \mathbf{A}_S$. Letting $\mathbf{\Lambda}_o, \mathbf{V}_o$ be as in Definition 5, with probability at least $1-\delta$,*

$$\|\mathbf{\Lambda}_o^{1/2}\mathbf{V}_o^T\bar{\mathbf{S}}\bar{\mathbf{S}}^T\mathbf{V}_o\mathbf{\Lambda}_o^{1/2} - \mathbf{\Lambda}_o\|_2 \leq \epsilon n.$$

Proof. Define $\mathbf{E} = \mathbf{\Lambda}_o^{1/2}\mathbf{V}_o^T\bar{\mathbf{S}}\bar{\mathbf{S}}^T\mathbf{V}_o\mathbf{\Lambda}_o^{1/2} - \mathbf{\Lambda}_o$. For all $i \in [n]$, let $\mathbf{V}_{o,i}$ be the i^{th} row of $\mathbf{V}_o$ and define the matrix valued random variable

$$\mathbf{Y}_i = \begin{cases} \frac{n}{s}\mathbf{\Lambda}_o^{1/2}\mathbf{V}_{o,i}\mathbf{V}_{o,i}^T\mathbf{\Lambda}_o^{1/2}, & \text{with probability } s/n \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Define $\mathbf{Q}_i = \mathbf{Y}_i - \mathbb{E}[\mathbf{Y}_i]$. Observe that $\mathbf{Q}_1, \ldots, \mathbf{Q}_n$ are independent random variables and that $\sum_{i=1}^{n}\mathbf{Q}_i = \mathbf{\Lambda}_o^{1/2}\mathbf{V}_o^T\bar{\mathbf{S}}\bar{\mathbf{S}}^T\mathbf{V}_o\mathbf{\Lambda}_o^{1/2} - \mathbf{\Lambda}_o = \mathbf{E}$. Further, observe that $\|\mathbf{Q}_i\|_2 \leq \max\left(1, \frac{n}{s}-1\right)\cdot\|\mathbf{\Lambda}_o^{1/2}\mathbf{V}_{o,i}\mathbf{V}_{o,i}^T\mathbf{\Lambda}_o^{1/2}\|_2 \leq \max\left(1, \frac{n}{s}-1\right)\cdot\|\mathbf{\Lambda}_o^{1/2}\mathbf{V}_{o,i}\|_2^2$. Now, $\|\mathbf{\Lambda}_o^{1/2}\mathbf{V}_{o,i}\|_2^2 \leq \frac{1}{\epsilon\sqrt{\delta}}$ by Lemma 11. Thus, $\|\mathbf{Q}_i\|_2 \leq \frac{n}{\epsilon\sqrt{\delta}s}$. The variance $\mathbf{Var}(\mathbf{E}) \overset{\text{def}}{=} \mathbb{E}(\mathbf{E}\mathbf{E}^T) = \mathbb{E}(\mathbf{E}^T\mathbf{E}) = \sum_{i=1}^{n}\mathbb{E}[\mathbf{Q}_i^2]$ can be bounded as:

$$\begin{aligned}\sum_{i=1}^{n}\mathbb{E}[\mathbf{Q}_i^2] &= \sum_{i=1}^{n}\left[\frac{s}{n}\cdot\left(\frac{n}{s}-1\right)^2 + \left(1-\frac{s}{n}\right)\right]\cdot(\mathbf{\Lambda}_o^{1/2}\mathbf{V}_{o,i}\mathbf{V}_{o,i}^T\mathbf{\Lambda}_o\mathbf{V}_{o,i}\mathbf{V}_{o,i}^T\mathbf{\Lambda}_o^{1/2}) \\ &\preceq \sum_{i=1}^{n}\frac{n}{s}\cdot\|\mathbf{\Lambda}_o^{1/2}\mathbf{V}_{o,i}\|_2^2\cdot(\mathbf{\Lambda}_o^{1/2}\mathbf{V}_{o,i}\mathbf{V}_{o,i}^T\mathbf{\Lambda}_o^{1/2}).\end{aligned} \tag{5}$$

Again by Lemma 11, $\|\mathbf{\Lambda}_o^{1/2}\mathbf{V}_{o,i}\|_2^2 \leq \frac{1}{\epsilon\sqrt{\delta}}$. Plugging back into (5) we can bound,

$$\sum_{i=1}^{n}\mathbb{E}[\mathbf{Q}_i^2] \preceq \sum_{i=1}^{n}\frac{n}{s}\cdot\frac{1}{\epsilon\sqrt{\delta}}\cdot(\mathbf{\Lambda}_o^{1/2}\mathbf{V}_{o,i}\mathbf{V}_{o,i}^T\mathbf{\Lambda}_o^{1/2}) = \frac{n}{s\epsilon\sqrt{\delta}}\mathbf{\Lambda}_o \preceq \frac{n^2}{s\epsilon\sqrt{\delta}}\cdot\mathbf{I}.$$

Since $\mathbf{Q}_i^2$ is PSD, this establishes that $\|\mathbf{Var}(\mathbf{E})\|_2 \leq \frac{n^2}{s\epsilon\sqrt{\delta}}$. We then apply Theorem 9 (the matrix Bernstein inequality) with $L = \frac{n}{s\epsilon\sqrt{\delta}}$, $v = \frac{n^2}{s\epsilon\sqrt{\delta}}$, and $d \leq \frac{1}{\epsilon^2\delta}$ since there are at most $\frac{\|\mathbf{A}\|_F^2}{\delta\epsilon^2 n^2} \leq \frac{1}{\epsilon^2\delta}$ outlying eigenvalues with magnitude $\geq \sqrt{\delta}\epsilon n$ in $\mathbf{\Lambda}_o$. This gives:

$$\mathbb{P}\left(\|\mathbf{E}\|_2 \geq \epsilon n\right) \leq \frac{2}{\epsilon^2\delta} \cdot \exp\left(\frac{-\epsilon^2 n^2/2}{v + L\epsilon n/3}\right) \leq \frac{2}{\epsilon^2\delta} \cdot \exp\left(\frac{-\epsilon^2 n^2/2}{\frac{n^2}{s\epsilon\sqrt{\delta}} + \frac{\epsilon n^2}{3s\epsilon\sqrt{\delta}}}\right)$$
$$\leq \frac{2}{\epsilon^2\delta} \cdot \exp\left(\frac{-s\epsilon^3\sqrt{\delta}}{4}\right).$$

Thus, if we set $s \geq \frac{c\log(1/(\epsilon\delta))}{\epsilon^3\sqrt{\delta}}$ for large enough c, then the probability is bounded above by δ, completing the proof. ◀

We cannot prove an analogous leverage score bound to Lemma 11 for the interior eigenvectors of $\mathbf{A}$ appearing in $\mathbf{V}_m$. Thus we cannot apply a matrix Bernstein bound as in Lemma 12. However, we can use Theorem 6 to show that the spectral norm of the random principal submatrix $\mathbf{A}_{m,S}$ is not too large, and thus that the eigenvalues of $\mathbf{A}_S = \mathbf{A}_{o,S} + \mathbf{A}_{m,S}$ are close to those of $\mathbf{A}_{o,S}$.

▶ **Lemma 13** (Spectral norm bound – sampled middle eigenvalues)**.** *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be symmetric with $\|\mathbf{A}\|_\infty \leq 1$. Let $\mathbf{A}_m$ be as in Definition 5. Let S be sampled as in Algorithm 1. If $s \geq \frac{c\log n}{\epsilon^2\delta}$ for some sufficiently large constant c, then with probability at least $1-\delta$, $\|\mathbf{A}_{m,S}\|_2 \leq \epsilon s$.*

Proof. Let $\mathbf{A}_m = \mathbf{D}_m + \mathbf{H}_m$ where $\mathbf{D}_m$ is the matrix of diagonal elements and $\mathbf{H}_m$ the matrix of off-diagonal elements. Let $\mathbf{S} \in \mathbb{R}^{n\times|S|}$ be the binary sampling matrix with $\mathbf{A}_{m,S} = \mathbf{S}^T\mathbf{A}_m\mathbf{S}$. From Theorem 6, we have for some constant C,

$$\mathbb{E}_2[\|\mathbf{A}_{m,S}\|_2] \leq C\left[\log n \cdot \mathbb{E}_2[\|\mathbf{S}^T\mathbf{H}_m\mathbf{S}\|_\infty] + \sqrt{\frac{s\log n}{n}}\mathbb{E}_2[\|\mathbf{H}_m\mathbf{S}\|_{1\to2}] + \frac{s}{n}\|\mathbf{H}_m\|_2\right] + \mathbb{E}_2[\|\mathbf{S}^T\mathbf{D}_m\mathbf{S}\|]. \tag{6}$$

Considering the various terms in (6), we have $\|\mathbf{S}^T\mathbf{H}_m\mathbf{S}\|_\infty \leq \|\mathbf{A}_m\|_\infty$ and $\|\mathbf{S}^T\mathbf{D}_m\mathbf{S}\|_2 = \|\mathbf{S}^T\mathbf{D}_m\mathbf{S}\|_\infty \leq \|\mathbf{A}_m\|_\infty$. We also have

$$\|\mathbf{H}_m\|_2 \leq \|\mathbf{A}_m\|_2 + \|\mathbf{D}_m\|_2 \leq \|\mathbf{A}_m\|_2 + \|\mathbf{A}_m\|_\infty \leq \epsilon\delta^{1/2}n + \|\mathbf{A}_m\|_\infty$$

and

$$\|\mathbf{H}_m\mathbf{S}\|_{1\to2} \leq \|\mathbf{A}_m\mathbf{S}\|_{1\to2} \leq \|\mathbf{A}_m\|_{1\to2} \leq \sqrt{n}.$$

The final bound follows since $\mathbf{A}_m = \mathbf{V}_m\mathbf{V}_m^T\mathbf{A}$, where $\mathbf{V}_m\mathbf{V}_m^T$ is an orthogonal projection matrix. Thus, $\|\mathbf{A}_m\|_{1\to2} \leq \|\mathbf{A}\|_{1\to2} \leq \sqrt{n}$ by our assumption that $\|\mathbf{A}\|_\infty \leq 1$. Plugging all these bounds into (6) we have, for some constant C,

$$\mathbb{E}_2[\|\mathbf{A}_{m,S}\|_2] \leq C\left[\log n \cdot \|\mathbf{A}_m\|_\infty + \sqrt{\log n \cdot s} + s \cdot \epsilon\delta^{1/2}\right]. \tag{7}$$

It remains to bound $\|\mathbf{A}_m\|_\infty$. We have $\mathbf{A} = \mathbf{A}_m + \mathbf{A}_o$ and thus by triangle inequality,

$$\|\mathbf{A}_m\|_\infty \leq \|\mathbf{A}\|_\infty + \|\mathbf{A}_o\|_\infty = 1 + \|\mathbf{A}_o\|_\infty. \tag{8}$$

Writing $\mathbf{A}_o = \mathbf{V}_o \mathbf{\Lambda}_o \mathbf{V}_o^T$ (see Definition 5), and letting $\mathbf{V}_{o,i}$ denote the i^{th} row of $\mathbf{V}_o$, the $(i,j)^{\text{th}}$ element of $\mathbf{A}_o$ has magnitude

$$|\mathbf{A}_{o,i,j}| = |\mathbf{V}_{o,i} \mathbf{\Lambda}_o \mathbf{V}_{o,j}^T| \leq \|\mathbf{V}_{o,i}\|_2 \cdot \|\mathbf{\Lambda}_o \mathbf{V}_{o,j}^T\|_2,$$

by Cauchy-Schwarz. From Lemma 11, we have $\|\mathbf{V}_{o,i}\|_2 \leq \frac{1}{\epsilon \delta^{1/2} \sqrt{n}}$. Also, from (2), $\|\mathbf{\Lambda}_o \mathbf{V}_{o,j}^T\|_2 = \|[\mathbf{A}\mathbf{V}_o]_j\|_2 \leq \sqrt{n}$. Overall, for all i,j we have $\mathbf{A}_{o,i,j} \leq \frac{1}{\epsilon \delta^{1/2} \sqrt{n}} \cdot \sqrt{n} = \frac{1}{\epsilon \delta^{1/2}}$, giving $\|\mathbf{A}_o\|_\infty \leq \frac{1}{\epsilon \delta^{1/2}}$. Plugging back into (8) and in turn (7), we have for some constant C,

$$\mathbb{E}_2[\|\mathbf{A}_{m,S}\|_2] \leq C \left[\frac{\log n}{\epsilon \delta^{1/2}} + \sqrt{s \log n} + s \epsilon \delta^{1/2} \right].$$

Setting $s \geq \frac{c \log n}{\epsilon^2 \delta}$ for sufficiently large c, all terms in the right hand side of the above equation are bounded by $\epsilon \sqrt{\delta} s$ and so

$$\mathbb{E}_2[\|\mathbf{A}_{m,S}\|_2] \leq 3 \epsilon \sqrt{\delta} s$$

Thus, by Markov's inequality, with probability at least $1 - \delta$, we have $\|\mathbf{A}_{m,S}\|_2 \leq 3\epsilon s$. We can adjust ϵ by a constant to obtain the required bound. ◀

3.2 Main Accuracy Bounds

We now restate our main result, and give its proof via Lemmas 12 and 13.

▶ **Theorem 1** (Sublinear Time Eigenvalue Approximation). *Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be symmetric with $\|\mathbf{A}\|_\infty \leq 1$ and eigenvalues $\lambda_1(\mathbf{A}) \geq \ldots \geq \lambda_n(\mathbf{A})$. Let $S \subseteq [n]$ be formed by including each index independently with probability s/n as in Algorithm 1. Let $\mathbf{A}_S$ be the corresponding principal submatrix of $\mathbf{A}$, with eigenvalues $\lambda_1(\mathbf{A}_S) \geq \ldots \geq \lambda_{|S|}(\mathbf{A}_S)$.*

For all $i \in [|S|]$ with $\lambda_i(\mathbf{A}_S) \geq 0$, let $\tilde{\lambda}_i(\mathbf{A}) = \frac{n}{s} \cdot \lambda_i(\mathbf{A}_S)$. For all $i \in [|S|]$ with $\lambda_i(\mathbf{A}_S) < 0$, let $\tilde{\lambda}_{n-(|S|-i)}(\mathbf{A}) = \frac{n}{s} \cdot \lambda_i(\mathbf{A}_S)$. For all other $i \in [n]$, let $\tilde{\lambda}_i(\mathbf{A}) = 0$. If $s \geq \frac{c \log(1/(\epsilon\delta)) \cdot \log^3 n}{\epsilon^3 \delta}$, for large enough constant c, then with probability $\geq 1 - \delta$, for all $i \in [n]$, $\lambda_i(\mathbf{A}) - \epsilon n \leq \tilde{\lambda}_i(\mathbf{A}) \leq \lambda_i(\mathbf{A}) + \epsilon n$.

Proof. Let $\mathbf{S} \in \mathbb{R}^{n \times |S|}$ be the binary sampling matrix with a single one in each column such that $\mathbf{S}^T \mathbf{A} \mathbf{S} = \mathbf{A}_S$. Let $\bar{\mathbf{S}} = \sqrt{n/s} \cdot \mathbf{S}$. Following Definition 5, we write $\mathbf{A} = \mathbf{A}_o + \mathbf{A}_m$. By Fact 1 we have that the nonzero eigenvalues of $\frac{n}{s} \cdot \mathbf{A}_{o,S} = \bar{\mathbf{S}}^T \mathbf{V}_o \mathbf{\Lambda}_o \mathbf{V}_o^T \bar{\mathbf{S}}$ are identical to those of $\mathbf{\Lambda}_o^{1/2} \mathbf{V}_o^T \bar{\mathbf{S}} \bar{\mathbf{S}}^T \mathbf{V}_o \mathbf{\Lambda}_o^{1/2}$ where $\mathbf{\Lambda}_o^{1/2}$ is the square root matrix of $\mathbf{\Lambda}_o$ such that $\mathbf{\Lambda}_o^{1/2} \mathbf{\Lambda}_o^{1/2} = \mathbf{\Lambda}_o$.

Note that $\mathbf{\Lambda}_o$ is Hermitian. However $\mathbf{\Lambda}_o^{1/2}$ may be complex, and hence $\mathbf{\Lambda}_o^{1/2} \mathbf{V}_o^T \bar{\mathbf{S}} \bar{\mathbf{S}}^T \mathbf{V}_o \mathbf{\Lambda}_o^{1/2}$ is *not necessarily Hermitian*, although it does have real eigenvalues. Thus, we can apply the perturbation bound of Fact 4 to $\mathbf{\Lambda}_o$ and $\mathbf{\Lambda}_o^{1/2} \mathbf{V}_o^T \bar{\mathbf{S}} \bar{\mathbf{S}}^T \mathbf{V}_o \mathbf{\Lambda}_o^{1/2}$ to claim for all $i \in [n]$, and some constant C,

$$|\lambda_i(\mathbf{\Lambda}_o^{1/2} \mathbf{V}_o^T \bar{\mathbf{S}} \bar{\mathbf{S}}^T \mathbf{V}_o \mathbf{\Lambda}_o^{1/2}) - \lambda_i(\mathbf{\Lambda}_o)| \leq C \log n \|\mathbf{\Lambda}_o^{1/2} \mathbf{V}_o^T \bar{\mathbf{S}} \bar{\mathbf{S}}^T \mathbf{V}_o \mathbf{\Lambda}_o^{1/2} - \mathbf{\Lambda}_o\|_2.$$

By Lemma 12 applied with error $\frac{\epsilon}{2C \log n}$, with probability at least $1 - \delta$, for any $s \geq \frac{c \log(1/(\epsilon\delta)) \cdot \log^3 n}{\epsilon^3 \sqrt{\delta}}$ (for a large enough constant c) we have $\|\mathbf{\Lambda}_o^{1/2} \mathbf{V}_o^T \bar{\mathbf{S}} \bar{\mathbf{S}}^T \mathbf{V}_o \mathbf{\Lambda}_o^{1/2} - \mathbf{\Lambda}_o\|_2 \leq \frac{\epsilon n}{2C \log n}$. Thus, for all i,

$$\left| \lambda_i(\mathbf{\Lambda}_o^{1/2} \mathbf{V}_o^T \bar{\mathbf{S}} \bar{\mathbf{S}}^T \mathbf{V}_o \mathbf{\Lambda}_o^{1/2}) - \lambda_i(\mathbf{\Lambda}_o) \right| < \frac{\epsilon n}{2}. \tag{9}$$

We note that the conceptual part of the proof is essentially complete: the nonzero eigenvalues of $\frac{n}{s} \cdot \mathbf{A}_{o,S}$ are identical to those of $\mathbf{\Lambda}_o^{1/2} \mathbf{V}_o^T \bar{\mathbf{S}} \bar{\mathbf{S}}^T \mathbf{V}_o \mathbf{\Lambda}_o^{1/2}$, which we have shown well approximate those of $\mathbf{\Lambda}_o$ and in turn $\mathbf{A}_o$. i.e., the non-zero eigenvalues of $\frac{n}{s} \cdot \mathbf{A}_{o,S}$ approximate all outlying

eigenvalues of $\mathbf{A}$. It remains to carefully argue how these approximations should be "lined up" given the presence of zero eigenvalues in the spectrum of these matrices. We also must account for the impact of the interior eigenvalues in $\mathbf{A}_{m,S}$, which is limited by the spectral norm bound of Lemma 13. The rest of the argument is completed in Theorem 1 of [10]. ◀

▶ Remark. The proof of Lemma 12 and consequently, Theorem 1 can be modified to give better bounds for the case when the eigenvalues of $\mathbf{A}_o$ lie in a bounded range – between $\epsilon^a\sqrt{\delta}n$ and $\epsilon^b n$ where $0 \leq b \leq a \leq 1$. See Theorem 9 in Appendix C of [10] for details. For example, if all the top eigenvalues are equal, one can show that $s = \tilde{O}\left(\frac{\log^2 n}{\epsilon^2}\right)$ suffices to give $\pm\epsilon n$ error, nearly matching the lower bound of [5]. This indicates that improving Theorem 1 in general requires tackling the case when the outlying eigenvalues in $\mathbf{\Lambda}_o$ have a wide range.

4 Conclusion

We present efficient algorithms for estimating all eigenvalues of a symmetric matrix with bounded entries up to additive error ϵn, by reading just a $\text{poly}(\log n, 1/\epsilon) \times \text{poly}(\log n, 1/\epsilon)$ random principal submatrix. We give improved error bounds of $\epsilon\sqrt{\text{nnz}(\mathbf{A})}$ and $\epsilon\|\mathbf{A}\|_F$ when the rows/columns are sampled with probabilities proportional to their sparsities or squared ℓ_2 norms, respectively (see Section 4 and Appendix E of [10]). We also perform numerical simulations which demonstrate the effectiveness of our algorithms in practice (see Section 5 of [10]).

Our work leaves several open questions. In particular, it is open if our query complexity for $\pm\epsilon n$ approximation can be improved, possibly to $\tilde{O}(\log^c n/\epsilon^4)$ total entries using principal submatrix queries or $\tilde{O}(\log^c /\epsilon^2)$ entries using general queries. The later bound is open even when $\mathbf{A}$ is PSD, a setting where we know that sampling a $O(1/\epsilon^2) \times O(1/\epsilon^2)$ principal submatrix (with $O(1/\epsilon^4)$ total entries) does suffice. Additionally, it is open if we can achieve sample complexity independent of n, by removing all $\log n$ factors, as have been done for the easier problem of testing positive semidefiniteness [5]. See Section 1.4 for more details. Finally, it would be interesting to identify additional assumptions on $\mathbf{A}$ or on the sampling model where stronger approximation guarantees (e.g., relative error) can be achieved in sublinear time.

References

1 Dimitris Achlioptas and Frank McSherry. Fast computation of low-rank matrix approximations. *Journal of the ACM (JACM)*, 54(2):9–es, 2007.

2 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 32nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2021.

3 Alexandr Andoni and Huy L Nguyên. Eigenvalues of a matrix in the streaming model. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2013.

4 Arturs Backurs, Piotr Indyk, Cameron Musco, and Tal Wagner. Faster kernel matrix algebra via density estimation. *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

5 Ainesh Bakshi, Nadiia Chepurko, and Rajesh Jayaram. Testing positive semi-definiteness via random submatrices. *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2020.

6 Maria-Florina Balcan, Yi Li, David P Woodruff, and Hongyang Zhang. Testing matrix rank, optimally. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2019.

7 Itai Benjamini, Oded Schramm, and Asaf Shapira. Every minor-closed property of sparse graphs is testable. *Advances in Mathematics*, 223(6):2200–2218, 2010.

8 Serge Bernstein. Sur l'extension du théorème limite du calcul des probabilités aux sommes de quantités dépendantes. *Mathematische Annalen*, 97(1):1–59, 1927.

9 Rajendra Bhatia. *Matrix analysis*. Springer Science & Business Media, 2013.

10 Rajarshi Bhattacharjee, Gregory Dexter, Petros Drineas, Cameron Musco, and Archan Ray. Sublinear time eigenvalue approximation via random sampling. *arXiv preprint*, 2021. `arXiv:2109.07647`.

11 Vladimir Braverman, Aditya Krishnan, and Christopher Musco. Linear and sublinear time spectral density estimation. *Proceedings of the 54th Annual ACM Symposium on Theory of Computing (STOC)*, 2022.

12 Nadiia Chepurko, Kenneth L Clarkson, Lior Horesh, Honghao Lin, and David P Woodruff. Quantum-inspired algorithms from randomized numerical linear algebra. *arXiv*, 2020. `arXiv:2011.04125`.

13 David Cohen-Steiner, Weihao Kong, Christian Sohler, and Gregory Valiant. Approximating the spectrum of a graph. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2018.

14 James Demmel, Ioana Dumitriu, Olga Holtz, and Robert Kleinberg. Fast matrix multiplication is stable. *Numerische Mathematik*, 2007.

15 Kun Dong, Austin R Benson, and David Bindel. Network density of states. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2019.

16 Petros Drineas and Ravi Kannan. Fast monte-carlo algorithms for approximate matrix multiplication. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2001.

17 Talya Eden and Will Rosenbaum. On sampling edges almost uniformly. *SIAM Symposium on Simplicty in Algorithms (SOSA)*, 2018.

18 Noureddine El Karoui. The spectrum of kernel random matrices. *The Annals of Statistics*, 38(1):1–50, 2010.

19 Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.

20 Semyon Aranovich Gershgorin. Uber die abgrenzung der eigenwerte einer matrix. *Izvestiya Rossiyskoy akademii nauk. Seriya matematicheskaya*, 6:749–754, 1931.

21 Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

22 Alex Gittens and Joel A Tropp. Tail bounds for all eigenvalues of a sum of random matrices. *arXiv*, 2011. `arXiv:1104.4513`.

23 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC)*, 1997.

24 Oded Goldreich and Dana Ron. Approximating average parameters of graphs. *Random Structures & Algorithms*, 32(4):473–493, 2008.

25 Leslie Greengard and John Strain. The fast gauss transform. *SIAM Journal on Scientific and Statistical Computing*, 1991.

26 Ming Gu and Stanley C Eisenstat. A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem. *SIAM Journal on Matrix Analysis and Applications*, 1995.

27 Moritz Hardt and Eric Price. The noisy power method: A meta algorithm with applications. *Advances in Neural Information Processing Systems 27 (NIPS)*, 2014.

28 Jonas Helsen, Francesco Battistel, and Barbara M Terhal. Spectral quantum tomography. *Quantum Information*, 2019.

29 Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, USA, 2nd edition, 2012.

30 Robert Krauthgamer and Ori Sasson. Property testing of data dimensionality. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2003.

31 Ruipeng Li, Yuanzhe Xi, Lucas Erlandson, and Yousef Saad. The eigenvalues slicing library (EVSL): Algorithms, implementation, and software. *SIAM Journal on Scientific Computing*, 2019.

32 Yi Li, Huy L Nguyễn, and David P Woodruff. On sketching matrix norms and the top singular vector. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2014.

33 Yi Li, Zhengyu Wang, and David P Woodruff. Improved testing of low rank matrices. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2014.

34 Yi Li and David P Woodruff. On approximating functions of the singular values in a stream. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*, 2016.

35 Lin Lin, Yousef Saad, and Chao Yang. Approximating spectral densities of large matrices. *SIAM Review*, 2016.

36 Milena Mihail and Christos Papadimitriou. On the eigenvalue power law. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 254–262. Springer, 2002.

37 Deanna Needell, William Swartworth, and David P Woodruff. Testing positive semidefiniteness using linear measurements. In *Proceedings of the 63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2022.

38 Mark Rudelson and Roman Vershynin. Sampling from large matrices: An approach through geometric functional analysis. *Journal of the ACM (JACM)*, 2007.

39 Yousef Saad. *Numerical methods for large eigenvalue problems: revised edition.* SIAM, 2011.

40 Levent Sagun, Leon Bottou, and Yann LeCun. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv*, 2016. `arXiv:1611.07476`.

41 RN Silver and H Röder. Densities of states of mega-dimensional Hamiltonian matrices. *International Journal of Modern Physics C*, 1994.

42 G. W. Stewart and Ji guang Sun. *Matrix Perturbation Theory.* Academic Press, 1990.

43 Ewin Tang. Quantum-inspired classical algorithms for principal component analysis and supervised clustering. *arXiv*, 2018. `arXiv:1811.00414`.

44 Joel A Tropp. Norms of random submatrices and sparse approximation. *Comptes Rendus Mathematique*, 2008.

45 Joel A. Tropp. The random paving property for uniformly bounded matrices. *Studia Mathematica*, 185:67–82, 2008.

46 Joel A Tropp. An introduction to matrix concentration inequalities. *arXiv*, 2015. `arXiv:1501.01571`.

47 Madeleine Udell and Alex Townsend. Why are big data matrices approximately low rank? *SIAM Journal on Mathematics of Data Science*, 1(1):144–160, 2019.

48 Lin-Wang Wang. Calculating the density of states and optical-absorption spectra of large quantum systems by the plane-wave moments method. *Physical Review B*, 1994.

49 Alexander Weiße, Gerhard Wellein, Andreas Alvermann, and Holger Fehske. The kernel polynomial method. *Reviews of Modern Physics*, 2006.

50 Hermann Weyl. The asymptotic distribution law of the eigenvalues of linear partial differential equations (with an application to the theory of cavity radiation). *Mathematical Annals*, 1912.

51 Zhewei Yao, Amir Gholami, Qi Lei, Kurt Keutzer, and Michael W Mahoney. Hessian-based analysis of large batch training and robustness to adversaries. *arXiv*, 2018. `arXiv:1802.08241`.

Streaming k-Edit Approximate Pattern Matching via String Decomposition

Sudatta Bhattacharya
Computer Science Institute of Charles University, Prague, Czech Republic

Michal Koucký
Computer Science Institute of Charles University, Prague, Czech Republic

Abstract

In this paper we give an algorithm for streaming k-edit approximate pattern matching which uses space $\widetilde{O}(k^2)$ and time $\widetilde{O}(k^2)$ per arriving symbol. This improves substantially on the recent algorithm of Kociumaka, Porat and Starikovskaya [22] which uses space $\widetilde{O}(k^5)$ and time $\widetilde{O}(k^8)$ per arriving symbol. In the k-edit approximate pattern matching problem we get a pattern P and text T and we want to identify all substrings of the text T that are at edit distance at most k from P. In the streaming version of this problem both the pattern and the text arrive in a streaming fashion symbol by symbol and after each symbol of the text we need to report whether there is a current suffix of the text with edit distance at most k from P. We measure the total space needed by the algorithm and time needed per arriving symbol.

2012 ACM Subject Classification Theory of computation → Pattern matching; Theory of computation → Sketching and sampling

Keywords and phrases Approximate pattern matching, edit distance, streaming algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.22

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `http://arxiv.org/abs/2305.00615`

Funding *Sudatta Bhattacharya*: Partially supported by the Grant Agency of the Czech Republic under the grant agreement no. 19-27871X.
Michal Koucký: Partially supported by the Grant Agency of the Czech Republic under the grant agreement no. 19-27871X.

Acknowledgements We thank Tomasz Kociumaka for pointing to us references for Corollary 3. We thank anonymous reviewers for helpful comments.

1 Introduction

Pattern matching is a classical problem of finding occurrences of a given pattern P in text T. It can be solved in time linear in the size of the pattern and text [21, 4, 20]. The classical algorithms use space that is proportional to the pattern size. In a surprising work [25], Porat and Porat were the first to design a pattern matching algorithm that uses less space. They designed an *on-line* algorithm that pre-processes the pattern P into a small data structure, and then it receives the text symbol by symbol. After receiving each symbol of the text, the algorithm is able to report whether the pattern matches the current suffix of the text. The algorithm uses poly-logarithmic amount of memory for storing the data structure and processing the text. This represents a considerable achievement in the design of pattern matching algorithms.

Porat and Porat also gave a small-space online algorithm that solves approximate pattern matching up-to Hamming distance k, *k-mismatch approximate pattern matching*. In this problem we are given the pattern P and a parameter k, and we should find all substrings of the

text T that are at Hamming distance at most k from P. Their algorithm uses $\widetilde{O}(k^3)$ space, and requires $\widetilde{O}(k^2)$ time per arriving symbol of the text. Subsequently this was improved to space $\widetilde{O}(k)$ and time $\widetilde{O}(\sqrt{k})$ [10]. There has been a series of works [5, 9, 10, 19, 18, 15, 27, 26, 17] on online and streaming pattern matching, and the line of work culminated in the work of Clifford, Kociumaka and Porat [11] who gave a fully *streaming* algorithm with similar parameters as [10].

In the streaming setting, also the pattern arrives symbol by symbol and we do not have the space to store all of it at once. An important feature of the algorithm of Clifford, Kociumaka and Porat is that their algorithm not only reports the k-mismatch occurrences of the pattern but for each k-mismatch occurrence of P it can also output the full information about the difference between P and the current suffix of the text, so called *mismatch information*.

Beside approximate pattern matching with respect to Hamming distance, researchers also consider approximate pattern matching with respect to other similarity measures such as edit distance. Edit distance $\mathrm{ED}(x, y)$ of two strings x and y is the minimum number of insertions, deletions and substitutions needed to transform x into y. The *k-edit approximate pattern matching problem* is a variant of the approximate pattern matching where we should find all substrings of T that are at edit distance at most k from P. Since there could be quadratically many such substrings, we usually only require to report for each position in T whether there is a substring of T ending at that position that has edit distance at most k from P. In the streaming version of the problem we want to output the minimal distance of P to a current suffix of the text after receiving each symbol of T. Again we assume that the text as well as the pattern arrive symbol by symbol, and we are interested in how much space the algorithm uses, and how much time it takes to process each symbol.

Starikovskaya [27] proposed a streaming algorithm for the k-edit pattern matching problem, which uses $\widetilde{O}(k^8\sqrt{m})$ space and takes $\widetilde{O}(k^2\sqrt{m} + k^{13})$ time per arriving symbol. Here, we denote $m = |P|$ and $n = |T|$. Recently, using a very different technique Kociumaka, Porat and Starikovskaya [22] constructed a streaming algorithm, which uses $\widetilde{O}(k^5)$ space and $\widetilde{O}(k^8)$ amortized time per arriving symbol of the text.

In this work we substantially improve on the result of Kociumaka, Porat and Starikovskaya. We give a streaming algorithm for k-edit approximate pattern matching that uses $\widetilde{O}(k^2)$ space and $\widetilde{O}(k^2)$ time per arriving symbol.

▶ **Theorem 1.** *Given integer $k \geq 0$, there exists a randomized streaming algorithm for the k-edit approximate pattern matching problem that uses $\widetilde{O}(k^2)$ bits of space and takes $\widetilde{O}(k^2)$ time per arriving symbol of the text.*

We speculate that some amortization techniques could bring the time complexity of our k-edit approximate pattern matching algorithm further down. However, it seems unlikely to achieve complexity below $\widetilde{O}(k)$ per arriving symbol as one could then solve the plain edit distance problem in sub-quadratic time contradicting the Strong Exponential Time Hypothesis (SETH) [2]. It is an interesting open question to achieve smaller space complexity than $\widetilde{O}(k^2)$. Currently, all known sketching techniques for edit distance that people use for k-edit approximate pattern matching give sketches of size $\Omega(k^2)$.

The technique of Kociumaka, Porat and Starikovskaya [22] for edit distance pattern matching to large extent emulates the inner working of Hamming approximate pattern matching algorithms. To that effect Kociumaka, Porat and Starikovskaya had to design a *rolling* sketch for edit distance where multiple sketches can be "homomorphically" combined into one. This requires sophisticated machinery. Here we use a somewhat different approach. We use a recent locally consistent decomposition of strings which preserves edit distance of Bhattacharya and Koucký [3]. The decomposition in essence translates edit distance

to Hamming distance. Hence, we apply the k-mismatch approximate pattern matching algorithm of Clifford, Kociumaka and Porat [11] on the stream of symbols coming from the decomposition as a black box. Bhattacharya and Koucký [3] also constructed a rolling sketch with limited update abilities, namely adding and deleting a symbol. We do not use that sketch here.

1.1 Related work

Landau and Vishkin [23] gave the first algorithm for the *k-mismatch approximate pattern matching* problem which runs in time $O(k(m \log m + n))$ and takes $O(k(m+n))$ amount of space. This was then improved to $O(m \log m + kn)$ time and $O(m)$ space by Galil and Giancarlo [13]. Later, Amir, Lewenstein and Porat [1] proposed two algorithms running in time $O(n\sqrt{k \log k})$ and $\widetilde{O}(n+k^3(n/m))$. The latter was improved by Clifford, Fontaine, Porat, Sach and Starikovskaya [10] who gave an $\widetilde{O}(n+k^2(n/m))$ time algorithm. Charalampopoulos, Kociumaka and Wellnitz, in their FOCS'20 paper [7], also proposed an $\widetilde{O}(n + k^2(n/m))$ time algorithm with slightly better *polylog* factors. An $\widetilde{O}(n + kn/\sqrt{m})$ time algorithm was given by Gawrychowski and Uznański [16], which showed a nice tradeoff between the $O(n\sqrt{k \log k})$ and $\widetilde{O}(n + k^2(n/m))$ running times. Not only that, they also showed that their algorithm is essentially optimal upto *polylog* factors, by proving a matching conditional lower bound. The *polylog* factors in the running time were then improved further by a randomized algorithm by Chan, Golan, Kociumaka, Kopelowitz and Porat [6], with running time $O(n + kn(\sqrt{\log m/m}))$. This problem is thus quite well studied.

For the edit distance counterpart of the problem however, there is still a significant gap between the best upper bound and the known conditional lower bound. Landau and Vishkin [24] proposed an $O(nk)$ time algorithm for the problem. This algorithm is still the state of the art for larger values of k. Cole and Hariharan [12] gave an algorithm running in time $O(n + m + k^4(n/m))$(this runs faster if $m \geq k^3$). In their unified approach paper [7], Charalampopoulos, Kociumaka and Wellnitz also proposed an algorithm running in time $O(n + m + k^4(n/m))$. The same authors in their FOCS'22 paper [8] gave an algorithm running in time $O(n + k^{3.5}\sqrt{\log m \log kn/m})$, finally improving the bound after 20 years. For the lower bound, Backurs and Indyk [2] proved that a truly subquadratic time algorithm for computing edit distance would falsify SETH. This would imply that an algorithm for the k-edit approximate pattern matching which is significantly faster than $O(n + k^2(n/m))$ is highly unlikely.

Online *k-mismatch approximate pattern matching* problem was first solved by Benny Porat and Ely Porat in 2009 [25]. They gave an online algorithm with running time $\widetilde{O}(k^2)$ and space $\widetilde{O}(k^3)$ per arriving symbol of the text. Clifford, Fontaine, Porat, Sach and Starikovskaya in their SODA'16 paper [10], improved it to $\widetilde{O}(k^2)$ space and $O(\sqrt{k} \log k + poly(\log(n)))$ time per arriving symbol of the text. Clifford, Kociumaka and Porat [11] proposed a randomized streaming algorithm which uses $O(k \log (m/k))$ space and $O(\log (m/k)(\sqrt{k \log k} + \log^3 m))$ time per arriving symbol. The space upper bound is optimal up-to logarithmic factors, matching the communication complexity lower bound. All these algorithms use some form of rolling sketch.

In the streaming model, Starikovskaya proposed a randomized algorithm [27] for the k-edit approximate pattern matching problem, which takes $O(k^8\sqrt{m} \log^6 m)$ space and $O((k^2\sqrt{m} + k^{13}) \log^4 m)$ time per arriving symbol. Kociumaka, Porat and Starikovskaya [22] proposed an improved randomized streaming algorithm, which takes $\widetilde{O}(k^5)$ space and $\widetilde{O}(k^8)$ amortized time per arriving symbol of the text.

2 Notations and preliminaries

We use a standard notation. For any string $x = x_1x_2x_2 \dots x_n$ and integers p, q, $x[p]$ denotes x_p, $x[p, q]$ represents substring $x' = x_p \dots x_q$ of x, and $x[p, q) = x[p, q - 1]$. If $q < p$, then $x[p, q]$ is the empty string ε. $x[p, \dots]$ represents $x[p, |x|]$, where $|x|$ is the length of x. "$\cdot$"-operator is used to denote concatenation, e.g $x \cdot y$ is the concatenation of two strings x and y. For strings x and y, $\mathrm{ED}(x, y)$ is the minimum number of modifications (*edit operations*) required to change x into y, where a single modification can be adding a character, deleting a character or substituting a character in x. All logarithms are based-2 unless stated otherwise. For integers $p > q$, $\sum_{i=p}^{q} a_i = 0$ by definition regardless of a_i's.

2.1 Grammars

We will use the following definitions from [3]. They are taken essentially verbatim. Let $\Sigma \subseteq \Gamma$ be two alphabets and $\# \notin \Gamma$. A *grammar* G is a set of *rules* of the type $c \to ab$ or $c \to a^r$, where $c \in (\Gamma \cup \{\#\}) \setminus \Sigma$, $a, b \in \Gamma$ and $r \in \mathbb{N}$. c is the *left hand side* of the rule, and ab or a^r is the *right hand side* of the rule. $\#$ is the starting symbol. The size $|G|$ of the grammar is the number of rules in G. We only consider grammars where each $a \in \Gamma \cup \{\#\}$ appears on the left hand side of at most one rule of G, we call such grammars *deterministic*. The $\mathrm{eval}(G)$ is the string from Σ^* obtained from $\#$ by iterative rewriting of the intermediate results by the rules from G. If the rewriting process never stops or stops with a string not from Σ^*, $\mathrm{eval}(G)$ is undefined. We use $\mathrm{eval}(G_1, G_2, \dots, G_t)$ to denote the concatenation $\mathrm{eval}(G_1) \cdot \mathrm{eval}(G_2) \cdots \mathrm{eval}(G_t)$. Using a depth-first traversal of a deterministic grammar G we can calculate its *evaluation size* $|\mathrm{eval}(G)|$ in time $O(|G|)$. Given a deterministic grammar G and an integer m less or equal to its evaluation size, we can construct in time $O(|G|)$ another grammar G' of size $O(|G|)$ such that $\mathrm{eval}(G') = \mathrm{eval}(G)[m, \dots]$. G' will use some new auxiliary symbols.

We will use the following observation of Ganesh, Kociumaka, Lincoln and Saha [14]:

▶ **Proposition 2** ([14]). *There is an algorithm that on input of two grammars G_x and G_y of size at most m computes the edit distance k of $\mathrm{eval}(G_x)$ and $\mathrm{eval}(G_y)$ in time $O((m + k^2) \cdot \mathrm{poly}(\log(m + n)))$, where $n = |\mathrm{eval}(G_x)| + |\mathrm{eval}(G_y)|$.*

We remark that the above algorithm can be made to output also full information about edit operations that transform $\mathrm{eval}(G_x)$ to $\mathrm{eval}(G_y)$. We will also use the following proposition which can be obtained from Landau-Vishkin algorithm [23] see e.g. a combination of Lemma 6.2 and Theorem 7.13 in [7]:

▶ **Corollary 3.** *For every pair of grammars G_x and G_y representing strings x and y, respectively, and given a parameter k we can find in time $O((m + k^2) \cdot \mathrm{poly}(\log(m + n)))$, where $n = |x| + |y|$ and $m = |G_x| + |G_y|$, the length of a suffix of x with the minimum edit distance to y among all the suffixes of x, provided that the edit distance of the suffix and y is at most k. If the edit distance of all the suffixes of x to y is more than k then the algorithm stops in the given time and reports that no suffix was found.*

3 Decomposition algorithm

Bhattacharya and Koucký [3] give a string decomposition algorithm (*BK-decomposition algorithm*) that splits its input string into blocks, each block represented by a small grammar. With high probability over the choice of randomness of the algorithm, two strings of length

at most n and edit distance at most k are decomposed so that the number of blocks is the same and at most k corresponding pairs of blocks differ. The edit distance between the two strings corresponds to the sum of edit distances of differing pairs of blocks.

More specifically, the BK-decomposition algorithm gets two parameters n and k, $k \leq n$, and an input x. It selects at random pair-wise independent functions $C_1, \ldots, C_L$ and S-wise independent functions $H_0, \ldots, H_L$ from certain hash families, and using those hash functions it decomposes x into blocks, and outputs a grammar for each of the block. We call the sequence of the produced grammars the *BK-decomposition of* x. Here, parameters $L = \lceil \log_{3/2} n \rceil + 3$ and $S = O(k \log^3 n \log^* n)$. As shown in [3], the algorithm satisfies the following property.

▶ **Proposition 4** (Theorem 3.1 [3]). *Let x be a string of length at most n. The BK-decomposition algorithm outputs a sequence of grammars $G_1, \ldots, G_s$ such that for n large enough:*

1. *With probability at least $1 - 2/n$, $x = \mathrm{eval}(G_1, \ldots, G_s)$.*
2. *With probability at least $1 - 2/\sqrt{n}$, for all $i \in \{1, \ldots, s\}$, $|G_i| \leq S$.*

The randomness of the algorithm is over the random choice of functions $C_1, \ldots, C_L$ and $H_0, \ldots, H_L$.

The functions $C_1, \ldots, C_L$ can be described using $O(\log^2 n)$ bits in total and the S-wise independent functions $H_0, \ldots, H_L$ can be described using $O(S \log^2 n)$ bits in total. We also need the following special case of Theorem 3.12 [3].

▶ **Proposition 5** (Theorem 3.12 [3]). *Let $u, x, y \in \Gamma^*$ be strings such that $|ux|, |y| \leq n$ and $\mathrm{ED}(x, y) \leq k$. Let $G_1^x, \ldots, G_s^x$ and $G_1^y, \ldots, G_{s'}^y$ be the sequence of grammars output by the BK-decomposition algorithm on input ux and y respectively, using the same choice of random functions $C_1, \ldots, C_L$ and $H_0, \ldots, H_L$. With probability at least $1 - 1/5$ the following is true: There exist an integer $r \geq 1$, such that*

$$x = \mathrm{eval}(G^x_{s-s'+1})[r, \ldots] \cdot \mathrm{eval}(G^x_{s-s'+2}, \ldots, G^x_s) \quad \& \quad y = \mathrm{eval}(G^y_1, \ldots, G^y_{s'}),$$

and

$$\mathrm{ED}(x, y) = \mathrm{ED}(\mathrm{eval}(G^x_{s-s'+1})[r, \ldots], \mathrm{eval}(G^y_1)) + \sum_{i=2}^{s'} \mathrm{ED}(\mathrm{eval}(G^x_{s-s'+i}), \mathrm{eval}(G^y_i)).$$

The grammars for x can be built incrementally. For a fixed choice of functions C_i, H_i, and a string x we say that grammars $G_1^x, \ldots, G_t^x$ are *definite* in its BK-decomposition $G_1^x, \ldots, G_s^x$ if for any string z and the BK-decomposition $G_1^{xz}, \ldots, G_{s'}^{xz}$ of xz obtained using the same functions C_i, H_i, $G_1^x = G_1^{xz}$, $\ldots$, $G_t^x = G_t^{xz}$. It turns out that all, but $\widetilde{O}(1)$ last grammars in the BK-decomposition of x are always definite. The following claim appears in [3]:

▶ **Proposition 6** (Lemma 4.2 [3]). *Let n and k be given and $R = O(\log n \log^* n)$ be a suitably chosen parameter. Let $x, z \in \Gamma^*$, $|xz| \leq n$. Let $H_0, \ldots, H_L, C_1, \ldots, C_L$ be given. Let $G_1^x, G_2^x, \ldots, G_s^x$ be the output of the BK-decomposition algorithm on input x, and $G_1^{xz}, G_2^{xz}, \ldots, G_{s'}^{xz}$ be the output of the decomposition algorithm on input xz using the given hash functions.*

1. *$G_i^x = G_i^{xz}$ for all $i = 1 \ldots, s - R$.*
2. *$|x| \leq \sum_{i=1}^{\min(s+R, s')} |\mathrm{eval}(G_i^{xz})|$.*

The following claim bounds the resources needed to update BK-decomposition of x when we append a symbol a to it.

▶ **Proposition 7** (Theorem 5.1 [3]). *Let $k \leq n$ be given and $R = O(\log n \log^* n)$ be a suitably chosen parameter. Let functions $C_1, \ldots, C_L$ and $H_0, \ldots, H_L$ be given. Let $a \in \Sigma$ and $x \in \Sigma^*$ be of length at most n, and let $G_1^x, \ldots, G_s^x$ be the grammars output by the BK-decomposition algorithm on input x using functions $C_1, \ldots, C_L, H_0, \ldots, H_L$.* UpdateActiveGrammars($G^x_{s-\min(s,R+1)+1}, \ldots, G^x_s, a$) *outputs a sequence of grammars $G'_1, \ldots, G'_{t'}$ such that $G^x_1, \ldots, G^x_{s-\min(s,R+1)}, G'_1, \ldots, G'_{t'}$ is the sequence that would be output by the BK-decomposition algorithm on $x \cdot a$ using the same functions $C_1, \ldots, C_L, H_0, \ldots, H_L$. The update algorithm runs in time $\widetilde{O}(k)$ and outputs $t' \leq 4RL$ grammars.*

3.1 Encoding a grammar

Let S and $M = O(S \log n) = O(k \log^4 n \log^* n)$ be parameters determined by the BK-decomposition algorithm. [3] shows that each grammar of size at most S can be encoded as a string of size M over some polynomial-size alphabet $\{1, \ldots, 2\alpha\}$, where the integer α can be chosen so that $2M/\alpha \leq 1/n$. The encoding Enc satisfies that if two grammars differ, their encodings differ in every coordinate. The encoding is randomized, and one needs $O(\log n)$ random bits to select the encoding function. The encoding can be calculated in time linear in M, and given $\mathrm{Enc}(G)$ we can decode G in time $O(M)$. The encoding satisfies:

▶ **Proposition 8.** *Let G, G' be two grammars of size at most S output by BK-decomposition algorithm. Let encoding* Enc *be chosen at random.*

1. $\mathrm{Enc}(G) \in \{1, \ldots, 2\alpha\}^M$.
2. *If $G = G'$ then* $\mathrm{Enc}(G) = \mathrm{Enc}(G')$.
3. *If $G \neq G'$ then with probability at least $1 - (2M/\alpha)$,* $\mathrm{Ham}(\mathrm{Enc}(G), \mathrm{Enc}(G')) = M$, *that is they differ in every symbol.*

3.2 k-mismatch approximate pattern matching

Clifford, Kociumaka and Porat [11] design a streaming algorithm for *k-mismatch approximate pattern matching* with the following properties. The algorithm first reads a pattern P symbol by symbol, and then it reads a text T symbol by symbol. Upon reading each symbol of the text it reports whether the word formed by the last received $|P|$ symbols of the text are within Hamming distance at most k from the pattern. If they are within Hamming distance at most k we can request the algorithm to report the mismatch information between the current suffix of the text and the pattern. The parameters k and n are given to the algorithm at the beginning, where n is an upper bound on the total length of the pattern and the text. By *mismatch information* between two strings x and y of the same length we understand $\mathrm{MIS}(x, y) = \{(i, x[i], y[i]);\ i \in \{1, \ldots, |x|\}$ *and* $x[i] \neq y[i]\}$. So the Hamming distance of x and y is $\mathrm{Ham}(x, y) = |\mathrm{MIS}(x, y)|$. Clifford, Kociumaka and Porat [11] give the following main theorem.

▶ **Proposition 9** ([11]). *There exists a streaming k-*mismatch approximate pattern matching *algorithm which uses $O(k \log n \log(n/k))$ bits of space and takes $O((\sqrt{k \log k} + \log^3 n) \log(n/k))$ time per arriving symbol. The algorithm is randomised and its answers are correct with high probability, that is it errs with probability inverse polynomial in n. For each reported occurrence, the mismatch information can be reported on demand in $O(k)$ time.*

4 Algorithm overview

Now we provide the high-level view of how we proceed. We will take the pattern P and apply on it the BK-decomposition algorithm. That will give us grammars $G_1^P, G_2^P, \ldots, G_r^P$ encoding the pattern. This has to be done incrementally as the symbols of P arrive. Then we will incrementally apply the BK-decomposition algorithm on the text T.

We will not store all the grammars in memory, instead we will use the *K-mismatch approximate pattern matching algorithm* of Clifford, Kociumaka and Porat [11] (*CKP-match algorithm*) on the grammars. Here $K = k \cdot M$, where M is the encoding size of each grammar. For a suitable parameter $R = \widetilde{O}(1)$, we will feed the grammars $G_1^P, \ldots, G_{r-R}^P$ to the CKP-match algorithm as a pattern. In particular, we will encode each grammar by the encoding function Enc from Section 3.1, and we will feed the encoding into the CKP-match algorithm symbol by symbol.

Then as the symbols of the text T will arrive, we will incrementally build the grammars for T while maintaining only a small set of *active* grammars. Grammars that become *definite* will be fed into the CKP-match algorithm as its input text. (Again each one of the grammars encoded by Enc.) The CKP-match algorithm will report *K-mismatch* occurrences of our pattern in the text. Each *K-mismatch* occurrence corresponds to a match of the pattern grammars to the text grammars, with up-to k differing pairs of grammars. We will recover the differing pairs of grammars and calculate their overall edit distance. We will combine this edit distance with the edit distance of the last R grammars of the pattern from the last R grammars of the text. (The last R grammars of the text contain the active grammars which were not fed into the CKP-match algorithm, yet.) If the total edit distance of the match does not exceed the threshold k, we report it as an k-edit occurrence of P in T. If required we can also output the edit operations that transform the pattern into a suffix of T. (Among the current suffixes of T we pick the one which gives the smallest edit distance from P.)

The success probability of our scheme in reporting a particular occurrence of P in T is some constant $\geq 1/2$. Thus, we run the processes in parallel $O(\log n)$ times with independently chosen randomness to achieve small error-probability.

We describe our algorithm in more details next.

5 Description of the algorithm

Now we describe one run of our algorithm. The algorithm receives parameters n and k, based on them it sets parameters $L = O(\log n)$, $R = O(\log n \log^* n)$, $S = O(k \log^3 n \log^* n)$, $M = O(k \log^4 n \log^* n)$, $K = k \cdot M = O(k^2 \log^4 n \log^* n)$. Then it chooses at random pairwise independent functions $C_1, \ldots, C_L$ and S-wise independent functions $H_0, \ldots, H_L$ needed by the BK-decomposition algorithm. It also selects the required randomness for the encoding function Enc. It initializes the CKP-match algorithm for *K-mismatch* approximate pattern matching on strings of length at most $n \cdot M$.

There are two phases of the algorithm. In the first phase the algorithm receives a pattern P symbol by symbol and incrementally builds a sequence of grammars $G_1^P, \ldots, G_r^P$ representing the pattern P. All but the last R grammars are encoded using Enc and sent to our instance of CKP-match algorithm as its pattern (symbol by symbol of each encoding). In the second phase our algorithm receives an input text T symbol by symbol. It will incrementally build a sequences of grammars $G_1^T, G_2^T, \ldots$ representing the received text. Whenever one of the grammars becomes *definite* it is encoded by Enc and sent to our instance of CKP-match algorithm as the next part of its input text (symbol by symbol).

In the first phase, our algorithm uses the procedure given by Proposition 7 to construct the grammars $G_1^P, \ldots, G_r^P$ incrementally by adding symbols of P. The algorithm maintains a buffer of $2R$ *active* grammars which are updated by the addition of each symbol. Whenever the number of active grammars exceeds $2R$ we encode the *oldest* (left-most) grammars that are definite and pass them to our instance of CKP-match algorithm as the continuation of its pattern. The precise details of updating the grammars of the pattern are similar to that of updating them for text which we will elaborate on more. After the input pattern ends, we keep only R grammars $G_{r-R+1}^P, \ldots, G_r^P$, and we send all the other grammars to the CKP-match algorithm. Then we announce to the CKP-match algorithm the end of its input pattern. So the CKP-match algorithm received as its pattern encoding of grammars $G_1^P, \ldots, G_{r-R}^P$ in this order. (In the case we end up with fewer than $R+1$ grammars representing P ($r \leq R$), we apply a *naïve* pattern matching algorithm without need for the CKP-match algorithm. We leave this simple case as an exercise to the reader.) For the rest of this description we assume that $r > R$.

In the second phase, the algorithm will receive the input text T symbol by symbol. It will incrementally build a sequence of grammars representing the text using the algorithm from Proposition 7. We will keep at most R *active* grammars $G_1^a, \ldots, G_t^a$ on which the algorithm from Proposition 7 will be applied. The active grammars represent a current suffix of T. The prefix of T up-to that suffix is represented by grammars $G_1^T, \ldots, G_s^T$ which are definite. Out of those definite grammars we will explicitly store only the last R in a buffer, the other grammars will not be stored explicitly. (They will be used to calculate the current edit distance and to run the update algorithm from Proposition 7.) The encoding of all the definite grammars will be fed into the CKP-match algorithm as its input text whenever we detect that a grammar is definite.

As the algorithm proceeds over the text it calculates a sequence of integers $m_1, m_2, \ldots, m_s$, where the algorithm stores only the last R of them in a buffer. Each value m_i is the minimal edit distance of $\mathrm{eval}(G_1^P, \ldots, G_{r-R}^P)$ (a prefix of the pattern) to any suffix of $\mathrm{eval}(G_1^T, \ldots, G_i^T)$ (a suffix of a prefix of the text) if the edit distance is less than k. m_i is considered infinite otherwise. (Values $m_1, \ldots, m_{r-R-1}$ are all considered to be infinite.) The value m_i will be calculated after G_i^T becomes definite and we send the grammar to our CKP-match algorithm. (The CKP-match algorithm will facilitate its calculation.) Values m_i will be used to calculate the edit distance of the current suffix of the input text received by the algorithm. See Fig. 1 for an illustration.

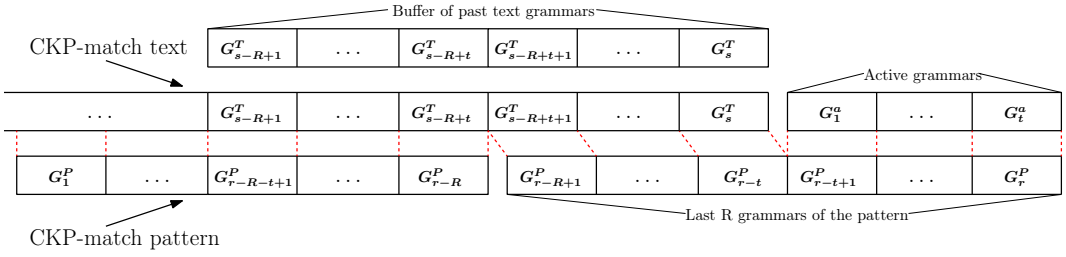


Figure 1 The alignment of text and pattern grammars after arrival of some text symbol. The pattern P is represented by grammars $G_1^P, \ldots, G_r^P$. Grammars $G_1^P, \ldots, G_{r-R}^P$ are encoded by Enc and sent to the CKP-match algorithm as its pattern. The current text T is represented by the sequence of grammars $G_1^T, \ldots, G_s^T, G_1^a, \ldots, G_t^a$. Grammars $G_1^T, \ldots, G_s^T$ are encoded and committed to the CKP-match algorithm as its text. Grammars $G_1^a, \ldots, G_t^a$ are active grammars of the text, and might change as more symbols are added to the text.

We are ready to describe the basic procedures performed by the algorithm.

Symbol arrival. Upon receiving the next symbol a of the input text, our algorithm invokes the algorithm from Proposition 7 on the $R+1$ grammars $G^T_{s-R+t}, \ldots, G^T_s, G^a_1, \ldots, G^a_t$ to append the symbol a. From the algorithm we receive back grammars $G^T_{s-R+t}, \ldots, G^T_s, G'^a_1, \ldots, G'^a_{t'}$, where $t' < 4RL$. (Here, $\mathrm{eval}(G'^a_1, \ldots, G'^a_{t'}) = \mathrm{eval}(G^a_1, \ldots, G^a_t) \cdot a$. The grammars $G^T_{s-R+t}, \ldots, G^T_s$ received from the algorithm are discarded as they are definite and should not change. The update algorithm needs them to have the proper context for compression.) If $t' > R$ then grammars $G'^a_1, \ldots, G'^a_{t'-R}$ become definite and we will *commit* each of them to the CKP-match algorithm as explained further. We will commit them in order $G'^a_1, \ldots, G'^a_{t'-R}$. The remaining grammars $G'^a_{t'-R+1}, \ldots, G'^a_{t'}$ are relabelled as $G^a_1, \ldots, G^a_t$ and become the active grammars for the addition of the next symbol.

At this point our algorithm can output the minimal possible edit distance of the pattern to any suffix of the text received up-to this point. We explain below how such query is calculated.

Committing a grammar. When a grammar G becomes definite the algorithm commits the grammar as follows. Thus far, grammars $G_1, \ldots, G_s$ were committed and the sequence of values $m_1, \ldots, m_s$ was calculated. We set $G_{s+1} = G$, calculate encoding $\mathrm{Enc}(G_{s+1})$ and send the encoding symbol by symbol to our CKP-match algorithm. At this point we can calculate m_{s+1} using the mismatch information provided by our CKP-match algorithm. If $s + 1 < r - R$ then we set m_{s+1} to ∞ otherwise we continue as follows to calculate m_{s+1}.

We query our CKP-match algorithm for the Hamming distance between encoding of $G^P_1, \ldots, G^P_{r-R}$ (the pattern to the CKP-match algorithm) and the encoding of $G^T_{s-r+R+2}$, $G^T_{s-r+R+3}, \ldots, G^T_{s+1}$ (the current suffix of the text of the CKP-match algorithm). If the Hamming distance is less than $K = k \cdot M$, then we let the CKP-match algorithm to recover the mismatch information. By the design of the encoding function, if two grammars differ then their encodings differ in all M positions (unless the encoding function Enc fails which happens only with negligible probability.) Hence, the mismatch information consists of encoding of up-to k pairs of grammars, with their indexes relative to the pattern. Thus, from the mismatch information we recover pairs of grammars $(G_1, G'_1), \ldots, (G_{k'}, G'_{k'})$, for some $k' \leq k$ where G_i come from the text and G'_i come from the pattern.

If (G_1, G'_1) is not the very first grammar pair $(G^T_{s-r+R+2}, G^P_1)$ (which we recognize by their index in the mismatch information) then we compute the edit distance for each pair of strings $\mathrm{eval}(G_i)$ and $\mathrm{eval}(G'_i)$, $i = 1, \ldots, k'$. We set m_{s+1} to be the sum of those distances.

If (G_1, G'_1) is the pair (G^T_{s-R+2}, G^P_1) then we apply the algorithm from Corollary 3 to calculate the minimal edit distance between any suffix of $\mathrm{eval}(G_1)$ and the string $\mathrm{eval}(G'_1)$. For $i = 2, \ldots, k'$, we compute the edit distance of $\mathrm{eval}(G_i)$ and $\mathrm{eval}(G'_i)$. We set m_{s+1} to be the sum of the k' calculated values.

However, if the CKP-match algorithm declares that the Hamming distance of its pattern to its current suffix is more than K, we set $m_{s+1} = \infty$.

Finally, we discard G_{s-r+R} from the buffer of the last R committed grammars, and we discard m_{s-R+2} from the buffer of values m_i. We set s to be $s + 1$. This finishes the process of committing a single grammar G, and a next grammar might be committed.

Pattern edit distance query. After we process the arrival of a new symbol, update the active grammars as described above and commit grammars as necessary, the algorithm is ready to answer the edit distance query on the current suffix of the text T and the pattern P. At this point grammars $G^T_1, \ldots, G^T_s$ were already committed to the CKP-match algorithm. There are current active grammars $G^a_1, \ldots, G^a_t$ which were not committed to the CKP-match

algorithm, and there are R grammars $G^P_{r-R+1}, \ldots, G^P_r$ of the input pattern that were not committed to the CKP-match algorithm as part of its pattern. To answer the edit distance query we will compare the edit distance of those last R grammars of pattern P with the last grammars of the text, and we will combine this with a certain value m_i, namely m_{s-R+t}.

Let $d = R-t$. If $d > 0$, for $i = 1, \ldots, d$ compute the edit distance of each pair $\mathrm{eval}(G^T_{s-d+i})$ and $\mathrm{eval}(G^P_{r-R+i})$. (Each grammar G^T_{s-d+i} is available in the buffer of the last R committed grammars.) For $i = d+1, \ldots, R$, compute the edit distance of each pair $\mathrm{eval}(G^a_{i-d})$ and $\mathrm{eval}(G^P_{r-R+i})$. Sum those R values together with m_{s-d}. If the sum is less than k output it, otherwise output ∞.

Since we are running $O(\log n)$ independent copies of our algorithm, each of the copies produces an estimate on the edit distance and we output the smallest estimate. That is the correct value with high probability.

6 Correctness of the algorithm

In this section we argue that the algorithm produces a correct output. First we analyze the probability of certain bad events happening when the algorithm fails and then we argue the correctness of the output assuming none of the bad events happens. There are several sources of failure in our algorithm.

1. The BK-decomposition algorithm might produce a decomposition of either the pattern or some suffix of the text with a grammar that is too big or with grammars that do not represent expected strings. (A failure of Proposition 4.)
2. The BK-decomposition algorithm produces a correct decomposition of the pattern and all suffixes of the text but grammars of some suffix of the text T and the pattern P do not align well. (A failure of Proposition 5.)
3. The encoding function Enc fails for some pair of grammars produced by the BK-decomposition algorithm that the CKP-match algorithm is supposed to compare. (A failure of Proposition 8.)
4. BK-decomposition algorithm does not fail but the CKP-match algorithm fails to identify a *K-mismatch* occurrence of its pattern or fails to produce correct mismatch information. (A failure of Proposition 9.)

The failure probability of events 1), 3) and 4) will be each bounded by inverse polynomial in n, where n is the parameter sent to those algorithms as an upper bound on the length of the processed strings. Thus, if we expect our algorithm to process a text and a pattern of size at most N, we can set the parameter n for the BK-decomposition algorithm to be N^4 and for the CKP-algorithm to be $N^4 \cdot M = \widetilde{O}(N^5)$, where M is calculated from $n = N^4$ and k of the BK-decomposition algorithm. (Parameter k for the BK-decomposition algorithm is set to k, and for the CKP-algorithm to $K = k \cdot M = \widetilde{O}(k^2)$.) We will run $2 \log N$ independent copies of our algorithm on the same text and pattern. Next we calculate the probability of failure in case 1), 3) and 4) in a particular copy of the algorithm.

Event 1. There is one pattern P of length at most N, the probability of either of the two conditions in Proposition 4 failing on P is at most $4/\sqrt{n} = 4/N^2$. The probability of failure of Proposition 4 on any the at most N prefixes of the text T is at most $N \cdot 4/\sqrt{n} = 4/N$. Thus the probability of the bad event 1) happening is at most $4/N + 4/N^2$.

Event 3. There are at most N grammars of the pattern encoded by Enc and there are at most N grammars of the text encoded by Enc and committed. Thus there are at most N^2 pairs of grammars on which Proposition 8 could fail by encoding two distinct grammars

by strings of Hamming distance less than M (failure in the third part of Proposition 8). Given our setting of parameters, the probability of the bad event 3) happening is at most $N^2/n = 1/N^2$.

Event 4. The probability that the CKP-match algorithm fails during its execution is at most $1/n = 1/N^4$.

Thus, the probability of a failure of 1), 3) or 4) is at most $5/N$, for N large enough. We run $2\log N$ copies of the algorithm so the probability that any of the copies fails because of events 1), 3), or 4) is at most $10\log N/N$.

If none of the events 1), 3) and 4) occurs during the execution of the algorithm then the pattern and the text are correctly decomposed into grammars by the BK-decomposition, the grammars are properly encoded by Enc, and the CKP-match algorithm correctly identifies all the occurrences of the pattern grammars in the committed text grammars, and for each of the occurrences we correctly recover the differing pairs of pattern and text grammars. Assuming this happens, we want to argue that with a high probability our algorithm will correctly identify k-edit occurrences of the pattern P in the text T.

After receiving a prefix of the text $T[1,\ell]$, $\ell \le N$, we want to determine whether some suffix of $T[1,\ell]$ has edit distance at most k from the pattern P. Let a be such that $T[a,\ell]$ has the minimal distance from P. Clearly, if the edit distance between $T[a,\ell]$ and P is at most k then $a \in \{\ell - |P| - k + 1, \dots, \ell - |P| + k + 1\}$. By Proposition 5 applied on $u = T[1, a-1]$, $x = T[a,\ell]$ and $y = P$, each of the $2\log N$ copies of our algorithm has probability at least $4/5$ that the grammars of T are well aligned with grammars of P. Being well aligned means that $T[a,\ell]$ is a suffix of $\mathrm{eval}(G^T_{s-r+t+1}, \dots, G^T_s, G^a_1, \dots, G^a_t)$ and

$$\begin{aligned}\mathrm{ED}(T[a,\ell], P) &= \mathrm{ED}(\mathrm{eval}(G^T_{s-r+t+1})[b,\dots], \mathrm{eval}(G^P_1)) \\ &+ \sum_{i=2}^{r-t} \mathrm{ED}(\mathrm{eval}(G^T_{s-r+t+i}), \mathrm{eval}(G^P_i)) \\ &+ \sum_{i=r-t+1}^{r} \mathrm{ED}(\mathrm{eval}(G^a_{i-r+t}), \mathrm{eval}(G^P_i)),\end{aligned}$$

for appropriate b. Moreover, the minimality of a implies that

$$\begin{aligned}\mathrm{ED}(T[a,\ell], P) &= \min_b \mathrm{ED}(\mathrm{eval}(G^T_{s-r+t+1})[b,\dots], \mathrm{eval}(G^P_1)) \\ &+ \sum_{i=2}^{r-t} \mathrm{ED}(\mathrm{eval}(G^T_{s-r+t+i}), \mathrm{eval}(G^P_i)) \\ &+ \sum_{i=r-t+1}^{r} \mathrm{ED}(\mathrm{eval}(G^a_{i-r+t}), \mathrm{eval}(G^P_i)).\end{aligned}$$

Notice, regardless of whether Proposition 5 fails or not, the right-hand-side of the last equation is always at least $\mathrm{ED}(T[a,\ell], P)$ since it is an upper-bound on the true edit distance of P to some suffix of T. We will argue that each copy of the algorithm outputs the right-hand-side value of that equation if it has value at most k, and ∞ otherwise. Moreover, if at least one of the copies of our algorithm has $T[a,\ell]$ and P well aligned, then the minimum among the values output by the different copies of our algorithm is $\mathrm{ED}(T[a,\ell], P)$.

Since we have $2\log N$ copies of the algorithm, the probability that none of the decompositions aligns $T[a,\ell]$ and P well is at most $(1/5)^{2\log N} < 1/N^4$. This upper-bounds the probability of error of outputting a wrong value of $\min_b \mathrm{ED}(T[b,\ell], P)$ after receiving ℓ symbols of the text. As there will be at most N distinct values of ℓ, the probability of outputting

a wrong estimate of the edit distance of P to some suffix of T is at most $N \cdot 1/N^4 = 1/N^3$, conditioned on none of the bad events 1), 3) or 4) happening. Overall, the probability of a failure of our algorithm is at most $O(\log N/N) \leq 1/\sqrt{N}$, for N large enough, and it could be made an arbitrary small polynomial in N by choosing the parameters differently (n vs N).

It remains to argue that the copy of our algorithm which aligns $T[a,\ell]$ and P well, outputs their edit distance. Consider the copy of the algorithm that aligns grammars of $T[a,\ell]$ and P well. After arrival of the symbol $T[\ell]$ and updating the grammars, there are active grammars $G_1^a,\ldots,G_t^a$, committed grammars $G_1^T,\ldots,G_s^T$ and the pattern grammars $G_1^P,\ldots,G_r^P$. If $\mathrm{ED}(T[a,\ell],P)$ is at most k then the number of grammars in which P differs from the last r grammars of T is at most k. Thus the CKP-match algorithm can identify the differing grammars when computing the value m_{s-R+t} which is set to

$$\begin{aligned} m_{s-R+t} &= \min_b \mathrm{ED}(\mathrm{eval}(G^T_{s-r+t+1})[b,\ldots],\mathrm{eval}(G_1^P)) \\ &+ \sum_{i=2}^{r-R} \mathrm{ED}(\mathrm{eval}(G^T_{s-r+t+i}),\mathrm{eval}(G_i^P)). \end{aligned}$$

Since, $m_{s-R+t} \leq \mathrm{ED}(T[a,\ell],P) \leq k$, we have the true value of m_{s-R+t}. Thus,

$$\begin{aligned} \mathrm{ED}(T[a,\ell],P) &= m_{s-R+t} \\ &+ \sum_{i=r-R+1}^{r-t} \mathrm{ED}(\mathrm{eval}(G^T_{s-r+t+i}),\mathrm{eval}(G_i^P)) \\ &+ \sum_{i=r-t+1}^{r} \mathrm{ED}(\mathrm{eval}(G^a_{i-r+t}),\mathrm{eval}(G_i^P)). \end{aligned}$$

That is precisely how we evaluate the edit distance query of our algorithm.

If $\mathrm{ED}(T[a,\ell],P) > k$ then we will output a value $> k$ as we output some upper bound on the edit distance. Any value $> k$ is treated as the infinity.

7 Time complexity of the algorithm

In the first phase, we incrementally construct the grammars for the pattern P, using the BK-decomposition algorithm from Proposition 7 on each symbol of P at a time. Updating the active grammars for each new symbol takes $\widetilde{O}(k)$ time, committing each of the possible $\widetilde{O}(1)$ definite grammars to the CKP-match algorithm takes $\widetilde{O}(M \cdot \sqrt{K}) = \widetilde{O}(k^2)$. Thus the time needed per arriving symbol of the pattern is $\widetilde{O}(k^2)$.

For each symbol of the text that arrives during the second phase of the algorithm we need to update the active grammars of the text, update m_s, and evaluate the edit distance of the pattern from the current suffix of text. This includes parts *Symbol arrival, Committing a grammar* and *Pattern edit distance query* of the algorithm.

Symbol arrival. Appending a symbol using the BK-decomposition algorithm from Proposition 7 takes $\widetilde{O}(k)$ time.

Committing a grammar. Encoding the grammar takes $O(M)$ time using the algorithm from Proposition 8, and committing it to the CKP-match algorithm takes time $\widetilde{O}(k^2)$, as in the pattern case.

Querying the CKP-match algorithm for Hamming distance K takes $O(K) = \widetilde{O}(k^2)$ time. This recovers at most k pairs of distinct grammars $(G_i, G_i^{'})$, $1 \leq i \leq k$. Computing edit distance k_i of each pair of strings $\mathrm{eval}(G_i)$ and $\mathrm{eval}(G_i^{'})$, takes $\widetilde{O}(S + k_i^2) = \widetilde{O}(k + k_i^2)$ time using Proposition 2. If $\sum_i k_i \leq k$, the total time for the edit distance computation is bounded

by $\widetilde{O}(k^2)$. If the computation runs for longer we can stop it as we know m_s is larger than k. Running the algorithm from Corollary 3 on the first pair of distinct grammars to compute the minimum edit distance between any suffix of eval(G_1) and the string eval(G_1') takes $\widetilde{O}(S + k^2)$ time. Thus committing a grammar takes time at most $\widetilde{O}(k^2)$ where the longest time takes the minimization algorithm on the first pair of grammars.

Pattern edit distance query. This step requires the alignment of the last R grammars of the pattern with the appropriate grammars of the text and computing their edit distances. Using Proposition 2, computing edit distances of R pairs of grammars takes $R \times \widetilde{O}(k^2) = \widetilde{O}(k^2)$ time.

As there are at most $\widetilde{O}(1)$ committed grammars after processing each new symbol, the total time of this step is $\widetilde{O}(k^2)$ per arriving symbol.

8 Space complexity of the algorithm

During either phase of the algorithm, we store $O(RL) = \widetilde{O}(1)$ active and updated grammars and buffer last $O(R)$ committed grammars. This requires space $\widetilde{O}(k)$. Furthermore, the CKP-match algorithm requires $\widetilde{O}(K) = \widetilde{O}(k^2)$ space. The edit distance algorithm of Proposition 2 cannot use more space than its running time so each invocation uses at most $\widetilde{O}(k^2)$ space. Similarly, Corollary 3 uses space $\widetilde{O}(k^2)$. Thus our algorithm uses space at most $\widetilde{O}(k^2)$ at any point during its computation.

References

1. Amihood Amir, Moshe Lewenstein, and Ely Porat. Faster algorithms for string matching with k mismatches. *Journal of Algorithms*, 50(2):257–275, 2004.
2. Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless seth is false). In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing STOC*, pages 51–58, 2015.
3. Sudatta Bhattacharya and Michal Koucký. Locally consistent decomposition of strings with applications to edit distance sketching. In *Proceedings of the 55th Annual ACM SIGACT Symposium on Theory of Computing, STOC(to appear)*, 2023. `arXiv:2302.04475`.
4. Robert S Boyer and J Strother Moore. A fast string searching algorithm. *Communications of the ACM*, 20(10):762–772, 1977.
5. Dany Breslauer and Zvi Galil. Real-time streaming string-matching. *ACM Transactions on Algorithms (TALG)*, 10(4):1–12, 2014.
6. Timothy M Chan, Shay Golan, Tomasz Kociumaka, Tsvi Kopelowitz, and Ely Porat. Approximating text-to-pattern hamming distances. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 643–656, 2020.
7. Panagiotis Charalampopoulos, Tomasz Kociumaka, and Philip Wellnitz. Faster approximate pattern matching: A unified approach. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 978–989. IEEE, 2020.
8. Panagiotis Charalampopoulos, Tomasz Kociumaka, and Philip Wellnitz. Faster pattern matching under edit distance: a reduction to dynamic puzzle matching and the seaweed monoid of permutation matrices. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 698–707. IEEE, 2022.
9. Raphaël Clifford, Allyx Fontaine, Ely Porat, Benjamin Sach, and Tatiana Starikovskaya. Dictionary matching in a stream. In *Algorithms-ESA 2015: 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 361–372. Springer, 2015.
10. Raphaël Clifford, Allyx Fontaine, Ely Porat, Benjamin Sach, and Tatiana Starikovskaya. The k-mismatch problem revisited. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms SODA*, pages 2039–2052. SIAM, 2016.

11 Raphaël Clifford, Tomasz Kociumaka, and Ely Porat. The streaming k-mismatch problem. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 1106–1125. SIAM, 2019. doi:10.1137/1.9781611975482.68.

12 Richard Cole and Ramesh Hariharan. Approximate string matching: A simpler faster algorithm. *SIAM Journal on Computing*, 31(6):1761–1782, 2002.

13 Zvi Galil and Raffaele Giancarlo. Improved string matching with k mismatches. *ACM SIGACT News*, 17(4):52–54, 1986.

14 Arun Ganesh, Tomasz Kociumaka, Andrea Lincoln, and Barna Saha. How compression and approximation affect efficiency in string distance measures. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2867–2919, 2022. doi:10.1137/1.9781611977073.112.

15 Paweł Gawrychowski and Tatiana Starikovskaya. Streaming dictionary matching with mismatches. *Algorithmica*, pages 1–21, 2019.

16 Pawel Gawrychowski and Przemyslaw Uznanski. Towards unified approximate pattern matching for hamming and l_1 distance. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2018.

17 Shay Golan, Tomasz Kociumaka, Tsvi Kopelowitz, and Ely Porat. The Streaming k-Mismatch Problem: Tradeoffs Between Space and Total Time. In Inge Li Gørtz and Oren Weimann, editors, *31st Annual Symposium on Combinatorial Pattern Matching (CPM 2020)*, volume 161 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:15, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CPM.2020.15.

18 Shay Golan, Tsvi Kopelowitz, and Ely Porat. Towards optimal approximate streaming pattern matching by matching multiple patterns in multiple streams. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

19 Shay Golan and Ely Porat. Real-time streaming multi-pattern search for constant alphabet. In *25th Annual European Symposium on Algorithms (ESA 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

20 Richard M. Karp and Michael O. Rabin. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31(2):249–260, 1987. doi:10.1147/rd.312.0249.

21 Donald E. Knuth, James H. Morris Jr., and Vaughan R. Pratt. Fast pattern matching in strings. *SIAM J. Comput.*, 6(2):323–350, 1977.

22 Tomasz Kociumaka, Ely Porat, and Tatiana Starikovskaya. Small-space and streaming pattern matching with k edits. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 885–896. IEEE, 2022.

23 Gad M Landau and Uzi Vishkin. Efficient string matching with k mismatches. *Theoretical Computer Science*, 43:239–249, 1986.

24 Gad M Landau and Uzi Vishkin. Fast parallel and serial approximate string matching. *Journal of algorithms*, 10(2):157–169, 1989.

25 Benny Porat and Ely Porat. Exact and approximate pattern matching in the streaming model. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 315–323. IEEE, 2009.

26 Jakub Radoszewski and Tatiana Starikovskaya. Streaming k-mismatch with error correcting and applications. *Information and Computation*, 271:104513, 2020.

27 Tatiana Starikovskaya. Communication and streaming complexity of approximate pattern matching. In *28th Annual Symposium on Combinatorial Pattern Matching (CPM 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

On Computing the Vertex Connectivity of 1-Plane Graphs

Therese Biedl
David R. Cheriton School of Computer Science, University of Waterloo, Canada

Karthik Murali
School of Computer Science, Carleton University, Ottawa, Canada

Abstract

A graph is called *1-plane* if it has an embedding in the plane where each edge is crossed at most once by another edge. A crossing of a 1-plane graph is called an $\times$*-crossing* if there are no other edges connecting the endpoints of the crossing (apart from the crossing pair of edges). In this paper, we show how to compute the vertex connectivity of a 1-plane graph G without $\times$-crossings in linear time.

To do so, we show that for any two vertices u, v in a minimum separating set S, the distance between u and v in an auxiliary graph $\Lambda(G)$ (obtained by planarizing G and then inserting into each face a new vertex adjacent to all vertices of the face) is small. It hence suffices to search for a minimum separating set in various subgraphs Λ_i of $\Lambda(G)$ with small diameter. Since $\Lambda(G)$ is planar, the subgraphs Λ_i have small treewidth. Each minimum separating set S then gives rise to a partition of Λ_i into three vertex sets with special properties; such a partition can be found via Courcelle's theorem in linear time.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases 1-Planar Graph, Connectivity, Linear Time, Treewidth

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.23

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2212.06782` [2]

Funding *Therese Biedl*: Supported by NSERC.
Karthik Murali: Research done in part for the author's Master's thesis at University of Waterloo [26].

1 Introduction

The class of *planar graphs*, which are graphs that can be drawn on the plane without crossings, is fundamental to both graph theory and graph algorithms. Many problems can be more efficiently solved in planar graphs than in general graphs. However, real-world graphs, such as social networks and biological networks, are typically non-planar. But they are often *near-planar*, i.e., close to planar in some sense. One such graph class is the *1-planar graphs*, i.e., graphs that can be drawn on the plane such that each edge is crossed at most once. Introduced in 1965 [28], both structural and algorithmic properties of 1-planar graphs have been studied extensively, see [17, 21] for overviews.

In this paper, we look at the problem of vertex connectivity for 1-planar graphs. The problem of *vertex connectivity* is fundamental in graph theory: given a connected graph G, what is the size (denoted by $\kappa(G)$) of the smallest *separating set*, i.e., set of vertices whose removal makes G disconnected? Vertex connectivity has many applications, e.g. in network reliability and for measuring social cohesion.

Known Results. For an n-vertex m-edge graph G, one can test in linear (i.e. $O(m+n)$) time whether $\kappa(G) \geq 1$ with a graph traversal algorithm. In 1969, Kleitman [20] showed how to test $\kappa(G) \leq k$ in time $O(k^2nm)$. Subsequently, [29] and [18] presented linear-time algorithms to decide k-connectivity for $k = 2$ and $k = 3$ respectively. (Some errors in [18] were corrected in [15].) For $\kappa(G) = 4$, the first $O(n^2)$ algorithm was by Kanevsky and Ramachandran [19]. For $\kappa(G) \in O(1)$, the first $O(n^2)$ algorithm was by Nagamochi and Ibaraki [27]. For general k and m, the fastest running times are $\tilde{O}(n^\omega + nk^\omega)$ [25] and $\tilde{O}(kn^2)$ [16]. (Here, $\tilde{O}(g(n)) = O(g(n)\log^c n)$ for some constant c, and $\omega < 2.372$ is the matrix multiplication exponent.) Both algorithms are randomized and are correct with high probability. The fastest deterministic algorithm takes time $O(m \cdot (n + \min\{k^{5/2}, kn^{3/4}\}))$ [12].

Recent breakthroughs brought the run-time to test k-connectivity down to $\widehat{O}(m + \min\{n^{1.75}k^{1+k/2}, n^{1.9}k^{2.5}\})$ when $k < n^{1/8}$ [13], and to $\tilde{O}(m + nk^3)$ for randomized algorithms [11]. (Here $\widehat{O}(g(n)) = O(g(n)^{1+o(1)})$.) Very recently, Li et al. [24] showed that in fact $\widehat{O}(m)$ run-time can be achieved for randomized algorithms, independent of k. On the other hand, the problem of obtaining a deterministic linear time algorithm for deciding vertex connectivity is still open.

Vertex Connectivity in Planar Graphs. Any simple planar graph G has at most $3n - 6$ edges, hence has a vertex with at most five distinct neighbours, so $\kappa(G) \leq 5$. Since $\kappa(G) \leq 3$ can be tested in linear time, it only remains to test whether $\kappa(G) = 4$ or $\kappa(G) = 5$. In 1990, Laumond [23] gave a linear time algorithm to compute $\kappa(G)$ for maximal planar graphs. In 1999, Eppstein gave an algorithm to test vertex connectivity of all planar graphs in linear time [8]. His algorithm inspired the current work, and so we review it briefly here. Given a planar graph G with a fixed planar embedding (a *plane* graph), let the *radialization* be the plane graph obtained by adding a new vertex inside each face of G and connecting this *face vertex* to all vertices on the boundary of the face. The subgraph formed by the newly added edges is called the *radial graph* $R(G)$ [10]. The following is known.

▶ **Theorem 1** (attributed to Nishizeki in [8]). *Let S be a minimal separating set of a plane graph G. Then there is a cycle C in $R(G)$ with $V(C) \cap V(G) = S$ such that there are vertices of G inside and outside C.*

It hence suffices to find a shortest cycle C in $R(G)$ for which $V(C) \cap V(G)$ is a separating set of G. Since $\kappa(G) \leq 5$, this reduces to the problem of testing the existence of a bounded-length cycle (with some separation properties) in a planar graph. Eppstein solves this in linear time by modifying his planar subgraph isomorphism algorithm suitably [8].

Our Results. In this paper, we consider testing vertex connectivity of near-planar graphs, a topic that to our knowledge has not been studied before. We focus on 1-planar graphs that come with a fixed 1-planar embedding (*1-plane graphs*), since testing 1-planarity is NP-hard [14]. Since a simple 1-planar graph G has at most $4n-8$ edges [3], we have $\kappa(G) \leq 7$. For technical reasons we assume that G has no $\times$-*crossing*, i.e., a crossing without other edges among the endpoints of the crossing edges.

Let G be a 1-plane graph without $\times$-crossings. Inspired by Eppstein's approach, we define a planar auxiliary graph $\Lambda(G)$, and show that G has a separating set of size k if and only if $\Lambda(G)$ has a *co-separating triple* with size- and diameter-restrictions. (Roughly speaking, "co-separating triple" means that the vertices of $\Lambda(G)$ can be partitioned into three sets (A, X, B), such that X separates A, B in $\Lambda(G)$ while simultaneously $X \cap V(G)$ separates $A \cap V(G)$ and $B \cap V(G)$ in G. Detailed definitions are in Section 2.)

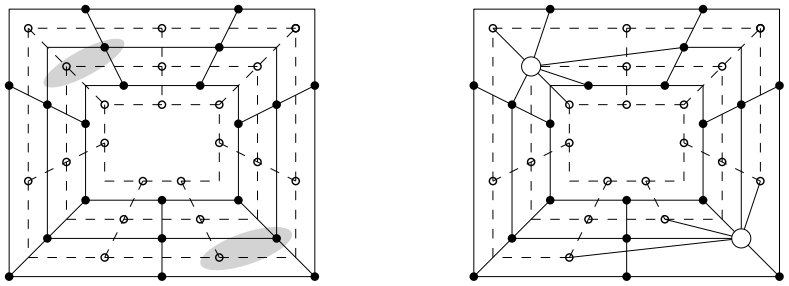


Figure 1 Theorem 2 does not hold for 1-plane graphs with ×-crossings.

▶ **Theorem 2.** *Let G be a 1-plane graph without ×-crossings. Then G has a separating set of size at most k if and only if $\Lambda(G)$ has a co-separating triple (A, X, B) where $|X \cap V(G)| \leq k$ and the subgraph of $\Lambda(G)$ induced by X has diameter at most $4k$.*

Let S be a minimum separating set of G and let (A, X, B) be the co-separating triple derived from S with this theorem. Since vertices of X are close to each other in $\Lambda(G)$, we can project (A, X, B) onto a subgraph $\Lambda_i \subseteq \Lambda(G)$ of diameter $O(|S|)$ to obtain a co-separating triple (A_i, X, B_i) of Λ_i. Conversely, we will show that with a suitable definition of subgraph Λ_i every co-separating triple (A_i, X, B_i) of Λ_i can be extended into a co-separating triple (A, X, B) of $\Lambda(G)$ from which we can obtain $X \cap V(G)$ as a separating set of G. Since Λ_i is planar and has diameter $O(|S|)$, it has treewidth $O(|S|)$. Using standard approaches for graphs of small treewidth, and by $\kappa(G) \leq 7$, we can search for (A_i, X, B_i) in linear time. Therefore we will obtain:

▶ **Theorem 3.** *The vertex connectivity of a 1-plane graph without ×-crossings can be computed in linear time.*

Limitations. We briefly discuss here the difficulty with ×-crossings. Figure 1(a) shows two copies of a graph that are interleaved to produce a 1-planar embedding such that each crossing is an ×-crossing. When these two graphs are fused together by identifying two pairs of vertices that are diametrically opposite to each other (shown by grey blobs in Figure 1(a)), we get the graph G in Figure 1(b). The two fused vertices form a separating set of G. Moreover, this is the only minimum separating set since both graphs in Figure 1(a) are 3-connected. This example can be extended (by adding more concentric layers and more vertices within each layer) to show that the distance between the two fused vertices can be made arbitrarily large even in $\Lambda(G)$. Thus Theorem 2 fails to hold for graphs with ×-crossings, and in consequence our techniques to test vertex connectivity cannot be extended to them.

Organization of the Paper. In Section 2, we lay out the preliminaries, defining co-separating triples and $\Lambda(G)$ for 1-plane graphs. In Section 3, we generalize Theorem 1 to the class of *full 1-plane graphs*, which are 1-plane graphs where the endpoints of each crossing induce the complete graph K_4. Using this result we prove Theorem 2 in Section 4, and turn it into a linear-time algorithm in Section 5. We summarize in Section 6.

2 Preliminaries

We assume familiarity with graphs, see e.g. [7]. All graphs in this paper are assumed to be connected and have no loops. A *separating set* of a graph G is a set S of vertices such that $G \setminus S$ is disconnected; we use the term *flap* for a connected component of $G \setminus S$. Set S *separates* two sets A, B if there is no path connecting A and B in $G \setminus S$. The *vertex connectivity* of G, denoted $\kappa(G)$, is the cardinality of a minimum separating set. For any vertex set A, an *A-vertex* is a vertex that belongs to A; we also write *G-vertex* for a vertex of $V(G)$.

A drawing of a graph in the plane is called *good* if edges are simple curves that intersect only if they properly cross or have a common endpoint, any two edges intersect at most once, and no three edges cross in a point. A *1-planar graph* is a graph that has a good drawing in the plane where each edge is crossed at most once; such a drawing is called a *1-planar drawing* and a graph with a given 1-planar drawing is called a *1-plane graph*. Throughout the paper, we assume that we are given a 1-plane graph G.

Let $\{(u, v), (w, x)\}$ be a crossing in G, i.e., edges (u, v) and (w, x) cross each other. The vertices $\{u, v, w, x\}$ are called *endpoints of the crossing*; these are four distinct vertices since the drawing is good. Two endpoints are called *consecutive* if they are not the two ends of (u, v) or (w, x). We distinguish six types of crossings by whether consecutive endpoints are adjacent (see Figure 2). As in [9], we call a crossing *full* if $\{u, v, w, x\}$ induces K_4, and *almost-full* if $\{u, v, w, x\}$ induces K_4 minus one edge.[1] We call it *bowtie* if $\{u, v, w, x\}$ induces a cycle, *arrow* if $\{u, v, w, x\}$ induces $K_{1,3}$ plus one edge, *chair* if $\{u, v, w, x\}$ induces a path of length three (the *length* of a path is its number of edges), and the crossing is an $\times$-crossing otherwise (no edges connecting consecutive endpoints of the crossing). For an almost-full crossing, there are exactly two consecutive non-adjacent endpoints; we call these the *wing tips* and the other two endpoints the *spine-vertices*. For an arrow crossing, depending on whether an endpoint is adjacent to zero or two of its consecutive endpoints, we call it the *tail* or *tip*; the other two endpoints are the *base vertices*.

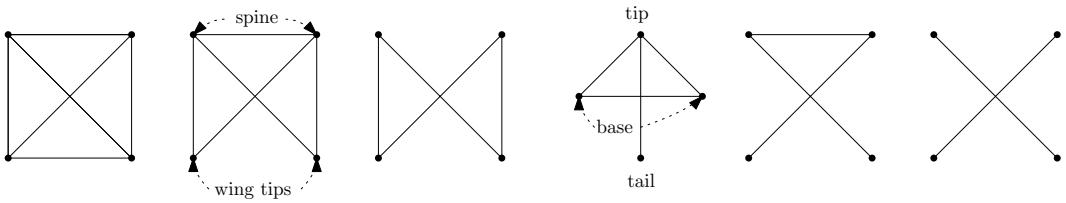


Figure 2 Types of crossings. From left to right: Full, almost-full, bowtie, arrow, chair and $\times$.

The *planarization* of G, denoted $G^\times$, is obtained by replacing any crossing $\{(u, v), (w, x)\}$ with a *dummy vertex*, i.e., remove the crossing edges and insert (at the point where the crossing used to be) a new vertex adjacent to all of u, v, w, x. The resulting drawing is *planar*, i.e., has no crossings. In a planar drawing Γ, a *face* is a maximal region of $\mathbb{R}^2 \setminus \Gamma$. Drawing Γ defines at each vertex v the *rotation* $\rho(v)$, which is the circular list of incident edges and faces, ordered clockwise.

[1] If G has parallel edges, then "$\{u, v, w, x\}$ induces graph H" is intended to mean "the underlying simple graph of the graph induced by $\{u, v, w, x\}$ is H".

Pre-processing. In Figure 2, we assumed that any edge (u, x) between consecutive endpoints of a crossing is actually drawn near that crossing, i.e., $G^\times$ contains a face incident to (u, x) and the dummy vertex of the crossing. (We call such a face a *kite face* and the edge a *kite edge*.) In general this may not be true. But if (u, x) exists elsewhere in the drawing, then we can duplicate it and insert it as a kite edge. This affects neither 1-planarity nor vertex connectivity nor crossing type, so assume from now on that at any crossing all edges among consecutive endpoints exist as kite edges. Since we never create a face of $G^\times$ that is bounded by two edges, graph G continues to have at most $4n - 8$ edges.

Radial Planarization. Recall that Eppstein [8] used the radialization to compute the vertex connectivity of a planar graph. We now generalize this concept to our 1-plane graph G as follows. The *radial planarization*, denoted $\Lambda(G)$, is obtained by first planarizing G to obtain $G^\times$, and then radializing $G^\times$ (see Figure 3). In other words, we add a *face vertex* f inside each face F of $G^\times$, and for every incidence of F with a vertex v we add an edge (v, f), drawn inside F and inserted in the drawing of $\Lambda(G)$ such that it bisects the occurrence of F in the rotation $\rho(v)$. (Repeated incidences of F with v give rise to parallel edges (f, v).) As in [10], we use the term *radial graph* for the subgraph $R(G)$ of $\Lambda(G)$ formed by the edges incident with face vertices. Note that $\Lambda(G)$ has three types of vertices: G-vertices, dummy vertices that replace crossings of G, and face vertices. For a cycle C in $\Lambda(G)$, we define the shortcut $\mathbf{V}_G(C) := V(G) \cap V(C)$ for the G-vertices of C.

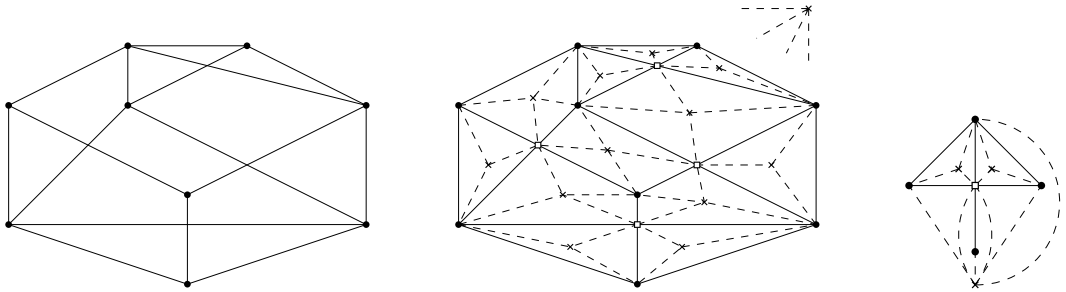


Figure 3 (a) A 1-plane graph G. (b) Its radial planarization $\Lambda(G)$. (c) $\Lambda(G)$ if G is an arrow crossing. Face vertices are crosses, dummy vertices are white squares, edges of $R(G)$ are dashed.

Co-separating Triple. We now clarify what it means to be separating in G and $\Lambda(G)$ simultaneously. We will actually give this definition for an arbitrary graph Λ that shares some vertices with G (since it will later be needed for graphs derived from $\Lambda(G)$).

▶ **Definition 4** (Co-separating triple). *Let Λ be a graph that shares some vertices with G. A partition of the vertices of Λ into three sets (A, X, B) is called a* co-separating triple of Λ *if it satisfies the following properties:*

1. *Each of A, X and B contains at least one G-vertex.*
2. *For any two vertices $a \in A$ and $b \in B$, there is no edge (a, b) in either $E(\Lambda)$ or $E(G)$.*

We say that a co-separating triple of Λ has *diameter* d if any two vertices of X have distance at most d in Λ. When $\Lambda = \Lambda(G)$, then all G-vertices belong to Λ, and since A and B both contain G-vertices the following is immediate:

▶ **Observation 5.** *If (A, X, B) is a co-separating triple of $\Lambda(G)$, then X is a separating set of $\Lambda(G)$ and $X \cap V(G)$ is a separating set of G.*

3 Full 1-Plane Graphs

In this section, we study *full 1-plane graphs*, i.e., 1-plane graphs where all crossings are full. In this case, we will find a co-separating triple (A, X, B) that has a special form (this will be needed in Section 4): Vertex set X contains no dummy vertices, forms a cycle C in $R(G)$, and A and B are exactly the vertices inside and outside this cycle in the planar drawing of $\Lambda(G)$. (In other words, we generalize Theorem 1.)

▶ **Theorem 6.** *Let G be a full 1-plane graph and S be a minimal separating set of G. Then there is a cycle C in $R(G)$ such that C does not visit dummy vertices, $\mathbf{V}_G(C) = S$, and there are vertices of G inside and outside C.*

Proof. The broad idea is to take a maximal path in $R(G)$ that alternates between S-vertices and face vertices with suitable properties, and then close it up into a cycle C of $R(G)$ that separates two vertices of G. Hence C automatically does not visit dummy vertices and $\mathbf{V}_G(C) = S$ since S is minimal. We explain the details now.

Call a face F of $G^\times$ a *transition face* if F is either incident to an edge between two S-vertices, or F is incident to vertices from different flaps of $G \setminus S$. The corresponding face vertex in $\Lambda(G)$ is called a *transition-face vertex*. By walking along the boundary of a transition face, one can easily show the following (see the full version [2]):

▷ Claim 7. Every transition face F contains at least two vertices of S or two incidences with the same vertex of S.

In consequence, any transition-face vertex has (in $R(G)$) two edges to vertices of S. Vice versa, any S-vertex v has (in $R(G)$) two transition-face neighbours, because in the planarization $G^\times$ we transition at v from edges leading to one flap of $G \setminus S$ to edges leading to another flap and back; this can happen only at transition faces. See the full version [2] for a proof of the following claim (and for the formal definition of "clockwise between").

▷ Claim 8. Let (v, t_1) and (v, t_2) be two edges of G such that $v \in S$ and t_1 and t_2 are in different flaps of $G \setminus S$. Then there exists a transition face incident to v that is clockwise between (v, t_1) and (v, t_2).

With this, it is obvious that we can find a simple cycle C that alternates between transition-face vertices and S-vertices, but we need to ensure that C has vertices of G inside and outside, and for this, we choose C more carefully. Formally, let v_1 be an arbitrary S-vertex. Let $P = v_1 \dots v_k$ be a simple path that alternates between transition-face vertices and S-vertices and that is maximal in the following sense: $v_k \in S$, and for any transition-face vertex v_{k+1} adjacent to v_k and any S-vertex v_{k+2} adjacent to v_{k+1}, at least one of v_{k+1}, v_{k+2} already belongs to P. Since $v_k \in S$ and S is minimal, v_k has neighbours t_1, t_2 in different flaps of $G \setminus S$. Applying Claim 8 twice gives a transition face F clockwise between (v_k, t_1) and (v_k, t_2), and a transition face F' clockwise between (v_k, t_2) and (v_k, t_1). If $k > 1$, then (up to renaming of t_1, t_2, F') we may assume that v_{k-1} is the face vertex of F'. Hence for $k > 1$, edge (v_k, f) (where f is the face vertex of F) is not on P, and the same holds vacuously also if $k = 1$. We have cases:

- In the first case, $f \in P$, say $f = v_i$ for some $1 \le i \le k-1$ (Figure 4(a)). Then $C := v_i v_{i+1} \dots v_k v_i$ is a cycle with t_1 and t_2 on opposite sides.
- In the second case $f \notin P$. By Claim 7, $R(G)$ contains at least two edges e, e' that connect f to S-vertices. Up to renaming we may assume that $e = (v_k, f)$. Consider extending P via e and e'. By maximality of P the result is non-simple, and by $f \notin P$ therefore $e' = (f, v_i)$ for some $1 \le i \le k$.

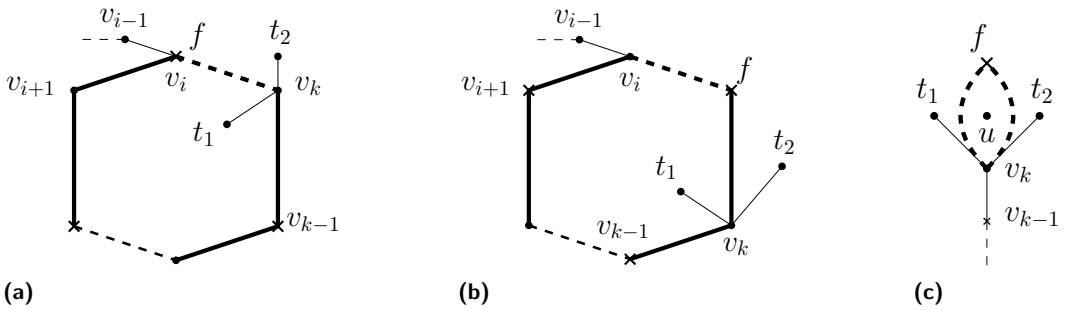


Figure 4 Constructing the cycle C (bold) in $R(G)$.

If $1 \leq i \leq k-1$ (Figure 4(b)), then define simple cycle $C := v_i v_{i+1} \dots v_k f v_i$ and observe that it has t_1 and t_2 on opposite sides. Otherwise ($i = k$) both edges e, e' connect v_k to f (Figure 4(c)), and we let C be the cycle consisting of e and e'. There can be parallel edges incident to f only because face F of $G^\times$ was incident to v_k repeatedly. Edges e, e' were then added to $\Lambda(G)$ in such a way that at least one other G-vertex on the boundary of F lies between those incidences on either side of cycle C.

So in either case we have constructed a cycle C in $R(G)$ with $\mathbf{V}_G(C) \subseteq S$ that does not visit dummy vertices and with two G-vertices (say a and b) inside and outside C. To show that $\mathbf{V}_G(C) = S$, it is sufficient (by minimality of S) to show that $\mathbf{V}_G(C)$ is separating in G. To see this, fix any path π from a to b in G and let π_Λ be the corresponding path in $\Lambda(G)$ obtained by replacing crossings by dummy vertices. Path π_Λ must intersect C, but it uses only G-vertices and dummy vertices while C uses only G-vertices and face vertices, so π_Λ intersects C in a vertex of G and $\mathbf{V}_G(C)$ hence separates a from b in G. ◀

4 1-Plane Graphs Without $\times$-Crossings

In this section, we prove Theorem 2: Minimal separating sets correspond to co-separating triples with small diameter. One direction is easy: If $\Lambda(G)$ has a co-separating triple (A, X, B) with $|V(G) \cap X| \leq k$, then from Observation 5, G has a separating set of size at most k.

Proving the other direction is harder, and we first give an outline. For the rest of this section, fix a minimal separating set S, and two arbitrary flaps ϕ_1, ϕ_2 of $G \setminus S$. We first augment graph G to G_{aug} by adding more edges; this is done to reduce the types of crossings that can exist and thereby the number of cases. (Augmenting the graph is *only* used as a tool to prove Theorem 2; the vertex connectivity algorithm does not use it.) We then find a cycle C for $\Lambda(G_{\text{aug}})$ with vertices of G inside and outside C such that all vertices of S are in the neighbourhood of C. To do so we temporarily modify G_{aug} further to make it a full 1-plane graph G^+_{aug}, appeal to Theorem 6, and show that the resulting cycle can be used for C. By setting $X_{\text{aug}} = V(C) \cup S$, this cycle gives a co-separating triple $(A_{\text{aug}}, X_{\text{aug}}, B_{\text{aug}})$ of $\Lambda(G_{\text{aug}})$, and using C we can argue that the diameter of the graph induced by X_{aug} is small. Finally we undo the edge-additions to transfer the co-separating triple from $\Lambda(G_{\text{aug}})$ to $\Lambda(G)$.

Augmentation. We define the *augmentation* of G with respect to S, ϕ_1, ϕ_2 to be the graph $G_{\text{aug}} := G_{\text{aug}}(S, \phi_1, \phi_2)$ obtained as the result of the following iterative process:

For any two consecutive endpoints u, x of a crossing, if there is no kite edge (u, x) and it could be added without connecting flaps ϕ_1, ϕ_2, then add the kite edge, update the flaps ϕ_1 and ϕ_2 (because they may have grown by merging with other flaps), and repeat.

By construction S remains a separating set with flaps ϕ_1, ϕ_2 in G_{aug}, and it is minimal since adding edges cannot decrease connectivity. Also, one can easily show the following properties of crossings in G_{aug} (here not having $\times$-crossings is crucial), see Figure 5 for an illustration, and the full version [2] for details.

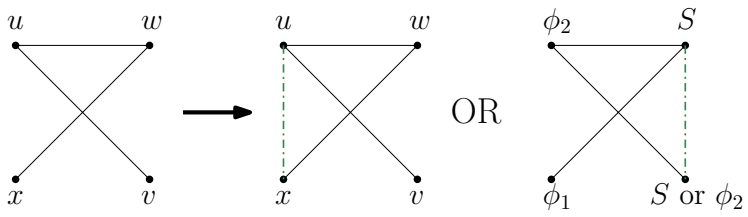


Figure 5 At a chair crossing, we can always add an edge to create an arrow-crossing.

▶ **Observation 9.** *The crossings of $G_{\text{aug}} = G_{\text{aug}}(S, \phi_1, \phi_2)$ have the following properties:*

1. *Any crossing is full, almost-full or an arrow crossing.*
2. *At any almost-full crossing, the spine vertices belong to S and the wing tips belong to ϕ_1 and ϕ_2.*
3. *At any arrow crossing, the tip belongs to S, the tail belongs to one of ϕ_1, ϕ_2, and the base vertices belong to the other of ϕ_1, ϕ_2.*

Extending Theorem 1? Note that we expanded Theorem 1 (for plane graphs) to Theorem 6 (for full 1-plane graphs), but as we illustrate now, it cannot be expanded to 1-plane graphs without $\times$-crossings. One example for this is the graph that exists of exactly one arrow crossing (see Figure 3(c)), because the tip is a separating set, but there is no 2-cycle in $R(G)$ that contains the tip. For an example with higher connectivity, consider Figure 6. The figure shows a 1-plane graph where each crossing is an arrow crossing or a chair crossing. The graph is 4-connected and a minimum separating set S is shown by vertices marked with white disks. One can verify that in the radial planarization of the graph, there is no 8-cycle in $R(G)$ that contains all vertices in S. (This example will also be used later as running example for our approach.)

Cycle C in $\Lambda(G_{\text{aug}})$. So we cannot hope to find a cycle C in $\Lambda(G_{\text{aug}})$ with G-vertices on both sides that goes *exactly* through S. But we can find a cycle C that is "adjacent" to all of S. To make this formal, define for a cycle C in $\Lambda(G_{\text{aug}})$ the set $\mathbf{V}_\times(C)$ to be the set of all vertices of C that are vertices of $G^\times_{\text{aug}}$, i.e., they are G-vertices or dummy vertices of $\Lambda(G_{\text{aug}})$.

▶ **Lemma 10.** *There is a cycle C in $\Lambda(G_{\text{aug}})$ that uses only edges of $R(G_{\text{aug}})$ and such that*
(1) *every vertex in $\mathbf{V}_\times(C)$ is either in S or is a dummy vertex adjacent to an S-vertex,*
(2) *every S-vertex is either in $\mathbf{V}_\times(C)$ or adjacent to a dummy vertex in $\mathbf{V}_\times(C)$,*
(3) *there are G-vertices that are not in S both inside and outside C, and*
(4) *S separates G-vertices inside C from G-vertices outside C in G_{aug}.*

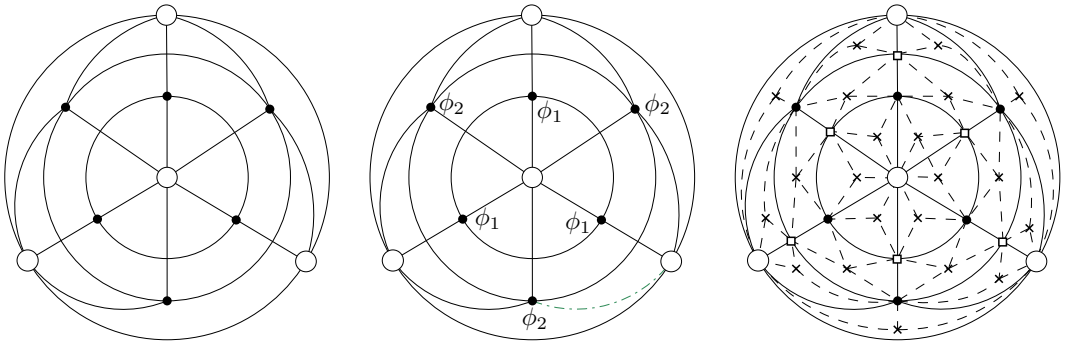


Figure 6 A 4-connected 1-plane graph G, its augmentation G_{aug} with respect to the minimum separating set S (white disks, the added edge is green/dot-dashed), and its radial planarization $\Lambda(G_{\text{aug}})$. Note that the unique chair crossing in G becomes an arrow crossing in G_{aug}. The radial graph does not have an 8-cycle passing through S.

Proof. As outlined, we first convert G_{aug} to a full 1-plane graph G^+_{aug} as follows (see Figure 7 for the abstract construction and Figure 8(a) for the running example): At every almost-full crossing and every arrow crossing, replace the crossing with a dummy vertex. At every arrow crossing, furthermore add a *base edge*, which connects the base vertices and is inserted so that it forms a full crossing. Since every crossing of G_{aug} is full, almost-full or arrow, all crossings of G^+_{aug} are full. We use $D := V(G^+_{\text{aug}}) \setminus V(G)$ for the new vertices and note that every vertex in D is adjacent to an S-vertex and corresponds to a dummy vertex in $\Lambda(G_{\text{aug}})$.

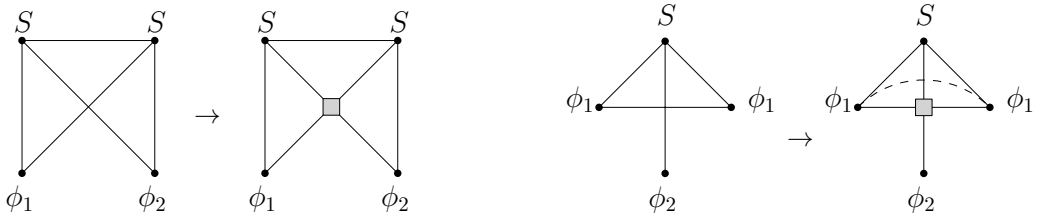


Figure 7 From G_{aug} to G^+_{aug}. Vertices in D are grey squares, the base edge is dashed.

Define $S^+ := S \cup D$ and observe that this is a separating set of G^+_{aug} since no edge of G^+_{aug} connects ϕ_1 with ϕ_2. Apply Theorem 6 to G^+_{aug} and a subset of S^+ that is minimally separating. This gives a cycle C in $R(G^+_{\text{aug}})$ such that $\mathbf{V}_{G^+_{\text{aug}}}(C) \subseteq S^+$, C does not visit dummy vertices of G^+_{aug}, and there are G^+_{aug}-vertices inside and outside C. See Figure 8(b). We claim that C satisfies all conditions, for which we first need to show that it actually is a cycle in $\Lambda(G_{\text{aug}})$. The only difference between $\Lambda(G_{\text{aug}})$ and $\Lambda(G^+_{\text{aug}})$ is at each base edge: Here $\Lambda(G^+_{\text{aug}})$ has an extra vertex c (the dummy vertex for the crossing created by the base edge) and the four incident face vertices, while $\Lambda(G_{\text{aug}})$ has only the two face vertices of the kite faces at the arrow crossing. But by Theorem 6 cycle C does not visit c, so C also is a cycle of $\Lambda(G_{\text{aug}})$.

To see that C satisfies (1), observe that $\mathbf{V}_\times(C) = \mathbf{V}_{G^+_{\text{aug}}}(C) \subseteq S^+ = S \cup D$, and every vertex of D is a dummy vertex of $\Lambda(G_{\text{aug}})$ that is adjacent to an S-vertex. Next we show (3). We know that there exist two vertices $a, b \in V(G^+_{\text{aug}})$ inside and outside C. If $a \in S \cup D$, then inspection of Figure 7 shows that a has a neighbour a' in $\phi_1 \cup \phi_2$ (hence $a' \in V(G)$ but $a' \notin S \cup D$). By $\mathbf{V}_\times(C) \subseteq S \cup D$ therefore a' is on the same side of C as a. Up to renaming hence $a \notin S \cup D$, and likewise $b \notin S \cup D$. This proves (3).

Before proving (2) and (4), we first show that the same vertices a and b are separated (in G_{aug}) by the set S' consisting of all S-vertices that are in $\mathbf{V}_\times(C)$ or adjacent to $\mathbf{V}_\times(C) \cap D$. To do so, pick an arbitrary path π from a to b in G_{aug}. We define a path π^+ in G^+_{aug} that corresponds to π as follows: use the same set of edges of π, except if π used an edge (r, s) that is part of an almost-full or an arrow crossing. At an almost-full crossing, we replace (r, s) by a path r-d-s where $d \in D$ is the dummy vertex. At an arrow crossing we have two cases. If r, s were the base vertices, then we replace (r, s) by the base edge. If r, s were tip and tail, then we replace (r, s) by r-d-s where $d \in D$ is the dummy vertex.

Since π^+ is a path from inside C to outside C in G_{aug}, it contains a vertex $w \in \mathbf{V}_\times(C) \subseteq S \cup D$. If $w \in S$, then define $t := w$. If $w \in D$, then near w path π^+ must have had the form r-w-s for some $(r, s) \in \pi$, due to our construction of π^+. Furthermore, (r, s) either belongs to an almost-full crossing, or to an arrow crossing with (r, s) connecting the tip and tail. For both types of crossings, one of r, s belongs to S, and we define t to be this vertex. So we have found a vertex $t \in S$ on π that is either on $\mathbf{V}_\times(C)$ or adjacent to a dummy vertex $d \in D \cap \mathbf{V}_\times(C)$. Therefore $t \in S'$ and so any path from a to b intersects S'. So $S' \subseteq S$ is a separating set of G_{aug}, hence by minimality $S' = S$, which proves (2). Also the G-vertices a and b are inside and outside C and separated by S, which proves (4). ◀

Notice that this lemma immediately implies a co-separating triple $(A_{\text{aug}}, X_{\text{aug}}, B_{\text{aug}})$ of G_{aug}: Fix such a cycle C, let $X_{\text{aug}} = V(C) \cup S$ and let A_{aug} and B_{aug} be the sets of vertices of $\Lambda(G_{\text{aug}}) \setminus X_{\text{aug}}$ inside and outside C respectively. See Figure 8(c). Clearly this is a partition, and by Lemma 10(3), both A_{aug} and B_{aug} have a G-vertex. As A_{aug} and B_{aug} are on opposite sides of cycle C, $V(C) \subseteq X_{\text{aug}}$ separates A_{aug} and B_{aug} in $\Lambda(G_{\text{aug}})$, and by Lemma 10(4), $S \subseteq X_{\text{aug}}$ separates $A_{\text{aug}} \cap V(G)$ and $B_{\text{aug}} \cap V(G)$ in G_{aug}. Therefore there can be no edge (a, b) with $a \in A_{\text{aug}}$ and $b \in B_{\text{aug}}$ in either $\Lambda(G_{\text{aug}})$ or G_{aug}.

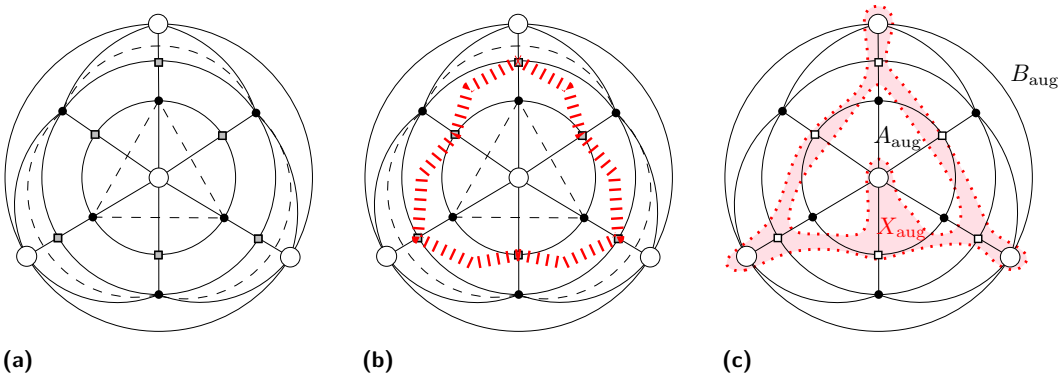


Figure 8 Finding a co-separating triple for the graph G from Figure 6. (a) Graph G^+_{aug}; vertices in D are grey squares and base edges are dashed. (b) Cycle C for the minimal cutting set D; we do not show the face-vertices. Note that every vertex of S is adjacent (in G_{aug}) to a dummy-vertex on C. (c) The resulting co-separating triples $(A_{\text{aug}}, X_{\text{aug}}, B_{\text{aug}})$.

Small Diameter. In order to prove Theorem 2, we first argue that the subgraph of $\Lambda(G_{\text{aug}})$ induced by X_{aug} has small diameter. Clearly the diameter of this graph is in $O(|C|)$ since all vertices of X_{aug} are on C or adjacent to it by Lemma 10(2). However, $|C|$ may not be in $O(|S|)$, which is why we need a more careful analysis to bound the length of a walk connecting two vertices of X_{aug}. Furthermore, to transfer the diameter-bound to $\Lambda(G)$ later, we need to exclude the edges that were added in G_{aug} from such walks. Write $E_{\text{aug}} := E(G_{\text{aug}}) \setminus E(G)$

(in Figure 6 the unique edge in E_{aug} is green/dash-dotted). Recall that edges in E_{aug} are kite edges, hence connect two vertices of G and have no crossing, therefore these edges also exist in $\Lambda(G_{\text{aug}})$.

▶ **Lemma 11.** *For any two vertices $u, v \in X_{aug}$, there is a walk W from u to v in $\Lambda(G_{\text{aug}})$ that has length at most $4|S|$ and does not use edges of E_{aug}.*

Proof. Since $u, v \in X_{\text{aug}}$, they are either in $\mathbf{V}_\times(C)$ (recall that this includes dummy vertices of $\Lambda(G_{\text{aug}})$ on C), or face vertices on C, or in S. In the latter two cases they are within distance one of some vertex in $\mathbf{V}_\times(C)$. So there exist vertices $u', v' \in \mathbf{V}_\times(C)$ that are within distance at most one of u and v, respectively. Enumerate one of the paths between u', v' along cycle C as $x_0, \ldots, x_{2t}$ with $x_0 = u'$ and $x_{2t} = v'$. (Observe that the vertices in $\mathbf{V}_\times(C)$ are exactly the even-indexed ones since C uses edges of $R(G_{\text{aug}})$.) Each vertex x_{2i} for $i = 0, \ldots, t$ is either in S (then set $s_{2i} := x_{2i}$), or by Lemma 10(1) it is a dummy vertex that has a neighbour $s_{2i} \in S$. Define π to be the walk

$$u, u'{=}x_0, s_0, x_0, x_1, x_2, s_2, x_2, x_3, \ldots, x_{2i-1}, x_{2i}, s_{2i}, x_{2i}, x_{2i+1}, \ldots, s_{2t}, x_{2t}{=}v', v,$$

i.e., it is the walk from u to v via C with detours at even-indexed vertices to reach an S-vertex. Observe that π has two properties: (1) At most three consecutive vertices in π do not belong to S, and at the ends there are at most two consecutive vertices not in S; (2) if y, z are two consecutive vertices of π that are different, then at least one of them is a face vertex or a dummy vertex, hence $(y, z) \notin E_{\text{aug}}$. We call a walk that satisfies (1) and (2) an *S-hopping walk*.

Let W be the shortest S-hopping walk from u to v. Observe that W can visit any S-vertex at most once, for otherwise we could find a shorter S-hopping walk by omitting the part between a repeated S-vertex. Since W contains at most three vertices between any two S-vertices, and at most two vertices not in S at the beginning and end, it has length at most $4|S|$. ◀

From G_{aug} to G. We now show how to transform $(A_{\text{aug}}, X_{\text{aug}}, B_{\text{aug}})$ into a co-separating triple (A, X, B) of $\Lambda(G)$ of small diameter. Recall that $G_{\text{aug}} = G \cup E_{\text{aug}}$, so $\Lambda(G_{\text{aug}})$ is obtained from $\Lambda(G)$ by inserting the edges E_{aug} and splitting any face vertex of a face that was divided by an edge in E_{aug}. We undo this in two parts. First, remove the edges of E_{aug} from $\Lambda(G_{\text{aug}})$. This does not affect the separation properties of $(A_{\text{aug}}, X_{\text{aug}}, B_{\text{aug}})$ since we only remove edges, and it maintains the diameter since the walks of Lemma 11 do not use E_{aug}. The second step is to identify face vertices that belong to the same face of G. Define sets A, B, X to be the same as $A_{\text{aug}}, B_{\text{aug}}, X_{\text{aug}}$ except that face vertices that were identified need to be replaced. To do so, observe that A_{aug} and B_{aug} are on opposite sides of C, and so no face vertex of A_{aug} can get identified with a face vertex of B_{aug} unless they both get identified with a face vertex of C. Thus the resulting face vertices come in three kinds: entirely composed of face vertices of A_{aug} (add these to A), entirely composed of face vertices of B_{aug} (add these to B), and containing a face vertex of C (add these to X).

Clearly A, B, X contain vertices of G since $A_{\text{aug}}, B_{\text{aug}}, X_{\text{aug}}$ did and we only identified face vertices. Assume for contradiction that (a, b) is an edge of G or $\Lambda(G)$ for some $a \in A$ and $b \in B$. Then (a, b) is not an edge in G_{aug} or $\Lambda(G_{\text{aug}})$. Thus at least one of a, b must be a face vertex that resulted from identifications. But with our choice of A and B then there was some edge (a', b') in $\Lambda(G_{\text{aug}})$ connecting vertices in A_{aug} and B_{aug}, a contradiction. Thus (A, X, B) is the desired co-separating triple and Theorem 2 holds.

5 Computing Vertex Connectivity in Linear Time

In this section, we show how to use Theorem 2 to obtain a linear time algorithm for finding the vertex connectivity of 1-plane graphs without $\times$-crossings. The crucial insight is that we only need to find the smallest k for which there is a co-separating triple (A, X, B) with $|X \cap V(G)| = k$. Moreover, the subgraph of $\Lambda(G)$ induced by X has small diameter. Therefore we can create some subgraphs $\Lambda_1, \Lambda_2, \ldots$ of $\Lambda(G)$ that have small treewidth and X belongs to at least one subgraph. (We assume that the reader is familiar with treewidth and its implications for algorithms; see for example [4] or the full version [2].) We can search for a co-separating triple within these subgraphs using standard approaches for graphs of bounded treewidth, quite similar to the planar subgraph isomorphism algorithm by Eppstein [8].

The Graphs Λ_i. As a first step, we perform a breadth-first search in $\Lambda(G)$ starting at an arbitrary vertex (the *root*); let T be the resulting BFS-tree. For $j = 1, 2, \ldots$ let V_j (the *jth layer*) be the vertices at distance $j-1$ from the root, and let d be the largest index where V_j is non-empty. Define $V_j := \emptyset$ for any index $j < 1$ or $j > d$. For any $a \leq b$, the notation $\Lambda[V_a \cup \cdots \cup V_b]$ will be a shortcut for the subgraph of $\Lambda(G)$ induced by $V_a \cup \cdots \cup V_b$.

Assume that we know the size $k \leq 7$ of the separating set that we seek (we will simply try all possibilities for k later). Define $w = 4k + 2 \leq 30$, so we know that any two vertices in X (of some putative co-separating triple (A, X, B) that satisfies the conclusion of Theorem 2) have distance at most $w - 2$ in $\Lambda(G)$. Hence X belongs to $V_{i+1} \cup \ldots \cup V_{i+w-2}$ for some $i \in \{0, \ldots, d-w+2\}$. Thus we will search for X within $\Lambda[V_{i+1} \cup \ldots \cup V_{i+w-2}]$, but to guarantee that there are vertices representing A and B we also keep two extra layers above and below (i.e., layers $V_{i-1}, V_i, V_{i+w-1}, V_{i+w}$). Furthermore, we add an edge set U_{i-1} ('upper edges') within V_{i-1} and an edge set L_{i+w} ('lower edges') within V_{i+w} that have some special properties.

▷ Claim 12. For $i \in \{0, \ldots, d-w+2\}$, there exist sets of edges U_{i-1} (connecting vertices of V_{i-1}) and L_{i+w} (connecting vertices of V_{i+w}) such that the following holds:

1. Two vertices $u, v \in V_{i-1}$ can be connected via edges of U_{i-1} if and only if there exists a path in $\Lambda[V_1 \cup \cdots \cup V_{i-1}]$ that connects u and v.
2. Two vertices $u, v \in V_{i+w}$ can be connected via edges of L_{i+w} if and only if there exists a path in $\Lambda[V_{i+w} \cup \cdots \cup V_d]$ that connects u and v.
3. The graph $\Lambda_i := \Lambda[V_{i-1} \cup \ldots \cup V_{i+w}] \cup U_{i-1} \cup L_{i+w}$ is planar and has radius at most $w+2$.

Furthermore, $\sum_{j=0}^{d-w+2} |U_j| + |L_j| \in O(n)$ and we can compute these sets in time $O(n)$.

Proof. For U_{i-1}, this is easy. If $i = 0, 1$ then V_{i-1} is empty and $U_{-1} = \emptyset$ works. Otherwise, pick an arbitrary vertex r_{i-1} in V_{i-1}, and let U_{i-1} be the edges that connect r_{i-1} to all other vertices of V_{i-1}. In consequence, all vertices of V_{i-1} can be connected within U_{i-1}, but this is appropriate since they can all be connected within the BFS-tree T, using only layers $1, \ldots, i-1$ of T. Graph $\Lambda[V_{i-1} \cup \cdots \cup V_{i+w}] \cup U_{i-1}$ is planar, because it can be obtained from the planar graph $\Lambda[V_1 \cup \cdots \cup V_{i+w}]$ by first contracting every vertex in layers $2, \ldots, i-2$ into its parent in T (yielding one super-node at the root), and then contracting this super-node into r_{i-1}.

For L_{i+w}, existence likewise is easy (and was argued by Eppstein [8]): Simply contract any edge of $\Lambda[V_{i+w} \cup \cdots \cup V_d]$ that has at least one endpoint not in V_{i+w}, and let L_{i+w} be the edges within V_{i+w} that remain at the end. However, it is not obvious how one could implement contraction in overall linear time; we give an alternate approach for this in the full version [2].

Since both U_{i-1} and L_{i+w} can be seen as obtained via contractions, graph Λ_i is planar. To prove the radius-bound, define r to be r_{i-1} if $i \geq 2$ and to be the root of T otherwise. Any vertex $v \in \Lambda_i$ has distance at most $w+2$ from r, because we can go upward from v in T at most $w+1$ times until we either reach a vertex v in V_{i-1} (which is $r=r_{i-1}$ or adjacent to it due to U_{i-1}), or $i \in \{0,1\}$ and we reach the root of T (which is r). ◁

An example of graph Λ_i is given in the full version [2].

Co-separating Triples in Λ_i. We continue to assume that we know $k \leq 7$ (and hence $w = 4k+2 \leq 30$). Crucial for the correctness of our search for a co-separating triple in $\Lambda(G)$ is that it suffices to search in Λ_i for all i.

▶ **Lemma 13.** *There exists a co-separating triple (A, X, B) of $\Lambda(G)$ with diameter k if and only if there exists an index $i \in \{0, \ldots, w-d+2\}$ and a co-separating triple (A_i, X, B_i) of Λ_i with diameter k for which $X \subseteq \Lambda[V_{i+1} \cup \cdots \cup V_{i+w-2}]$.*

Proof. Let (A, X, B) be a co-separating triple of $\Lambda(G)$. Since X has diameter at most $w-2$, all vertices of X lie in the layers $i+1, \ldots, i+w-2$ for some index i. Let A_i and B_i be subsets of A and B restricted to the vertices of Λ_i. We now show that (A_i, X, B_i) is a co-separating triple of Λ_i. Clearly these sets partition Λ_i. Condition 1 ('each set contains a G-vertex') clearly holds for X. To see that it holds for A_i, consider graph G in which $X \cap V(G)$ separates non-empty sets $A \cap V(G)$ and $B \cap V(G)$. Since G is connected, there exists an edge (a, x) with $a \in A \cap V(G)$ and $x \in X \cap V(G)$. This edge may or may not exist in $\Lambda(G)$, but if it does not then it got replaced by a-c-x with a dummy vertex c. So a has distance at most two from a vertex in $V_{i+1} \cup \cdots \cup V_{i+w-2}$ and hence belongs to $V_{i-1} \cup \cdots \cup V_{i+w}$, and so to Λ_i and to A_i. The argument is symmetric for B_i.

Now we argue Condition 2 ('no edges between A_i and B_i in Λ_i or G'). Fix two vertices $a \in A_i$ and $b \in B_i$. Since $A_i \subseteq A$ and $B_i \subseteq B$, there is no edge (a, b) in either $\Lambda(G)$ or G. So we are done unless (a, b) is an edge of U_{i-1} or L_{i+w}. Assume $(a, b) \in L_{i+w}$, the other case is similar. By Claim 12(2), there exists a path π in $\Lambda[V_{i+w} \cup \cdots \cup V_d]$ connecting a and b. No vertex of π belongs to X, and so the vertices of π either all belong to A or all belong to B since (A, X, B) is co-separating. This contradicts $a \in A$ and $b \in B$.

We now prove the other direction. Let (A_i, X, B_i) be a co-separating triple of some Λ_i with $X \subseteq \Lambda[V_{i+1} \cup \cdots \cup V_{i+w-2}]$. We define A and B as follows. Begin with all vertices in A_i and B_i, respectively; with this all vertices in Λ_i belong to one of A, X, B. Now consider any vertex v that does not belong to Λ_i, so either $v \in V_1 \cup \cdots \cup V_{i-2}$ or $v \in V_{i+w+1} \cup \cdots \cup V_d$. Assume the latter (the other case is similar), and let K be the component of $\Lambda[V_{i+w} \cup \cdots \cup V_d]$ that contains v. By Claim 12(2), there exists a component K' of graph (V_{i+w}, L_{i+w}) that contains exactly the vertices of $K \cap V_{i+w}$. The vertices of K' must either all be in A_i, or they must all be in B_i, because they are in layer V_{i+w} (so not in X) and they are connected via L_{i+w}. Assign v (and actually all vertices of K) to A if $V(K') \subseteq A_i$, and to B otherwise.

We now show that partition (A, X, B) is a co-separating triple of $\Lambda(G)$. Clearly Condition 1 holds since already A_i and B_i contain G-vertices. To show Condition 2, consider two vertices $a \in A$ and $b \in B$ and assume for contradiction that there is an edge (a, b) in either G or $\Lambda(G)$. This means that at least one of a, b is not in $V_{i-1} \cup \cdots \cup V_{i+w}$, else edge (a, b) would contradict that (A_i, X, B_i) was co-separating in Λ_i. Say $a \in V_{i+w+1} \cup \cdots \cup V_d$, all other cases are similar. With this it is impossible that (a, b) is an edge of $\Lambda(G)$: Such an edge would put a, b into the same component of $\Lambda[V_{i+w} \cup \cdots \cup V_d]$, but by construction of A and B we know that all vertices of such a component are put into the same set of A and B.

So (a, b) must be an edge of $G \setminus \Lambda(G)$, which means that it is crossed. Let c be the dummy vertex on (a, b). If $c \in A$ then (c, b) is an edge of $\Lambda(G)$ with endpoints in A and B, which we proved impossible already. Likewise $c \in B$ is impossible, so we must have $c \in X$. But then $c \in V_{i+1} \cup \cdots \cup V_{i+w-2}$, which puts its neighbour a into $V_i \cup \cdots \cup V_{i+w-1}$, a contradiction. ◀

Subroutine (To Find a Separating Set of Size k). We continue to assume that we know $k \leq 7$ (and hence $w = 4k + 2 \leq 30$). We also assume that edge-sets U_j and L_j have been computed already for all possible indices j, and that the edges of $\Lambda(G)$ have been split into $2d + 1$ sets $E_0, E_{0,1}, \ldots, E_{d-1,d}, E_d$ where E_j (for $j = 0, \ldots, d$) are all edges within layer V_j while $E_{j-1,j}$ (for $j = 1, \ldots, d$) are all edges connecting V_j to V_{j+1}. Perform the following for $i = 0, \ldots, d-w+2$:

1. Compute Λ_i. This takes time $O(|E(\Lambda_i)|)$ time: The vertices are $V_{i-1} \cup \cdots \cup V_{i+w}$, the edges are $U_{i-1} \cup E_{i-1,i} \cup E_i \cup \cdots \cup E_{i+w-1,i+w} \cup L_{i+w}$, and all these sets are pre-computed.
2. Since Λ_i is a planar graph with radius at most $w + 2$, it has treewidth $O(w)$ [8] and a corresponding tree decomposition $\mathcal{T}$ can be found in $O(|E(\Lambda_i)|)$ time. We may also assume that $\mathcal{T}$ has $O(|\Lambda_i|)$ bags.
3. We want to express Condition 2 of a co-separating triple as a condition in a single graph, and so define Λ_i^+ as follows: Begin with graph Λ_i, and add to it any edge (v, w) of G that is crossed (so is replaced in $\Lambda(G)$ by a path v-c-w via dummy vertex c) and for which v, w, c all belong to Λ_i.
4. Create a tree decomposition $\mathcal{T}^+$ of Λ_i^+ as follows. Begin with $\mathcal{T}$. For any bag Y and any dummy vertex $c \in Y$, add to Y all neighbours of c that belong to Λ_i. One can argue that this is a tree decomposition of Λ_i^+ of width $O(5w) = O(1)$, and can be computed in $O(|\mathcal{T}|) = O(|\Lambda_i|)$ time, see the full version [2].
5. Test whether Λ_i has a co-separating triple (A, X, B) for which X contains exactly k vertices of G and lies within $V_{i+1} \cup \cdots \cup V_{i+w-2}$. One can show (see the full version [2]) that this can be expressed in *monadic second-order logic*, using graph Λ_i^+ for defining adjacencies. By Courcelle's famous theorem [6], since Λ_i^+ has a tree decomposition of constant width, therefore the test can be done in $O(|\mathcal{T}^+|) = O(|\Lambda_i|)$ time.
6. If we find such a co-separating triple, then break (and output $X \cap V(G)$ as a separating set of size k), else try the next i.

The run-time for one index i is hence $O(|E(\Lambda_i)|)$. To bound the total run-time, we must bound $\sum_{i=0}^{d} |E(\Lambda_i)|$. Since each Λ_i uses $w+2$ consecutive layers, any edge of $\Lambda(G)$ belong to at most $w + 2$ sets in $E(\Lambda_0), \ldots, E(\Lambda_{d-w+1})$. Any edge in $U_0, \ldots, U_d, L_0, \ldots, L_d$ belongs to exactly one set in $E(\Lambda_0), \ldots, E(\Lambda_{d-w+1})$. Therefore $\sum_{i=0}^{d-w+1} |E(\Lambda_i)| \leq (w + 2)|E(\Lambda(G)| + \sum_{i=0}^{d-w+1}(|U_i| + |L_i|) \in O(w|E(\Lambda(G))|) + O(n)$, which by $w \in O(1)$ and $|E(\Lambda(G)| \in O(n)$ shows that the total run-time is linear.

The Final Algorithm. The algorithm for testing vertex-connectivity hence proceeds as follows. First pre-process G and duplicate edges to become kite edges where required. Then compute $\Lambda(G)$, the BFS-tree and the layers, and the edge-sets $E_j, E_{j,j+1}$, U_j and L_j for $j = 0, \ldots, d+2$. All this takes $O(n)$ time since $\Lambda(G)$ has $O(n)$ edges. For $k = 1, \ldots, 7$, run the sub-routine to test whether there exists a separating set of size k; this will necessarily find the minimum such set. Each such run takes time $O(n)$, and since there is a constant number of them the overall time is linear and Theorem 3 holds.

6 Outlook

In this paper, we showed that the vertex connectivity of a 1-plane graph G without $\times$-crossings can be computed in linear time. The main insight is that the distance (in an auxiliary graph) between any two vertices of a minimum separating set of G must be bounded. We close with some open questions. First, can we deal with $\times$-crossings?

▶ **Open problem 1.** *Can the vertex connectivity of an arbitrary 1-plane graph be computed in linear time?*

In our 'bad example' (Figure 1), *all* crossings were $\times$-crossings. As a first step towards Problem 1, could we at least compute the vertex connectivity in linear time if the number of $\times$-crossings is bounded by a constant?

Throughout the paper, we assumed that the input came with a fixed 1-planar embedding. We did this since testing 1-planarity is NP-hard [14]. However, it might be easier to test whether there exists a 1-planar embedding without $\times$-crossing; all the existing NP-hardness proofs of 1-planarity that we are aware of [14, 22, 1, 5] have $\times$-crossings in the 1-planar drawings.

▶ **Open problem 2.** *Is it NP-hard to test whether a given graph has a 1-planar drawing without $\times$-crossing?*

The crucial ingredient for our result was the structural property that vertices of a separating set are close in some sense. Are there similar structural properties for edge connectivity or bisections? Are there similar results for other classes of near-planar graphs?

References

1 Christopher Auer, Franz J Brandenburg, Andreas Gleißner, and Josef Reislhuber. 1-planarity of graphs with a rotation system. *Journal of Graph Algorithms and Applocations*, 19(1):67–86, 2015. doi:10.7155/jgaa.00347.

2 Therese Biedl and Karthik Murali. On computing the vertex connectivity of 1-plane graphs. *CoRR*, abs/2212.06782, 2022. doi:10.48550/arXiv.2212.06782.

3 Rainer Bodendiek, Heinz Schumacher, and Klaus Wagner. Bemerkungen zu einem Sechsfarbenproblem von G Ringel. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 53:41–52, 1983. doi:10.1007/BF02941309.

4 Hans Bodlaender and Arie Koster. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, 51(3):255–269, 2008. doi:10.1093/comjnl/bxm037.

5 Sergio Cabello and Bojan Mohar. Adding one edge to planar graphs makes crossing number and 1-planarity hard. *SIAM Journal on Computing*, 42(5):1803–1829, 2013. doi:10.1137/120872310.

6 Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and computation*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

7 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

8 David Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, 3(3):1–27, 1999. doi:10.7155/jgaa.00014.

9 Igor Fabrici, Jochen Harant, Tomás Madaras, Samuel Mohr, Roman Soták, and Carol T Zamfirescu. Long cycles and spanning subgraphs of locally maximal 1-planar graphs. *Journal of Graph Theory*, 95(1):125–137, 2020. doi:10.1002/jgt.22542.

10 Fedor V Fomin and Dimitrios M Thilikos. New upper bounds on the decomposability of planar graphs. *Journal of Graph Theory*, 51(1):53–81, 2006. doi:10.1002/jgt.20121.

11 Sebastian Forster, Danupon Nanongkai, Liu Yang, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Computing and testing small connectivity in near-linear time and queries via fast local cut algorithms. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 2046–2065. SIAM, 2020. doi:10.1137/1.9781611975994.126.

12 Harold N Gabow. Using expander graphs to find vertex connectivity. *Journal of the ACM*, 53(5):800–844, 2006. doi:10.1145/1183907.1183912.

13 Yu Gao, Jason Li, Danupon Nanongkai, Richard Peng, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Deterministic graph cuts in subquadratic time: Sparse, balanced, and k-vertex. *CoRR*, abs/1910.07950, 2019. arXiv:1910.07950.

14 Alexander Grigoriev and Hans L Bodlaender. Algorithms for graphs embeddable with few crossings per edge. *Algorithmica*, 49(1):1–11, 2007. doi:10.1007/s00453-007-0010-x.

15 Carsten Gutwenger and Petra Mutzel. A linear time implementation of SPQR-trees. In *Graph Drawing*, volume 1984 of *Lecture Notes in Computer Science*, pages 77–90. Springer, 2000. doi:10.1007/3-540-44541-2_8.

16 Monika R Henzinger, Satish Rao, and Harold N Gabow. Computing vertex connectivity: new bounds from old techniques. *Journal of Algorithms*, 34(2):222–250, 2000. doi:10.1006/jagm.1999.1055.

17 Seok-Hee Hong and Takeshi Tokuyama, editors. *Beyond Planar Graphs, Communications of NII Shonan Meetings*. Springer, 2020. doi:10.1007/978-981-15-6533-5.

18 John E Hopcroft and Robert Endre Tarjan. Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2(3):135–158, 1973. doi:10.1137/0202012.

19 Arkady Kanevsky and Vijaya Ramachandran. Improved algorithms for graph four-connectivity. *Journal of Computer and System Sciences*, 42(3):288–306, 1991. doi:10.1016/0022-0000(91)90004-O.

20 Daniel Kleitman. Methods for investigating connectivity of large graphs. *IEEE Transactions on Circuit Theory*, 16(2):232–233, 1969. doi:10.1109/TCT.1969.1082941.

21 Stephen Kobourov, Giuseppe Liotta, and Fabrizio Montecchiani. An annotated bibliography on 1-planarity. *Computer Science Review*, 25:49–67, 2017. doi:10.1016/j.cosrev.2017.06.002.

22 Vladimir P Korzhik and Bojan Mohar. Minimal obstructions for 1-immersions and hardness of 1-planarity testing. *Journal of Graph Theory*, 72(1):30–71, 2013. doi:10.1002/jgt.21630.

23 Jean-Paul Laumond. Connectivity of plane triangulations. *Information Processing Letters*, 34(2):87–96, 1990. doi:10.1016/0020-0190(90)90142-K.

24 Jason Li, Danupon Nanongkai, Debmalya Panigrahi, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Vertex connectivity in poly-logarithmic max-flows. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 317–329, 2021. doi:10.1145/3406325.3451088.

25 Nathan Linial, László Lovász, and Avi Wigderson. Rubber bands, convex embeddings and graph connectivity. *Combinatorica*, 8(1):91–102, 1988. doi:10.1007/BF02122557.

26 Karthik Murali. Testing vertex connectivity of bowtie 1-plane graphs. Master's thesis, David R. Cheriton School of Computer Science, University of Waterloo, 2022. Available at https://uwspace.uwaterloo.ca/.

27 Hiroshi Nagamochi and Toshihide Ibaraki. A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica*, 7(5&6):583–596, 1992. doi:10.1007/BF01758778.

28 Gerhard Ringel. Ein Sechsfarbenproblem auf der Kugel. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 29(1):107–117, 1965. doi:10.1007/BF02996313.

29 Robert E Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972. doi:10.1137/0201010.

Fault-Tolerant ST-Diameter Oracles

Davide Bilò
Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Italy

Keerti Choudhary
Department of Computer Science and Engineering, Indian Institute of Technology Delhi, India

Sarel Cohen
School of Computer Science, Tel-Aviv-Yaffo Academic College, Israel

Tobias Friedrich
Hasso Plattner Institute, Universität Potsdam, Germany

Simon Krogmann
Hasso Plattner Institute, Universität Potsdam, Germany

Martin Schirneck
Faculty of Computer Science, Universität Wien, Austria

Abstract

We study the problem of estimating the ST-diameter of a graph that is subject to a bounded number of edge failures. An *f-edge fault-tolerant ST-diameter oracle* (f-FDO-ST) is a data structure that preprocesses a given graph G, two sets of vertices S, T, and positive integer f. When queried with a set F of at most f edges, the oracle returns an estimate $\widehat{D}$ of the ST-diameter $\operatorname{diam}(G-F, S, T)$, the maximum distance between vertices in S and T in $G-F$. The oracle has stretch $\sigma \geqslant 1$ if $\operatorname{diam}(G-F, S, T) \leqslant \widehat{D} \leqslant \sigma \operatorname{diam}(G-F, S, T)$. If S and T both contain all vertices, the data structure is called an *f-edge fault-tolerant diameter oracle* (f-FDO). An *f-edge fault-tolerant distance sensitivity oracles* (f-DSO) estimates the pairwise graph distances under up to f failures.

We design new f-FDOs and f-FDO-STs by reducing their construction to that of *all-pairs* and *single-source* f-DSOs. We obtain several new tradeoffs between the size of the data structure, stretch guarantee, query and preprocessing times for diameter oracles by combining our black-box reductions with known results from the literature.

We also provide an information-theoretic lower bound on the space requirement of approximate f-FDOs. We show that there exists a family of graphs for which any f-FDO with sensitivity $f \geqslant 2$ and stretch less than $5/3$ requires $\Omega(n^{3/2})$ bits of space, regardless of the query time.

2012 ACM Subject Classification Theory of computation → Shortest paths; Theory of computation → Data structures design and analysis; Theory of computation → Cell probe models and lower bounds

Keywords and phrases diameter oracles, distance sensitivity oracles, space lower bounds, fault-tolerant data structures

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.24

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2305.03697

1 Introduction

The diameter, i.e., the largest distance between any two vertices, is one of the most fundamental graph parameters for it measures how fast information can spread in a network. The problem of approximating the diameter of a given graph in a time-efficient manner has been extensively studied [1, 3, 4, 25, 26, 27, 43, 44, 45]. Here, we investigate the diameter computation problem in the fault-tolerant model. The interest in this setting stems from the

fact that most real-world networks are prone to errors. These failures, though unpredictable, are transient due to some simultaneous repair process that is undertaken in these applications. This has motivated the research on designing fault-tolerant oracles for various graph problems in the past decade. An *f-edge/vertex fault-tolerant oracle* is a compact data structure that can quickly report the desired solution or graph property of the network on occurrence of up to f edge/vertex failures. The parameter f that describes the degree of robustness against errors is known as the *sensitivity* of the oracle. A lot of work has been done in designing fault-tolerant structures for various problems like connectivity [20, 32, 33], finding shortest paths [2, 12, 36], and distance sensitivity oracles [5, 7, 14, 24, 30, 31, 34, 47].

While the fault-tolerant model has been studied a lot for distances, the landscape of fault-tolerant diameter oracles is far less explored. For a given graph $G = (V, E)$ and two sets $S, T \subseteq V$ of vertices, an *f-edge fault-tolerant diameter oracle* (f-FDO-ST) is a data structure that stores information about G after a preprocessing step. When queried with a set F of at most f edges, the oracle returns an upper bound of the ST-diameter $\operatorname{diam}(G - F, S, T) = \max_{s \in S, t \in T} d_{G-F}(s, t)$ of $G - F$. This is the maximum among all s-t-distances for $s \in S$ and $t \in T$ under the condition that none of the shortest paths can use an edge in the query set F. We say that the oracle has a stretch of $\sigma \geqslant 1$ if the value $\widehat{D}$ returned upon query F satisfies $\operatorname{diam}(G - F, S, T) \leqslant \widehat{D} \leqslant \sigma \operatorname{diam}(G - F, S, T)$. When $S = T = V$, the data structure is called an *f-edge fault-tolerant diameter oracle* (f-FDO).

The problem of designing f-FDOs was originally raised by Henzinger, Lincoln, Neumann, and Vassilevska Williams [40] and has recently been studied by Bilò, Cohen, Friedrich, and Schirneck [17] and the same authors together with Choudhary [15], see also Section 1.1.

The problem of designing f-FDO-ST can be seen as a generalisation of the Bichromatic Diameter, a problem in which the two sets S and T form a partition of V. The latter problem is motivated by several related, well-studied problems in computational geometry, e.g., Bichromatic Diameter on point sets (commonly known as Bichromatic Farthest Pair), where one seeks to determine the farthest pair of points in a given set of points in space. The problem of Bichromatic Diameter was studied by Dalirrooyfard, Vassilevska Williams, Vyas, and Wein [28].

Given the plethora of work on distance oracles and the close connection between the distance and the diameter problem, a natural question is if we can convert the results on distance computation under failures into analogous oracles for the diameter without sacrificing much on our performance parameters.

▶ **Question.** *Are there black-box reductions from fault-tolerant diameter oracles to fault-tolerant distance oracles without considerable overhead in stretch, query time, and space?*

In this work, we present several such reductions and, from them, conclude trade-offs between the space, stretch, preprocessing, and query time for diameter oracles. In more detail, our techniques for obtaining upper bounds is by presenting reductions to the problem of constructing *f-edge fault-tolerant distance sensitivity oracles* (f-DSOs) in two widely studied categories. The *all-pairs* variant can be queried with any pair of vertices $s, t \in V$ and set $F \subseteq E$ of f failures and reports (an estimate) of the distance $d_{G-F}(s, t)$ between s and t in $G - F$. In the *single-source* variant, the source s is fixed and the set of allowed queries consists of the target vertices t together with a set F of failures.

For the regular diameter ($S = T = V$), we provide two theorems showing that both all-pairs and single-source f-DSOs can be used to construct f-FDOs.

Table 1 Properties of the f-FDOs obtained via Theorem 1 using all-pairs f-DSOs from the literature. The applicable graph class (un-/directed, un-/weighted) is determined by the f-DSO. W denotes the maximum edge weight for graphs with arbitrary positive weights, M is the maximum edge weight for integer weighted graphs. The parameter $k \geqslant 1$ is a positive integer, $\varepsilon > 0$ a positive real, $\alpha \in [0, 1]$ is a real number in the unit interval, and $\omega < 2.37286$ denotes the matrix multiplication exponent.

Sensitivity	Stretch	Space	Query time	Preprocessing Time	Ref.
1	2	$\widetilde{O}(n^2)$	$O(1)$	$\widetilde{O}(mn)$	[10, 11]
1	2	$\widetilde{O}(n^2)$	$O(1)$	$\widetilde{O}(n^{2.5794}M + mn)$	[37]
1	$1 + (2k-1)(1+\varepsilon)$	$\widetilde{O}(k^5 n^{1+1/k}\varepsilon^{-4})$	$O(k)$	$O(kmn^{1+1/k})$	[8]
2	2	$\widetilde{O}(n^2)$	$\widetilde{O}(1)$	$\mathsf{poly}(n)$	[32]
$f = o(\frac{\log n}{\log\log n})$	2	$\widetilde{O}(n^{3-\alpha})$	$\widetilde{O}(f^2 n^{2-(1-\alpha)/f})$	$O(n^{\omega+1-\alpha}M)$	[47]
$f = o(\frac{\log n}{\log\log n})$	$2+\varepsilon$	$O(fn^{2+o(1)}(\log W)\varepsilon^{-f})$	$\widetilde{O}(f^7 \log\log W)$	$O(fn^{5+o(1)}(\log W)\varepsilon^{-f})$	[23]
$f \geqslant 1$	2	$O(fn^4)$	$f^{O(f)}$	$n^{O(f)}$	[34]
$f \geqslant 1$	2	$O(n^{2+\alpha}M)$	$\widetilde{O}(f^4 n^{2-\alpha}M + f^{2+\omega}nM)$	$\widetilde{O}(n^{\omega+(3-\omega)\alpha}M + mn)$	[18]
$f \geqslant 1$	$1 + (8k-2)(f+1)$	$O\left(fkn^{1+1/k}\log(nW)\right)$	$\widetilde{O}(f^3)$	$\mathsf{poly}(n)$	[24]

▶ **Theorem 1.** *Let G be a (undirected or directed) graph with n vertices, m edges, and possibly positive edge weights. Given access to an f-DSO $\mathcal{D}$ for G with stretch $\sigma \geqslant 1$, preprocessing time $\mathtt{P}$, space $\mathtt{S}$, and query time $\mathtt{Q}$, one can construct an f-FDO for G with stretch $1+\sigma$, preprocessing time $O(\mathtt{P} + mn\log n)$, space $O(\mathtt{S})$, and query time $O(f^2\mathtt{Q})$.*

In Section 1.2, we review existing all-pairs f-DSOs. By applying the reduction stated in Theorem 1 we obtain new f-FDOs as listed in Table 1.

The following theorem shows how we can use the single-source variant of distance sensitivity oracles to construct f-FDOs.

▶ **Theorem 2.** *Let G be a (undirected or directed) graph with n vertices, m edges, and possibly positive edge weights. Given access to a single-source f-DSO $\mathcal{D}$ for G with stretch $\sigma \geqslant 1$, preprocessing time $\mathtt{P}$, space $\mathtt{S}$, and query time $\mathtt{Q}$, one can construct an f-FDO for G with stretch $2(1+\sigma)$, preprocessing time $O(\mathtt{P})$, space $O(\mathtt{S})$, and query time $O(f\mathtt{Q})$.*

Section 1.3 discusses single-source f-DSOs from the literature. Together with Theorem 2 they give new f-FDOs, summarized in Table 2.

The main technical contribution of this work, however, is a novel fault-tolerant data structure for the more general ST-diameter problem that was introduced and studied in recent years. For example, Backurs, Roditty, Segal, Vassilevska Williams, and Wein [4] proved that for any undirected graph one can compute a 3-approximation of the ST-diameter in $O(mn)$ time. They also provided a randomized algorithm that computes a 2-approximation of the ST-diameter in $\widetilde{O}(m\sqrt{n})$ time.[1] Dalirrooyfard, Vassilevska Williams, Vyas, and Wein [28] studied the problem of computing the bi-chromatic ST-diameter, the special case of ST-diameter problem where the sets S and T form a partition of V. Similar to f-FDOs, we explore the problem of designing compact oracles that report the ST-diameter of a graph after occurrences of up to f failures. We present reductions between f-DSOs and f-FDO-STs, as stated in the following theorem. To the best of our knowledge, our paper is the first work that provides some results on f-FDO-STs, for general values of f.

[1] For a non-negative function $g(n, m, f)$, we write $\widetilde{O}(g)$ for $O(g \cdot \mathsf{polylog}(n))$.

Table 2 Properties of the f-FDOs obtained via Theorem 2 using single-source f-DSOs from the literature. The applicable graph class (un-/directed, un-/weighted) is determined by the single-source f-DSO. W denotes the maximum edge weight for graphs with arbitrary positive weights, M is the maximum edge weight for integer weighted graphs. The parameter $\varepsilon > 0$ is a positive real and $\omega < 2.37286$ denotes the matrix multiplication exponent.

Sensitivity	Stretch	Space	Query time	Preprocessing Time	Ref.
1	4	$\widetilde{O}(n^{3/2})$	$\widetilde{O}(1)$	$\widetilde{O}(mn^{1/2}+n^2)$	[16, 38]
1	4	$\widetilde{O}(n^{3/2}M^{1/2})$	$\widetilde{O}(1)$	$\widetilde{O}(n^{\omega}M)$	[16]
1	$4+\varepsilon$	$\widetilde{O}(n(\log W)\varepsilon^{-1})$	$O(\log\log_{1+\varepsilon}(nW))$	$\mathsf{poly}(n)$	[5, 8, 13]
1	6	$O(n)$	$O(1)$	$\widetilde{O}(mn)$	[13]
$f \geqslant 1$	$4f+4$	$\widetilde{O}(fn)$	$\widetilde{O}(f^3)$	$\widetilde{O}(fm)$	[14]

▶ **Theorem 3.** *Let $G=(V,E)$ be an undirected graph with n vertices, m edges, and possibly positive edge weights. Let $S,T \subseteq V$ be two non-empty sets. Given access to an f-DSO for G with stretch $\sigma \geqslant 1$, preprocessing time $\mathtt{P}$, space $\mathtt{S}$, and query time $\mathtt{Q}$, one can compute an f-FDO-ST for G with preprocessing time $\mathtt{P}+\widetilde{O}(mn+n|S||T|)$ and stretch $1+3\sigma$. Additionally, the f-FDO-ST has the following properties.*

- *If the sensitivity is $f=o(\log n)$, the oracle requires $\mathtt{S}+O(n^{3/2}(2^f+\log n))$ space and has a query time of $O(f^2(2^f+\mathtt{Q}))$.*
- *If $f=\Omega(\log n)$, the oracle requires $\mathtt{S}+O(n^2)$ space and has a query time of $O(f^2(f+\mathtt{Q}))$.*

Some more remarks on the preprocessing time stated in Theorem 3 may be in order. The reduction itself takes time $\mathtt{P}+O(mn+n^2\log n+n|S||T|)$ to compute but requires that the shortest paths in G are unique. The total preprocessing time depends on how this condition is achieved. It is always possible to guarantee unique shortest paths either by randomly perturbing the edge weights with sufficiently small values, see [41], or by using a more complex deterministic method, also known as *lexicographic perturbation* [19, 21, 39]. While the first method increases the preprocessing only by a constant factor, it makes the preprocessing procedure randomized. Lexicographic perturbation, in turn, increases the time by an additive $O(mn+n^2\log^2 n)$ term [19]. By applying the reduction stated in Theorem 3 to existing all-pairs f-DSOs we obtain the f-FDOs listed in Table 3.

In addition, we present improved constructions of f-FDO-STs for the important case of a single source or target, i.e., when $|S|=1$ or $|T|=1$, or when one is only given access to single-source f-DSOs. In the following, for the sake of readability, when $S=\{s\}$, we will use "sT-diameter" instead of "ST-diameter" or "$\{s\}T$-diameter", same for the oracles.

▶ **Theorem 4.** *Let $G=(V,E)$ be an undirected graph with n vertices, m edges, and possibly positive edge weights. Let $s \in V$ be a vertex and $T \subseteq V$ a non-empty set. Given a single-source f-DSO for G with preprocessing time $\mathtt{P}$, space $\mathtt{S}$, query time $\mathtt{Q}$, and stretch σ, one can compute an f-FDO-sT for G with preprocessing time $\mathtt{P}+O(m+n\log n)$, space $\mathtt{S}+O(n)$, query time $O(f^2+f\mathtt{Q})$, and stretch $1+2\sigma$. For unweighted graphs, the preprocessing time can be improved to $\mathtt{P}+O(m)$.*

Table 4 shows the f-fault-tolerant sT-diameter-oracle obtained from Theorem 4.

Table 3 Properties of the f-FDO-ST for undirected graphs obtained via Theorem 3 using all-pairs f-DSOs from the literature. The preprocessing time is omitted due to space reasons. W denotes the maximum edge weight for graphs with arbitrary positive weights, M is the maximum edge weight for integer weighted graphs. The parameter $k \geqslant 1$ is a positive integer, $\varepsilon > 0$ a positive real, $\alpha \in [0, 1]$ is a real number in the unit interval, and $\omega < 2.37286$ denotes the matrix multiplication exponent.

Sensitivity	Stretch	Space	Query time	Ref.
1	4	$\widetilde{O}(n^2)$	$O(1)$	[10, 11, 37]
1	$1+(6k-3)(1+\varepsilon)$	$\widetilde{O}(n^{3/2}+k^5n^{1+1/k}\varepsilon^{-4})$	$O(1)$	[8]
2	4	$\widetilde{O}(n^2)$	$\widetilde{O}(1)$	[32]
$f=o(\frac{\log n}{\log\log n})$	4	$\widetilde{O}(n^{3-\alpha})$	$\widetilde{O}(f^2n^{2-(1-\alpha)/f})$	[47]
$f=o(\frac{\log n}{\log\log n})$	$4+\varepsilon$	$O(fn^{2+o(1)}(\log W)\varepsilon^{-f})$	$\widetilde{O}(f^2 2^f + f^7 \log\log W)$	[23]
$f=o(\log n)$	4	$O(n^{2+\alpha}M)$	$\widetilde{O}(f^4n^{2-\alpha}M + f^{2+\omega}nM)$	[18]
$f=o(\log n)$	4	$O(fn^4)$	$f^{O(f)}$	[34]
$f=o(\log n)$	$1+(24k-6)(f+1)$	$O(n^{3/2+o(1)}+fkn^{1+1/k}\log(nW))$	$\widetilde{O}(f^2 2^f)$	[24]

Table 4 Properties of the f-FDO-sT for undirected graphs obtained via Theorem 4 using single-source f-DSOs from the literature.

Sensitivity	Stretch	Space	Query time	Preprocessing Time	Ref.
1	3	$\widetilde{O}(n^{3/2})$	$\widetilde{O}(1)$	$\widetilde{O}(mn^{1/2}+n^2)$	[16, 38]
1	3	$\widetilde{O}(n^{3/2}M^{1/2})$	$\widetilde{O}(1)$	$\widetilde{O}(n^\omega M)$	[16]
1	$3+\varepsilon$	$\widetilde{O}(n(\log W)\varepsilon^{-1})$	$O(\log\log_{1+\varepsilon}(nW))$	$\mathsf{poly}(n)$	[5, 8, 13]
1	5	$O(n)$	$O(1)$	$\widetilde{O}(mn)$	[13]
$f \geqslant 1$	$4f+3$	$\widetilde{O}(fn)$	$\widetilde{O}(f^3)$	$\widetilde{O}(fm)$	[14]

▶ **Theorem 5.** *Let $G=(V,E)$ be an undirected graph with n vertices, m edges, and possibly positive edge weights. Let S, T be two non-empty subsets of V. Given a single-source f-DSO for G with preprocessing time $\mathtt{P}$, space $\mathtt{S}$, query time $\mathtt{Q}$, and stretch σ, one can compute an f-FDO-ST for G with preprocessing time $O(\mathtt{P}+m+n\log n)$, space $O(\mathtt{S}+n)$, query time $O(f^2+f\mathtt{Q})$, and stretch $2+5\sigma$. For unweighted graphs, the preprocessing time can be improved to $O(\mathtt{P}+m)$*

Table 5 corresponds to the oracles obtained via Theorem 5.

We also prove an information-theoretic lower bound on the space requirement of approximate f-FDOs that support $f \geqslant 2$ edge failures. Note that the lower bound in Theorem 6 holds independently of the query time. It is known from work of Bilò, Cohen, Friedrich, and Schirneck [17] that f-FDOs with stretch $\sigma < 1.5$ require $\Omega(n^2)$ bits of space, and in our work we complement this result by proving that f-FDOs with stretch $\sigma < 5/3$ require $\Omega(n^{1.5})$ bits of space. Obtaining $\Omega(n^2)$ lower bound for f-FDOs with stretch $\sigma < 2$ for undirected unweighted graphs is an interesting open problem.

▶ **Theorem 6.** *Let n be a positive integer. Any f-FDO or f-FDO-ST for n-vertex graphs with sensitivity $f \geqslant 2$ and stretch $\frac{5}{3}-\varepsilon$ for any $\varepsilon > 0$ requires $\Omega(n^{3/2})$ bits of space.*

Table 5 Properties of the fault-tolerant ST-diameter oracles (f-FDO-ST) obtained via the reduction in Theorem 5 using single-source distance sensitivity oracles (f-DSOs) from the literature. W denotes the maximum edge weight for graphs with arbitrary positive weights, M is the maximum edge weight for integer weighted graphs. The parameter $\varepsilon > 0$ is a positive real and $\omega < 2.37286$ denotes the matrix multiplication exponent.

Sensitivity	Stretch	Space	Query time	Preprocessing Time	Ref.
1	7	$\widetilde{O}(n^{3/2})$	$\widetilde{O}(1)$	$\widetilde{O}(mn^{1/2}+n^2)$	[16, 38]
1	7	$\widetilde{O}(n^{3/2}M^{1/2})$	$\widetilde{O}(1)$	$\widetilde{O}(n^{\omega}M)$	[16]
1	$7+\varepsilon$	$\widetilde{O}(n(\log W)\varepsilon^{-1})$	$O(\log\log_{1+\varepsilon}(nW))$	$\mathsf{poly}(n)$	[5, 8, 13]
1	12	$O(n)$	$O(1)$	$\widetilde{O}(mn)$	[13]
$f \geqslant 1$	$10f+7$	$\widetilde{O}(fn)$	$\widetilde{O}(f^3)$	$\widetilde{O}(fm)$	[14]

Outline. This work is structured as follows. In the remainder of this section, we review the literature focusing on diameter oracles and distance sensitivity oracles. We then fix our notations and some preliminaries in Section 2. Section 3 presents our constructions of f-FDO-ST, for the general case of $S, T \subseteq V$. In Section 4 we consider the special case of a single source, that is, of f-FDO-sT. In Section 5 we prove the space lower bound. The proofs of Theorems 1 and 2 follow from similar ideas as discussed in Section 3 and are deferred to the full version of the paper.

1.1 Related Work on Fault-Tolerant Diameter Oracles

Fault-tolerant diameter oracles were introduced by Henzinger, Lincoln, Neumann, and Vassilevska Williams [40]. They showed that for a single failure in unweighted directed graphs, one can compute in time $\widetilde{O}(mn + n^{1.5}\sqrt{Dm/\varepsilon})$, where $\varepsilon \in (0,1]$ and D is the diameter of the graph, a 1-FDO with $1+\varepsilon$ stretch that has $O(m)$ space, constant query time. Bilò, Cohen, Friedrich, and Schirneck [17] showed that one can improve the preprocessing time to $\widetilde{O}(mn + n^2/\varepsilon)$, which is nearly optimal under certain conditional hardness assumptions for combinatorial algorithms (see [40]). They also showed that fast matrix multiplication reduces the preprocessing time for dense graphs to $\widetilde{O}(n^{2.5794} + n^2/\varepsilon)$.

Bilò, Choudhary, Cohen, Friedrich, and Schirneck [15] addressed the problem of computing 1-FDOs with $o(m)$ space. They showed that for unweighted directed graphs with diameter $D = \omega(n^{5/6})$, there is a 1-FDO with $\widetilde{O}(n)$ space, $1 + \frac{n^{5/6}}{D} = 1 + o(1)$ stretch, and $O(1)$ query time. It has a preprocessing time of $O(mn)$. In the same work it was also shown that for graphs with diameter $D = \omega((n^{4/3}\log n)/(\varepsilon\sqrt{m}))$ and any $\varepsilon > 0$, there is a $(1+\varepsilon)$-stretch 1-FDO, with preprocessing time $O(mn)$, space $o(m)$, and constant query time.

For *undirected* graphs the space requirement can be reduced. There is a folklore construction that combines the DSO by Bernstein and Karger [11] with the observation that in undirected graphs the eccentricity of an arbitrary vertex is a 2-approximation of the diameter. This results in an 1-FDO with stretch 2 and constant query time that takes only $O(n)$ space, details can be found in [17, 40].

For $f > 1$ edge failures in undirected graphs with non-negative edge weights, Bilò et al. [17] presented an f-FDO with $(f+2)$ stretch, $O(f^2\log^2 n)$ query time, $\widetilde{O}(fn)$ space, and $\widetilde{O}(fm)$ preprocessing time. A lower bound in that work showed that f-FDO with finite stretch must have $\Omega(fn)$ space, nearly matching their construction.

Table 6 Existing f-sensitive all-pairs distance oracles for undirected graphs. The parameter $k \geqslant 1$ is a positive integer, $\varepsilon > 0$ a positive real, $\alpha \in [0,1]$ is a real number in the unit interval, and $\omega < 2.37286$ denotes the matrix multiplication exponent.

Sensitivity	Stretch	Space	Query time	Preprocessing Time	Ref.
1	1	$\widetilde{O}(n^2)$	$O(1)$	$\widetilde{O}(mn)$	[10, 11]
1	1	$\widetilde{O}(n^2)$	$O(1)$	$\widetilde{O}(n^{2.5794}M)$	[37]
1	$(2k-1)(1+\varepsilon)$	$\widetilde{O}(k^5 n^{1+1/k}\varepsilon^{-4})$	$O(k)$	$O(kmn^{1+1/k})$	[8]
2	1	$\widetilde{O}(n^2)$	$\widetilde{O}(1)$	$\mathsf{poly}(n)$	[32]
$f = o(\frac{\log n}{\log\log n})$	1	$\widetilde{O}(n^{3-\alpha})$	$\widetilde{O}(n^{2-(1-\alpha)/f})$	$O(n^{\omega+1-\alpha}M)$	[47]
$f = o(\frac{\log n}{\log\log n})$	$1+\varepsilon$	$O(fn^{2+o(1)}(\log W)\varepsilon^{-f})$	$\widetilde{O}(f^5 \log\log W)$	$O(fn^{5+o(1)}(\log W)\varepsilon^{-f})$	[23]
$f \geqslant 1$	1	$O(fn^4)$	$f^{O(f)}$	$n^{O(f)}$	[34]
$f \geqslant 1$	1	$O(n^{2+\alpha}M)$	$\widetilde{O}(f^2 n^{2-\alpha}M + f^\omega nM)$	$\widetilde{O}(n^{\omega+(3-\omega)\alpha}M)$	[18]
$f \geqslant 1$	$(8k-2)(f+1)$	$O\left(fkn^{1+1/k}\log(nW)\right)$	$\widetilde{O}(f^3)$	$\mathsf{poly}(n)$	[24]

We are not aware of any $O(n)$-sized, constant-stretch FDOs for *directed* graphs with arbitrary diameter in the literature prior to this work, not even for sensitivity ($f = 1$). Also, no non-trivial f-FDOs with $o(f)$-stretch were known. To the best of our knowledge, we are the first to study the problem of general f-FDO-STs with $S, T \neq V$.

We now discuss the known information-theoretic lower bounds for FDOs. Bilò, Cohen, Friedrich, Schirneck [17] showed that any FDO with stretch $\sigma < 3/2$ for undirected unweighted graphs requires $\Omega(m)$ bits of space, even for $f = 1$. They also extended the same lower bound of $\Omega(m)$ bits to edge-weighted graphs and $\sigma < 2$. Bilò, Choudhary, Cohen, Friedrich, and Schirneck [15] extended this result to directed graphs. In particular, they showed that for directed unweighted graphs with diameter $D = O(\sqrt{n}/m)$, any FDO with stretch better than $\left(\frac{3}{2} - \frac{1}{D}\right)$ requires $\Omega(m)$ bits of space. They further proved that for directed graphs any f-FDO requires $\Omega(2^{f/2}n)$ bits of space, as long as $2^{f/2} = O(n)$.

1.2 All-Pairs Distance Sensitivity Oracles

The first distance-sensitive oracle was in the context of directed graphs [29]. It maintained exact distances and was capable of handling a single edge failure. The space requirement of this oracle is $O(n^2 \log n)$ and its query time is $O(\log n)$. This was later generalized to handle a single vertex or edge failure in [30]. Demetrescu, Thorup, Chowdhury, and Ramachandran [30] presented an exact 1-sensitive distance oracle of size $O(n^2 \log n)$, $O(1)$ query time and $\widetilde{O}(mn^2)$ preprocessing time. Later, in two consecutive papers, Bernstein and Karger improved the preprocessing time (while keeping the space and query time unchanged), first to $O(n^2\sqrt{m})$ in [10] and then to $\widetilde{O}(mn)$ in [11]. Baswana and Khanna [8] considered approximate 1-DSOs for unweighted graphs. More precisely, they presented a data structure of size $O(k^5 n^{1+1/k}\frac{\log^3 n}{\varepsilon^4})$, $(2k-1)(1+\varepsilon)$ stretch and $O(k)$ query time. Duan and Pettie [32] considered the case of two failures (vertices or edges) with exact distances. The size of their oracle is $O(n^2 \log^3 n)$, the query time is $O(\log n)$ and the construction time is polynomial.

Using fast matrix multiplication, Weimann and Yuster [47] presented, for any parameter $\alpha \in [0,1]$, a DSO that can handle up to $O(\log n/\log\log n)$ edges or vertices failures with $\widetilde{O}(n^{2-(1-\alpha)/f})$ query time and $O(Mn^{\omega+1-\alpha})$ preprocessing time for directed graphs with integer weights in the range $[-M, M]$, where $\omega < 2.373$ is the matrix multiplication exponent. In [35], Grandoni and Vassilevska Williams presented a distance sensitivity oracle with

Table 7 Existing f-sensitive single-source distance oracles for undirected graphs W denotes the maximum edge weight for graphs with arbitrary positive weights, M is the maximum edge weight for integer weighted graphs. The parameter $\varepsilon > 0$ is a positive real and $\omega < 2.37286$ denotes the matrix multiplication exponent.

Sensitivity	Stretch	Space	Query time	Preprocessing Time	Ref.
1	1	$\widetilde{O}(n^{3/2})$	$\widetilde{O}(1)$	$\widetilde{O}(mn^{1/2}+n^2)$	[16, 38]
1	1	$\widetilde{O}(n^{3/2}M^{1/2})$	$\widetilde{O}(1)$	$\widetilde{O}(n^{\omega}M)$	[16]
1	$1+\varepsilon$	$\widetilde{O}(n(\log W)\varepsilon^{-1})$	$O(\log\log_{1+\varepsilon}(nW))$	$\mathsf{poly}(n)$	[5, 8, 13]
1	2	$O(n)$	$O(1)$	$\widetilde{O}(mn)$	[13]
$f \geqslant 1$	$2f+1$	$\widetilde{O}(fn)$	$\widetilde{O}(f^2)$	$\widetilde{O}(fm)$	[14]

subcubic $\widetilde{O}(Mn^{\omega+1/2}+Mn^{\omega+\alpha(4-\omega)})$ preprocessing time and sublinear $\widetilde{O}(n^{1-\alpha})$ query time. Van den Brand and Saranurak [18] presented a distance-sensitive oracle that can handle $f \geqslant \log n$ updates (where an update is an edge insertion or deletion), with $\widetilde{O}(Mn^{\omega+(3-\omega)\mu})$ preprocessing time, $\widetilde{O}(Mn^{2-\mu}f^2+Mnf^{\omega})$ update time, and $\widetilde{O}(Mn^{2-\mu}f+Mnf^2)$ query time, where the parameter $\mu \in [0,1]$ can be chosen. Chechik and Cohen [22] presented a 1-DSO with with subcubic $\widetilde{O}(Mn^{2.873})$ preprocessing time and $\widetilde{O}(1)$ query time. This was improved by Ren [42] and later by Gu and Ren [37], who obtained a 1-DSO with $\widetilde{O}(Mn^{2.5794})$ preprocessing time and constant query time. Recently Duan and Ren [34] presented an exact f-DSO with $O(fn^4)$ space, $f^{O(f)}$ query time, and $n^{O(f)}$ preprocessing time.

In Table 6 we summarize several of the above f-DSOs for undirected graphs.

1.3 Related Work on Single-Source Distance Sensitivity Oracles

First, we discuss undirected graphs. Baswana and Khanna [8] showed that unweighted undirected graphs can be preprocessed in $O(m\sqrt{n/\varepsilon})$ time to compute a $(1+\varepsilon)$-stretch single-source edge/vertex fault-tolerant distance-oracle of size $O(n\log n+n/\varepsilon^3)$ and constant query time. For weighted graphs, they showed the construction of an $O(n\log n)$ size oracle which can report 3-approximate distances on single failure in $O(1)$ time. Bilò, Gualà, Leucci, and Proietti [13] showed that for a single edge failure in weighted graphs we can compute an $O(n)$-size oracle with stretch 2 and constant query time. Also, a construction is provided that has $1+\varepsilon$ stretch, with $O\big(\varepsilon^{-1}n\log(1/\varepsilon)\big)$ size and $O\big(\varepsilon^{-1}\log n\log(1/\varepsilon)\big)$ query time. All the results stated till now are for a single edge or vertex failure only. For multiple failures, Bilò, Gualà, Leucci, and Proietti [14] gave a construction of size $O(fn\log^2 n)$, computable in $\widetilde{O}(mf)$ time that reports $(2f+1)$-stretched distances in $O(f^2\log^2 n)$ time.

Bilò, Cohen, Friedrich, and Schirneck [16] presented several additional single-source DSOs. For undirected unweighted graphs, they presented a single-source DSO that has size $O(n^{3/2})$, query time $\widetilde{O}(1)$ and $\widetilde{O}(m\sqrt{n}+n^2)$ preprocessing time. For graphs with integer edge weights in the range $[1,M]$ and using fast matrix multiplication, they presented a single-source DSO with $O(M^{1/2}n^{3/2})$ space, $\widetilde{O}(1)$ query time and $\widetilde{O}(Mn^{\omega})$ preprocessing time. For sparse graphs with $m=O(M^{3/7}n^{7/4})$ they presented a single-source DSO with the same size, $\widetilde{O}(1)$ query time, and subquadratic $\widetilde{O}(M^{7/8}m^{1/2}n^{11/8})$ preprocessing time.

For directed graphs, Baswana, Choudhary, Hussain, and Roditty [5] showed that we can preprocess directed weighted graphs with edge weights in range $[1,W]$ to compute an oracle of $\widetilde{O}(\varepsilon^{-1}n\log W)$ size that reports $(1+\varepsilon)$-approximate distances on single edge/vertex

failure in $\widetilde{O}(\log \log_{1+\varepsilon}(nW))$ time. Gupta and Singh [38] designed exact distance oracles of $\widetilde{O}(n^{3/2})$ size that on single edge/vertex failure in directed/undirected unweighted graphs reports distances in $\widetilde{O}(1)$ time. In Table 7 we summarize several of the above f-DSOs for undirected graphs.

2 Preliminaries

For a given graph $G = (V, E)$, possibly with positive edge weights, we denote by $d_G(u, v)$ the distance in G from vertex $u \in V$ to vertex $v \in V$. Given two non-empty subsets $S, T \subseteq V$, the *ST-diameter* of G is defined as $\mathrm{diam}(G, S, T) = \max_{s \in S, t \in T} d_G(s, t)$. With a little abuse of notation, when $S = \{s\}$ (resp., $T = \{t\}$), we also use $\mathrm{diam}(G, s, T)$ (resp., $\mathrm{diam}(G, S, t)$) as a shorthand of $\mathrm{diam}(G, \{s\}, T)$ (resp., $\mathrm{diam}(G, S, \{t\})$) for the sT-diameter (resp., St-diameter). Moreover, if $S = T = V$, we use $\mathrm{diam}(G)$ instead of $\mathrm{diam}(G, V, V)$.

For a given set $F \subseteq E$, we denote by $G - F$ the graph obtained from G by removing all the edges of F. If H is a subgraph of G, we use $V(H)$ and $E(H)$ for the vertices and edges of H, respectively. An *f-edge fault-tolerant distance sensitivity oracle* (f-DSO) with *stretch* $\sigma \geqslant 1$ is a data structure that answers queries (u, v, F) with $u, v \in V$ and $F \subseteq E$ with $|F| \leqslant f$. It returns an estimate $\widehat{d}_{G-F}(u, v)$ of the distance from u to v in $G - F$ such that $d_{G-F}(u, v) \leqslant \widehat{d}_{G-F}(u, v) \leqslant \sigma \cdot d_{G-F}(u, v)$. An *$f$-edge fault-tolerant ST-diameter oracle* (f-FDO-ST) with stretch σ returns, upon query $F \subseteq E$ with $|F| \leqslant f$, an estimate $\widehat{D} = \widehat{D}(F, S, T)$ of the ST-diameter of $G - F$ such that $\mathrm{diam}(G-F, S, T) \leqslant \widehat{D} \leqslant \sigma \cdot \mathrm{diam}(G-F, S, T)$. If $S = \{s\}$ is a singleton or $S = T = V$ are both the whole vertex set, we abbreviate such oracles for as f-FDO-sT and f-FDO, respectively.

3 ST-Diameter Oracles

We start by showing how to use distance sensitivity oracles to design data structures for the fault-tolerant ST-diameter, i.e., the ST-diameter of $G - F$ after a set of edges $F \subseteq E$ failed. The maximum number f of supported failures is called the sensitivity of the data structure. The result is formally stated in Theorem 3.

In the following, we assume that the shortest paths in G are made unique. This way, we can identify a shortest path with its endpoints, which enabled saving both in the time-efficiency of the preprocessing and the space-efficiency of the resulting data structure. In particular, it allows for a subquadratic (in n) space overhead over the underlying f-DSO. However, the precise way how to make the paths unique influences the nature of the preprocessing. As discussed in Section 1, one can ensure a unique shortest path in a random fashion by slightly perturbing the edge weights. Alternatively, lexicographic perturbation [19, 21, 39] provides a deterministic procedure but adds an $O(mn + n^2 \log^2 n)$ term to the running time.

Let $\pi_{u,v}$ denote the (unique) shortest path in G from u to v. Fix a set $F \subseteq E$ of at most f edges and recall that we use $V(F)$ to denote the set of endpoints of edges in F. Our f-DSO-ST uses a data structure to map S and T into two suitable subsets S' and T' of $V(F)$, respectively. A vertex $v \in V(F)$ belongs to S' (resp., T') if there exists a shortest path $\pi_{s,t}$ from some $s \in S$ to some $t \in T$ such that v is a vertex on $\pi_{s,t}$ and the subpath $\pi_{s,v}$ (resp., $\pi_{v,t}$) of $\pi_{s,t}$ from s to v (resp., from t to v) contains no vertex of $V(F)$ other than v. Note that $\pi_{s,v}$ (resp., $\pi_{v,t}$) is completely contained in $G - F$, whence $d_{G-F}(s, v) = d_G(s, v)$ (analogously for $d_{G-F}(v, t)$). The sizes of $S', T' \subseteq V(F)$ are in $O(f)$.

3.1 Query Algorithm

Before describing the data structure, we present the query algorithm. Let $\mathcal{D}$ denote the f-DSO with stretch $\sigma \geqslant 1$ that is assumed in Theorem 3. Given the query F, our diameter oracle computes the two sets S' and T'. Next, for every two vertices u and v such that $u \in S'$ and $v \in T'$, it queries $\mathcal{D}$ with the triple (u, v, F) to obtain a σ-approximation of $d_{G-F}(u, v)$. The f-FDO-ST returns the value $\widehat{D} = \operatorname{diam}(G, S, T) + \max_{(u,v)\in S'\times T'} \mathcal{D}(u, v, F)$.

Given S' and T', the time needed to compute $\widehat{D}$ is $O(f^2\mathtt{Q})$, where $\mathtt{Q}$ is the query time of the f-DSO $\mathcal{D}$. The value $\operatorname{diam}(G, S, T)$ can be precomputed.

▶ **Lemma 7.** *The f-FDO-ST has a stretch of $1 + 3\sigma$.*

Proof. Let $s \in S$ and $t \in T$ be two arbitrary vertices. We first show that $d_{G-F}(s, t) \leqslant \widehat{D}$, that is, the returned value never underestimates the ST-diameter of $G - F$. We only need to prove the case in which some of the failing edges in F belong to $\pi_{s,t}$ as otherwise $d_{G-F}(s, t) = d_G(s, t) \leqslant \operatorname{diam}(G, S, T) \leqslant \widehat{D}$. Thus, let x_s (resp., x_t) be the vertex of $V(F)$ that is closest to s (resp., t) in $\pi_{s,t}$. By definition of S', T', we have $x_s \in S'$ and $x_t \in T'$ and thus $d_{G-F}(s, x_s) = d_G(s, x_s)$ and $d_{G-F}(x_t, t) = d_G(x_t, t)$. Moreover, it holds that $d_{G-F}(s, x_s) + d_{G-F}(x_t, t) = d_G(s, x_s) + d_G(x_t, t) \leqslant \operatorname{diam}(G, S, T)$ as π_{s,x_s} and $\pi_{x_t,t}$ are vertex-disjoint. Using the triangle inequality twice and the fact that $\max_{(u,v)\in S'\times T'} \mathcal{D}(u, v, F)$ never underestimates $\operatorname{diam}(G-F, S', T')$, we get

$$\begin{aligned} d_{G-F}(s, t) &\leqslant d_{G-F}(s, x_s) + d_{G-F}(x_s, x_t) + d_{G-F}(x_t, t) \\ &\leqslant \operatorname{diam}(G, S, T) + \operatorname{diam}(G-F, S', T') \leqslant \widehat{D}. \end{aligned}$$

We now prove that $\widehat{D} \leqslant (1 + 3\sigma) \cdot \operatorname{diam}(G-F, S, T)$. Let $u \in S'$ and $v \in T'$ be arbitrary. There are $s \in S$ and $t \in T$ such that $d_{G-F}(s, u), d_{G-F}(v, t) \leqslant \operatorname{diam}(G, S, T)$. We arrive at

$$\begin{aligned} \mathcal{D}(u, v, F) &\leqslant \sigma\, d_{G-F}(u, v) \leqslant \sigma\big(d_{G-F}(u, s) + d_{G-F}(s, t) + d_{G-F}(t, v)\big) \\ &\leqslant \sigma\big(\operatorname{diam}(G, S, T) + \operatorname{diam}(G-F, S, T) + \operatorname{diam}(G, S, T)\big) \\ &\leqslant 3\sigma \operatorname{diam}(G-F, S, T), \end{aligned}$$

thus $\widehat{D} = \operatorname{diam}(G, S, T) + \max_{u\in S', v\in T'} \mathcal{D}(u, v, F) \leqslant (1 + 3\sigma) \operatorname{diam}(G - F, S, T)$. ◀

3.2 Data Structure for the Sets S' and T' for Large Sensitivity

Recall that, given the failure set F, the set S' contains all $v \in V(F)$ such that there are $s \in T$ and $t \in T$ for which v is the closest vertex to s on $V(F) \cap E(\pi_{s,t})$, analogously for T'. We now describe the data structure that computes the sets S' and T', focusing on S' since the case of T' follows in the same fashion.

The construction algorithm depends on the sensitivity f. Suppose first that $f = \Omega(\log n)$. For each vertex $v \in V$, the data structure stores the shortest-path tree T_v of G rooted at v and mark some of its vertices. Namely, all $s \in S$ are marked for which there is a $t \in T$ such that v lies on the path $\pi_{s,t}$. For every two vertices $s \in S$ and $t \in T$, $\pi_{s,t}$ contains v if and only if $d_G(s, t) = d_G(s, v) + d_G(v, t)$. We used here that the paths are unique. It suffices to compute the all-pairs distances in G in time $O(mn + n^2 \log n)$ time[2] and use them to mark the vertices of T_v for *all* v with the obvious $O(n|S||T|)$-time algorithm.

[2] The time needed for this step reduces to $O(mn)$ in case G is unweighted or has only small integer or even floating point weights (in exponent-mantissa representation) using Thorup's algorithm [46].

Additionally, each vertex u of T_v is annotated with the value $count_v(u)$, the number of marked vertices in the subtree $(T_v)_u$ rooted at u. For a fixed tree T_v, all values $count_v(u)$ are computable in $O(n)$ time in a bottom-up fashion. Finally, we store, for each T_v, a data structure that supports least common ancestor (LCA) queries in constant time. Such structures can be built in time and space that is linear in the size of the tree [9]. The time needed to construct the data structure is $O(mn + n^2 \log n + n|S||T|)$ and the space is $O(n^2)$.

To answer a query F, the algorithm scans all the vertices $v \in V(F)$ and decides which of them to include in S'. The graph $T_v - F$ is a collection of rooted trees. (Possibly some of the trees degenerated to isolated vertices.) We observe that $v \in S'$ if and only if $T_v - F$ contains a marked vertex that is still reachable from v. To check this condition, the algorithm computes the set F_0 of all the edges $\{u, w\} \in F$ that are contained in T_v. This is the case if and only if the LCA of u and w in T_v is either u or w.

Next, we define a notion of domination for edges in F_0. We say that an edge $\{u, w\} \in F_0$, where u is the parent of w in T_v, is *dominated* by another edge $\{a, b\} \in F_0$, where a is the parent of b in T_v, if $\{u, w\}$ is in the subtree of T_v rooted at b. This is equivalent to b being the LCA of b and u. The query algorithm removes all dominated edges from F_0, which can be done in $O(|F_0|^2) = O(f^2)$ time.

Recall that $count_v(v)$ is the overall number of marked vertices in T_v. Evidently, some vertex in $T_v - F$ is reachable from v iff they are in the same connected component. Thus, there is a marked vertex reachable from v if and only if $count_v(v)$ is strictly larger than the number of marked vertices contained in those components of $T_v - F$ that do *not* contain v. Indeed, the difference between those two values is exactly the *number* of marked vertices reachable from v. Each connected component of $T_v - F$ that does not contain v is a tree T' rooted at some vertex $w \in V(F_0)\backslash\{v\}$. Let u be the parent of w in T_v. Compared to the full subtree $(T_v)_u$ rooted at u, T' is missing those subtrees "further down" that are rooted at some other vertex b whose parent a is a vertex of T'. Those are exactly the edges $\{a, b\} \in F_0$ that are dominated by $\{u, w\}$. Accordingly, the value $count_v(u)$ counts the marked vertices in T' and additionally those in the subtrees rooted at the vertices b. By removing all dominated edges from F_0, we avoid any double counting and ensure that $count_v(v) - \sum_{u \in V(F_0)} count_v(u)$ is indeed the quantity we are interested in. It can be computed in time $O(f)$ for each v.

3.3 Small Sensitivity

We now modify the data structure in the case where the sensitivity $f = o(\log n)$ is sublogarithmic. If so, the information of all the trees T_v can be stored in a more compact way. For every vertex $v \in V$, we define a new representation $\mathcal{T}_v$ of the tree T_v by first removing unnecessary parts and then replacing long paths with single edges. This corresponds to the two steps of the compression described below. For the first one, we need the following definition. We say a subtree $\mathcal{T}_v$ of T_v preserves the *source-to-leaf reachability* if, for every set $F \subseteq E$ of up to f failing edges, there is a marked vertex of T_v that is reachable from the source v in $T_v - F$ if and only if there is a leaf of $\mathcal{T}_v$ that is reachable from v in $\mathcal{T}_v - F$.

The first compression step. We first describe how to preserve the source-to-leaf reachability. We select a set $\mathcal{L}_v \subseteq S$ of at most 2^f marked vertices and set $\mathcal{T}_v$ as the smallest subtree of T_v that contains v and $\mathcal{L}_v$. We say that a marked vertex s of T_v is *relevant* if there is no marked vertex $s' \neq s$ that is contained in the path from v to s in T_v.

We compute $\mathcal{L}_v$ as follows. We construct a DAG G_v that is obtained from a copy of T_v in which each edge (u, u'), with u being the parent of u' in T_v, is directed from u to u'. The DAG is augmented with a dummy sink vertex x that contains an incoming directed edge

from each relevant vertex s of T_v. We then run the algorithm of Baswana, Choudhary, and Roditty [6] to compute a subgraph H_v of G_v such that (i) the in-degree of each vertex of H_v is at most 2^f and (ii) for every possible set F of at most f edge failures, each vertex u is reachable from v in the graph $G_v - F$ iff u is reachable from v in $H_v - F$.

The set $\mathcal{L}_v$ of marked vertices corresponds to the tails of the edges in H_v that enter the sink x. As x has in-degree of at most 2^f in H_v, the size of $\mathcal{L}_v$ is $O(2^f)$. Moreover, $\mathcal{L}_v$ is the set of leaves of $\mathcal{T}_v$. The following lemma proves the correctness of our selection algorithm.

▶ **Lemma 8.** *For every $F \subseteq E(G)$, with $|F| \leqslant f$, there is a marked vertex of T_v that is reachable from v in $T_v - F$ iff there is a vertex of $\mathcal{L}_v$ that is reachable from v in $\mathcal{T}_v - F$.*

Proof. Fix a set F of at most f failing edges of G. As $\mathcal{T}_v$ is a subtree of T_v, if there is a vertex in $\mathcal{L}_v$ that is reachable from v in $\mathcal{T}_v - F$, then the same marked vertex is reachable from v in $T_v - F$. To prove the other direction, let X be the set of all marked vertices that are reachable from v in $T_v - F$. We prove that $X \cap \mathcal{L}_v \neq \emptyset$. Let $s \in X$ be a marked vertex that is reachable from v in $T_v - F$. Let $s^* \in S$ be the vertex closest to v in the path from v to s in T_v (possibly, $s^* = s$). We have that s^* is relevant and is reachable from v in $T_v - F$. This implies that the sink x is reachable from v in $G_v - F$ via the path that goes through s^*. As a consequence, x is also reachable in $H_v - F$. Hence, there is a vertex in $\mathcal{L}_v$ that is also reachable from v in $\mathcal{T}_v - F$. Therefore, $X \cap \mathcal{L}_v \neq \emptyset$. ◀

The second compression step. After the first compression step, the tree $\mathcal{T}_v$ contains at most 2^f leaves. However, it might still be the case that the number of vertices of $\mathcal{T}_v$ is large due to the presence of very long paths connecting two consecutive *branch vertices*, i.e., vertices of $\mathcal{T}_v$ with two or more children. The second step of compressing $\mathcal{T}_v$ allows us to represent long paths between consecutive branch vertices in a more compact way.

Let x and y be two consecutive branch vertices in $\mathcal{T}_v$, i.e., x is an ancestor of y in $\mathcal{T}_v$ and the internal vertices of the path P from x to y are not branch vertices. We say that P is *long* if it contains at least $\sqrt{n}$ edges. If the path P is long, we substitute the path P in $\mathcal{T}_v$ with a *representative* edge between x and y (so we also remove all the internal vertices of P from the tree) and we add the path P to the set $\mathcal{P}$ of long paths. So, in every tree $\mathcal{T}_v$, we replace every long path between two consecutive branch vertices with a representative edge. We observe that $\mathcal{P}$ can be computed in $O(n^2)$ time. Moreover, we observe that $\mathcal{P}$ contains $O(n^{3/2})$ paths as each tree T_v contributes with at most $\sqrt{n}$ long paths.

Next, we use the algorithm given in [2] to hit all the long paths in $\mathcal{P}$ with a set Z of $O(\sqrt{n}\log n)$ *pivot* vertices in $O(|\mathcal{P}|\sqrt{n}) = O(n^2)$ time, where a path is *hit* if we select a pivot vertex that belongs to the path. For each pivot $z \in Z$, we store the shortest-path tree T_z of G rooted at z. By construction, each long path $P \in \mathcal{P}$ between two consecutive branch vertices x and y of a tree $\mathcal{T}_v$ is contained in T_z, for some $z \in Z$ that hits P; moreover, a vertex $z \in Z$ that hits P is also the least-common-ancestor of x and y in T_z.

The representative edge (x, y) in $\mathcal{T}_v$ stores a pointer to the tree T_z of any pivot z that hits P (ties can be arbitrarily broken). Clearly, after the second compression step, each tree $\mathcal{T}_v$ contains $O(2^f\sqrt{n})$ vertices. Therefore, the overall size needed to store all the trees $\mathcal{T}_v$ is $O(2^f n^{3/2})$. Moreover, storing the trees T_z for all the pivots in Z requires $O(n)$ space per tree, for a total of $O(n^{3/2}\log n)$ space. Hence, the overall size of our data structure is $O(n^{3/2}(2^f + \log n))$.

Now, given a set F of at most f failing edges, we describe how the query algorithm computes the set S' in $O(f^2 2^f)$ time. As before, for every $v \in V(F)$, we need to understand whether v must be added to S' or not. In the following, we fix $v \in V(F)$ and explain how to

check whether $v \in S'$ or not in $O(f2^f)$ time. We recall that v must be added to S' iff there is a marked vertex in $T_v - F$ that is still reachable from v. By Lemma 8, this is equivalent to having a leaf of $\mathcal{L}_v$ that is reachable from v in $\mathcal{T}_v - F$.

We visit the tree $\mathcal{T}_v$ and we remove from $\mathcal{T}_v$ all edges that correspond to edges in F. This can be easily done in $O(f)$ time for each non-representative edge using least-common-ancestor queries. For the representative edges we proceed as follows. We consider all the representative edges in $\mathcal{T}_v$. Let (x, y) be a representative edge of $\mathcal{T}_v$ and let z be the pivot of the tree T_z that is associated with the edge (x, y) in $\mathcal{T}_v$. We remove (x, y) from $\mathcal{T}_v$ iff there is a failing edge in F that is contained in the path P in T_z from x to y. We check whether P contains some edges of F in $O(f)$ time as follows. We look at all the failing edges in F and, for each failing edge $(u, u') \in F$, we check whether (u, u') is an edge of P using a constant number of *least-common-ancestor* queries in the tree T_z.[3] As each tree $\mathcal{T}_v$ contains $O(2^f)$ representative edges and we need $O(f)$ time to understand if a representative edge can be removed or not from the tree, we need $O(f2^f)$ to understand which are the representative edges that need to be removed from $\mathcal{T}_v$, for a fixed $v \in V(F)$.

Once all edges that represent F have been removed from $\mathcal{T}_v$, it is enough to check whether there is a vertex of $\mathcal{L}_v$ that is still reachable from v. This can be clearly done in $O(f^2)$ time per tree $\mathcal{T}_v$ using the values k_u, as already discussed for the case in which $f = \Omega(\log n)$. In particular, for every vertex u in $\mathcal{T}_v$, the value k_u is equal to the number of vertices of $\mathcal{L}_v$ that are contained in the subtree of $\mathcal{T}_v$ rooted at u.

4 Single-Source sT-Diameter Oracles

In the following theorem, we address the question of computing an sT-diameter oracle using a single-source DSO with source s. We restate the relevant theorem below. Its proof uses similar ideas as those shown in Section 3, but the single-source setting allows for a better preprocessing time, space, and stretch.

▶ **Theorem 4.** *Let $G = (V, E)$ be an undirected graph with n vertices, m edges, and possibly positive edge weights. Let $s \in V$ be a vertex and $T \subseteq V$ a non-empty set. Given a single-source f-DSO for G with preprocessing time $\mathtt{P}$, space $\mathtt{S}$, query time $\mathtt{Q}$, and stretch σ, one can compute an f-FDO-sT for G with preprocessing time $\mathtt{P} + O(m + n \log n)$, space $\mathtt{S} + O(n)$, query time $O(f^2 + f\mathtt{Q})$, and stretch $1 + 2\sigma$. For unweighted graphs, the preprocessing time can be improved to $\mathtt{P} + O(m)$.*

Proof. Let $\mathcal{D}$ denote the single-source f-DSO. The preprocessing algorithm for the f-FDO-sT first constructs $\mathcal{D}$ with source s. It also computes a shortest path tree T_s of G rooted at s. Each node $v \in V(T_s) = V$ is annotated with a pointer to its parent node and its respective number in the pre-order and post-order traversal of T_s. Similarly as above, the algorithm also computes the value $count(v)$ for every v, which is the number of descendants of v (including v itself) that are in T. Finally, it stores the maximum distance $C = \max_{t \in T} d_G(s, t)$ from the root among the vertices in the set T. The preprocessing takes total time $\mathtt{P} + O(m + n \log n)$ in general weighted graphs and, again, can be reduced to $\mathtt{P} + O(m)$ for certain classes of weights [46]. Storing the oracle and the tree takes $\mathtt{S} + O(n)$ space.

[3] We observe that (u, u') is on the path P iff one of the following two conditions hold: (i) the least-common-ancestor of u and x in T_z is u and the least-common-ancestor of u' and x in T_z is u'; (ii) the least-common-ancestor of u and y in T_z is u and the least-common-ancestor of u' and y in T_z is u'.

For the query, consider a set $F \subseteq E$ of up to f failing edges and let $F_0 = F \cap E(T_s)$ be those failures that are in the tree. Consider the collection of rooted (sub-)trees $T_s - F_0$. Define X_F to be the set of roots of those trees that contain some vertex from T. For some $v \in V$, let $\mathcal{D}(v, F)$ be the σ-approximation of the replacement distance $d_{G-F}(s, v)$ computed by the DSO $\mathcal{D}$. Our sT-diameter oracle answers the query F by reporting the value

$$\widehat{D} = C + \max_{x \in X_F} \mathcal{D}(x, F).$$

Regarding the correctness of that answer, consider a vertex $t \in T$. Let $x \in X_F$ be the root of the subtree of T_s that contains t. There is a path from s to t in $G - F$ of length at most $d_{G-F}(s, x) + d_G(x, t) \leqslant d_{G-F}(s, x) + d_G(s, t) \leqslant \mathcal{D}(x, F) + C$. Hence, we have $d_{G-F}(s, t) \leqslant C + \max_{x \in X_F} \mathcal{D}(x, F)$, that is, $\operatorname{diam}(G-F, s, T) \leqslant \widehat{D}$. We next prove $\widehat{D} \leqslant (1 + 2\sigma) \cdot \operatorname{diam}(G-F, s, T)$. Let $x_0 \in X_F$ be the maximizer of $\mathcal{D}(x, F)$, and $t \in T$ be in the tree in $T_s - F_0$ that is rooted in x_0. Then, we have $d_{G-F}(s, x_0) \leqslant d_{G-F}(s, t) + d_G(t, x_0) \leqslant d_{G-F}(s, t) + d_G(t, s) \leqslant 2 \cdot d_{G-F}(s, t)$. We used here that G is undirected so that we can go "up" the tree from t to x_0. From this, we get

$$\widehat{D} = C + \mathcal{D}(x_0, F) \leqslant C + \sigma \cdot d_{G-F}(s, x_0) \leqslant C + 2\sigma \cdot d_{G-F}(s, t) \leqslant (1 + 2\sigma) \cdot \operatorname{diam}(G - F, s, T).$$

Given X_F, computing $\widehat{D}$ takes time $O(f\mathtt{Q})$. It remains to show how to compute X_F from F in $O(f^2)$ time. Recall that we know the parent of every non-root node in T_s. We use it to first obtain F_0 from F in time $O(f)$ as an edge $\{a, b\}$ is in T_s iff a is parent of b or vice versa.

For each edge $e \in F_0$, let $b(e)$ be the endpoint of e that is farther from the source s. Next, define $B_0 = \{b(e) \mid e \in F_0\} \cup \{s\}$. Every root in X_F is either the source s or the "lower" endpoint of a failing edge, i.e., $X_F \subseteq B_0$. For each $b \in B_0$, let $B_0(b)$ be the closest proper descendants of b in B_0, if any. That is, on the paths in T_s between b and any $b' \in B_0(b)$ there is no other vertex from B_0. We can compute the sets $B_0(b)$ for all $b \in B_0$ simultaneously in total time $O(|B_0|^2) = O(f^2)$ as follows. A vertex is a proper ancestor of b' iff its pre-order number is strictly smaller than that of b' and its post-order number is strictly larger. So finding those takes time $O(|B_0|)$ for each $b' \in B_0$. Then, b' is in the set $B_0(b)$ for the proper ancestor b with the highest pre-order number.

Finally, observe that a vertex $b \in B_0$ lies in X_F if and only if there is at least one vertex of T that falls into the subtree of T_s rooted at v but not in any of the subtrees rooted at (proper) descendants of b in B_0. To check this condition via the counts, we only need to consider the immediate descendants in $B_0(b)$. If the element of T is in some lower subtree, then it is also accounted for by an immediate descendant. In summary, some $b \in B_0$ is in X_F iff $\mathit{count}(b) - \sum_{b' \in B_0(b)} \mathit{count}(b') > 0$. This proves that X_F is computable in time $O(f^2)$. ◀

We now handle multiple sources, that is, we build an f-FDO-ST for a general set S. The next result is a straightforward reduction to the sT-case. As it turns out, it is enough to construct the sT-diameter oracle for two arbitrary vertices $s \in S$ and $t \in T$. Due to lack of space, the proof of Lemma 9 is deferred to the full version of the paper.

▶ **Lemma 9.** *Let $G = (V, E)$ be an undirected graph with n vertices, m edges, and possibly positive edge weights. Let $S, T \subseteq V$ be non-empty sets of vertices, and $s \in S$ and $t \in T$ be two vertices. Suppose one is given access to an f-FDO-sT and an f-FDO-tS for G with respective preprocessing times $\mathtt{P}_{sT}$ and $\mathtt{P}_{tS}$, space requirements $\mathtt{S}_{sT}$ and $\mathtt{S}_{tT}$, query times $\mathtt{Q}_{sT}$ and $\mathtt{Q}_{sT}$, and stretches σ_{sT} and σ_{tS}. Then, one can compute an f-FDO-ST for G with preprocessing time $\mathtt{P}_{sT} + \mathtt{P}_{tS}$, space $\mathtt{S}_{sT} + \mathtt{S}_{tS}$, query time $\mathtt{Q}_{sT} + \mathtt{Q}_{tT}$, and stretch $\sigma_{sT} + \sigma_{tS} + \min(\sigma_{sT}, \sigma_{tS})$.*

Table 8 Conditions for the presence of edges between the vertex sets of graph G in Section 5. The symbol $\oplus$ stands for the exclusive or. All conditions are symmetric with respect to the index pairs (i,x), (j,y), and (k,z), whence H is undirected.

Set Pair	Vertex Pair	Edge Condition
$A \times A$		independent set
$B \times B$	$b[i,j,k],\ b[x,y,z]$	$(i=x) \oplus (j=y)$
$C \times C$	$c[i,j,k],\ c[x,y,z]$	$(i=x) \oplus (j=y)$
$D \times D$		independent set
$A \times B$	$a[i,j],\ b[x,y,z]$	$(i=x) \wedge (z=0)$
$B \times C$	$b[i,j,k],\ c[x,y,z]$	$(i=x) \wedge (j=y)$
$C \times D$	$c[i,j,k],\ d[x,y]$	$(j=y) \wedge (k=0)$

Combining Theorem 4 and Lemma 9 gives a reduction from f-FDO-ST to single-source f-DSOs. However, it results in a data structure with a stretch of $3+6\sigma$, where σ is the original stretch of the f-DSO We can improve this by not treating Lemma 9 as a black box.

▶ **Theorem 5.** *Let $G=(V,E)$ be an undirected graph with n vertices, m edges, and possibly positive edge weights. Let S,T be two non-empty subsets of V. Given a single-source f-DSO for G with preprocessing time $\mathtt{P}$, space $\mathtt{S}$, query time $\mathtt{Q}$, and stretch σ, one can compute an f-FDO-ST for G with preprocessing time $O(\mathtt{P}+m+n\log n)$, space $O(\mathtt{S}+n)$, query time $O(f^2+f\mathtt{Q})$, and stretch $2+5\sigma$. For unweighted graphs, the preprocessing time can be improved to $O(\mathtt{P}+m)$*

Proof. Let $s \in S$ and $t \in T$ be arbitrary. The preprocessing algorithm of the f-FDO-ST uses the single-source f-DSO twice, once for source s and once for t, to construct an f-FDO-sT $\mathcal{D}_{sT}$ and an f-FDO-tS $\mathcal{D}_{tS}$ both with stretch $1+2\sigma$, as described in Theorem 4.

For a set $F \subseteq E$ of at most f edge failures, let $\mathcal{D}_{sT}(F)$ and $\mathcal{D}_{tS}(F)$ be the respective $(1+2\sigma)$-approximations of $\operatorname{diam}(G-F,s,T)$ and $\operatorname{diam}(G-F,t,S)$. Further, let $\mathcal{D}_{st}(F)$ be a σ-approximation of $d_{G-F}(s,t)$, obtained from the DSO with source s. The query algorithm outputs $\widehat{D} = \mathcal{D}_{tS}(F) + \mathcal{D}_{st}(F) + \mathcal{D}_{sT}(F)$. Let $(s_0,t_0) \in S \times T$. We have

$$\begin{aligned} d_{G-F}(s_0,t_0) &\leqslant d_{G-F}(s_0,t) + d_{G-F}(t,s) + d_{G-F}(s,t_0) \\ &\leqslant \mathcal{D}_{tS}(F) + \mathcal{D}_{st}(F) + \mathcal{D}_{sT}(F) \leqslant (2+5\sigma)\cdot \operatorname{diam}(G-F,S,T). \end{aligned}$$ ◀

5 Space Lower Bound

Recall that Theorem 6 states a space lower bound for f-FDOs and f-FDO-STs with sensitivity $f \geqslant 2$ in that if they have stretch better than $5/3$, they must take $\Omega(n^{3/2})$ space. The theorem is implied by the following lemma, which we prove in this section.

▶ **Lemma 10.** *For infinitely many n, there is a graph $G=(V,E)$ with n vertices (and two sets $S,T \subseteq V$) such that any data structure that decides for any pair of edges $e,e' \in E$, whether $G-\{e,e'\}$ has diameter (resp., ST-diameter) 3 or 5 requires $\Omega(n^{3/2})$ bits of space.*

We first construct an auxiliary graph H. Let $n=6N$ for some N which is a perfect square. In the following, indices i,j range over the set $[\sqrt{N}]$ and k ranges over $\{0,1\}$ Define four pairwise disjoint sets of vertices $A=\{a[i,j]\}_{i,j}$, $B=\{b[i,j,k]\}_{i,j,k}$, $C=\{c[i,j,k]\}_{i,j,k}, D=\{d[i,j]\}_{i,j}$ with respective cardinalities $N, 2N, 2N$, and N. The vertex set of H is $V(H)=A\cup B\cup C\cup D$. The edges in H are shown in Table 8 and are defined

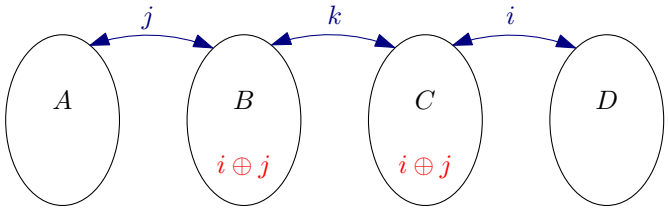


Figure 1 Visual representation of the graph H. Each vertex corresponds to a tuple $[i, j]$ or $[i, j, k]$ and belongs to one of the sets A, B, C, or D. To move from vertex $a[i, j]$ to $b[i', j', k']$ it must be the case that $i = i'$ and $k' = 0$, but one can jump from any j to any j'. This is marked by the blue edge labeled j between sets A and B, analogously for the other pairs of sets. When moving inside the sets B or C *either* the first index $i \neq i'$ *or* the second one $j \neq j'$ changes, marked by the red labels. Sets A and D have no internal edges.

depending on the relations among the indices of the participating vertices. For example, some edge $\{b[i, j, k],\ b[x, y, z]\}$ between elements of B and C exists if and only if *either* i and x are equal *or* j and y are equal, while $k, z \in \{0, 1\}$ can be arbitrary. Note that the number of edges in E is $\Theta(N^{3/2}) = \Theta(n^{3/2})$.

▶ **Lemma 11.** *The diameter of H is at most* 3.

Proof. To verify that the diameter of H is at most 3, we give explicit paths of length at most 3 between all possible vertex pairs from the sets A, B, C, and D. Note that all paths below are reversible as the edges are undirected. The symbol $\overline{x}$ stands for any index from $[\sqrt{N}]$ except x, analogously for $\overline{y}$.

- For vertices $a[i, j], a[x, y] \in A$, we distinguish two cases depending on whether the first indices $i \neq x$ are different or not. In the first case, the vertices are joined by the path $(a[i, j], b[i, y, 0], b[x, y, 0], a[x, y])$. In the second case, the middle two vertices are the same, thus the path shortens to $(a[i, j], b[x, y, 0], a[x, y])$.
- Symmetrically, for vertices $d[i, j], d[x, y] \in D$, the cases are defined with respect two the second indices, i.e., whether $j \neq y$. The paths are $(d[i, j], c[x, j, 0], c[x, y, 0], d[x, y])$ and $(d[i, j], c[x, y, 0], d[x, y])$, respectively.
- For vertices $b[i, j, k], b[x, y, z] \in B$, the generic path is $(b[i, j, k], b[x, j, k], b[x, y, z])$. If $i = x$, then the first two vertices are the same; if $j = y$, the last two are. The argument for vertices $c[i, j, k], c[x, y, z] \in C$ is the same.
- For the vertex pair $(a[i, j], b[x, y, z]) \in A \times B$, the ley point is that any edge inside of B changes exactly one of the first two indices. If $i \neq x$, the path is $(a[i, j], b[i, y, 0], b[x, y, z])$, otherwise it is $(a[i, j], b[x, \overline{y}, 0], b[x, y, z])$.
- The pair $(d[i, j], c[x, y, z]) \in D \times C$ is handled symmetrically. If $j \neq y$, the path is $(d[i, j], c[x, j, 0], c[x, y, z])$, otherwise it is $(d[i, j], b[\overline{x}, y, 0], b[x, y, z])$.
- Vertex pair $(a[i, j], c[x, y, z]) \in A \times C$: path $(a[i, j], b[i, y, 0], c[i, y, z], c[x, y, z])$. Note that if $i = x$ the last two vertices are the same. Vertex pair $(d[i, j], b[x, y, z]) \in D \times B$: path $(d[i, j], c[x, j, 0], b[x, j, z], b[x, y, z])$.
- Vertex pair $(a[i, j], d[x, y]) \in A \times D$: path $(a[i, j], b[i, y, 0], c[i, y, 0], d[x, y])$.
- Vertex pair $(b[i, j, k], c[x, y, z]) \in B \times C$: the path $(b[i, j, k], b[x, j, k], c[x, j, k], c[x, y, z])$ possibly shortens if consecutive vertices are the same. ◀

Consider an arbitrary binary $\sqrt{N} \times \sqrt{N} \times \sqrt{N}$ matrix (tensor) M. We build a supergraph $G \supseteq H$ embedding the information about the entries of M in the fault-tolerant diameter of G under dual failures, i.e., $\mathrm{diam}(G-F)$ with $|F| = 2$. The number of possible matrices M will then imply the space lower bounds for diameter oracles for G.

The graph G contains all vertices and edges of H and the following additional edges.

- For all $i, j, y \in [\sqrt{N}]$, if $M[i, j, y] = 1$, then add $\{a[i, j], b[i, y, 1]\}$ as an edge of G.
- For all $i, x, y \in [\sqrt{N}]$, if $M[i, x, y] = 1$, then add $\{c[i, y, 1], d[x, y]\}$.

Note that the diameter of G remains at most 3.

Consider any four indices $i, j, x, y \in [\sqrt{N}]$ such that $i \neq x$ and $j \neq y$. We define two sets F, F' both containing pairs of vertices in $V = V(H)$. First, let $F \subseteq E(H) \subseteq E$ contain $e_1 = \{a[i, j], b[i, y, 0]\}$ and $e_2 = \{c[i, y, 0], d[x, y]\}$. Secondly, let F' be the set comprising the two pairs $e'_1 = \{a[i, j], b[i, y, 1]\}$ and $e'_2 = \{c[i, y, 1], d[x, y]\}$. Note that the elements of F' are only edges of G if the entries $M[i, j, y]$ and $M[i, x, y]$ are 1.

▶ **Lemma 12.** *For any four indices $i, j, x, y \in [\sqrt{N}]$ such that $i \neq x$ and $j \neq y$, the diameter of $G - (F \cup F')$ is at least 5.*

Proof. We show that the distance between $a[i, j]$ and $d[x, y]$ in $G - (F \cup F')$ is at least 5. Contrarily, assume that $P = (a[i, j], w_1, w_2, w_3, d[x, y])$ is a path of length at most 4. P must pass across sets $A \to B$, $B \to C$, and $C \to D$ and change the indices from (i, j) to (x, y).

The neighborhood of $a[i, j]$ in $G - (F \cup F')$ is the set

$$\Big\{b[i, \overline{y}, 0] \mid \overline{y} \in [\sqrt{N}] \setminus \{y\}\Big\} \cup \Big\{b[i, \overline{y}, 1] \mid \overline{y} \in [\sqrt{N}] \setminus \{y\} \wedge M[i, j, \overline{y}]\} = 1\Big\}.$$

The index i cannot change on the first edge $\{a[i, j], w_1\}$ of P and, since the edges $e_1 = \{a[i, j], b[i, y, 0]\} \in F$ and $e'_1 = \{a[i, j], b[i, y, 1]\} \in F'$ are missing, the second index of w_1 must differ from y. Symmetrically, the change of j cannot take place on the last edge $\{w_3, d[x, y]\}$ and the first index of w_3 must differ from x. At least one of the edges $\{w_1, w_2\}$ or $\{w_2, w_3\}$ passes from B to C, w.l.o.g. let this be $\{w_1, w_2\}$. This edge (already present in H) cannot change any of the indices. We are left with $\{w_2, w_3\}$. If P has strictly less than 4 edges, then $w_2 = w_3$. Otherwise, either both endpoints w_2 and w_3 are in B, both are in C or there is exactly one in either. None of those cases allows one to make the *two* necessary changes to the indices simultaneously. ◀

▶ **Lemma 13.** *The diameter of $(G - F) \cup F'$ is 3.*

Proof. The proof is very similar to that of Lemma 11, only that every time the edge $e_1 = \{a[i, j], b[i, y, 0]\} \in F$ (respectively, $e_2 = \{c[i, y, 0], d[x, y]\}$) has been used, it is replaced by $e'_1 = \{a[i, j], b[i, y, 1]\} \in F'$ (respectively, by $e'_2 = \{c[i, y, 1], d[x, y]\}$). ◀

▶ **Lemma 14.** *The diameter of $G - F$ is at most 3 if $M[i, j, y] = M[i, x, y] = 1$, and at least 5 if $M[i, j, y] = M[i, x, y] = 0$.*

Proof. The diameter of graph $G - F$ is at least 5 if neither vertex pair in F' is an edge of G by Lemma 12. This is only true if $M[i, j, y] = M[i, x, y] = 0$. Conversely, by Lemma 13, the diameter is at most 3 if both edges in F' lie in G, i.e., if $M[i, j, y] = M[i, x, y] = 1$. ◀

We now finish the proof of Lemma 10. Suppose there exists a data structure that distinguishes whether after any two edges fail the diameter of the resulting graph is bounded by 3 or at least 5. We can use it to infer the entry $M[i, j, y]$ for any triple $(i, j, y) \in [\sqrt{N}]^3$

of indices such that i and j differ from each other, and j and y differ. We compute the edges in F with respect to the indices $i \neq x = j \neq y$ and apply Lemma 14 to check whether $M[i,j,y] = M[i,x,y] = 1$ or $M[i,j,y] = M[i,x,y] = 0$. For the assertion in Lemma 10 about the ST-diameter, we choose $S = A$ and $T = D$. Since there are $2^{\sqrt{N}(\sqrt{N}-1)^2} = 2^{\Omega(n^{3/2})}$ collections of possible answers, the oracle must take $\Omega(n^{3/2})$ bits of space.

References

1 Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast Estimation of Diameter and Shortest Paths (Without Matrix Multiplication). *SIAM Journal on Computing*, 28:1167–1181, 1999. doi:10.1137/S0097539796303421.

2 Noga Alon, Shiri Chechik, and Sarel Cohen. Deterministic Combinatorial Replacement Paths and Distance Sensitivity Oracles. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming, (ICALP)*, pages 12:1–12:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.12.

3 Bertie Ancona, Monika Henzinger, Liam Roditty, Virginia Vassilevska Williams, and Nicole Wein. Algorithms and Hardness for Diameter in Dynamic Graphs. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 13:1–13:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.13.

4 Arturs Backurs, Liam Roditty, Gilad Segal, Virginia Vassilevska Williams, and Nicole Wein. Toward Tight Approximation Bounds for Graph Diameter and Eccentricities. *SIAM Journal on Computing*, 50:1155–1199, 2021. doi:10.1137/18M1226737.

5 Surender Baswana, Keerti Choudhary, Moazzam Hussain, and Liam Roditty. Approximate Single-Source Fault Tolerant Shortest Path. *ACM Transactions on Algorithms*, 16:44:1–44:22, 2020. doi:10.1145/3397532.

6 Surender Baswana, Keerti Choudhary, and Liam Roditty. Fault-Tolerant Subgraph for Single-Source Reachability: General and Optimal. *SIAM Journal on Computing*, 47:80–95, 2018. doi:10.1137/16M1087643.

7 Surender Baswana and Telikepalli Kavitha. Faster Algorithms for Approximate Distance Oracles and All-Pairs Small Stretch Paths. In *Proceedings of the 47th Symposium on Foundations of Computer Science (FOCS)*, pages 591–602, 2006. doi:10.1109/FOCS.2006.29.

8 Surender Baswana and Neelesh Khanna. Approximate Shortest Paths Avoiding a Failed Vertex: Near Optimal Data Structures for Undirected Unweighted Graphs. *Algorithmica*, 66:18–50, 2013. doi:10.1007/s00453-012-9621-y.

9 Michael A. Bender and Martin Farach-Colton. The LCA problem revisited. In Gaston H. Gonnet, Daniel Panario, and Alfredo Viola, editors, *LATIN 2000: Theoretical Informatics, 4th Latin American Symposium, Punta del Este, Uruguay, April 10-14, 2000, Proceedings*, volume 1776 of *Lecture Notes in Computer Science*, pages 88–94. Springer, 2000. doi:10.1007/10719839_9.

10 Aaron Bernstein and David R. Karger. Improved Distance Sensitivity Oracles via Random Sampling. In *Proceedings of the 19th Symposium on Discrete Algorithms (SODA)*, pages 34–43, 2008. URL: https://dl.acm.org/citation.cfm?id=1347082.1347087.

11 Aaron Bernstein and David R. Karger. A Nearly Optimal Oracle for Avoiding Failed Vertices and Edges. In *Proceedings of the 41st Symposium on Theory of Computing (STOC)*, pages 101–110, 2009. doi:10.1145/1536414.1536431.

12 Davide Bilò, Keerti Choudhary, Luciano Gualà, Stefano Leucci, Merav Parter, and Guido Proietti. Efficient Oracles and Routing Schemes for Replacement Paths. In *Proceedings of the 35th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 13:1–13:15, 2018. doi:10.4230/LIPIcs.STACS.2018.13.

13 Davide Bilò, Luciano Gualà, Stefano Leucci, and Guido Proietti. Compact and Fast Sensitivity Oracles for Single-Source Distances. In Piotr Sankowski and Christos D. Zaroliagis, editors, *Proceedings of the 24th European Symposium on Algorithms (ESA)*, pages 13:1–13:14, 2016. doi:10.4230/LIPIcs.ESA.2016.13.

14 Davide Bilò, Luciano Gualà, Stefano Leucci, and Guido Proietti. Multiple-Edge-Fault-Tolerant Approximate Shortest-Path Trees. *Algorithmica*, 84:37–59, 2022. doi:10.1007/s00453-021-00879-8.

15 Davide Bilò, Keerti Choudhary, Sarel Cohen, Tobias Friedrich, and Martin Schirneck. Deterministic Sensitivity Oracles for Diameter, Eccentricities and All Pairs Distances. In *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 22:1–22:19, 2022. doi:10.4230/LIPIcs.ICALP.2022.22.

16 Davide Bilò, Sarel Cohen, Tobias Friedrich, and Martin Schirneck. Near-Optimal Deterministic Single-Source Distance Sensitivity Oracles. In *Proceedings of the 29th European Symposium on Algorithms (ESA)*, pages 18:1–18:17, 2021. doi:10.4230/LIPIcs.ESA.2021.18.

17 Davide Bilò, Sarel Cohen, Tobias Friedrich, and Martin Schirneck. Space-Efficient Fault-Tolerant Diameter Oracles. In *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 18:1–18:16, 2021. doi:10.4230/LIPIcs.MFCS.2021.18.

18 Jan van den Brand and Thatchaphol Saranurak. Sensitive Distance and Reachability Oracles for Large Batch Updates. In *Proceedings of the 60th Symposium on Foundations of Computer Science (FOCS)*, pages 424–435, 2019. doi:10.1109/FOCS.2019.00034.

19 Sergio Cabello, Erin W. Chambers, and Jeff Erickson. Multiple-Source Shortest Paths in Embedded Graphs. *SIAM J. Comput.*, 42:1542–1571, 2013. doi:10.1137/120864271.

20 Diptarka Chakraborty and Keerti Choudhary. New Extremal Bounds for Reachability and Strong-Connectivity Preservers Under Failures. In *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 25:1–25:20, 2020. doi:10.4230/LIPIcs.ICALP.2020.25.

21 Abraham Charnes. Optimality and Degeneracy in Linear Programming. *Econometrica*, 20:160–170, 1952.

22 Shiri Chechik and Sarel Cohen. Distance Sensitivity Oracles with Subcubic Preprocessing Time and Fast Query Time. In *Proccedings of the 52nd Symposium on Theory of Computing (STOC)*, pages 1375–1388, 2020. doi:10.1145/3357713.3384253.

23 Shiri Chechik, Sarel Cohen, Amos Fiat, and Haim Kaplan. $(1+\varepsilon)$-Approximate f-Sensitive Distance Oracles. In *Proceedings of the 28th Symposium on Discrete Algorithms (SODA)*, pages 1479–1496, 2017. doi:10.1137/1.9781611974782.96.

24 Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. f-Sensitivity Distance Oracles and Routing Schemes. *Algorithmica*, 63:861–882, 2012. doi:10.1007/s00453-011-9543-0.

25 Shiri Chechik, Daniel H. Larkin, Liam Roditty, Grant Schoenebeck, Robert E. Tarjan, and Virginia Vassilevska Williams. Better Approximation Algorithms for the Graph Diameter. In *Proceedings of the 25th Symposium on Discrete Algorithms (SODA)*, pages 1041–1052, 2014. doi:10.1137/1.9781611973402.78.

26 Keerti Choudhary and Omer Gold. Extremal Distances in Directed Graphs: Tight Spanners and Near-Optimal Approximation Algorithms. In *Proceedings of the 31st Symposium on Discrete Algorithms (SODA)*, pages 495–514, 2020. doi:10.1137/1.9781611975994.30.

27 Pierluigi Crescenzi, Roberto Grossi, Leonardo Lanzi, and Andrea Marino. On Computing the Diameter of Real-World Directed (Weighted) Graphs. In Ralf Klasing, editor, *Proceedings of the 11th Symposium on Experimental Algorithms (SEA)*, pages 99–110, 2012. doi:10.1007/978-3-642-30850-5_10.

28 Mina Dalirrooyfard, Virginia Vassilevska Williams, Nikhil Vyas, and Nicole Wein. Tight Approximation Algorithms for Bichromatic Graph Diameter and Related Problems. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 47:1–47:15, 2019. doi:10.4230/LIPIcs.ICALP.2019.47.

29 Camil Demetrescu and Mikkel Thorup. Oracles for Distances Avoiding a Link-Failure. In *Proceedings of the 13th Symposium on Discrete Algorithms (SODA)*, pages 838–843, 2002. URL: https://dl.acm.org/citation.cfm?id=545381.545490.

30 Camil Demetrescu, Mikkel Thorup, Rezaul A. Chowdhury, and Vijaya Ramachandran. Oracles for Distances Avoiding a Failed Node or Link. *SIAM Journal on Computing*, 37:1299–1318, 2008. doi:10.1137/S0097539705429847.

31 Ran Duan, Yong Gu, and Hanlin Ren. Approximate Distance Oracles Subject to Multiple Vertex Failures. In *Proceedings of the 32nd Symposium on Discrete Algorithms (SODA)*, pages 2497–2516, 2021. doi:10.1137/1.9781611976465.148.

32 Ran Duan and Seth Pettie. Dual-Failure Distance and Connectivity Oracles. In *Proceedings of the 20th Symposium on Discrete Algorithms (SODA)*, pages 506–515, 2009. URL: http://dl.acm.org/citation.cfm?id=1496770.1496826.

33 Ran Duan and Seth Pettie. Connectivity Oracles for Failure Prone Graphs. In Leonard J. Schulman, editor, *Proceedings of the 42nd Symposium on Theory of Computing (STOC)*, pages 465–474, 2010. doi:10.1145/1806689.1806754.

34 Ran Duan and Hanlin Ren. Maintaining Exact Distances under Multiple Edge Failures. In *Proceedings of the 54th Symposium on Theory of Computing (STOC)*, pages 1093–1101, 2022. doi:10.1145/3519935.3520002.

35 Fabrizio Grandoni and Virginia Vassilevska Williams. Improved Distance Sensitivity Oracles via Fast Single-Source Replacement Paths. In *Proceedings of the 53rd Symposium on Foundations of Computer Science (FOCS)*, pages 748–757, 2012. doi:10.1109/FOCS.2012.17.

36 Fabrizio Grandoni and Virginia Vassilevska Williams. Faster Replacement Paths and Distance Sensitivity Oracles. *ACM Transaction on Algorithms*, 16:15:1–15:25, 2020. doi:10.1145/3365835.

37 Yong Gu and Hanlin Ren. Constructing a Distance Sensitivity Oracle in $O(n^{2.5794}M)$ Time. In *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 76:1–76:20, 2021. doi:10.4230/LIPIcs.ICALP.2021.76.

38 Manoj Gupta and Aditi Singh. Generic Single Edge Fault Tolerant Exact Distance Oracle. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming, (ICALP)*, pages 72:1–72:15, 2018. doi:10.4230/LIPIcs.ICALP.2018.72.

39 David Hartvigsen and Russell Mardon. The All-Pairs Min Cut Problem and the Minimum Cycle Basis Problem on Planar Graphs. *SIAM J. Discret. Math.*, 7:403–418, 1994. doi:10.1137/S0895480190177042.

40 Monika Henzinger, Andrea Lincoln, Stefan Neumann, and Virginia Vassilevska Williams. Conditional Hardness for Sensitivity Problems. In *Proceedings of the 8th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 26:1–26:31, 2017. doi:10.4230/LIPIcs.ITCS.2017.26.

41 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching Is as Easy as Matrix Inversion. *Comb.*, 7:105–113, 1987. doi:10.1007/BF02579206.

42 Hanlin Ren. Improved Distance Sensitivity Oracles with Subcubic Preprocessing Time. *Journal of Computer and System Sciences*, 123:159–170, 2022. doi:10.1016/j.jcss.2021.08.005.

43 Liam Roditty. Approximating the Diameter. In Ming-Yang Kao, editor, *Encyclopedia of Algorithms*, pages 116–117. Springer, New York City, NY, USA, 2016. doi:10.1007/978-1-4939-2864-4_566.

44 Liam Roditty and Virginia Vassilevska Williams. Fast Approximation Algorithms for the Diameter and Radius of Sparse Graphs. In *Proceedings of the 45th Symposium on Theory of Computing (STOC)*, pages 515–524, 2013. doi:10.1145/2488608.2488673.

45 Frank W. Takes and Walter A. Kosters. Determining the Diameter of Small World Networks. In Craig Macdonald, Iadh Ounis, and Ian Ruthven, editors, *Proceedings of the 20th Conference on Information and Knowledge Management (CIKM)*, pages 1191–1196, 2011. doi:10.1145/2063576.2063748.

46 Mikkel Thorup. Undirected Single-Source Shortest Paths with Positive Integer Weights in Linear Time. *Journal of the ACM*, 46:362–394, 1999. doi:10.1145/316542.316548.

47 Oren Weimann and Raphael Yuster. Replacement Paths and Distance Sensitivity Oracles via Fast Matrix Multiplication. *ACM Transactions on Algorithms*, 9:14:1–14:13, 2013. doi:10.1145/2438645.2438646.

Isoperimetric Inequalities for Real-Valued Functions with Applications to Monotonicity Testing

Hadley Black
Department of Computer Science, University of California at Los Angeles, CA, USA

Iden Kalemaj
Department of Computer Science, Boston University, MA, USA

Sofya Raskhodnikova
Department of Computer Science, Boston University, MA, USA

Abstract

We generalize the celebrated isoperimetric inequality of Khot, Minzer, and Safra (SICOMP 2018) for Boolean functions to the case of real-valued functions $f : \{0,1\}^d \to \mathbb{R}$. Our main tool in the proof of the generalized inequality is a new Boolean decomposition that represents every real-valued function f over an arbitrary partially ordered domain as a collection of Boolean functions over the same domain, roughly capturing the distance of f to monotonicity and the structure of violations of f to monotonicity.

We apply our generalized isoperimetric inequality to improve algorithms for testing monotonicity and approximating the distance to monotonicity for real-valued functions. Our tester for monotonicity has query complexity $\widetilde{O}(\min(r\sqrt{d}, d))$, where r is the size of the image of the input function. (The best previously known tester makes $O(d)$ queries, as shown by Chakrabarty and Seshadhri (STOC 2013).) Our tester is nonadaptive and has 1-sided error. We prove a matching lower bound for nonadaptive, 1-sided error testers for monotonicity. We also show that the distance to monotonicity of real-valued functions that are α-far from monotone can be approximated nonadaptively within a factor of $O(\sqrt{d \log d})$ with query complexity polynomial in $1/\alpha$ and the dimension d. This query complexity is known to be nearly optimal for nonadaptive algorithms even for the special case of Boolean functions. (The best previously known distance approximation algorithm for real-valued functions, by Fattal and Ron (TALG 2010) achieves $O(d \log r)$-approximation.)

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Mathematics of computing → Probabilistic algorithms

Keywords and phrases Isoperimetric inequalities, property testing, monotonicity testing

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.25

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2011.09441

Funding *Hadley Black*: This work was supported by NSF Grant CCF-1553605 and Boston University's Data Science Initiative.
Iden Kalemaj: This work was supported by NSF award CCF-1909612 and Boston University's Dean's Fellowship.
Sofya Raskhodnikova: This work was supported by NSF award CCF-1909612.

Acknowledgements We thank Ramesh Krishnan Pallavoor Suresh for useful discussions.

1 Introduction

We investigate the structure of real-valued functions over the domain $\{0,1\}^d$, the d-dimensional hypercube. Our main contribution is a generalization of a powerful tool from the analysis of Boolean functions, specifically, isoperimetric inequalities[1], to the case of real-valued functions. Isoperimetric inequalities for the undirected hypercube were studied by Margulis [34] and Talagrand [40]. Chakrabarty and Seshadhri [19] had a remarkable insight to develop a directed analogue of the Margulis inequality. This beautiful line of work culminated in the directed analogue of the Talagrand inequality proved by Khot, Minzer, and Safra [32]. We refer to this as the KMS inequality. As Khot, Minzer, and Safra explain in their celebrated work, the Margulis inequality follows from the Talagrand inequality and, more generally, the directed analogue of the Talagrand inequality implies all the other inequalities we mentioned. We generalize all these inequalities to the case of real-valued functions[2].

For the directed case, we prove a generalization of the KMS inequality for functions $f\colon \{0,1\}^d \to \mathbb{R}$. To generalize the *undirected* isoperimetric inequalities, we give a property testing interpretation of the Talagrand inequality. With this interpretation, it is easy to show a generalization of the undirected Talagrand inequality to the case of real-valued functions.

Our proofs of the new isoperimetric inequalities reduce the general case to the Boolean case. Our main tool for generalizing the KMS inequality is a new Boolean decomposition theorem that represents every real-valued function f over an arbitrary partially ordered domain as a collection of Boolean functions over the same domain, roughly capturing the distance of f to monotonicity and the structure of violations of f to monotonicity.

We apply our generalized isoperimetric inequality to improve algorithms for testing monotonicity and approximating the distance to monotonicity for real-valued functions. Our algorithm for testing monotonicity is nonadaptive and has 1-sided error. An algorithm is *nonadaptive* if its input queries do not depend on answers to previous queries. A property testing algorithm has *1-sided error* if it always accepts all inputs with the property it is testing. We show that our algorithm for testing monotonicity is optimal among nonadaptive, 1-sided error testers. Our distance approximation algorithm is nonadaptive. Its query complexity is nearly optimal for nonadaptive algorithms, even for the special case of Boolean functions.

1.1 Isoperimetric Inequalities for Real-Valued Functions

We view the domain of functions $f\colon \{0,1\}^d \to \mathbb{R}$ as the vertices of a d-dimensional hypercube. For the directed isoperimetric inequalities, the edges of the hypercube are ordered pairs (x,y), where $x,y \in \{0,1\}^d$ and there is a unique[3] $i \in [d]$ such that $x_i = 0, y_i = 1$, and $x_j = y_j$ for all coordinates $j \in [d] \setminus \{i\}$. This defines a natural partial order on the domain: $x \preceq y$ if $x_i \leq y_i$ for all coordinates $i \in [d]$ or, equivalently, if there is a directed path from x to y in the hypercube. A function $f\colon \{0,1\}^d \to \mathbb{R}$ is *monotone* if $f(x) \leq f(y)$ whenever $x \preceq y$. The distance to monotonicity of a function $f\colon \{0,1\}^d \to \mathbb{R}$, denoted $\varepsilon(f)$, is the minimum of

[1] We discuss isoperimetric inequalities that study the size of the "boundary" between the points on which the function takes value 0 and the points on which it takes value 1. The boundary size is defined in terms of the edges of the d-dimensional hypercube with vertices labeled by the values of the function. The edges of the hypercube might be directed or undirected, depending on the type of the inequality.

[2] Following our initial manuscript, [10] and [14] proved generalizations of the KMS inequality to Boolean functions over hypergrids. We remark that our techniques also extend these inequalities to real-valued functions, $f\colon [n]^d \to \mathbb{R}$. See Section 1.3 for more discussion.

[3] Given a positive integer $\ell \in \mathbb{Z}^+$, we let $[\ell]$ denote the set $\{1, 2, \ldots, \ell\}$.

$|\{x \in \{0,1\}^d \colon f(x) \neq g(x)\}|/2^d$ over all monotone functions $g \colon \{0,1\}^d \to \mathbb{R}$. An edge (x, y) is *violated* by f if $f(x) > f(y)$. Let $\mathcal{S}_f^-$ be the set of violated edges. For $x \in \{0,1\}^d$, let $I_f^-(x)$ be the number of *outgoing* violated edges incident on x, specifically,

$$I_f^-(x) = \left|\left\{y \colon (x,y) \in \mathcal{S}_f^-\right\}\right|.$$

Our main result is the following isoperimetric inequality.

▶ **Theorem 1.1** (Isoperimetric Inequality). *There exists a constant $C > 0$, such that for all functions $f \colon \{0,1\}^d \to \mathbb{R}$,*

$$\mathop{\mathbb{E}}_{\boldsymbol{x} \sim \{0,1\}^d}\left[\sqrt{I_f^-(\boldsymbol{x})}\right] \geq C \cdot \varepsilon(f). \tag{1}$$

Theorem 1.1 is a generalization of the celebrated inequality of Khot, Minzer, and Safra [32] that was strengthened by Pallavoor et al. [36], who proved (1) for the special case of Boolean functions $f \colon \{0,1\}^d \to \{0,1\}$. We show that the same inequality holds for real-valued functions without any dependence on the size of the image of the function. In addition, the constant C is only a factor of 2 smaller than the constant in the inequality of Pallavoor et al.

Applications to monotonicity testing and distance approximation rely on a stronger, "robust" version of Theorem 1.1. The robust version considers an arbitrary 2-coloring $\texttt{col} \colon \mathcal{S}_f^- \to \{\text{red}, \text{blue}\}$ of the violated edges. The color of an edge is used to specify whether the edge is counted towards the lower or the upper endpoint. Let $I^-_{f,\text{red}}(x)$ be the number of *outgoing* red violated edges incident on x, and $I^-_{f,\text{blue}}(x)$ be the number of *incoming* blue violated edges incident on x, specifically,

$$I^-_{f,\text{red}}(x) = \left|\left\{y \colon (x,y) \in \mathcal{S}_f^-,\, \texttt{col}(x,y) = \text{red}\right\}\right|;$$
$$I^-_{f,\text{blue}}(y) = \left|\left\{x \colon (x,y) \in \mathcal{S}_f^-,\, \texttt{col}(x,y) = \text{blue}\right\}\right|.$$

Our next theorem is a generalization of the robust isoperimetric inequality for Boolean functions established by Khot, Minzer, and Safra and strengthened by Pallavoor et al. As before, the constant C is only a factor of 2 smaller for the real-valued case than for the Boolean case.

▶ **Theorem 1.2** (Robust Isoperimetric Inequality). *There exists a constant $C > 0$, such that for all functions $f \colon \{0,1\}^d \to \mathbb{R}$ and colorings $\texttt{col} \colon \mathcal{S}_f^- \to \{\text{red}, \text{blue}\}$,*

$$\mathop{\mathbb{E}}_{\boldsymbol{x} \sim \{0,1\}^d}\left[\sqrt{I^-_{f,\text{red}}(\boldsymbol{x})}\right] + \mathop{\mathbb{E}}_{\boldsymbol{y} \sim \{0,1\}^d}\left[\sqrt{I^-_{f,\text{blue}}(\boldsymbol{y})}\right] \geq C \cdot \varepsilon(f).$$

Note that Theorem 1.2 implies Theorem 1.1 by considering the coloring where all violated edges are red. Therefore, we only present a proof of Theorem 1.2.

1.1.1 Boolean Decomposition

Our main technical contribution is the Boolean decomposition (Theorem 1.3). It allows us to prove Theorem 1.2 by reducing the general case of real-valued functions to the special case of Boolean functions. Theorem 1.3 states that every non-monotone function f can be decomposed into Boolean functions $f_1, f_2, \ldots, f_k$ that collectively preserve the distance to monotonicity of f and violate a subset of the edges violated by f. Crucially, they violate edges in vertex-disjoint subgraphs of the hypercube.

Our Boolean decomposition works for functions over any partially ordered domain. We represent such a domain by a directed acyclic graph (DAG). For a DAG $\mathcal{G}$, we denote its vertex set by $V(\mathcal{G})$ and its edge set by $E(\mathcal{G})$. A DAG $\mathcal{G}$ determines a natural partial order on its vertex set: for all $x, y \in V(\mathcal{G})$, we have $x \preceq y$ if and only if $\mathcal{G}$ contains a path from x to y. A function $f\colon V(\mathcal{G}) \to \mathbb{R}$ is *monotone* if $f(x) \leq f(y)$ whenever $x \preceq y$. An edge (x, y) of $\mathcal{G}$ is *violated* by f if $f(x) > f(y)$. The definitions of $\varepsilon(f)$, the distance of f to monotone, and $\mathcal{S}_f^-$, the set of violated edges, are the same as for the special case of the hypercube.

▶ **Theorem 1.3** (Boolean Decomposition). *Suppose $\mathcal{G}$ is a DAG and $f\colon V(\mathcal{G}) \to \mathbb{R}$ is a function over the vertices of $\mathcal{G}$ that is not monotone. Then, for some $k \geq 1$, there exist Boolean functions $f_1, \ldots, f_k\colon V(\mathcal{G}) \to \{0,1\}$ and vertex-disjoint (induced) subgraphs $\mathcal{H}_1, \ldots, \mathcal{H}_k$ of $\mathcal{G}$ for which the following hold:*

1. $2\sum_{i=1}^{k} \varepsilon(f_i) \geq \varepsilon(f)$.
2. $\mathcal{S}_{f_i}^- \subseteq \mathcal{S}_f^- \cap E(\mathcal{H}_i)$ *for all* $i \in [k]$.

We derive Theorem 1.2 from Theorem 1.3 in Section 2 and prove Theorem 1.3 in Section 3.

A natural first attempt to proving Theorem 1.1 is to try reducing to the special case of Boolean functions (the KMS inequality) via a thresholding argument. Given $f\colon \{0,1\}^d \to \mathbb{R}$ and $t \in \mathbb{R}$, define $h_t\colon \{0,1\}^d \to \{0,1\}$ to be $h_t(x) = 1$ iff $f(x) > t$. Clearly, this can only reduce the left-hand side of (1) since the influential edges of h_t are a subset of the influential edges of f. Thus, if there exists some $t \in \mathbb{R}$ such that $\varepsilon(h_t) = \Omega(\varepsilon(f))$, then applying the KMS inequality to h_t would show that the inequality also holds for f. In fact, this technique easily allows us to reduce the *undirected* inequality for the real-valued case to the Boolean case, without any significant additional ideas (see Section 7 of the full version [11] for details). However, in the directed setting, a simple argument shows that there exists f for which $\varepsilon(h_t) \leq \varepsilon(f)/r$ for all $t \in \mathbb{R}$, where r is the size of the image of f. Thus, we use additional ideas to prove Theorem 1.1 by a reduction to the KMS inequality. The highly structured decomposition of Theorem 1.3 gives a collection of vertex-disjoint subgraphs $\mathcal{H}_1, \ldots, \mathcal{H}_k$ of the directed hypercube where, in each $\mathcal{H}_i$, an independent "variable thresholding rule" can be applied, yielding the Boolean function f_i. The "threshold" for each vertex x in $\mathcal{H}_i$ depends on the values of the function at a particular set of vertices reachable from x.

The Boolean decomposition is quite powerful: in addition to enabling us to prove the new isoperimetric inequality, it can be used to easily derive a lower bound on the number of edges violated by a real-valued function directly from the bound for the Boolean case, without relying on Theorem 1.2. This bound is used to analyze the edge tester for monotonicity whose significance is described in Section 1.2. The early works on monotonicity testing [30, 25, 39] have shown that $|\mathcal{S}_f^-| \geq \varepsilon(f) \cdot 2^d$ for every Boolean function f on the domain $\{0,1\}^d$. In other words, the number of edges violated by f is at least the number of points on which the value of the function has to change to make it monotone. This bound was generalized to the case of real-valued functions by [25, 39] who showed that $|\mathcal{S}_f^-| \geq (\varepsilon(f)/\lceil \log r \rceil) \cdot 2^d$ for every real-valued function f on the domain $\{0,1\}^d$ and with image size r. (The size of the image of f is the number of distinct values it takes.) Chakrabarty and Seshadhri [17] improved this bound by a factor of $\Theta(\log r)$, thus removing the dependence on the size of the image of the function. Our Boolean decomposition of a real-valued function f in terms of Boolean functions $f_1, \ldots, f_k$, given by Theorem 1.3, yields this result of [17] as an immediate corollary of the special case for Boolean functions:

$$|\mathcal{S}_f^-| \geq \sum_{i=1}^{k} |\mathcal{S}_{f_i}^-| \geq \sum_{i=1}^{k} \varepsilon(f_i) \cdot 2^d \geq \varepsilon(f) \cdot 2^{d-1},$$

where the inequalities follow by first applying Item 2 of Theorem 1.3, then applying the bound for the Boolean case, and, finally, applying Item 1 of Theorem 1.3.

1.1.2 Undirected Isoperimetric Inequality for Real-Valued Functions

The original isoperimetric inequality of Talagrand [40] treats the domain $\{0,1\}^d$ as an undirected hypercube. An undirected edge $\{x,y\}$ is *influential* if $f(x) \neq f(y)$. Let $I_f(x)$ be the number of influential edges $\{x,y\}$ incident on $x \in \{0,1\}^d$ for which $f(x) > f(y)$. This definition ensures that each influential edge is counted towards $I_f(x)$ for exactly one vertex x. The variance $\mathrm{var}(f)$ of a Boolean function is defined as $p_0(1-p_0)$, where p_0 is the probability that $f(x) = 0$ for a uniformly random point x in the domain. Talagrand [40] proved the following.

▶ **Theorem 1.4** (Talagrand Inequality [40]). *For all functions* $f\colon \{0,1\}^d \to \{0,1\}$,

$$\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\{0,1\}^d}\left[\sqrt{I_f(\boldsymbol{x})}\right] \geq \sqrt{2}\ \mathrm{var}(f). \tag{2}$$

Before generalizing Theorem 1.4 to real-valued functions, we reinterpret it using a property testing notion. Observe that the natural definition of the variance of a real-valued function results in a quantity that depends on specific values of the function, whereas whether an edge is influential depends only on whether the values on its endpoints are different and not on the specific values themselves. So, variance is not a suitable notion for generalizing this inequality. We replace the variance of f with the distance of f to constant, denoted $\mathtt{dist}(f, \mathbf{const})$, i.e., the minimum of $\Pr_{\boldsymbol{x}\sim\{0,1\}^d}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})]$ over all constant functions $g\colon \{0,1\}^d \to \mathbb{R}$. For a Boolean function f, the distance to constant is $\min\{p_0, (1-p_0)\}$ and, therefore, the left-hand side of (2) is at least $\mathtt{dist}(f, \mathbf{const})/\sqrt{2}$. Next, we state our generalization of Talagrand's inequality. See Section 7 of the full version [11] for the proof.

▶ **Theorem 1.5** (Undirected Isoperimetric Inequality). *For all functions* $f\colon \{0,1\}^d \to \mathbb{R}$,

$$\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\{0,1\}^d}\left[\sqrt{I_f(\boldsymbol{x})}\right] \geq \frac{\mathtt{dist}(f, \mathbf{const})}{2\sqrt{2}}.$$

Note that natural generalizations of the Margulis inequality and the inequality of Chakrabarty and Seshadhri to the real range follow from Theorem 1.2 (for the the special case of Boolean functions, the implication is discussed in [32], and it holds for the real range for the same reasons).

1.2 Applications of Our Isoperimetric Inequality for Real-Valued Functions

We apply our generalized isoperimetric inequality (Theorem 1.2) to improve algorithms for testing monotonicity and approximating the distance to monotonicity for real-valued functions.

1.2.1 Monotonicity Testing

Monotonicity of functions, first studied in the context of property testing by Goldreich et al. [30], is one of the most widely investigated properties in this model [26, 25, 39, 33, 29, 1, 28, 31, 3, 38, 2, 7, 15, 12, 17, 18, 19, 13, 16, 21, 5, 23, 35, 8, 32, 20, 9]. A function is ε-far from monotone if its distance to monotonicity is at least ε; otherwise, it is ε-close to monotone. An ε-tester for monotonicity is a randomized algorithm that, given a parameter $\varepsilon \in (0,1)$ and oracle access to a function f, accepts with probability at least $2/3$ if f is monotone and rejects with probability at least $2/3$ if f is ε-far from monotone. Prior to

our work, the best monotonicity tester for real-valued functions was the *edge tester*. The edge tester, introduced by [30], queries the values of f on the endpoints of uniformly random edges of the hypercube and rejects if it finds a violated edge. As we discussed in Section 1.1, a series of works [30, 25, 39, 17] proved lower bounds on $|\mathcal{S}_f^-|$, the number of violated edges, resulting in the tight analysis of the edge tester for both Boolean and real-valued functions: $O(d/\varepsilon)$ queries are sufficient (and also necessary, e.g., for $f(x) = 1 - x_1$, the anti-dictator function). For many years, it remained open whether an $o(d)$-query tester for monotonicity existed, until a sequence of breakthroughs [19, 22, 32] designed testers for Boolean functions with query complexity $\widetilde{O}(d^{7/8}), \widetilde{O}(d^{5/6})$, and finally $\widetilde{O}(\sqrt{d})$. Prior to our work, the same question remained open for functions with image size, r, greater than 2.

We show that when r is small compared to d, monotonicity can be tested with $o(d)$ queries. (Note that $r \leq 2^d$.)

▶ **Theorem 1.6.** *There exists a nonadaptive, 1-sided error ε-tester for monotonicity of functions $f\colon \{0,1\}^d \to \mathbb{R}$ that makes $\widetilde{O}\Big(\min\big(\frac{r\sqrt{d}}{\varepsilon^2}, \frac{d}{\varepsilon}\big)\Big)$ queries and works for all functions f with image size r.*

The proof of Theorem 1.6 (in Section 4) heavily relies on the generalized isoperimetric inequality of Theorem 1.2. We extend several other combinatorial properties of Boolean functions to real-valued functions. In particular, the persistence of a vertex $x \in \{0,1\}^d$ is a key combinatorial concept in the analysis. A vertex $x \in \{0,1\}^d$ is τ-persistent if, with high probability, a random walk that starts at x and takes τ steps in the d-dimensional directed hypercube ends at a vertex y for which $f(y) \leq f(x)$. As we show, the upper bound on the number of vertices which are not τ-persistent grows linearly with the distance τ *and* the image size r. For the tester analysis, one needs to carefully choose the distance parameter τ for which many vertices are τ-persistent. In particular, this value of τ also depends on the image size r, resulting in the linear dependence on r in the query complexity of the tester.

1.2.2 Our Lower Bound for Testing Monotonicity

We show that our monotonicity tester is optimal among nonadaptive, 1-sided error testers.

▶ **Theorem 1.7.** *There exists a constant $\varepsilon > 0$, such that for all $d, r \in \mathbb{N}$, every nonadaptive, 1-sided error ε-tester for monotonicity of functions $f\colon \{0,1\}^d \to [r]$ requires $\Omega(\min(r\sqrt{d}, d))$ queries.*

We prove Theorem 1.7 by generalizing a construction of Fischer et al. [29] that showed that nonadaptive, 1-sided error monotonicity testers of Boolean functions must make $\Omega(\sqrt{d})$ queries. We refer the reader to Section 6 of the full version [11] for the proof. Blais et al. [12] demonstrated that every tester for monotonicity over the d-dimensional hypercube domain requires $\Omega(\min(d, r^2))$ queries. Our lower bound is stronger when $r \in [2, \sqrt{d}]$, although it applies only to nonadaptive, 1-sided error algorithms.

1.2.3 Approximating the Distance to Monotonicity

Motivated by the desire to handle noisy inputs, Parnas et al. [38] generalized the property testing model to tolerant testing. There is a direct connection between tolerant testing of a property and approximating the distance to the property with additive and multiplicative error in the sense that these problems can be reduced to each other with the right setting of parameters and have the same query complexity up to logarithmic factors (see, e.g., [38, Claim 2] and [36, Theorem A.1]). One clean way to state distance approximation guarantees

is to replace the additive error α with the promise that the input function is α-far from the property, as specified in the following definition. A randomized *c-approximation algorithm for the distance to monotonicity,* where $c > 1$, is given a parameter $\alpha \in (0, 1)$ and oracle access to a function $f\colon \{0,1\}^d \to \mathbb{R}$ that is α-far from monotone. It outputs an estimate $\hat{\varepsilon}$ that, with probability at least $2/3$, satisfies $\varepsilon(f) \leq \hat{\varepsilon} \leq c \cdot \varepsilon(f)$.

Fattal and Ron [27] studied the problem of approximating the distance to monotonicity for real-valued functions over the hypergrid domain $[n]^d$. For the special case of the hypercube domain, they give an $O(d \log r)$-approximation algorithm for functions with image size r that makes $\text{poly}(d, 1/\alpha)$ queries. Theorem 1.2 allows us to improve on their result, by showing that the algorithm of Pallavoor et al. [36] for approximating the distance to monotonicity of Boolean functions also works for real-valued functions, without any loss in the approximation guarantee.

▶ **Theorem 1.8.** *There exists a nonadaptive $O(\sqrt{d \log d})$-approximation algorithm for the distance to monotonicity that, given a parameter $\alpha \in (0, 1)$ and oracle access to a function $f\colon \{0,1\}^d \to \mathbb{R}$ that is α-far from monotone, makes $\text{poly}(d, 1/\alpha)$ queries.*

Pallavoor et al. prove that this approximation ratio is nearly optimal for nonadaptive algorithms, even for the special case of Boolean functions. We also note that, by the connection between tolerant testing and erasure-resilient testing observed by Dixit et al. [24], our Theorem 1.8 implies the existence of an erasure-resilient ε-tester for monotonicity of functions $f : \{0,1\}^d \to \mathbb{R}$ that can handle up to $\Theta(\varepsilon/\sqrt{d \log d})$ erasures with query complexity $\text{poly}(d, 1/\varepsilon)$. The tester of Dixit et al. could handle only $O(\varepsilon/d)$ erasures. For the proof of Theorem 1.8, we refer the reader to Section 5 of the full version [11].

1.3 Other Prior Work on Monotonicity Testing and Open Questions

The query complexity of monotonicity testing of Boolean functions over the hypercube has been resolved for nonadaptive testers by Chen et al. [21, 23] who proved a lower bound of $\widetilde{\Omega}(\sqrt{d})$. For adaptive testers, the best lower bound known to date is $\widetilde{\Omega}(d^{1/3})$, also shown by [23]. It is an open question whether adaptive algorithms can do better than nonadaptive ones for functions over the hypercube domain, both in the case of Boolean functions and, more generally, for functions with small image size. As we mentioned before, there is a lower bound of $\Omega(d)$ for functions with image size $\Omega(\sqrt{d})$ [12].

Monotonicity testing has also been studied for functions on other types of domains, including general partially ordered domains [29], with particular attention to the hypergrid domain $[n]^d$. (It has also been investigated in the context where the distance to monotonicity is the normalized L_p distance instead of the Hamming distance [6], but we focus our attention here on the Hamming distance.) When $d = 1$, monotonicity testing on the hypergrid $[n]$ is equivalent to testing sortedness of n-element arrays. This problem was introduced by Ergun et al. [26]. Its query complexity has been completely pinned down in terms of n and ε by [26, 28, 18, 4]: it is $\Theta(\frac{\log(\varepsilon n)}{\varepsilon})$. Pallavoor et al. [35, 37] considered the setting when the tester is given an additional parameter r, the number of distinct elements in the array, and obtained an $O((\log r)/\varepsilon)$-query algorithm. There are also lower bounds for this setting: $\Omega(\log r)$ for nonadaptive algorithms by [13] and $\Omega(\frac{\log r}{\log \log r})$ for all testers for the case when $r = n^{1/3}$ by [4].

For general d, Black et al. [8, 9] gave an $\widetilde{O}(d^{5/6})$-query tester for Boolean functions $f\colon [n]^d \to \{0,1\}$. For real-valued functions, Chakrabarty and Seshadhri [17, 18] proved basically matching upper and lower bounds of $O((d \log n)/\varepsilon)$ and $\Omega((d \log n - \log \varepsilon^{-1})/\varepsilon)$. However, their lower bound only applies for functions with a large image. Pallavoor et al. [35]

gave an $O(\frac{d}{\varepsilon} \cdot \log \frac{d}{\varepsilon} \cdot \log r)$-query tester, where r, the size of the image, is given to the tester as a parameter. It remains open whether there is an $\widetilde{O}(\sqrt{d})$-query tester for Boolean functions on the hypergrid domain.

1.3.1 Discussion of Results Published After our Initial Manuscript

Recently, in independent works, [10] and [14] showed generalizations of the isoperimetric inequality of [32] to Boolean functions on general hypergrids (see [10, Theorem 1.4] and [14, Theorem 1.3]). These works obtain $\widetilde{O}(n\sqrt{d}/\varepsilon^2)$-query and $\widetilde{O}(n^3\sqrt{d}/\varepsilon^2)$-query nonadaptive, 1-sided error monotonicity testers, respectively, for such functions. Our Boolean decomposition (Theorem 1.3) implies that these isoperimetric inequalities also hold for functions $f\colon [n]^d \to \mathbb{R}$ by the approach described in Section 2. We also believe that this should imply the existence of an $\widetilde{O}(rn\sqrt{d}/\varepsilon^2)$-query tester for functions $f\colon [n]^d \to [r]$. A possible approach to proving this could be to generalize the analysis given in Section 7 of [10] to the case of range $[r]$. Presumably, this would follow the same approach as in our Section 4 in which we prove Theorem 1.6, but adapted to hypergrids. Since [10, 14] were published well after our initial manuscript, we will refrain from going into further details on their relationship with our results.

2 Directed Talagrand Inequality for Real-Valued Functions

In this section, we use our Boolean decomposition (Theorem 1.3) to prove Theorem 1.2, which easily implies the non-robust version (Theorem 1.1) as we point out in the introduction. Let $f\colon \{0,1\}^d \to \mathbb{R}$ be a non-monotone function over the d-dimensional hypercube and let $\texttt{col}\colon \mathcal{S}_f^- \to \{\text{red}, \text{blue}\}$ be an arbitrary 2-coloring of $\mathcal{S}_f^-$. Given $x \in \{0,1\}^d$ and a subgraph $\mathcal{H}$ of the d-dimensional hypercube, we define the quantities

$$I^-_{f,\text{red},\mathcal{H}}(x) = \left|\left\{y\colon (x,y) \in \mathcal{S}_f^- \cap E(\mathcal{H}),\, \texttt{col}(x,y) = \text{red}\right\}\right|;$$
$$I^-_{f,\text{blue},\mathcal{H}}(y) = \left|\left\{x\colon (x,y) \in \mathcal{S}_f^- \cap E(\mathcal{H}),\, \texttt{col}(x,y) = \text{blue}\right\}\right|.$$

Let $f_1, \ldots, f_k\colon \{0,1\}^d \to \{0,1\}$ be the Boolean functions and $\mathcal{H}_1, \ldots, \mathcal{H}_k$ be the vertex-disjoint subgraphs of the d-dimensional hypercube that are guaranteed by Theorem 1.3. Let C' denote the constant from the robust Boolean isoperimetric inequality (Theorem 2.7 of [36]) that is hidden by Ω. We have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\{0,1\}^d}\left[\sqrt{I^-_{f,\text{red}}(\boldsymbol{x})}\right] + \mathop{\mathbb{E}}_{\boldsymbol{y}\sim\{0,1\}^d}\left[\sqrt{I^-_{f,\text{blue}}(\boldsymbol{y})}\right] \\
&\geq \mathop{\mathbb{E}}_{\boldsymbol{x}}\left[\sqrt{I^-_{f,\text{red},\bigcup_{i=1}^k \mathcal{H}_i}(\boldsymbol{x})}\right] + \mathop{\mathbb{E}}_{\boldsymbol{y}}\left[\sqrt{I^-_{f,\text{blue},\bigcup_{i=1}^k \mathcal{H}_i}(\boldsymbol{y})}\right] && (3)\\
&= \sum_{i=1}^k \left(\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[\sqrt{I^-_{f,\text{red},\mathcal{H}_i}(\boldsymbol{x})}\right] + \mathop{\mathbb{E}}_{\boldsymbol{y}}\left[\sqrt{I^-_{f,\text{blue},\mathcal{H}_i}(\boldsymbol{y})}\right]\right) && (4)\\
&\geq \sum_{i=1}^k \left(\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[\sqrt{I^-_{f_i,\text{red},\mathcal{H}_i}(\boldsymbol{x})}\right] + \mathop{\mathbb{E}}_{\boldsymbol{y}}\left[\sqrt{I^-_{f_i,\text{blue},\mathcal{H}_i}(\boldsymbol{y})}\right]\right) && (5)\\
&= \sum_{i=1}^k \left(\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[\sqrt{I^-_{f_i,\text{red}}(\boldsymbol{x})}\right] + \mathop{\mathbb{E}}_{\boldsymbol{y}}\left[\sqrt{I^-_{f_i,\text{blue}}(\boldsymbol{y})}\right]\right) && (6)
\end{aligned}$$

$$\geq \sum_{i=1}^{k} C' \cdot \varepsilon(f_i) \tag{7}$$

$$\geq \frac{C' \cdot \varepsilon(f)}{2}. \tag{8}$$

The inequality (3) holds because $\bigcup_{i=1}^{k} \mathcal{H}_i$ is a subgraph of the d-dimensional hypercube. The equality (4) holds because the $\mathcal{H}_i$'s are vertex-disjoint. The inequality (5) holds since $\mathcal{S}^-_{f_i} \subseteq \mathcal{S}^-_f$ and the equality (6) holds since $\mathcal{S}^-_{f_i} \subseteq E(\mathcal{H}_i)$ (these are both by item 2 of Theorem 1.3). Finally, (7) is due to [36, Theorem 2.7] and (8) is due to item 1 of Theorem 1.3.

3 Boolean Decomposition: Proof of Theorem 1.3

In this section, we prove Boolean Decomposition (Theorem 1.3). Our results consider any partially ordered domain, which we represent by a DAG $\mathcal{G}$. The *transitive closure* of $\mathcal{G}$, denoted $\mathrm{TC}(\mathcal{G})$, is the graph with vertex set $V(\mathcal{G})$ and edge set $\{(x, y) \colon x \prec y\}$. The *violation graph* of f is the graph $(V(\mathcal{G}), E')$, where E' is the set of edges of $\mathrm{TC}(\mathcal{G})$ violated by f.

In Section 3.1, we define the key notion of sweeping graphs and identify some of their important properties. In Section 3.2, we prove a general lemma that shows how to use a matching M in $\mathrm{TC}(\mathcal{G})$ to find vertex-disjoint sweeping graphs in $\mathcal{G}$ satisfying a "matching rearrangement" property. The techniques in Section 3.1 and Section 3.2 are inspired by the techniques of [8] used to analyze Boolean functions on the hypergrid domain, $[n]^d$. In Section 3.3, we apply our matching decomposition lemma to a carefully chosen matching to obtain the subgraphs $\mathcal{H}_1, \ldots, \mathcal{H}_k$. Finally, in Section 3.4, we define the Boolean functions $f_1, \ldots, f_k$ and complete the proof of Theorem 1.3.

3.1 Sweeping Graphs and Their Properties

Given a graph $\mathcal{G}$ and two subgraphs $\mathcal{H}_1$ and $\mathcal{H}_2$, we define the union $\mathcal{H}_1 \cup \mathcal{H}_2$ to be the graph with vertex set $V(\mathcal{H}_1) \cup V(\mathcal{H}_2)$ and edge set $E(\mathcal{H}_1) \cup E(\mathcal{H}_2)$.

▶ **Definition 3.1** ((S,T)-Sweeping Graphs). *Given a DAG $\mathcal{G}$ and $s, t \in V(\mathcal{G})$, define $\mathcal{H}(s,t)$ to be the subgraph of $\mathcal{G}$ formed by the union of all directed paths in $\mathcal{G}$ from s to t. Given two disjoint subsets $S, T \subseteq V(\mathcal{G})$, define the* (S,T)-sweeping graph*, denoted $\mathcal{H}(S,T)$, to be the union of directed paths in $\mathcal{G}$ that start from some $s \in S$ and end at some $t \in T$. That is,*

$$\mathcal{H}(S,T) = \bigcup_{(s,t) \in S \times T} \mathcal{H}(s,t).$$

Note that if $s \not\preceq t$ then $\mathcal{H}(s,t) = \emptyset$.

We now prove three properties of sweeping graphs which we use in Section 3.4 to analyze our functions $f_1, \ldots, f_k$. Given disjoint sets $S, T \subseteq V(\mathcal{G})$ and $z \in V(\mathcal{H}(S,T))$, define the sets

$$S(z) = \{s \in S \colon s \preceq z\} \text{ and } T(z) = \{t \in T \colon z \preceq t\}.$$

▷ Claim 3.2 (Properties of Sweeping Graphs). Let $\mathcal{G}$ be a DAG and $S, T \subseteq V(\mathcal{G})$ be disjoint sets.

1. *(Property of Nodes in a Sweeping Graph):* If $z \in V(\mathcal{H}(S,T))$ then $S(z) \neq \emptyset$ and $T(z) \neq \emptyset$.
2. *(Property of Nodes Outside of a Sweeping Graph):* If $z \in V(\mathcal{G}) \setminus V(\mathcal{H}(S,T))$ then at most one of the following is true: (a) $\exists y \in V(\mathcal{H}(S,T))$ such that $z \prec y$, (b) $\exists x \in V(\mathcal{H}(S,T))$ such that $x \prec z$.
3. *(Sweeping Graphs are Induced):* If $x, y \in V(\mathcal{H}(S,T))$ and $(x,y) \in E(\mathcal{G})$ then $(x,y) \in E(\mathcal{H}(S,T))$.

Proof. Property 1 holds by definition of the sweeping graph $\mathcal{H}(S,T)$. If $z \in V(\mathcal{H}(S,T))$, then, by definition of $\mathcal{H}(S,T)$, there exist $s \in S$ and $t \in T$ for which z belongs to some directed path from s to t. That is, $z \in V(\mathcal{H}(s,t))$. Thus, $s \in S(z)$ and $t \in T(z)$, and property 1 holds.

We now prove property 2. Suppose, for the sake of contradiction, that there exist $x, y, z \in V(\mathcal{G})$ for which $x, y \in V(\mathcal{H}(S,T))$, $z \notin V(\mathcal{H}(S,T))$, and $x \prec z \prec y$. By property 1, there exist some $s \in S(x)$ and some $t \in T(y)$. Then $s \preceq x \prec z \prec y \preceq t$ and, consequently, z belongs to some directed path from s to t. Thus, $z \in V(\mathcal{H}(s,t))$, and so $z \in V(\mathcal{H}(S,T))$. This is a contradiction.

We now prove property 3. Suppose $x, y \in V(\mathcal{H}(S,T))$ and $(x,y) \in E(\mathcal{G})$. By property 1, there exist $s \in S$ and $t \in T$ for which $s \preceq x$ and $y \preceq t$. Since $(x,y) \in E(\mathcal{G})$, we have $x \prec y$ and so $s \preceq x \prec y \preceq t$. Thus, the edge (x,y) belongs to a directed path from s to t. That is, $(x,y) \in E(\mathcal{H}(s,t))$ and so $(x,y) \in E(\mathcal{H}(S,T))$. ◁

3.2 Matching Decomposition Lemma for DAGs

In this section, we prove the following matching decomposition lemma. Recall that $\mathrm{TC}(\mathcal{G})$ denotes the transitive closure of $\mathcal{G}$, which is the graph with vertex set $V(\mathcal{G})$ and edge set $\{(x,y) \colon x \prec y\}$. Consider a matching M in $\mathrm{TC}(\mathcal{G})$. We represent $M \colon S \to T$ as a bijection between two disjoint sets $S, T \subseteq V(\mathcal{G})$ of the same size for which $s \prec M(s)$ for all $s \in S$. For a set $S' \subseteq S$, define $M(S') = \{M(s) \colon s \in S'\}$. Note that for convenience we will sometimes abuse notation and represent M as the set of pairs, $\{(s, M(s)) \colon s \in S\}$, instead of as a bijection.

▶ **Lemma 3.3** (Matching Decomposition Lemma for DAGs). *For every DAG $\mathcal{G}$ and every matching $M \colon S \to T$ in $\mathrm{TC}(\mathcal{G})$, there exist partitions $(S_i \colon i \in [k])$ of S and $(T_i \colon i \in [k])$ of T, where $M(S_i) = T_i$ for all $i \in [k]$, and the following hold.*

1. (Sweeping Graph Disjointness): *$V(\mathcal{H}(S_i, T_i)) \cap V(\mathcal{H}(S_j, T_j)) = \emptyset$ for all $i \neq j \in [k]$.*
2. (Matching Rearrangement Property): *For all $i \in [k]$ and $(x,y) \in S_i \times T_i$, if $x \prec y$ then there exists a matching $\widehat{M} \colon S_i \to T_i$ in $\mathrm{TC}(\mathcal{G})$ for which $(x,y) \in \widehat{M}$.*

Proof. In Algorithm 1, we show how to construct partitions $(S_i \colon i \in [k])$ for S and $(T_i \colon i \in [k])$ for T from a matching M in $\mathrm{TC}(\mathcal{G})$. We use the following notion of conflicting pairs.

▶ **Definition 3.4** (Conflicting Pairs). *Given a DAG $\mathcal{G}$ and four disjoint sets $X, Y, X', Y' \subset V(\mathcal{G})$, we say the two pairs (X,Y) and (X',Y') conflict if $V(\mathcal{H}(X,Y)) \cap V(\mathcal{H}(X',Y')) \neq \emptyset$.*

Algorithm 1 Algorithm for constructing conflict-free pairs from a matching M.

Require: A DAG $\mathcal{G}$ and a matching $M \colon S \to T$ in $\mathrm{TC}(\mathcal{G})$.

$\mathcal{Q}_0 \leftarrow \{(\{x\}, \{y\}) \colon (x,y) \in M\}$ ▷ Initialize pairs using M
for $s \geq 0$ **do**
  if two pairs $(X,Y) \neq (X',Y') \in \mathcal{Q}_s$ *conflict* **then**
    $\mathcal{Q}_{s+1} \leftarrow (\mathcal{Q}_s \setminus \{(X,Y),(X',Y')\}) \cup \{(X \cup X', Y \cup Y')\}$ ▷ Merge conflicting pairs
  else
    $s^* \leftarrow s$ and **return** $\mathcal{Q}_{s^*}$ ▷ Terminate when there are no conflicts

The following observation is apparent and by design of Algorithm 1.

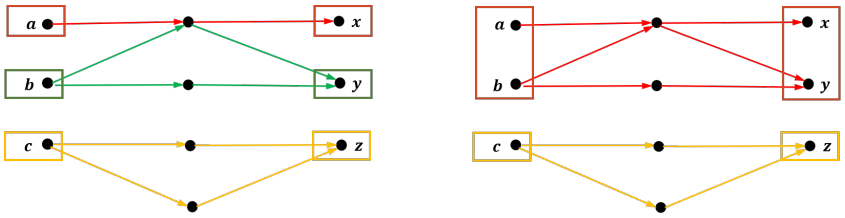


Figure 1 An illustration for Algorithm 1 with input matching $M = \{(a,x),(b,y),(c,z)\}$. We initialize $\mathcal{Q}_0 = \{(\{a\},\{x\}),(\{b\},\{y\}),(\{c\},\{z\})\}$. The pairs $(\{a\},\{x\})$ and $(\{b\},\{y\})$ conflict, so we merge them to obtain a new and final collection $Q_1 = \{(\{a,b\},\{x,y\}),(\{c\},\{z\})\}$.

▶ **Observation 3.5** (Loop Invariants of Algorithm 1). *For all $s \in \{0,1,\ldots,s^*\}$, (a) $M(X) = Y$ for all $(X,Y) \in \mathcal{Q}_s$, (b) $(X : (X,\cdot) \in \mathcal{Q}_s)$ is a partition of S, and (c) $(Y : (\cdot,Y) \in \mathcal{Q}_s)$ is a partition of T.*

Given a matching $M : S \to T$ in $\mathrm{TC}(\mathcal{G})$, we run Algorithm 1 to obtain the set $\mathcal{Q}_{s^*}$. See Fig. 1 for an illustration. Define $k = |\mathcal{Q}_{s^*}|$ and let $\{(S_i,T_i) : i \in [k]\}$ be the set of pairs in $\mathcal{Q}_{s^*}$. By Observation 3.5, $(S_i : i \in [k])$ is a partition of S, $(T_i : i \in [k])$ is a partition of T, and $M(S_i) = T_i$ for all $i \in [k]$. Item 1 of Lemma 3.3 holds since Algorithm 1 terminates at step s only when all pairs in $\mathcal{Q}_s$ are non-conflicting (recall Definition 3.4). Thus, to prove Lemma 3.3 it only remains to prove item 2. To do so, we prove the following Claim 3.6, that easily implies item 2. Note that while we only require Claim 3.6 to hold for the special case of $s = s^*$, using an inductive argument on s allows us to give a proof for all $s \in \{0,1,\ldots,s^*\}$.

▷ Claim 3.6 (Rematching Claim). For all $s \in \{0,1,\ldots,s^*\}$, pairs $(X,Y) \in \mathcal{Q}_s$, and $(x,y) \in X \times Y$, there exists a matching $\widehat{M} : X \setminus \{x\} \to Y \setminus \{y\}$ in $TC(\mathcal{G})$.

Proof. The proof is by induction on s. For the base case, if $s = 0$, then, by inspection of Algorithm 1, for $(X,Y) \in \mathcal{Q}_0$, we must have $X = \{x\}$ and $Y = \{y\}$. Thus, setting $\widehat{M} = \emptyset$ trivially proves the claim.

Now let $s > 0$. Fix some $(X,Y) \in \mathcal{Q}_s$ and $(x,y) \in X \times Y$. Let $(X_1,Y_1),(X_2,Y_2) \in \mathcal{Q}_{s-1}$ be the pairs of sets in $\mathcal{Q}_{s-1}$ for which $x \in X_1$ and $y \in Y_2$. First, if $(X_1,Y_1) = (X_2,Y_2)$, then by induction there exists a matching $\widehat{M}' : X_1 \setminus \{x\} \to Y_1 \setminus \{y\}$ in $\mathrm{TC}(\mathcal{G})$. Note that by definition of Algorithm 1, we must have $X_1 \subseteq X$ and $Y_1 \subseteq Y$. Then the required matching is $\widehat{M} = \widehat{M}' \cup M|_{X \setminus X_1}$ where $M|_{(\cdot)}$ denotes the restriction of the original matching M to the set $(\cdot)$. Suppose $(X_1,Y_1) \neq (X_2,Y_2)$. This is the interesting case, and we give an accompanying illustration in Fig. 2. By definition of Algorithm 1, it must be that (X_1,Y_1) and (X_2,Y_2) *conflict* (recall Definition 3.4) and were merged to form $X = X_1 \cup X_2$ and $Y = Y_1 \cup Y_2$. Thus, there exists some vertex $z \in V(\mathcal{H}(X_1,Y_1)) \cap V(\mathcal{H}(X_2,Y_2))$ and $x_1 \in X_1, y_1 \in Y_1, x_2 \in X_2, y_2 \in Y_2$ for which $x_1 \preceq z \preceq y_1$ and $x_2 \preceq z \preceq y_2$.

We now invoke the inductive hypothesis to get matchings $\widehat{M}_1 : X_1 \setminus \{x\} \to Y_1 \setminus \{y_1\}$ and $\widehat{M}_2 : X_2 \setminus \{x_2\} \to Y_2 \setminus \{y\}$ in $\mathrm{TC}(\mathcal{G})$. Observe that $x_2 \preceq z \preceq y_1$ and thus we can match x_2 and y_1. The required matching in $\mathrm{TC}(\mathcal{G})$ is $\widehat{M} = \widehat{M}_1 \cup \widehat{M}_2 \cup \{(x_2,y_1)\}$. ◁

We conclude the proof of Lemma 3.3 by showing that Claim 3.6 implies item 2. We are given $(S_i,T_i) \in \mathcal{Q}_{s^*}$ for some $i \in [k]$ and $(x,y) \in S_i \times T_i$ where $x \prec y$. By Claim 3.6, there exists a matching $\widehat{M}' : S_i \setminus \{x\} \to T_i \setminus \{y\}$ in $\mathrm{TC}(\mathcal{G})$. We set $\widehat{M} = \widehat{M}' \cup \{(x,y)\}$. Since $x \prec y$, the final matching $\widehat{M} : S_i \to T_i$ is a matching in $\mathrm{TC}(\mathcal{G})$ which contains the pair (x,y). ◀

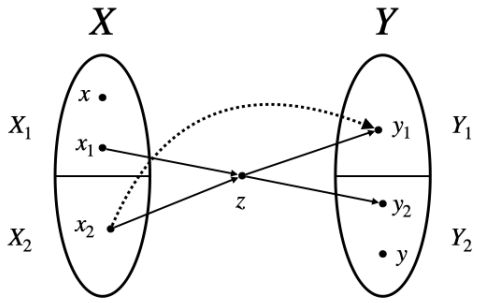


Figure 2 An illustration for the case of $(X_1, Y_1) \neq (X_2, Y_2)$ in the proof of Claim 3.6. The solid lines represent directed paths. The dotted line represents the pair (x_2, y_1) added to obtain the final matching $\widehat{M}$. The only vertices of $X \cup Y$ not participating in $\widehat{M}$ are x and y.

3.3 Specifying a Matching to Construct the Subgraphs $\mathcal{H}_1, \ldots, \mathcal{H}_k$

In this section, we apply Lemma 3.3 to a carefully chosen matching M in order to construct our vertex-disjoint subgraphs $\mathcal{H}_1, \ldots, \mathcal{H}_k$.

▶ **Definition 3.7** (Max-weight, Min-cardinality Matching). *A matching M in $\mathrm{TC}(\mathcal{G})$ is a max-weight, min-cardinality matching for f if M maximizes $\sum_{(x,y)\in M}(f(x) - f(y))$ and among such matchings minimizes $|M|$.*

Henceforth, let M denote a max-weight, min-cardinality matching. Let S and T denote the set of lower and upper endpoints, respectively, of M. We use the following well-known fact on matchings in the violation graph.

▶ **Fact 3.8** (Corollary 2 [29]). *For a DAG $\mathcal{G}$ and function $f\colon V(\mathcal{G}) \to \mathbb{R}$, the distance to monotonicity $\varepsilon(f)$ is equal to the size of the minimum vertex cover of the violation graph of f divided by $|V(\mathcal{G})|$.*

▶ **Fact 3.9.** *M is a matching in the violation graph of f that is also maximal. That is, (a) $f(x) > f(y)$ for all $(x, y) \in M$ and (b) $|M| \geq (\varepsilon(f) \cdot |V(\mathcal{G})|)/2$.*

Proof. First, for the sake of contradiction, suppose $f(x) \leq f(y)$ for some pair $(x, y) \in M$. Then we can set $M = M \setminus \{(x, y)\}$, which can only increase $\sum_{(x,y)\in M}(f(x) - f(y))$ and will decrease $|M|$ by 1. This contradicts the definition of M. Thus, $f(x) > f(y)$ for all $(x, y) \in M$ and so M is a matching in the *violation graph of f*. Second, since M maximizes $\sum_{(x,y)\in M}(f(x) - f(y))$, it must also be a *maximal* matching in the violation graph of f. Thus, (b) follows from Fact 3.8 and the fact that the size of any maximal matching is at least half the size of the minimum vertex cover. ◀

We now apply Lemma 3.3 to M, obtaining the partitions $(S_i\colon i \in [k])$ and $(T_i\colon i \in [k])$ for S and T, respectively, for which $M(S_i) = T_i$ for all $i \in [k]$. For each $i \in [k]$, let $\mathcal{H}_i = \mathcal{H}(S_i, T_i)$. We use the collection of sweeping graphs $\mathcal{H}_1, \ldots, \mathcal{H}_k$ to prove Theorem 1.3. Note that these subgraphs are all vertex-disjoint by item 1 of Lemma 3.3. We use item 2 of Lemma 3.3 to prove the following lemma regarding the (S_i, T_i) pairs. The proof crucially relies on the fact that M is a max-weight, min-cardinality matching.

▶ **Lemma 3.10** (Property of the Pairs (S_i, T_i)). *For all $i \in [k]$ and $(x, y) \in S_i \times T_i$, if $x \prec y$ then $f(x) > f(y)$.*

Proof. Suppose there exists $i \in [k]$, $x \in S_i$, and $y \in T_i$ for which $x \prec y$ and $f(x) \leq f(y)$. By item 2 of Lemma 3.3 there exists a matching $\widehat{M}\colon S \to T$ in $\mathrm{TC}(\mathcal{G})$ for which $(x, y) \in \widehat{M}$. In particular, since M and $\widehat{M}$ have identical sets of lower and upper endpoints,

$$\sum_{(s,t)\in\widehat{M}} (f(s) - f(t)) = \sum_{(s,t)\in M} (f(s) - f(t)) \text{ and } |\widehat{M}| = |M|.$$

Now set $\widehat{M}' = \widehat{M} \setminus \{(x, y)\}$ and observe that since $f(x) \leq f(y)$,

$$\sum_{(s,t)\in\widehat{M}'} (f(s) - f(t)) \geq \sum_{(s,t)\in M} (f(s) - f(t)) \text{ and } |\widehat{M}'| < |M|.$$

Therefore, M is not a max-weight, min-cardinality matching and this is a contradiction. ◀

3.4 Tying it Together: Defining the Boolean Functions $f_1, \ldots, f_k$

We are now equipped to define the functions $f_1, \ldots, f_k\colon V(\mathcal{G}) \to \{0, 1\}$ and complete the proof of Theorem 1.3. First, given $i \in [k]$ and $z \in V(\mathcal{G}) \setminus V(\mathcal{H}_i)$, we say that z is *below* $\mathcal{H}_i$ if there exists $y \in V(\mathcal{H}_i)$ for which $z \prec y$, and z is *above* $\mathcal{H}_i$ if there exists $x \in V(\mathcal{H}_i)$ for which $x \prec z$. Since $\mathcal{H}_i$ is the (S_i, T_i)-sweeping graph, by item 2 of Claim 3.2, vertex z cannot be both below and above $\mathcal{H}_i$, simultaneously. Second, given $z \in V(\mathcal{H}_i)$, we define the set $T_i(z) = \{t \in T_i\colon z \preceq t\}$. Note that by item 1 of Claim 3.2, $T_i(z) \neq \emptyset$ for all $z \in V(\mathcal{H}_i)$, and so the quantity $\max_{t\in T_i(z)} f(t)$ is always well-defined.

▶ **Definition 3.11.** *For each $i \in [k]$, define the function $f_i\colon V(\mathcal{G}) \to \{0, 1\}$ as follows. For every $z \in V(\mathcal{G})$,*

$$f_i(z) = \begin{cases} 1, & \text{if } z \in V(\mathcal{H}_i) \text{ and } f(z) > \max_{t\in T_i(z)} f(t), \\ 0, & \text{if } z \in V(\mathcal{H}_i) \text{ and } f(z) \leq \max_{t\in T_i(z)} f(t), \\ 1, & \text{if } z \notin V(\mathcal{H}_i) \text{ and } z \text{ is above } \mathcal{H}_i, \\ 0, & \text{if } z \notin V(\mathcal{H}_i) \text{ and } z \text{ is not above } \mathcal{H}_i. \end{cases}$$

See Fig. 3 for an illustration of the values of f_i. We first prove item 1 of Theorem 1.3. Recall that $M(S_i) = T_i$ for all $i \in [k]$. Let $M_i = M|_{S_i}$ denote the matching M restricted to S_i. Consider $x \in S_i$. By Lemma 3.10, $f(x) > f(y)$ for all $y \in T_i$ such that $x \prec y$. Thus, $f(x) > \max_{t\in T_i(x)} f(t)$ and so $f_i(x) = 1$. Now consider $y \in T_i$. Observe that $y \in T_i(y)$. Thus, clearly, $f(y) \leq \max_{t\in T_i(y)} f(t)$, and so $f_i(y) = 0$. Therefore, $f_i(x) = 1$ for all $x \in S_i$ and $f_i(y) = 0$ for all $y \in T_i$. In particular, $f_i(x) = 1 > 0 = f_i(M(x))$ for all $x \in S_i$ and so M_i is a matching in the violation graph of f_i. Thus, $\varepsilon(f_i) \geq \frac{|M_i|}{|V(\mathcal{G})|}$ for all $i \in [k]$. Then

$$\sum_{i=1}^{k} \varepsilon(f_i) \geq |V(\mathcal{G})|^{-1} \sum_{i=1}^{k} |M_i| = |V(\mathcal{G})|^{-1} \cdot |M| \geq |V(\mathcal{G})|^{-1} \cdot \frac{\varepsilon(f) \cdot |V(\mathcal{G})|}{2} = \frac{\varepsilon(f)}{2}$$

by the above argument and Fact 3.9. Thus, item 1 of Theorem 1.3 holds.

To prove item 2 of Theorem 1.3, we need to show that, for all $i \in [k]$, the following hold:

$$\mathcal{S}^-_{f_i} \subseteq E(\mathcal{H}_i) \text{ and } \mathcal{S}^-_{f_i} \subseteq \mathcal{S}^-_f.$$

We first prove that $\mathcal{S}^-_{f_i} \subseteq E(\mathcal{H}_i)$. Consider an edge $(x, y) \in E(\mathcal{G}) \setminus E(\mathcal{H}_i)$. We need to show that $f_i(x) \leq f_i(y)$. First, observe that if both $x, y \in V(\mathcal{H}_i)$, then by item 3 of Claim 3.2, we have $(x, y) \in E(\mathcal{H}_i)$. Thus, we only need to consider the following three cases. Recall that $f_i(x), f_i(y) \in \{0, 1\}$.

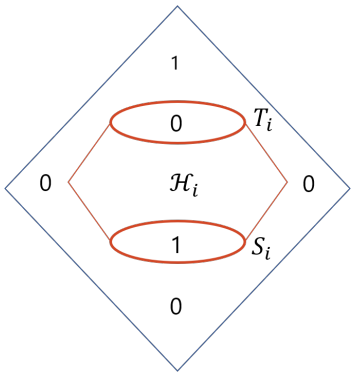


Figure 3 An illustration for the Boolean function f_i of Definition 3.11. The diamond represents the DAG $\mathcal{G}$ whose paths are directed from bottom to top. The hexagon represents the sweeping graph $\mathcal{H}_i = \mathcal{H}(S_i, T_i)$. The value of f_i is 1 for the vertices in S_i and 0 for the vertices in T_i. For vertices outside of $\mathcal{H}_i$, its value is 1 for the vertices that are above $\mathcal{H}_i$ and 0 for all other vertices.

1. $x \in V(\mathcal{H}_i)$, $y \notin V(\mathcal{H}_i)$: In this case, y is above $\mathcal{H}_i$, and so $f_i(y) = 1$. Thus, $f_i(x) \leq f_i(y)$.
2. $x \notin V(\mathcal{H}_i)$, $y \in V(\mathcal{H}_i)$: In this case, x is below $\mathcal{H}_i$, and so x is *not* above $\mathcal{H}_i$ by item 2 of Claim 3.2. Thus, $f_i(x) = 0$, and so $f_i(x) \leq f_i(y)$.
3. $x \notin V(\mathcal{H}_i)$, $y \notin V(\mathcal{H}_i)$: If x is above $\mathcal{H}_i$, then y is above $\mathcal{H}_i$ as well, and so $f_i(x) = f_i(y) = 1$. Otherwise, x is *not* above $\mathcal{H}_i$ and so $f_i(x) = 0$. Thus, $f_i(x) \leq f_i(y)$.

Therefore, $S^-_{f_i} \subseteq E(\mathcal{H}_i)$.

We now prove that $\mathcal{S}^-_{f_i} \subseteq \mathcal{S}^-_f$. Consider an edge $(x, y) \in \mathcal{S}^-_{f_i}$. Then $f_i(x) = 1$ and $f_i(y) = 0$. Since $S^-_{f_i} \subseteq E(\mathcal{H}_i)$, we get $(x, y) \in E(\mathcal{H}_i)$ and so $x, y \in V(\mathcal{H}_i)$. By definition of the functions f_i, it holds that $f(x) > \max_{t \in T_i(x)} f(t)$ and $f(y) \leq \max_{t \in T_i(y)} f(t)$. Since $x \prec y$, then $T_i(y) \subseteq T_i(x)$, because all vertices reachable from y are also reachable from x. Therefore,

$$f(x) > \max_{t \in T_i(x)} f(t) \geq \max_{t \in T_i(y)} f(t) \geq f(y).$$

Thus, $f(x) > f(y)$, and so $(x, y) \in \mathcal{S}^-_f$. As a result, $S^-_{f_i} \subseteq \mathcal{S}^-_f$ and item 2 of Theorem 1.3 holds. This concludes the proof of Theorem 1.3.

4 Testing Monotonicity of Real-Valued Functions

In this section, we prove Theorem 1.6. Some details have been omitted from this version. The omitted portion can be found in Section 4.3 of the full version [11]. We show that the tester of [32] for Boolean functions can be employed to test monotonicity of real-valued functions. The tester is simple: it queries two comparable vertices x and y and rejects if the pair exhibits a violation to monotonicity for f. The tester tries different values τ for the distance between x and y, that is, the number of coordinates on which they differ. The key step in the analysis of [32] (and in our analysis) is to show that for some choice of τ, the tester will detect a violation to monotonicity with high enough probability. The extra factor of r in the query complexity of our tester arises because we are forced to choose τ which is a factor of $(r-1)$ smaller than for the Boolean case. Intuitively, the reason for this is that as the walk length τ increases, the probability that the function value stays below a certain threshold decreases. We make this precise in Section 4.2.

We first define the distribution from which the tester samples x and y. Following this, we present the tester as Algorithm 2. Let p denote the largest integer for which $2^p \leq \sqrt{d/\log d}$. In Algorithm 2, we sample pairs of vertices at distance τ, where τ ranges over the powers of two up to 2^p.

▶ **Definition 4.1** (Pair Test Distribution). *Given parameters $b \in \{0,1\}$ and a positive integer τ, define the following distribution $\mathcal{D}_{\texttt{pair}}(b,\tau)$ over pairs $(x,y) \in (\{0,1\}^d)^2$. Sample $\boldsymbol{x}$ uniformly from $\{0,1\}^d$. Let $\boldsymbol{S} = \{i \in [d] \colon \boldsymbol{x}_i = b\}$. If $\tau > |\boldsymbol{S}|$, then set $\boldsymbol{y} = \boldsymbol{x}$. Otherwise, sample a uniformly random set $\boldsymbol{T} \subseteq \boldsymbol{S}$ of size $|\boldsymbol{T}| = \tau$. Obtain $\boldsymbol{y}$ by setting $\boldsymbol{y}_i = 1 - \boldsymbol{x}_i$ if $i \in \boldsymbol{T}$ and $\boldsymbol{y}_i = \boldsymbol{x}_i$ otherwise.*

Algorithm 2 Monotonicity Tester for $f \colon \{0,1\}^d \to \mathbb{R}$.

Require: Parameters $\varepsilon \in (0,1)$, dimension d, and image size r; oracle access to function $f \colon \{0,1\}^d \to \mathbb{R}$.

for all $b \in \{0,1\}$ and $\tau \in \{1,2,4,\ldots,2^p\}$ **do**
  repeat $\widetilde{O}\left(\min\left(\frac{r\sqrt{d}}{\varepsilon^2}, \frac{d}{\varepsilon}\right)\right)$ times:
    Sample $(\boldsymbol{x},\boldsymbol{y}) \sim \mathcal{D}_{\texttt{pair}}(b,\tau)$.
    if $b = 0$ and $f(\boldsymbol{x}) > f(\boldsymbol{y})$ **then reject**. ▷ if $b = 0$ then $\boldsymbol{x} \preceq \boldsymbol{y}$
    if $b = 1$ and $f(\boldsymbol{x}) < f(\boldsymbol{y})$ **then reject**. ▷ if $b = 1$ then $\boldsymbol{x} \succeq \boldsymbol{y}$
accept.

Our tester only uses comparisons between function values, not the values themselves. Thus, for the purposes of our analysis we can consider functions with the range $[r]$ w.l.o.g.

When $\tau = 1$, the algorithm is simply sampling edges from the d-dimensional hypercube. The distribution from which we sample is not the uniform distribution on edges, but following an argument from [32], we can assume that for $\tau = 1$, our tester has the same guarantees as the edge tester.

The choice of the distance parameter τ for which the rejection probability of the tester is high depends on the existence of a certain "good" bipartite subgraph of violated edges. Our analysis differs from the analysis of [32] both in how we obtain the "good" subgraph of violated edges and in the choice of the optimal distance parameter τ.

We extend the following definitions from [32]. Let $G(A,B,E_{AB})$ denote a directed bipartite graph with vertex sets A and B and all edges in E_{AB} *directed* from A to B.

▶ **Definition 4.2** ((K,Δ)-Good Graphs). *A directed bipartite graph $G(A,B,E_{AB})$ is (K,Δ)-good if for X,Y such that either $X = A$, $Y = B$ or $X = B$, $Y = A$, we have: (a) $|X| = K$. (b) Vertices in X have degree exactly Δ. (c) Vertices in Y have degree at most 2Δ. The graph G is (K,Δ)-left-good if $X = A$ and (K,Δ)-right-good if $X = B$.*

The *weight* of $x \in \{0,1\}^d$, denoted by $|x|$, is the number of coordinates of x with value 1.

▶ **Definition 4.3** (Persistence). *Given $f \colon \{0,1\}^d \to [r]$ and an integer $\tau \in \left[1, \sqrt{\frac{d}{\log d}}\right]$, a vertex $x \in \{0,1\}^d$ of weight in the range $\frac{d}{2} \pm O(\sqrt{d \log d})$ is* τ-right-persistent *for f if*

$$\Pr_{\boldsymbol{y}}[f(\boldsymbol{y}) \leq f(x)] > \frac{9}{10},$$

where $\boldsymbol{y}$ is obtained by choosing a uniformly random set $\boldsymbol{T} \subset \{i \in [d] \colon x_i = 0\}$ of size τ and setting $\boldsymbol{y}_i = 1$ if $i \in \boldsymbol{T}$ and $\boldsymbol{y}_i = x_i$ otherwise[4]. We define τ-left-persistence symmetrically.

We use the following technical claim implicitly shown in the analysis of the tester of [32].

[4] Note that $\tau \geq |\{i \in [d] \colon x_i = 0\}|$ by our assumption on x and τ.

▷ Claim 4.4 ([32]). Suppose there exists a (K, Δ)-right-good subgraph $G(A, B, E_{AB})$ of the directed d-dimensional hypercube, such that (a) $E_{AB} \subseteq \mathcal{S}_f^-$, (b) $K\sqrt{\Delta} = \Theta(\frac{\varepsilon(f)\cdot 2^d}{\log d})$, and (c) at least $\frac{99}{100}|B|$ of the vertices in B are $(\tau'-1)$-right-persistent for some τ' such that $\tau' \cdot \Delta \ll d$. Then there exists a constant $C' > 0$, such that for $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{D}_{\texttt{pair}}(0, \tau')$,

$$\Pr_{\boldsymbol{x},\boldsymbol{y}}[f(\boldsymbol{x}) > f(\boldsymbol{y})] \geq \frac{C' \cdot \tau'}{d} \cdot \frac{K}{2^d} \cdot \Delta.$$

The analogous claim holds given a (K, Δ)-left-good subgraph with many $(\tau'-1)$-left-persistent vertices in A and $(\boldsymbol{x}, \boldsymbol{y})$ drawn from $\mathcal{D}_{\texttt{pair}}(1, \tau')$.

In Section 4.1, we prove Lemma 4.6 which obtains a good subgraph for f satisfying conditions (a) and (b) of Claim 4.4. In Section 4.2, we prove Lemma 4.8 which gives an upper bound on the fraction of non-persistent vertices, enabling us to satisfy condition (c). The remainder of the proof of Theorem 1.6 is deferred to the full version [11]. In particular, in Section 4.3 of the full version, we use Lemma 4.6 and Lemma 4.8 to show that the conditions of Claim 4.4 are satisfied and then use this to prove Theorem 1.6.

4.1 Existence of a Good Bipartite Subgraph

In this section, we prove Lemma 4.6 on the existence of good bipartite subgraphs for real-valued functions, which was proved in [32] for the special case of Boolean functions. This lemma crucially relies on our isoperimetric inequality for real-valued functions (Theorem 1.2). We first state (without proof) a combinatorial result of [32], which we need for our lemma.

▶ **Lemma 4.5** (Lemma 6.5 of [32]). *Let $G(A, B, E_{AB})$ be a directed bipartite graph whose vertices have degree at most 2^s. Suppose in addition, that for any 2-coloring of its edges $\texttt{col} : E_{AB} \to \{\text{red}, \text{blue}\}$ we have*

$$\sum_{x \in A} \sqrt{\deg_{\text{red}}(x)} + \sum_{y \in B} \sqrt{\deg_{\text{blue}}(y)} \geq L, \tag{9}$$

where $\deg_{\text{red}}(x)$ denotes the number of red edges incident on x and $\deg_{\text{blue}}(y)$ denotes the number of blue edges incident on y. Then $G(A, B, E_{AB})$ contains a subgraph that is (K, Δ)-good with $K\sqrt{\Delta} \geq \frac{L}{8s}$.

We can now generalize Lemma 7.1 of [32].

▶ **Lemma 4.6.** *For all functions $f : \{0,1\}^d \to \mathbb{R}$, there exists a subgraph $G(A, B, E_{AB})$ of the directed, d-dimensional hypercube which is (K, Δ)-good, where $K\sqrt{\Delta} = \Theta(\frac{\varepsilon(f)\cdot 2^d}{\log d})$ and $E_{AB} \subseteq \mathcal{S}_f^-$.*

Proof. Our proof relies on Lemma 4.5. Condition (9) is clearly reminiscent of the isoperimetric inequality in Theorem 1.2. We want to partition the vertices in $\{0,1\}^d$ into sets A and B such that all the violated edges are directed from A to B and apply Theorem 1.2 to the resulting graph. In addition, we want (9) to hold for a big enough value of L. In the Boolean case, we can simply partition the vertices by function values. In contrast, for real-valued functions, a vertex $x \in \{0,1\}^d$ can be incident on both incoming and outgoing violated edges. To overcome this challenge we resort to the bipartiteness of the directed hypercube, where each edge is between a vertex with an odd weight and a vertex with an even weight. Partition $\mathcal{S}_f^-$ into two sets:

$$E_0 = \{(x, y) \in \mathcal{S}_f^- : |x| \text{ is even}\};$$

$E_1 = \{(x, y) \in \mathcal{S}_f^- : |x| \text{ is odd}\}$.

For $j \in \{0, 1\}$, let V_j and W_j denote the set of lower and upper endpoints, respectively, of the edges in E_j. We consider the two subgraphs $G_j(V_j, W_j, E_j)$ for $j \in \{0, 1\}$. Notice that the vertices in $V_0 \cup W_1$ have even weight and the vertices in $V_1 \cup W_0$ have odd weight. Obviously, V_0 and W_1 may not be disjoint, and similarly V_1 and W_0 may not be disjoint, and thus G_0 and G_1 may not be vertex-disjoint.

We quickly explain why we cannot simply use Lemma 4.5 with either G_0 or G_1. Fix a 2-coloring of the edges $E_0 \cup E_1$. By averaging, one of the graphs will have a high enough contribution to left-hand side of the isoperimetric inequality of Theorem 1.2. Assume this graph is G_0. As a result, condition (9) will hold for G_0 with $L = \Omega(\varepsilon \cdot 2^d)$. However, one cannot guarantee that condition (9) holds for *all* possible colorings of the edges of G_0. Our construction below describes how to combine G_0 and G_1 so that we can jointly "feed" them into Lemma 4.5.

We construct copies $\widehat{G}_0$ and $\widehat{G}_1$ of G_0 and G_1, so that $\widehat{G}_0$ contains a vertex labelled $(x, 0)$ for each vertex x of G_0, and $\widehat{G}_1$ contains a vertex $(x, 1)$ for each vertex x of G_1. For each edge (x, y) in G_0 we add an edge from $(x, 0)$ to $(y, 0)$ in $\widehat{G}_0$. We do the same for the edges of G_1. Note that each edge of $\mathcal{S}_f^-$ has exactly one copy, either in $\widehat{G}_0$ or $\widehat{G}_1$.

Let $\widehat{G}(\widehat{V}, \widehat{W}, \mathcal{S}_f^-)$ denote the union of the two vertex-disjoint graphs $\widehat{G}_0$ and $\widehat{G}_1$. That is,

$$\widehat{V} = \{(x, 0) \mid x \in V_0\} \cup \{(x, 1) \mid x \in V_1\},$$
$$\widehat{W} = \{(y, 0) \mid y \in W_0\} \cup \{(y, 1) \mid y \in W_1\}.$$

All the edges of $\widehat{G}$ are directed from $\widehat{V}$ to $\widehat{W}$. Although imprecise, we think of the edges of $\widehat{G}$ as $\mathcal{S}_f^-$, since each edge in $\mathcal{S}_f^-$ has exactly one copy in $\widehat{G}$.

Consider a 2-coloring $\texttt{col}\colon \mathcal{S}_f^- \to \{\text{red}, \text{blue}\}$. Observe that

$$\begin{aligned}
\sum_{(x,\cdot)\in\widehat{V}} \sqrt{I^-_{f,\text{red}}(x)} + \sum_{(y,\cdot)\in\widehat{W}} \sqrt{I^-_{f,\text{blue}}(x)} &= \sum_{x\in V_0\cup V_1} \sqrt{I^-_{f,\text{red}}(x)} + \sum_{y\in W_0\cup W_1} \sqrt{I^-_{f,\text{blue}}(y)} \\
&= \sum_{\substack{x\in\{0,1\}^d \\ |x| \text{ is even}}} \sqrt{I^-_{f,\text{red}}(x)} + \sqrt{I^-_{f,\text{blue}}(x)} + \sum_{\substack{x\in\{0,1\}^d \\ |x| \text{ is odd}}} \sqrt{I^-_{f,\text{red}}(x)} + \sqrt{I^-_{f,\text{blue}}(x)} \\
&= \sum_{x\in\{0,1\}^d} \sqrt{I^-_{f,\text{red}}(x)} + \sum_{y\in\{0,1\}^d} \sqrt{I^-_{f,\text{blue}}(y)} \geq C \cdot \varepsilon(f) \cdot 2^d,
\end{aligned}$$

where the inequality holds by Theorem 1.2.

By construction, $I^-_{f,\text{red}}(x) = \deg_{\text{red}}((x, \cdot))$ for all $(x, \cdot) \in \widehat{V}$ and $I^-_{f,\text{blue}}(y) = \deg_{\text{blue}}((y, \cdot))$ for all $(y, \cdot) \in \widehat{W}$. We have that condition (9) of Lemma 4.5 holds with $L = C \cdot \varepsilon(f) \cdot 2^d$. Thus, $\widehat{G}$ contains a subgraph $G_{\text{good}}(A, B, E_{AB})$ that is (K, Δ)-good with $K\sqrt{\Delta} \geq \frac{L}{8 \log d}$. Without loss of generality, assume $G_{\text{good}}(A, B, E_{AB})$ is (K, Δ)-right-good.

Let $G_{\text{good},0} = (A_0, B_0, E_{A_0B_0})$ denote the subgraph of G_{good} lying in $\widehat{G}_0$ and let $G_{\text{good},1} = (A_1, B_1, E_{A_1B_1})$ denote the subgraph of G_{good} lying in $\widehat{G}_1$. Since $B_0 \cap B_1 = \emptyset$, we know that either $|B_0| \geq K/2$ or $|B_1| \geq K/2$. Suppose $|B_0| \geq K/2$. Moreover, since $\widehat{G}_0$ and $\widehat{G}_1$ are vertex-disjoint subgraphs, the degree of a vertex of $A_0 \cup B_0$ in $G_{\text{good},0}$ is the same its degree in G_{good}. Thus, $G_{\text{good},0}$ is a $(K/2, \Delta)$-right-good subgraph of the d-dimensional directed hypercube for which $\frac{K}{2}\sqrt{\Delta} \geq \frac{L}{16 \log d}$.

By removing some vertices from B_0, and redefining K if necessary, we may assume that $K\sqrt{\Delta} = \Theta\left(\frac{\varepsilon(f)\cdot 2^d}{\log d}\right)$. This completes the proof of Lemma 4.6. ◀

4.2 Bounding the Number of Non-Persistent Vertices

We prove Lemma 4.8 that bounds the number of non-persistent vertices for a function f and a given distance parameter τ. All results in this section also hold for τ-left-persistence.

For a function $f\colon \{0,1\}^d \to \mathbb{R}$, we define I_f^- as $\frac{|\mathcal{S}_f^-|}{2^d}$.

▶ **Corollary 4.7** (Corollary of Theorem 6.6, Lemma 6.8 of [32]). *Consider a function $h\colon \{0,1\}^d \to \{0,1\}$ and an integer $\tau \in \left[1, \sqrt{\frac{d}{\log d}}\right]$. If $I_h^- \leq \sqrt{d}$ then*

$$\Pr_{\boldsymbol{x}\sim\{0,1\}^d}\left[\boldsymbol{x} \text{ is not } \tau\text{-right-persistent for } h\right] = O\left(\frac{\tau}{\sqrt{d}}\right). \tag{10}$$

We generalize the above result to functions with image size $r \geq 2$.

▶ **Lemma 4.8.** *Consider a function $f\colon \{0,1\}^d \to [r]$ and an integer $\tau \in \left[1, \sqrt{\frac{d}{\log d}}\right]$. If $I_f^- \leq \sqrt{d}$, then*

$$\Pr_{\boldsymbol{x}\sim\{0,1\}^d}\left[\boldsymbol{x} \text{ is not } \tau\text{-right-persistent for } f\right] = (r-1)\cdot O\left(\frac{\tau}{\sqrt{d}}\right).$$

Proof. For all $t \in [r]$, define the threshold function $h_t\colon \{0,1\}^d \to \{0,1\}$ as:

$$h_t(x) = \begin{cases} 1 & \text{if } f(x) > t, \\ 0 & \text{otherwise.} \end{cases}$$

Observe that for all $t \in [r]$, we have $\mathcal{S}_{h_t}^- \subseteq \mathcal{S}_f^-$, and thus $I_{h_t}^- \leq I_f^- \leq \sqrt{d}$. By Corollary 4.7, we have that (10) holds for $h = h_t$ for all $t \in [r]$. Next, we point out that a vertex $x \in \{0,1\}^d$ is τ-right-persistent for f if and only if x is τ-right-persistent for the Boolean function $h_{f(x)}$. Too see this, consider a vertex z such that $x \prec z$. First, note that $h_{f(x)}(x) = 0$. Second, note that $h_{f(x)}(z) = 1$ if and only if $f(z) > f(x)$ by definition of $h_{f(x)}$. Therefore, $f(z) \leq f(x)$ if and only if $h_{f(x)}(z) \leq h_{f(x)}(x)$. Finally, note that all vertices are persistent for h_r since $h_r(x) = 0$ for all $x \in \{0,1\}^d$. Using these observations, we have

$$\begin{aligned}
&\Pr_{\boldsymbol{x}\sim\{0,1\}^d}[\boldsymbol{x} \text{ is not } \tau\text{-right-persistent for } f] \\
&= \Pr_{\boldsymbol{x}\sim\{0,1\}^d}\left[\boldsymbol{x} \text{ is not } \tau\text{-right-persistent for } h_{f(\boldsymbol{x})}\right] \\
&\leq \Pr_{\boldsymbol{x}\sim\{0,1\}^d}[\exists t \in [r-1]\colon \boldsymbol{x} \text{ is not } \tau\text{-right-persistent for } h_t] \\
&\leq \sum_{t=1}^{r-1} \Pr_{\boldsymbol{x}\sim\{0,1\}^d}[\boldsymbol{x} \text{ is not } \tau\text{-right-persistent for } h_t] \\
&= \sum_{t=1}^{r-1} O\left(\frac{\tau}{\sqrt{d}}\right) = (r-1)\cdot O\left(\frac{\tau}{\sqrt{d}}\right),
\end{aligned}$$

where the second inequality is by the union bound and the last equality is due to the fact that (10) holds for all h_t, $t \in [r]$. ◀

References

1 Nir Ailon and Bernard Chazelle. Information theory in property testing and monotonicity testing in higher dimension. *Information and Computation*, 204(11):1704–1717, 2006.

2 Nir Ailon, Bernard Chazelle, Seshadhri Comandur, and Ding Liu. Estimating the distance to a monotone function. *Random Structures and Algorithms*, 31(3):371–383, 2007.

3 Tugkan Batu, Ronitt Rubinfeld, and Patrick White. Fast approximate PCPs for multidimensional bin-packing problems. *Information and Computation*, 196(1):42–56, 2005.

4 Aleksandrs Belovs. Adaptive lower bound for testing monotonicity on the line. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 31:1–31:10, 2018.

5 Aleksandrs Belovs and Eric Blais. A polynomial lower bound for testing monotonicity. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 1021–1032, 2016.

6 Piotr Berman, Sofya Raskhodnikova, and Grigory Yaroslavtsev. l_p-testing. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 164–173, 2014.

7 Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P. Woodruff. Transitive-closure spanners. *SIAM J. Comput.*, 41(6):1380–1425, 2012.

8 Hadley Black, Deeparnab Chakrabarty, and C. Seshadhri. A $o(d)\cdot$ polylog n monotonicity tester for Boolean functions over the hypergrid $[n]^d$. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2133–2151, 2018.

9 Hadley Black, Deeparnab Chakrabarty, and C. Seshadhri. Domain reduction for monotonicity testing: A $o(d)$ tester for Boolean functions in d-dimensions. In Shuchi Chawla, editor, *Proceedings, ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1975–1994, 2020.

10 Hadley Black, Deeparnab Chakrabarty, and C. Seshadhri. Directed isoperimetric theorems for boolean functions on the hypergrid and an $\widetilde{O}(n\sqrt{d})$ monotonicity tester. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, 2023.

11 Hadley Black, Iden Kalemaj, and Sofya Raskhodnikova. Isoperimetric inequalities for real-valued functions with applications to monotonicity testing. *CoRR*, abs/2011.09441, 2020. `arXiv:2011.09441`.

12 Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. *Computational Complexity*, 21(2):311–358, 2012.

13 Eric Blais, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Lower bounds for testing properties of functions over hypergrid domains. In *Proceedings, IEEE Conference on Computational Complexity (CCC)*, pages 309–320, 2014.

14 Mark Braverman, Subhash Khot, Guy Kindler, and Dor Minzer. Improved monotonicity testers via hypercube embeddings. In *Proceedings, Innovations in Theoretical Computer Science (ITCS)*, pages 25:1–25:24, 2023.

15 Jop Briët, Sourav Chakraborty, David García Soriano, and Ari Matsliah. Monotonicity testing and shortest-path routing on the cube. *Combinatorica*, 32(1):35–53, 2012.

16 Deeparnab Chakrabarty, Kashyap Dixit, Madhav Jha, and C. Seshadhri. Property testing on product distributions: Optimal testers for bounded derivative properties. *ACM Trans. on Algorithms*, 13(2):20:1–20:30, 2017.

17 Deeparnab Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 419–428, 2013.

18 Deeparnab Chakrabarty and C. Seshadhri. An optimal lower bound for monotonicity testing over hypergrids. *Theory of Computing*, 10:453–464, 2014.

19 Deeparnab Chakrabarty and C. Seshadhri. An $o(n)$ monotonicity tester for Boolean functions over the hypercube. *SIAM Journal on Computing*, 45(2):461–472, 2016.

20 Deeparnab Chakrabarty and C. Seshadhri. Adaptive Boolean monotonicity testing in total influence time. In *Proceedings, Innovations in Theoretical Computer Science (ITCS)*, pages 20:1–20:7, 2019.

21 Xi Chen, Anindya De, Rocco A. Servedio, and Li-Yang Tan. Boolean function monotonicity testing requires (almost) $n^{1/2}$ non-adaptive queries. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 519–528, 2015.

22 Xi Chen, Rocco A. Servedio, and Li-Yang Tan. New algorithms and lower bounds for monotonicity testing. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 286–295, 2014.

23 Xi Chen, Erik Waingarten, and Jinyu Xie. Beyond Talagrand functions: new lower bounds for testing monotonicity and unateness. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 523–536, 2017.

24 Kashyap Dixit, Sofya Raskhodnikova, Abhradeep Thakurta, and Nithin Varma. Erasure-resilient property testing. *SIAM Journal on Computing*, 47(2):295–329, 2018.

25 Yevgeniy Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, Dana Ron, and Alex Samorodnitsky. Improved testing algorithms for monotonicity. In *Proceedings of Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 97–108, 1999.

26 Funda Ergun, Sampath Kannan, Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spot-checkers. *J. Comput. System Sci.*, 60(3):717–751, 2000.

27 Shahar Fattal and Dana Ron. Approximating the distance to monotonicity in high dimensions. *ACM Trans. on Algorithms*, 6(3):52:1–52:37, 2010.

28 Eldar Fischer. On the strength of comparisons in property testing. *Information and Computation*, 189(1):107–116, 2004.

29 Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 474–483, 2002.

30 Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samorodnitsky. Testing monotonicity. *Combinatorica*, 20(3):301–337, 2000.

31 Shirley Halevy and Eyal Kushilevitz. Testing monotonicity over graph products. *Random Structures and Algorithms*, 33(1):44–67, 2008.

32 Subhash Khot, Dor Minzer, and Muli Safra. On monotonicity testing and Boolean isoperimetric-type theorems. *SIAM Journal on Computing*, 47(6):2238–2276, 2018.

33 Eric Lehman and Dana Ron. On disjoint chains of subsets. *Journal of Combinatorial Theory, Series A*, 94(2):399–404, 2001.

34 Grigory A. Margulis. Probabilistic characteristics of graphs with large connectivity. *Problemy Peredachi Informatsii*, 10(2):101–108, 1974.

35 Ramesh Krishnan S. Pallavoor, Sofya Raskhodnikova, and Nithin Varma. Parameterized property testing of functions. *ACM Trans. Comput. Theory*, 9(4):17:1–17:19, 2018.

36 Ramesh Krishnan S. Pallavoor, Sofya Raskhodnikova, and Erik Waingarten. Approximating the distance to monotonicity of Boolean functions. *Random Structures and Algorithms*, 60(2):233–260, 2022.

37 Ramesh Krishnan Pallavoor Suresh. *Improved Algorithms and New Models in Property Testing*. PhD thesis, Boston University, 2020.

38 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *J. Comput. Syst. Sci.*, 72(6):1012–1042, 2006.

39 Sofya Raskhodnikova. Monotonicity testing. *Masters Thesis, MIT*, 1999.

40 Michel Talagrand. Isoperimetry, logarithmic Sobolev inequalities on the discrete cube, and Margulis' graph connectivity theorem. *Geom. Func. Anal.*, 3(3):295–314, 1993.

The Geometry of Tree-Based Sorting

Guy E. Blelloch
Carnegie Mellon University, Pittsburgh, PA, USA

Magdalen Dobson
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

We study the connections between sorting and the binary search tree (BST) model, with an aim towards showing that the fields are connected more deeply than is currently appreciated. While any BST can be used to sort by inserting the keys one-by-one, this is a very limited relationship and importantly says nothing about parallel sorting. We show what we believe to be the first formal relationship between the BST model and sorting. Namely, we show that a large class of sorting algorithms, which includes mergesort, quicksort, insertion sort, and almost every instance-optimal sorting algorithm, are equivalent in cost to offline BST algorithms. Our main theoretical tool is the geometric interpretation of the BST model introduced by Demaine et al. [18], which finds an equivalence between searches on a BST and point sets in the plane satisfying a certain property. To give an example of the utility of our approach, we introduce the log-interleave bound, a measure of the information-theoretic complexity of a permutation π, which is within a $\lg \lg n$ multiplicative factor of a known lower bound in the BST model; we also devise a parallel sorting algorithm with polylogarithmic span that sorts a permutation π using comparisons proportional to its log-interleave bound. Our aforementioned result on sorting and offline BST algorithms can be used to show existence of an offline BST algorithm whose cost is within a constant factor of the log-interleave bound of any permutation π.

2012 ACM Subject Classification Theory of computation → Sorting and searching

Keywords and phrases binary search trees, sorting, dynamic optimality, parallelism

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.26

Category Track A: Algorithms, Complexity and Games

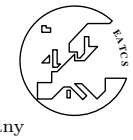
Related Version *Full Version*: `https://arxiv.org/abs/2110.11836`

Funding This work was supported by the National Science Foundation grants CCF-1901381, CCF-1910030, CCF-1919223, DGE1745016, and DGE2140739.

Acknowledgements We thank Kanat Tangwongsan for helpful discussions. We thank the anonymous reviewers for their useful comments.

1 Introduction

Comparison-based sorting and searching on a BST are among the most elementary, important, and well-studied algorithmic topics in all of theoretical computer science. It has long been observed that they are closely related: both enjoy better performance on sequences that are information-theoretically simpler, such as reversals of sorted lists, sequences with long runs of consecutive keys, or sequences composed from simple shuffles of sorted lists. Indeed, their respective searches for instance optimality have yielded the independent discovery of almost identical results [34]. Despite the extensive number of similar results throughout the literature, there is comparatively very little known about the *formal* relationship between sorting and the binary search tree (BST) model. In this paper, we present what we believe to be the first formal relation between the sorting cost model and the BST model. Our result shows that a large class of sorting algorithms, which includes mergesort, quicksort, insertion

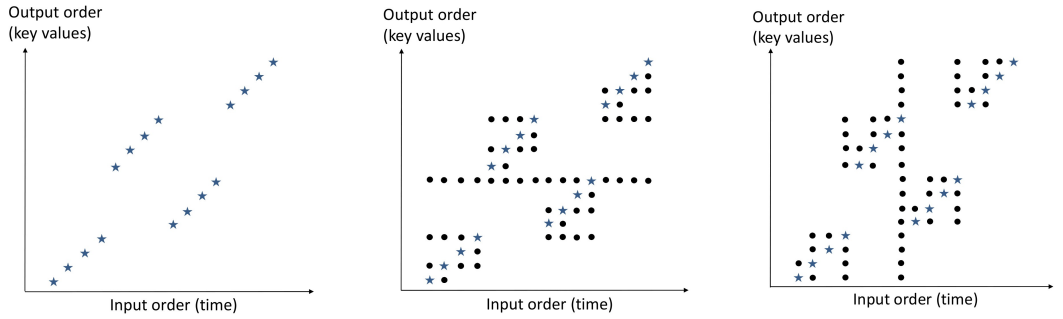


sort, and most adaptive sorting algorithms, are equivalent in cost to offline algorithms in the BST model. As this class is large and contains many well-studied algorithms, we find it interesting that we are able to show any kind of new and nontrivial theoretical result based on this characterization.

Binary Search Trees. The binary search tree is a fundamental data structure that stores an ordered universe of keys in a dynamic tree. A search for a key begins with a pointer to the root of the tree and at each step performs one of two unit-cost actions: move the pointer to a parent or child node, or perform a rotation. Rotations are key to understanding the power of the model, because they allow frequently queried elements to be kept close to the root of the tree as well as exploiting other kinds of order in the sequence of queried keys. The power of rotations is related to the concept of *instance optimality*, a general term which refers to the fact that an algorithm can enjoy improvements on its worst-case complexity on well-defined sets of "easy" inputs. Many BST algorithms perform better than their worst-case complexity (i.e. $\log n$ per operation) on various kinds of input [37, 19, 18, 3]. Beyond these specific improvements, the hope for a more general kind of instance optimality is crisply expressed by the *dynamic optimality conjecture* of Sleator and Tarjan [35], which states that there exists a binary search tree whose performance on any online sequence of searches is constant factor competitive with the best offline algorithm for that sequence. The dynamic optimality conjecture remains open. Another equally important open question is whether there is an offline efficient algorithm for calculating even an approximately optimal number of rotations for a given input sequence. These problems have been the subject of extensive work both in the past [38, 16, 15, 19, 21, 8] and at the present moment [26, 13, 7, 25, 23].

Sorting. Similarly to the BST model, where rotations are used to achieve instance optimality for particular classes of inputs, in comparison-based sorting an *adaptive sorting algorithm* performs fewer comparisons when the input is "closer" to sorted by some measure. In this field a *measure of disorder* for a list L is paired with an algorithm which is *optimal* for this measure. Here, optimal roughly means that sorting L only requires the number of comparisons needed to distinguish it from all other lists which are more presorted than L [34]. An accompanying notion is that a measure of disorder may be *superior* (inferior) to another measure – that is, always requires fewer comparisons for any given permutation. Mannila first formalized these ideas [31]. After this, many researchers devised new measures of disorder and corresponding optimal algorithms [17, 22, 24, 28, 29, 27, 33, 34]; furthermore, there was also interest in work-optimal parallel versions of optimal sorting algorithms [11, 30, 14]. It remains an open problem whether there exists a measure which is provably superior to any possible measure of disorder.

Arborally Satisfied Point Sets and Sorting. One of our most important tools in connecting BSTs and sorting is the geometric interpretation of BSTs [18, 20]. In this interpretation, an access sequence of n keys is represented as an $n \times n$ grid with time order (input order) on one axis (here the x axis) and key order (output order) on the other axis. Points are added to the grid to account for all keys that must be visited when searching or inserting the keys one at a time from left to right. Demaine et al. [18], and Derryberry, Sleator, and Wang [20] show that for any BST algorithm, the accesses plotted in the plane must satisfy the property that for every pair of points p, q (both original and added points), there is a monotonic path (e.g. up and right) from p to q consisting of horizontal and vertical segments with a point at



(a) The input sequence.

(b) Sequence arborally satisfied using quicksort.

(c) Sequence arborally satisfied using mergesort.

Figure 1 On the left, the input sequence plotted in the plane. In the middle, the input sequence arborally satisfied using accesses corresponding to quicksort. On the right, the input sequence arborally satisfied using accesses corresponding to mergesort.

each corner[1]. Demaine et al. refer to such a set of points as being *arborally satisfied*, and show that any such set of size m implies the sequence of n keys can be searched or inserted in cost $O(m)$ in the BST model.

We observe that the geometric approach is also useful for sorting, since unlike the BST model, it does not directly enforce an order of insertion. As some evidence of the utility of the geometric approach, consider the following two algorithms that can be used to arborally satisfy a set of accesses. The first algorithm starts by choosing a random point, then adding accesses to that point across its entire row of the point set. Then, it recurses above and below the row, and in each partition it picks a random point and adds new points to all locations in its row for which there is a key in the partition. It is not hard to verify the points added in this way are arborally satisfied: any point p can get to a point q by going up (or down) to the row that separated them, then across to column of q and up (down) to q. Second, consider another algorithm that adds points across the middle column, and then for the left and right, add points along their middle columns for all points in those halves. Recursing to the base case again gives an arborally satisfied set. See Figure 1 for an example of both of these algorithms. The attentive reader may have noticed that the accesses added in the first algorithm correspond to the comparisons made by the quicksort algorithm, and the accesses added in the second algorithm correspond to comparisons made by the mergesort algorithm. We will extend these ideas to more interesting algorithms in this paper.

We note that in addition to being of significant theoretical interest, taking advantage of locality in key sequences is widely practical for both search trees and sorting. Sleator and Tarjan won the ACM Kannelakis Theory and Practice award for their work on splay trees and its applications for reducing key search time in several widely used applications. Adaptive sorting algorithms are widely adopted in practice, including *timsort*, which is implemented as built-in libraries for Python, Java, Swift, and Rust, among other languages [2].

1.1 Our Results

In this paper we present specific results relating sorting and BSTs using arborally satisfied sets. Our first result is an explicit relation between the cost models of a broad set of sorting algorithms and BST algorithms. The specific set of algorithms, which we refer to as *tree-based*

[1] The papers have their own preferred definition, but the definitions are all equivalent.

sorting algorithms and which are formally defined in Section 3, are divided into two classes: *BST mergesorts* are sorting algorithms based on recursive merges of the input, where the keys being merged are stored in binary search trees; *BT partition sorts* are sorting algorithms based on recursive partitions of the input based on key ordering, where the keys are stored in a binary tree. In Section 3, we show that these two types of algorithms are "dual" to each other in the following way: a BST mergesort $\mathcal{A}$ sorting permutation π with cost $\mathcal{A}(\pi)$ implies the existence of binary tree (BT) partition sort $\mathcal{B}$ sorting π^{-1} with cost $\mathcal{A}(\pi)$, and vice versa.

The category of tree-based sorting algorithms is large. Both quicksort and mergesort on lists fall into the category of tree-based sorting algorithms, as lists are simply a special case of trees. Sorting by insertion into a BST is also a BST mergesort, since the merges can be carried out in any order. McIlroy's adaptive sorting algorithms – namely, insertion sort with exponential search and mergesort with exponential search – are BST mergesorts [32]. These algorithms were influential in the development of timsort [2], a mergesort algorithm that breaks the input into runs of increasing or decreasing keys and merges them based on certain ordering criteria; since it is also a mergesort on lists, timsort is a BST merge. In their capstone paper on adaptive sorting, Petersson and Moffat cover the three most powerful known adaptive sorting algorithms – local insertion sort, historical insertion sort, and regional insertion sort. Local insertion sort is a BST mergesort as it inserts into a BST with a single additional pointer, which can be converted to the BST model with constant overhead [13]. Historical and regional insertion sort are not BST mergesorts as presented by the authors, but independently discovered data structures would yield BST mergesorts with the same bounds [35, 21].

Theorem 1 shows that for any tree-based sorting algorithm that sorts access sequence π using $\mathcal{A}(\pi)$ accesses, there exists an offline algorithm in the BST model which searches for each key in π using $O(\mathcal{A}(\pi))$ accesses. The proof of Theorem 1 is subtle and nontrivial and requires several new insights relating to the geometric interpretation of the BST model. In the following statement, $\mathrm{OPT}_{\mathrm{BST}}(\pi)$ refers to the cost of the best offline algorithm.

▶ **Theorem 1.** *Let $\mathcal{A}$ be a tree-based sorting algorithm which sorts permutation π using $\mathcal{A}(\pi)$ accesses. Then $OPT_{BST}(\pi) \in O(\mathcal{A}(\pi))$.*

As some evidence for the utility of our approach, we introduce the log-interleave bound, a measure of the information-theoretic complexity of a permutation π. The log-interleave bound is an upper bound on the number of bits needed to encode π; it can also be understood from an algorithmic perspective as a mergesort with a more efficient merge step. Our main results on the log-interleave bound illustrate the connections between sorting and the BST model. In the statements of the following results, we use the notation $\mathrm{LIB}(\pi)$ to refer to the log-interleave bound of a permutation π. This will be defined formally in Section 4.

The first result is a proof that the log-interleave bound is within a $\lg \lg n$ multiplicative factor of the optimal offline BST algorithm on any permutation. Somewhat similarly to Demaine et al.'s proof of the closeness to optimality of tango trees [19], our proof shows closeness to optimality by comparing the log-interleave bound with Wilber's interleave bound [38], a lower bound in the BST model.

▶ **Theorem 17.** *For any permutation π, $IB(\pi) \leq LIB(\pi) \in O(\lg \lg n\ IB(\pi))$.*

Next, we show that there is a work optimal *parallel* sorting algorithm related to the log-interleave bound. The next result is a parallel mergesort featuring a merge step which combines recent work on parallel split and join of BSTs [4] with a BST from [9] and an analysis which shows that with this new merge step, the mergesort sorts a sequence π in

$O(\text{LIB}(\pi))$ work. While the span of this algorithm is greater than the span of a typical parallel sort and indeed may be open to improvement, all existing parallel sorting algorithms present guarantees only for very weak measures of disorder [14, 30].

▶ **Theorem 21.** *There exists a parallel mergesort which for any permutation π performs $O(LIB(\pi))$ work with polylogarithmic span.*

Finally, a corollary of Theorem 6 shows that there is an offline BST algorithm that incurs cost $O(\text{LIB}(\pi))$, and thus that the log-interleave bound is an upper bound in the BST model.

▶ **Corollary 22.** *There exists an offline BST algorithm $\mathcal{A}$ such that $\mathcal{A}(\pi) = O(LIB(\pi))$.*

Model of Computation. Our results for the parallel algorithms are given for the binary-fork-join model [5]. In this model a process can fork two child processes, which work in parallel and when both complete, the parent process continues. Costs are measured in terms of the work (total number of instructions across all processes) and span (longest dependence path among processes). Any algorithm in the binary forking model with W work and S span can be implemented on a CRCW PRAM with P processors in $O(W/P + S)$ time with high probability [1, 6], so the results here are also valid on the PRAM, maintaining work efficiency.

1.2 Related Work

Upper and Lower Bounds in the BST Model. The pursuit of dynamic optimality led to a string of work in both upper and lower bounds on the cost of a sequence of searches on a BST. Three important upper bounds in the literature are the *dynamic finger bound* [35, 16, 15, 13, 8, 23], the *working set bound* [35], and the *unified bound* [3, 21], which respectively state that accessing an element is fast if its key is close to the key of the previous search, if its key has been searched recently, and a combination of the two. There has also been significant work in lower bounding the cost of an access sequence in the BST model. Two such lower bounds, the *interleave bound* and the *funnel bound*, were introduced by Wilber in [38]; a recent work by Lecomte and Weinstein [26] affirmatively settled the 30-year open question of whether the funnel bound was tighter than the interleave bound, proving a $\lg \lg n$ multiplicative separation in some cases. Another lower bound, the *rectangle bound*, was introduced by Demaine et al. in [18].

Progress on Dynamic Optimality. The BST which comes closest to dynamic optimality is the *tango tree* of Demaine et al. [19], which has a competitive ratio of $O(\lg \lg n)$ with respect to the best offline algorithm. Wilber's interleave bound was vital in the analysis of the competitive ratio, since the authors showed that on any access sequence x, the tango tree uses $O(\lg \lg n \ \text{IB}(x))$ accesses, where $\text{IB}(x)$ represents the interleave bound of the sequence. In [37], Wang et al. introduce the multi-splay tree, a BST which achieves $O(\lg \lg n)$ optimality with better worst-case guarantees than the tango tree. Prominent candidates for a dynamically optimal algorithm include the splay tree, which was presented by Sleator and Tarjan at the same time as the dynamic optimality conjecture [35], and the Greedy algorithm presented in Demaine et al.'s geometric interpretation of the BST model [18].

Other Data Structures and the BST model. The *min-heap*, which stores a set of keys and supports inserting arbitrary elements and extracting and deleting the minimum element. Recently, Kozma and Saranurak show an explicit relation between the BST model and the

heap cost model [25], as well as proposing an analogue of the dynamic optimality conjecture for heaps. Specifically, they show that for every heapsort algorithm (that is, an algorithm which sorts a permutation π with cost $\mathcal{A}(\pi)$ by inserting its keys into a heap and repeatedly extracting the minimum element) corresponds to an insertion sort into a BST algorithm which incurs cost $\mathcal{A}(\pi)$ on the inverse permutation π^{-1}. Their insight came from relating the rotation operation in a BST to the link operation in a heap, which allowed them to relate the corresponding cost models.

Adaptive Sorting in Parallel. During the period of interest in the adaptive sorting model, researchers were also interested in work-optimal parallel sorting algorithms with polylogarithmic span. Unsurprisingly, such algorithms exist for practical measures Runs and Inv [11, 14]; one also exists for Osc, a generalization of Inv [30] that is still theoretically weak. To our knowledge there are no results on parallel sorting algorithms which are optimal with respect to any stronger measures.

2 Preliminaries

Terminology. Throughout this paper, we will use the terms **list**, **permutation**, and **access sequence** interchangeably to refer to some ordering of the keys $1, 2, \ldots, n$. The term access sequence is used in the literature on BSTs to denote a sequence of queries to a BST; unless otherwise stated, an access sequence is presumed not to contain repeated keys.

The Binary Search Tree Model. A binary tree (BT) is either a *leaf* or a *node* consisting of a left binary tree, a key and a right binary tree. A binary seach tree (BST) is a binary tree where the keys have a total order, and for each node in the tree all keys in its left subtree are less than its key, and all keys in its right tree are greater.

The following definition of the BST model is drawn from [35, 38, 19]. The model assumes an initial BST with keys $[1, 2, \ldots, n]$ and an access sequence $[x_1, x_2, \ldots, x_m]$ of searches, where $x_i \in \{1, 2, \ldots, n\}$. Each search starts at the root and at each node it visits, it may perform one of the following actions: (a) move to the right child, left child, or parent, or (b) perform a rotation of the node and its parent. Each of these actions has unit cost and the search must visit its specified key. We refer to an algorithm that decides on what actions to perform for each search as a **BST algorithm**. A BST algorithm may be offline – meaning it can see the entire sequence of queries ahead of time – or online, meaning that queries are revealed one at a time.

Wilber's Interleave Bound. Wilber's interleave bound is a lower bound on the cost of accessing any sequence in the BST model. Given an access sequence π consisting of the keys $x_1, x_2, \ldots, x_n$, fix a static binary tree P (meaning it will never be rotated) with the keys of π at the leaves in the order they appear in π. Calculate the interleave bound of π as follows: query the keys in π in sorted order. For each vertex v_j, label each element i of the sequence with R or L, depending on whether accessing i in P goes through the right or the left subtree of v_j, respectively (if i is in neither subtree, give it no label). The interleave bound of v_j, denoted $\mathrm{IB}(v_j)$, is the number of switches between R and L in the labels if the keys are queried in sorted order. The interleave bound of the entire access sequence π is calculated by summing over the interleave bounds of each vertex, so $\mathrm{IB}(\pi) = \sum_{v \in P} \mathrm{IB}(v_i)$. As the lower bound holds for an arbitrary tree P, the interleave bound of the sequence is usually understood to refer to the maximum over all static trees. See Figure 3 for an example calculation.

Arborally Satisfied Sets. In [21], Derryberry et al. formalize a connection between binary search trees and points in the plane satisfying a certain property. An access sequence can be plotted in the plane where one axis represents key values and the other axis represents the ordering of the search sequence (that is, time). In the context of sorting, these axes can also be referred to as input order and output order. In this work, we use the horizontal axis for time and the vertical access for keyspace. See Figure 1 for an example of an arborally satisfied set.

In addition to plotting the search sequence on the plane, one can also plot the key values of the nodes which a BST algorithm accesses (for search or rotations) while searching for a node. When searching for an element x_i which is inserted at time i, the values of the nodes in the search path are plotted on the same vertical. Demaine et al. [18] proved that such a plot satisfies the following property:

▶ **Definition 2.** *Given a set P of points in the plane, P is **arborally satisfied** if for every two points $x, y \in P$ that are not on the same vertical or horizontal, the rectangle defined by x and y contains at least one point in addition to x and y.*

As mentioned in Section 1, a useful equivalent definition of arboral satisfaction is that there must be a monotonic path (e.g. consisting only of moves up and to the right) between x and y with a point at every corner. Note that any valid search on a BST will only touch nodes in a subtree τ_i of tree T, where τ_i includes the root of T. We sometimes refer to such a subtree as a **top tree** of T.

Demaine et al. show that a BST can be used to arborally satisfy an access sequence plotted in the plane. However, one can also use an algorithm that directly places points in the plane rather than using a BST.

▶ **Definition 3.** *Given a set of points in the plane corresponding to an access sequence π, an **offline arboral satisfaction algorithm** adds points to the plane to make an arborally satisfied set. An **online arboral satisfaction algorithm** also adds points in the plane to form an arborally satisfied set, but accesses are revealed one by one in input order and the algorithm must produce an arborally satisfied set at each time.*

Demaine et al. show that a BST algorithm is equivalent to an arboral satisfaction algorithm, but they also show a more surprising result: an arboral satisfaction algorithm is equivalent to a BST algorithm. Specifically, they show that an offline (online) arboral satisfaction algorithm requiring $f(\pi)$ accesses to arborally satisfy a search sequence π can be transformed to an offline (online) BST algorithm requiring $O(f(\pi))$ accesses to search for the elements of a sequence π.

3 Tree-based Sorting

Towards the goal of unifying the BST model and sorting, we ask the following question: when can the costs of a sorting algorithm be related to the costs of a BST algorithm? Clearly not every comparison-based sorting algorithm should be relatable to the BST model: as mentioned in Section 1, for example, an algorithm that guesses and checks could take $O(n)$ steps for an arbitrary permutation. Our investigation is therefore limited to sorting over binary trees, and in particular it considers a class of mergesort and partition sort (related to quicksort) algorithms on binary trees.

3.1 Mergesort

We first consider mergesorts in which the sequences to be merged are represented as BSTs. "Mergesort" is interpreted here as merging based on any split of the input sequence, not just splits into equally-sized parts; thus insertion sort using a sequence of insertions into a BST is a special case of mergesort. The cost of these algorithms is measured in terms of the number of accesses to the tree required during the mergesort, which will always be at least as great as the number of comparisons. We capture the idea of a BST mergesort more formally before giving the theorem statement.

A merge will interleave contiguous subsequences from its two inputs. We refer to each of these subsequences as *blocks* and we refer to the ends of each block as *block boundaries*. The block boundaries need to be accessed to even verify that the merge is correct. The following defines a merge that examines some top part of two trees to generate its output.

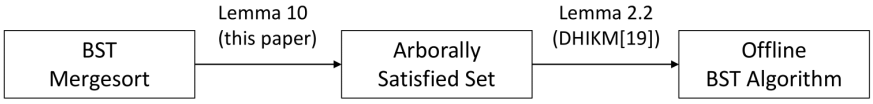
▶ **Definition 4.** *A* ***BST merge*** *takes two BSTs T_A and T_B, and for some top trees τ_a of T_A and τ_b of T_B, returns a BST T such that for some top tree τ of T, $\tau = \tau_a \cup \tau_b$, the subtrees of τ correspond to unchanged subtrees of τ_a and τ_b, and τ contains the block boundaries. The* number of accesses *used by the merge is $|\tau|$.*

▶ **Definition 5.** *A* ***BST mergesort*** *recursively splits the input sequence into two parts each of size at least 1, sorts each part with a BST mergesort, and executes a BST merge on the results. A mergesort on an input of size 1 returns its input. The* number of accesses *used by the mergesort is the sum of accesses across all merges.*

This leads to the main theorem.

▶ **Theorem 6.** *Let $\mathcal{A}$ be a BST mergesort algorithm which sorts permutation π using $\mathcal{A}(\pi)$ accesses. Then $OPT_{BST}(\pi) \in O(\mathcal{A}(\pi))$.*

Theorem 6 is proved by showing that for every BST mergesort algorithm, there is an offline BST algorithm that incurs the same cost as the BST mergesort within a constant factor. Our proof relies on the geometric interpretation of the BST model – instead of directly transforming a BST mergesort into an offline BST algorithm we use arborally satisfied sets as an intermediary. The key ingredient is a transformation of a BST merge algorithm to an offline arborally satisfied set algorithm, which is equivalent to an offline BST algorithm by Demaine et al.'s theorem [18]. The following graphic illustrates the chain of dependencies.

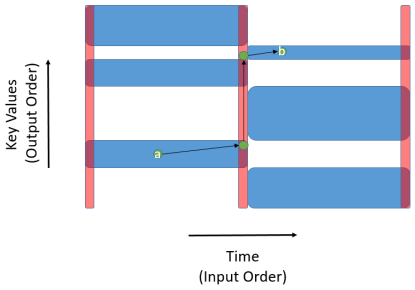


The main idea behind going from the mergesort to an arborally satisfied set is to transform a merge algorithm $\mathcal{M}$ into an algorithm that "merges" two arborally satisfied sets by concatenating them along the time axis (in this paper the x axis) and resolving any unsatisfied rectangles between the two sets, thus producing an arborally satisfied set. This "arboral" mergesort algorithm would by definition be an offline arborally satisfied set algorithm. Ideally this arboral merge would use the same number of accesses as a corresponding BST merge $\mathcal{M}$. The key idea behind our algorithm is to use the keys accessed during the tree merge to arborally satisfy the sets on the two sides, as well as add points to make it easier to satisfy the condition on future merges. In particular, the keys are added in

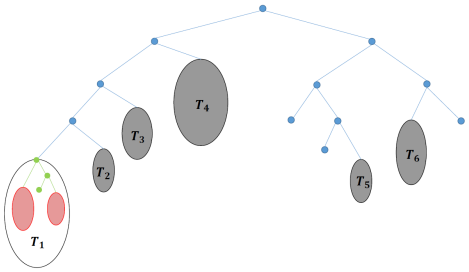
Algorithm 1 arboralMerge$(A, B, \mathcal{M})$
The arboral satisfaction algorithm; also illustrated in Figure 2a.

Input: Two arborally satisfied sets A, B; BST merge algorithm $\mathcal{M}$.
Output: An arborally satisfied set C consisting of the concatenation of A and B as well as additional accesses needed to arborally satisfy the concatenation.

```
1 if A == ∅ then return B ;
2 if B == ∅ then return A ;
3 C ← concatenate A and B on the time axis ;
4 S ← set of keys accessed when merging keys in A and B using M ;
5 access each key in S in the first, middle (rightmost column of A), and end columns of C ;
6 return C ;
```



(a) An illustration of an arboral merge.



(b) A BST separated into a top tree and auxiliary trees.

Figure 2 On the right, an illustration of the merging algorithm shown in Algorithm 1. The blue squares represent members of the two sets being merged, while the red columns (referred to as C_L, C_M, C_R in the proof of Theorem 6) illustrate the additional accesses necessary for the merge. A path drawn from $a \in A$ to $b \in B$ illustrates how the accesses in the middle column ensure the set is arborally satisfied. On the right, a BST with a top tree shown in blue and auxiliary trees T_1 through T_6, where the recursive structure is shown in T_1.

three columns: the leftmost column of the left set, the rightmost column of the right set, and a middle column – we will use the rightmost of the left set, although the leftmost of the right set would also work. Roughly, the accesses are placed down the middle to resolve unsatisfied rectangles, and they are placed down the leftmost and rightmost columns to restore invariants that are useful for future merges. These accesses must by definition form top trees of the two trees being merged. Algorithm 1 shows the merging routine, which is used as a sub-step in an arborally satisfied set algorithm which recursively splits the input set to singletons and then uses Algorithm 1 to merge sub-parts of the input; see Figure 2a for an illustration of the merge step. Since we are adding a constant number of points per access in the mergesort, the total points added is proportional to the cost of the mergesort, which in turn implies an offline BST algorithm with the same cost as the original BST mergesort.

The most difficult part of Theorem 6 is proving correctness of the arboral mergesort – that is, that each execution of the arboral merge routine produces an arborally satisfied set. This will require some more background on arborally satisfied sets.

▶ **Definition 7.** *A **treap** over a set of pairs S is a BST over the first coordinate of each $s \in S$ and a min-heap over the second coordinate. Ties over the second coordinate are permitted and may be broken arbitrarily.*

A treap with ties on the priorities can also be expressed as a **multi-treap** (for multi-node treap), where ties are stored in a **multi-node** that may have more than two children. Child relations must obey an underlying BST structure on the nodes stored within a multi-node; hence a multi-node may have at most one more child than the number of keys in the multi-node. A key component of our proof is that we will define a multi-treap with respect to each side of an arborally satisfied set and then relate these to the BST the mergesort will generate.

▶ **Definition 8.** *Given an arborally satisfied set A, let the left (right) priority be the distance from the left (right) boundary of the first point in the row (closer has higher priority). The* left (right) multi-treap *of A is the multi-treap defined by the (row, priority) pairs.*

Since there can be many points in any given column of an arborally satisfied set, multi-nodes of the multi-treaps can have more than two children.[2]

▶ **Definition 9.** *Given a BST T and an arborally satisfied set A with left (right) multi-treap H_A, T is* left (right) congruent with *A if there is some valid BST structure on H_A (forming a binary subtree within each multi-node) such that the tree structure on H_A is equal to T. T is* doubly *congruent with A if it is both left and right congruent.*

Note that many BSTs can be left (equivalently right) congruent with the same arborally satisfied set A due to equal priorities. Also many arborally satisfied sets can be left (right) congruent with the same BST T. It may seem unlikely that a BST is doubly congruent with a arborally satisfied set, but in our construction we will maintain double congruence, and in particular we will show that the point sets created by Algorithm 1 are doubly congruent to the corresponding tree.

The following observation will be useful for the proof of Theorem 6. It follows from a similar argument to Lemma 2.1 of Demaine et al. [18].

▶ **Observation 10.** *Consider an arborally satisfied set A that is double-congruent with a tree T. Then for any top tree τ of T, if the keys in τ are accessed along either the left or right column of A, or one past the left or right column, the resulting set of points is arborally satisfied.*

Proof. Begin with the case where the keys of τ are accessed one past the leftmost or rightmost column of A. Assume for the sake of contradiction that there exists an unsatisfied rectangle – that is, a rectangle with two points at its corners and no points contained within it – between access $a \in A$ and access $b \in \tau$. Consider the least common ancestor c of a and b, whose key value must be between those of a and b. Since by our assumption, there are no keys accessed between the rectangle defined by a and b, this contradicts the fact that τ is a continuous subtree of T, since c must be accessed in τ to reach b.

This leaves the case where accesses to τ are instead placed on the leftmost or rightmost column of A – that is, in addition to accesses that were already there. Consider an arbitrary access $a \in A$ and any access $b \in \tau$. If the accesses in τ are placed on a new column past the rightmost (leftmost) column of A, there is a monotonic path from a to b with accesses at every corner. If the accesses in τ are imposed on the rightmost (leftmost) column of A instead, the same accesses still form a monotonic path, since this only causes the elimination of one right (left) move. ◀

[2] Demaine et al. [18] define a similar notion when proving that for any arborally satisfied set there is a BST execution with equivalent cost, but only with respect to one side, and only when sweeping column by column.

We now prove correctness of the arboral mergesort algorithm.

▶ **Lemma 11.** *The arboral mergesort algorithm is correct: that is, it returns an arborally satisfied set.*

Proof. We will use the following inductive hypothesis on the arboral merge algorithm (Algorithm 1) to show correctness: Algorithm 1 returns an arborally satisfied set which is double-congruent to the tree T returned by the corresponding BST mergesort algorithm.

Base Case. When the set is just a single point, both arboral satisfaction and double congruence follow trivially.

Inductive Step. The inductive step is broken into several claims, and some new notation is called for. The two arborally satisfied sets being merged are A and B, and by the inductive hypothesis are both arborally satisfied and double-congruent to trees T_A and T_B, respectively. The additional accesses specified by the mergesort are added to three columns. Let C_L, C_R, and C_M denote the set of points which the arboral merge adds along the left, right, and middle columns respectively; see Figure 2a for an illustration. It will also be useful to denote the subset of a C_i ($i \in \{L, R, M\}$) consisting only of accesses to keys in A or B. These subsets are denoted by $C_i(A)$ or $C_i(B)$.

The merge will break A and B into contiguous blocks that are interleaved in key order. As assumed in the model, the block boundaries must be accessed by the merge, and therefore included in the C_i. In general the C_i will include other points as well. The inductive step has to show both that the resulting set is arborally satisfied and is doubly congruent to the merged tree T.

Arboral Satisfaction. Points within A or B are satisfied by the inductive hypothesis. Points both in C_L, C_R, C_M are satisfied by the fact that they access precisely the same keys. This leaves the two more interesting cases: (1) pairs of points one from the previously exiting points (in A or B) and one from the new boundaries C_L, C_R, C_M, and (2) pairs of points one from A and one for B. For the first case, A is double-congruent to tree T_A (by the inductive hypothesis), and $C_i(A)$ is a top tree of A (by construction), so we can apply Observation 10 for points in A and $C_i(A)$. Now since the boundaries of each block of A must be in $C_i(A)$, we can get from a point in A to a point in $C_i(B)$ using a monotonic path by going to a boundary point in $C_i(A)$ and then up or down the column to the point in $C_i(B)$. Symmetrically points in B can get to points in $C_i(B)$ and $C_i(A)$ by a monotonic path. For case (2) consider any point $a \in A$ and point $b \in B$. There is a monotonic path between a and b by composing the monotonic path between a and a boundary point in p_a in $C_m(A)$, between b and a boundary point p_b in $C_m(B)$, and between p_a and p_b, which are in the same column; see Figure 2a.

Double congruency to T. We will show that the set AB returned by the arboral merge algorithm is right congruent to T. Left congruency is true by symmetry. The keys in C_R (those accessed by the merge) correspond to a top tree τ of T. Since all keys in the column C_R have the same highest priority we can organize the root multi-node to match the structure of τ making those nodes congruent. Now consider the subtrees not in τ. They properly are lower in the tree and have equal or lower priority (only equal if they happen to be in the last row). Furthermore the subtrees are separated by keys in τ. Each such subtree either comes completely from T_A or completely from T_B and have the same structure as before the merge (they were not touched by the merge). Furthermore the points from T_A (T_B) only appear in

A (B). This implies the relative priorities have not change for those points when merging into AB. By induction, the trees T_A (T_B) were congruent to A (B) before the merge so the subtrees were congruent and remain congruent after the merge (neither the relative priorities nor tree structure have changed). This implies the whole tree T is congruent with AB. ◀

The proof of Theorem 6 now follows easily.

Proof of Theorem 6. The theorem follows once the cost of the arboral mergesort is shown to be $O(\mathcal{A}(\pi))$. Since the cost of $\mathcal{A}$ is dominated by the cost of each merge execution, and that cost is at most multiplied by six in each call to the arboral merge, the cost of the arboral mergesort is at most $6\mathcal{A}(\pi)$. When the arboral mergesort is transformed into an offline BST algorithm, the cost remains the same for a total cost of $O(\mathcal{A}(\pi))$. ◀

3.2 Partition Sort

We now consider a class of sorting algorithms motivated by quicksort, which we refer to as partition sorts. As in the case of mergesort, we limit ourselves to working with binary trees. However, in this case the trees are not ordered by key, but instead are ordered by input order. The algorithm is like quicksort in that it picks a pivot, partitions the keys on the pivot and recurses. Since we are interested in lower bounds (i.e. showing the cost of partition sort is at least as great as optimal BSTs), we can assume an oracle picks the perfect pivot (e.g., the median). As with mergesort, to achieve better than trivial $O(n \log n)$ bounds it is important that the partition need not visit the whole tree it is partitioning, but rather just some top tree. This allows for sending whole subsequences to the lesser or greater/equal side without visiting all nodes. More precisely here are the definitions of partition and partition sort.

▶ **Definition 12.** *A **BT partition** takes a BT T and for some top tree τ of T returns two BTs T_A, T_B with distinct keys such that for some top trees τ_a of T_A and τ_b of T_B, $\tau = \tau_a \cup \tau_b$, the other subtrees of T appear in either T_A or T_B unchanged, and τ contains the block boundaries of the partitioned output. Furthermore the preordering of the keys in T_A or T_B are a subsequence of the preordering in T (i.e. left-to-right ordering). We assume both partitions are non-empty. The* number of accesses *used by the partition is $|\tau|$.*

▶ **Definition 13.** *A **BT partition sort** on a BT tree T (1) partitions T into T_a and T_b such that for some key k all keys in T_a are less than k and all keys in T_b are greater or equal to k, (2) recurses on each partition, and (3) returns the left and right results appended. The recursion terminates when the tree is of size one. The number of accesses used by the partition sort is the sum of accesses across all partitions.*

Our goal is to show the following theorem, which has the same form as the result for mergesort.

▶ **Theorem 14.** *Let $\mathcal{A}$ be a BT partition sort algorithm that sorts permutation π using $\mathcal{A}(\pi)$ accesses, and let $OPT_{BST}(\pi)$ be the optimal cost of querying π with a BST algorithm. Then $OPT_{BST}(\pi) \in O(\mathcal{A}(\pi))$.*

Our approach is to show a one-to-one correspondence between the tree-based merge and partition sorts.

▶ **Lemma 15.** *For any BT partition sort algorithm $\mathcal{A}$ that sorts permutation π using $\mathcal{A}(\pi)$ accesses, there is a BST mergesort sort algorithm $\mathcal{B}$ that sorts permutation π^{-1} using $\mathcal{B}(\pi^{-1}) = \mathcal{A}(\pi)$ accesses, and vice versa.*

Proof. The idea is to consider running BST mergesort backwards, while reversing the role of time and key order. Consider undoing a merge – i.e. taking the merged tree and partitioning back into its two inputs. Reversing the roles of time and keys, this is equivalent to a BT partition where keys are time order and the partitions is on the first time of the right partition. In particular the size of the top tree and therefore access cost is identical. This continues to be true on the recursive calls. In both cases the base case is of size one. Hence the total access costs of the two algorithms are identical, one applied to the inverse permutation of the other. ◀

Since the size of arborally satisfied sets are invariant under rotation by 90 degrees, reversing the role does not affect the size of the set. Since the proof of Theorem 6 first showed how to map a BST mergesort to an arborally satisfied set and this implied the same bound on a offline BST, this remains true if we rotate the input, arborally satisfy it in the same way, and generate a BST. Theorem 14 follows. Taken together, Theorem 6 and Theorem 14 show the statement in Theorem 1. Although the duality of mergesort and quicksort has been recognized before we are not aware of any formal correspondence such as the one given here.

4 The Log-Interleave Bound

The following two sections contain results that illustrate the utility of the approach shown in Theorem 1: namely, that results in the sorting cost model can directly translate to interesting results in the BST model. In this section we propose an information-theoretic bound on both the cost of accessing a sequence in the BST model and sorting a list in the comparison model. Theorem 17 shows that the log-interleave bound is within a $\lg \lg n$ multiplicative factor of a known lower bound in the BST model. In the next section, we show that there exists a BST mergesort algorithm that sorts any permutation π in $O(\mathrm{LIB}(\pi))$ comparisons, and thus combined with Theorem 6 shows the existence of an offline BST algorithm with the same costs in the BST model.

The log-interleave bound can be thought of as an algorithmic perspective on Wilber's interleave bound. Let P be the static tree with the keys of a permutation π at the bottom. Consider sorting π via mergesort: clearly, each non-leaf vertex of P denotes a merge. The interleave bound charges unit cost for each switch between the right and left subtree during a merge. Another way of looking at this cost is that every continuous run of accesses to the left subtree incurs unit cost. Thus, a mergesort with a merge step that incurred unit cost for each consecutive run – as opposed to the standard mergestep which charges for the size of each run – would sort π using $O(\mathrm{IB}(\pi))$ comparisons.

Lecomte and Weinstein [26] and Chalermsoook et al. [12] independently show that the merge step described above does not exist. However, it is possible to charge the logarithm of the size of each consecutive run, as shown by Brown and Tarjan in [10]. This idea leads us to using such a merge step as an information-theoretic bound, which applies to sorting (sequentially and in parallel), and the BST model. The log-interleave bound is formally defined below; note its similarity to the interleave bound.

▶ **Definition 16.** *Given an access sequence π, fix a static binary tree P with the keys of π at the leaves. For each vertex v_j, query the descendants of v_j in sorted order, then label each with R or L depending on whether it is in the left or right subtree of v_j. Let $S(v_j)$ represent the decomposition of this labeling into the smallest possible number of runs of consecutive accesses to L or R in v_j. Then $LIB(v_j) = \sum_{r_i \in S(v_j)} \lg(|r_i|+1)$ and $LIB(\pi) = \sum_{v \in P} LIB(v_i)$.*

See Figure 3 for an example calculation.

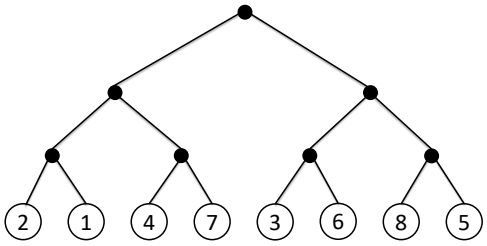


Figure 3 Consider accessing the keys 1-8 in order. For the vertex v_1 at the root of the tree shown here, the labeled access sequence is [L, L, L, R, L, R, R, R]. Since the access sequence switches between the left and right subtree three times, $\mathrm{IB}(v_1) = 3$. Similarly, $\mathrm{LIB}(v_1) = \lg 4 + \lg 2 + \lg 2 + \lg 4$ since the smallest possible decomposition of the labeled access sequence consists of [[L, L, L], [R], [L], [R, R, R]].

A natural question one might ask about a BST algorithm or an adaptive sorting algorithm is how far, in the worst case, is the cost of this algorithm from any known lower bounds? Or, in other words, how close is this algorithm to optimal? In this section, we will settle this question for the log-interleave bound in the BST model, ending in the following theorem:

▶ **Theorem 17.** *For any permutation π, $IB(\pi) \leq LIB(\pi) \in O(\lg\lg n \; IB(\pi))$.*

Our first step is to show that we cannot hope to do better than a $\lg\lg n$ separation; this is stated in the following lemma. This result is similar to the separation result in Theorem 2 of Lecomte and Weinstein [26]; furthermore, the result implies that an online BST algorithm using $\mathrm{LIB}(\pi)$ accesses cannot be dynamically optimal.

▶ **Lemma 18.** *There exists a permutation π such that $LIB(\pi) = \Theta(\lg\lg n \; IB(\pi))$.*

Proof. First we will need to define a particularly useful permutation. The *bit-reversal permutation* π_B on $n = 2^k$ keys is generated by taking a sorted list $[0, 1, \ldots, n]$, writing each key in binary, then reversing the bits of each key. For example, the bit-reversal permutation on 8 keys is $[0, 4, 2, 6, 1, 5, 3, 7]$. Let P be a static tree with keys of π_B at the bottom: then querying its keys in sorted order will switch between the left and right subtree of any $v_j \in P$ on each query. This implies that $\mathrm{IB}(\pi_B) = \Theta(n \lg n)$, thus showing that any BST algorithm will incur this cost when querying π_B.

Consider the permutation π obtained by splitting the sorted list into $n/\lg n$ segments of equal size, and then permuting those $n/\lg n$ segments according to the bit-reversal permutation π_B.

The interleave bound of π will be the same as for a list with $n/\lg n$ elements permuted according to the bit-reversal sequence – that is, $(n/\lg n)\lg(n/\lg n) = O(n)$. In π, every block is of size $\lg n$, so to calculate the log-interleave bound, we multiply by $\lg\lg n$ on all but the bottom $\lg\lg n$ levels. Thus, the log-interleave bound of π is $\Theta(n \lg\lg n)$ while $\mathrm{IB}(\pi) = O(n)$. ◀

Now we have shown there is no possibility of doing better than a $\lg\lg n$ separation, we show that this separation is tight. This starts with the following question: when are the interleave bound and the log-interleave bound farthest apart? It follows from the convexity of the logarithm that for each vertex v_j of a static tree, the interleave bound and the

log-interleave bound are farthest apart when v_j experiences long runs of consecutive accesses to its subtrees (e.g. the list [L, L, L, R, R, R] has fairly different values for its interleave bound and log-interleave bound, but the list [L, R, L, R, L, R] does not).

However, a completely sorted list π_S – translating to the longest run size possible for each vertex of P – has $\mathrm{IB}(\pi_S) = \mathrm{LIB}(\pi_S) = \Theta(n)$. This suggests there must be some intermediate value of the block size that maximizes the difference between the two bounds. As the reader might have inferred from Lemma 18, that size will turn out to be $\lg n$. The next lemma, whose proof follows directly from the convexity of the logarithm, formalizes this intuition.

▶ **Lemma 19.** *For a permutation π, let v be a vertex of the corresponding static tree P such that $IB(v) = S$. Then $LIB(v)$ will differ from $IB(v)$ by the greatest amount when each "run" of L or R in the labeled sequence is the same size.*

Now that we have established that the interleave bound and the log-interleave bound differ the most when all continuous runs are the same size, we move on to ask the following question: how large do the runs of the same size have to be to further maximize this difference? The next lemma shows this fact in the following way: in the inequality below, the expression $S \lg\left(\frac{n}{S}+1\right)$ bounds the log-interleave bound of any π such that $\mathrm{IB}(\pi) = S$. The left-hand expression $c\lg(\lg n+1)\left(S+\frac{n}{\lg n}\right)$ will directly suffice to prove Theorem 17. The proof of the lemma draws out the fact that the two expressions are closest to each other when the size of the continuous runs is $\lg n$.

▶ **Lemma 20.** *For a permutation π, let v be a vertex of the corresponding static tree P such that $IB(v) = S$. Furthermore, let the number of leaves below vertex v be n. Then for some constant c,*

$$c\lg(\lg n+1)\left(S+\frac{n}{\lg n}\right) \geq S\lg\left(\frac{n}{S}+1\right).$$

Proof. Assume for the sake of contradiction that

$$c\lg(\lg n+1)S + c\lg(\lg n+1)\frac{n}{\lg n} < S\lg\left(\frac{n}{S}+1\right).$$

This would imply that each added term is smaller than $S\lg\left(\frac{n}{S}+1\right)$. Begin by examining the case where the first term is smaller than the term on the right:

$$\begin{aligned} \lg(\lg n+1)S &< S\lg\left(\frac{n}{s}+1\right) \\ \implies \lg(\lg n+1) &< \lg\left(\frac{n}{S}+1\right) \\ \implies \lg n &< \frac{n}{S} \\ \implies S &< \frac{n}{\lg n}. \end{aligned}$$

This shows that when $S \geq \frac{n}{\lg n}$, we reach a contradiction and our claim holds. Now, when $S < \frac{n}{\lg n}$, the second term in the sum dominates. When the second term dominates, the expression reads

$$c\lg(\lg n+1)\cdot\frac{n}{\lg n} \geq \frac{n}{\lg n}\lg(\lg n+1)$$

which is self-evidently true for all $c \geq 1$. ◀

Now, these two lemmas are put together to prove Theorem 17.

Proof of Theorem 17. Let P be the static tree corresponding to π, and for each vertex v_i of P, let $S_i = \mathrm{IB}(v_i)$. By Lemma 19, we can assume that if the number of leaves below v_i is n_i, then $\mathrm{LIB}(v_i) = S_i \lg\left(\frac{n_i}{S} + 1\right)$. Then $\mathrm{IB}(\pi) = \sum_{i=1}^{n-1} S_i$. Next, we can use the upper bound on $\mathrm{LIB}(v_i)$ from Lemma 20 to upper bound $\mathrm{LIB}(\pi)$:

$$\begin{aligned}\mathrm{LIB}(\pi) &\leq \sum_{i=1}^{n-1} c \lg\lg n \left(S + \frac{n_i}{\lg n}\right) \\ &= c \lg\lg n \left(\mathrm{IB}(\pi) + \frac{1}{\lg n} \sum_{i=1}^{\lg n} 2^i \frac{n}{2^i}\right) \\ &= c \lg\lg n \left(\mathrm{IB}(\pi) + n\right) = O(\mathrm{IB}(\pi) \lg\lg n).\end{aligned}$$

◀

5 Adaptive Parallel Mergesort

In this section we present a parallel BST mergesort which sorts a permutation π using $O(\mathrm{LIB}(\pi))$ accesses, and the same amount of work. We refer to the algorithm as an *adaptive parallel mergesort*.

First we introduce the data structure used in our mergesort. Given a BST T and a key k, a **split** refers to returning two BSTs, one containing all keys from T which are greater than k, and one containing all keys which are less than k. Given two BSTs T_1, T_2 such that any key in T_1 is greater than every key in T_2, **join** returns a single BST T containing the union of the keys in T_1 and T_2. As previously stated, we assume keys are unique.

The tree used in our mergesort algorithm is a modified red-black tree described by Tarjan and Van Wyck in [36], which they call a *heterogeneous finger search tree*. These trees have the useful property that a key d in a heterogenous finger search tree with n elements can be accessed in time $O(\log(\min(d, n-d)+1))$. This property allowed Tarjan and Van Wyck to devise fast split and join algorithms for heterogenous finger search trees; split runs in amortized time $O(\lg(\min(|T_1|, |T_2|)+1))$ – that is, the logarithm of the size of the smaller tree returned. Join similarly is bounded by amortized time $O(\lg(\min(|T_1|, |T_2|)+1))$ – in this case, the size of the smaller of the two trees being joined together. The worst-case complexity of split and join is $O(\log \max |T_1|, |T_2|)$. As presented in [36], the heterogeneous finger search tree is not strictly a BST as it uses more than one pointer; however, work by [13] shows how it can be converted into using a single pointer with an additional constant factor loss.

The natural parallel algorithm to merge two trees is as follows: starting with two trees, split each tree using the other tree's root; then, recurse in parallel to merge the two left halves and the two right halves, respectively, joining the two at the end. This idea was presented by Blelloch et al. [4], and is shown here in Algorithm 2. This algorithm, however, does not meet the log-interleave bound even if we use heterogeneous finger search trees for the split and join. We therefore modify the algorithm as is shown in Algorithm 3 and illustrated in Figure 4, which follows the same idea with some small modifications. In addition to splitting the second tree T_2 into L_2 and R_2 based on the root of the first tree (T_1), it then splits T_1 by the maximum value of L_2 and the minimum value of R_2 to effectively break T_1 into three parts. The middle part need not be split recursively since it falls between two elements of T_2. This avoids redundant splits.

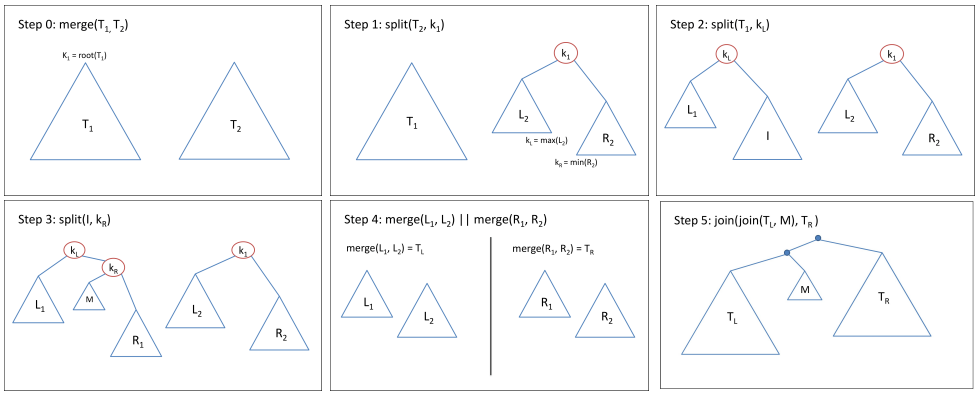


Figure 4 One round of our recursive merge algorithm. The nodes shown in red are the nodes used to split a tree; the small blue nodes denote merges.

Algorithm 2 union(T_1, T_2).
Blelloch et al.'s union algorithm. Here, the function *expose* refers to returning the root and its right and left subtrees.

Input: Two BSTs T_1, T_2 with disjoint keys.
Output: A BST containing the union of the keys of T_1 and T_2

if $T_1 =$ *Leaf* **then return** T_2 ;
else if $T_2 =$ *Leaf* **then return** T_1 ;
else
  $L_1, k_2, R_1 = \text{expose}(T_1)$;
  $L_2, R_2 = \text{split}(T_2, k)$;
  do in parallel
    $T_L = \text{union}(L_1, L_2)$;
    $T_R = \text{union}(R_1, R_2)$;
return *join*(T_L, k_2, T_R);

Algorithm 3 mergeHT(T_1, T_2)
Pseudocode for the merge step of our mergesort.

Input: Two BSTs T_1, T_2
Output: A BST containing the union of the keys of T_1 and T_2

if $T_1 =$ *Leaf* **then return** T_2 ;
else if $T_2 =$ *Leaf* **then return** T_1 ;
else
  $k = \text{root}(T_1)$;
  $L_2, R_2 = \text{split}(T_2, k)$;
  $k_1 = \max(L_2)$; $k_2 = \min(R_2)$;
  $L_1, I = \text{split}(T_1, k_1)$;
  $M, R_1 = \text{split}(I,\ k_2)$;
  do in parallel
    $T_L = \text{merge}(L_1, L_2)$;
    $T_R = \text{merge}(R_1, R_2)$;
return *join*(*join*(T_L, M), T_R);

It is not immediate our modified algorithm's work is bounded by the log-interleave bound, since the set of splits and joins it performs does not neatly correspond to the sums of block sizes at each level in the static tree used to calculate the log-interleave bound. We will show that this different sequence of splits and joins also performs within the log-interleave bound, culminating in the following theorem:

▶ **Theorem 21.** *There exists a parallel mergesort which for any permutation π performs $O(LIB(\pi))$ work with polylogarithmic span.*

Furthermore, since the algorithm proposed is a BST mergesort, it follows from Theorem 6 that there also exists an offline BST algorithm with the same cost in the BST model:

▶ **Corollary 22.** *There exists an offline BST algorithm $\mathcal{A}$ such that $\mathcal{A}(\pi) = O(LIB(\pi))$.*

The proofs of Theorem 21 and Corollary 22 are shown in the full version of the paper.

References

1 N. S. Arora, R. D. Blumofe, and C. G. Plaxton. Thread scheduling for multiprogrammed multiprocessors. *Theory of Computing Systems (TOCS)*, 34(2), April 2001.

2 Nicolas Auger, Vincent Jugé, Cyril Nicaud, and Carine Pivoteau. On the worst-case complexity of TimSort. In *European Symposium on Algorithms (ESA)*, volume 112. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

3 Mihai Badoiu, Richard Cole, Erik D. Demaine, and John Iacono. A unified access bound on comparison-based dynamic dictionaries. *Theoretical Computer Science*, 382(2), 2007.

4 Guy E. Blelloch, Daniel Ferizovic, and Yihan Sun. Just join for parallel ordered sets. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2016.

5 Guy E. Blelloch, Jeremy T. Fineman, Yan Gu, and Yihan Sun. Optimal parallel algorithms in the binary-forking model. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2020.

6 Robert D. Blumofe and Charles E. Leiserson. Scheduling multithreaded computations by work stealing. *J. ACM*, 46(5), 1999.

7 Prosenjit Bose, Jean Cardinal, John Iacono, Grigorios Koumoutsos, and Stefan Langerman. Competitive online search trees on trees. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2020.

8 Prosenjit Bose, Karim Douïeb, John Iacono, and Stefan Langerman. The power and limitations of static binary search trees with lazy finger. In *International Symposium on Algorithms and Computation (ISAAC)*, volume 8889. Springer, 2014.

9 Mark R Brown and Robert E Tarjan. A fast merging algorithm. *Journal of the ACM (JACM)*, 26(2), 1979.

10 Mark R. Brown and Robert Endre Tarjan. Design and analysis of a data structure for representing sorted lists. *SIAM J. Comput.*, 9(3):594–614, 1980.

11 Svante Carlsson and Jingsen Chen. An optimal parallel adaptive sorting algorithm. *Inf. Process. Lett.*, 39(4), 1991.

12 Parinya Chalermsook, Julia Chuzhoy, and Thatchaphol Saranurak. Pinning down the strong wilber 1 bound for binary search trees. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*, volume 176 of *LIPIcs*, pages 33:1–33:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.33.

13 Parinya Chalermsook, Mayank Goswami, László Kozma, Kurt Mehlhorn, and Thatchaphol Saranurak. Multi-finger binary search trees. In *International Symposium on Algorithms and Computation (ISAAC)*, volume 123. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

14 Jingsen Chen and Christos Levcopoulos. Improved parallel sorting of presorted sequences. In *Joint Conference on Vector and Parallel Processing (CONPAR - VAPP)*, volume 634. Springer, 1992.

15 Richard Cole. On the dynamic finger conjecture for splay trees. Part II: The proof. *SIAM Journal on Computing*, 30(1), 2000.

16 Richard Cole, Bud Mishra, Jeanette Schmidt, and Alan Siegel. On the dynamic finger conjecture for splay trees. Part I: Splay sorting log n-block sequences. *SIAM Journal on Computing*, 30(1), 2000.

17 Curtis R. Cook and Do Jin Kim. Best sorting algorithms for nearly sorted lists. *Communications of the ACM*, 23(11), 1980.

18 Erik D. Demaine, Dion Harmon, John Iacono, Daniel Kane, and Mihai Patrascu. The geometry of binary search trees. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2009.

19 Erik D. Demaine, Dion Harmon, John Iacono, and Mihai Patrascu. Dynamic optimality–almost. *SIAM Journal on Computing*, 37(1), 2007.

20 J. Derryberry, D. D. Sleator, and C. C. Wang. A lowerbound framework for binary search trees with rotations. Technical report, Technical Report CMU-CS-05-187, School of Computer Science, Carnegie Mellon University, 2005.

21 Jonathan Derryberry. *Adaptive Binary Search Trees*. PhD thesis, Carnegie Mellon University, 2009.

22 Vladimir Estivill-Castro and Derick Wood. A new measure of presortedness. *Information and Computation*, 83(1), 1989.

23 John Iacono and Stefan Langerman. Weighted dynamic finger in binary search trees. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2016.

24 Jyrki Katajainen, Christos Levcopoulos, and Ola Petersson. Local insertion sort revisited. In *International Symposium on Optimal Algorithms*, 1989.

25 László Kozma and Thatchaphol Saranurak. Smooth heaps and a dual view of self-adjusting data structures. In *ACM Symposium on Theory of Computing (STOC)*. ACM, 2018.

26 Victor Lecomte and Omri Weinstein. Settling the relationship between Wilber's bounds for dynamic optimality. In *European Symposium on Algorithms (ESA)*, 2020.

27 Christos Levcopoulos and Ola Petersson. Adaptive heapsort. *Journal of Algorithms*, 14(3), 1983.

28 Christos Levcopoulos and Ola Petersson. Sorting shuffled monotone sequences. In *Scandinavian Workshop on Algorithm Theory*, 1990.

29 Christos Levcopoulos and Ola Petersson. Splitsort–an adaptive sorting algorithm. *Information Processing Letters*, 39(4), 1991.

30 Christos Levcopoulos and Ola Petersson. Exploiting few inversions when sorting: Sequential and parallel algorithms. *Theor. Comput. Sci.*, 163(1&2), 1996.

31 Heikki Mannila. Measures of presortedness and optimal sorting algorithms. *IEEE Transactions on Computers*, C-34(4), 1985.

32 Peter McIlroy. Optimistic sorting and information theoretic complexity. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1993.

33 Alistair Moffat and Ola Petersson. Historical searching and sorting. In *International Symposium on Algorithms*, 1991.

34 Ola Petersson and Alistair Moffat. A framework for adaptive sorting. *Discrete Applied Mathematics*, 59, 1995.

35 Daniel Dominic Sleator and Robert Endre Tarjan. Self-adjusting binary search trees. *J. ACM*, 32(3), 1985.

36 Robert Endre Tarjan and Christopher J. Van Wyk. An $O(n \log \log n)$-time algorithm for triangulating a simple polygon. *SIAM J. on Computing*, 17(1), 1988.

37 Chengwen Chris Wang, Jonathan Derryberry, and Daniel Dominic Sleator. $O(\log \log n)$-competitive dynamic binary search trees. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*. ACM Press, 2006.

38 Robert Wilber. Lower bounds for accessing binary search trees with rotations. *SIAM Journal on Computing*, 18(1), 1989.

Parameterized Complexity of Binary CSP: Vertex Cover, Treedepth, and Related Parameters

Hans L. Bodlaender
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Carla Groenland
Mathematical Institute, Utrecht University, The Netherlands

Michał Pilipczuk
Institute of Informatics, University of Warsaw, Poland

Abstract

We investigate the parameterized complexity of Binary CSP parameterized by the vertex cover number and the treedepth of the constraint graph, as well as by a selection of related modulator-based parameters. The main findings are as follows:

- Binary CSP parameterized by the vertex cover number is W[3]-complete. More generally, for every positive integer d, Binary CSP parameterized by the size of a modulator to a treedepth-d graph is W[$2d+1$]-complete. This provides a new family of natural problems that are complete for odd levels of the W-hierarchy.
- We introduce a new complexity class XSLP, defined so that Binary CSP parameterized by treedepth is complete for this class. We provide two equivalent characterizations of XSLP: the first one relates XSLP to a model of an alternating Turing machine with certain restrictions on conondeterminism and space complexity, while the second one links XSLP to the problem of model-checking first-order logic with suitably restricted universal quantification. Interestingly, the proof of the machine characterization of XSLP uses the concept of *universal trees*, which are prominently featured in the recent work on parity games.
- We describe a new complexity hierarchy sandwiched between the W-hierarchy and the A-hierarchy: For every odd t, we introduce a parameterized complexity class S[t] with W[t] $\subseteq$ S[t] $\subseteq$ A[t], defined using a parameter that interpolates between the vertex cover number and the treedepth.

We expect that many of the studied classes will be useful in the future for pinpointing the complexity of various structural parameterizations of graph problems.

2012 ACM Subject Classification Theory of computation → W hierarchy

Keywords and phrases Parameterized Complexity, Constraint Satisfaction Problems, Binary CSP, List Coloring, Vertex Cover, Treedepth, W-hierarchy

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.27

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://doi.org/10.48550/arXiv.2208.12543

Funding *Carla Groenland*: Supported by the Marie Skłodowska-Curie grant GRAPHCOSY (number 101063180).
Michał Pilipczuk: This research is a part of the project BOBR that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 948057).

1 Introduction

The Binary Constraint Satisfaction Problem (BinCSP, for short) is a fundamental problem defined as follows. We are given an undirected graph $G = (V, E)$, called the *primal* or the *Gaifman graph*, where V is a set of variables, each with a prescribed domain of possible

values. Further, each edge uv of G corresponds to a binary constraint that restricts the possible pairs of values that can be assigned to u and v. The task is to decide whether every variable can be mapped to a value from its domain so that all the constraints are satisfied.

Due to immense modeling power, constraint satisfaction problems are of great importance in multiple applications, and the theoretical study of their complexity is a field on its own. In this work we are interested in parameterized algorithms for BinCSP, with a particular focus on structural parameters of the Gaifman graph. An example of such a result is a classic observation, usually attributed to Freuder [32]: using dynamic programming, BinCSP can be solved in time $n^{k+\mathcal{O}(1)}$, where n is the maximum size of a domain and k is the treewidth of the Gaifman graph. In the language of parameterized complexity, this means that BinCSP parameterized by treewidth is *slice-wise polynomial*, or in the complexity class XP.

The class XP is very general and just placing BinCSP parameterized by treewidth within XP does not provide much insight into the actual complexity of the problem. A more detailed study of the parameterizations of BinCSP by pathwidth and by treewidth was recently performed by Bodlaender, Groenland, Nederlof, and Swennenhuis in [11], and by Bodlaender, Groenland, Jacob, Pilipczuk, and Pilipczuk in [10]. In particular, as shown in [11], BinCSP parameterized by pathwidth is complete for XNLP: the class of all parameterized problems that can be solved by a nondeterministic Turing machine using $f(k)\log n$ space and $f(k)\cdot n^{\mathcal{O}(1)}$ time, where k is the parameter and f is a computable function. A "tree variant" of XNLP, called XALP, was studied in [10]; it can be defined using the same model of a Turing machine, except that the machine additionally has access to a stack of unbounded size that can be manipulated by pushing and popping. As proved in [10], BinCSP parameterized by treewidth is complete for XALP. All in all, the recent works [7, 9, 10, 11, 26] present a variety of problems on graphs with linear or tree-like structure that are complete for XNLP and XALP, respectively. This is an evidence that XNLP and XALP capture certain fundamental varieties of computational problems: those amenable to linearly and tree-structured dynamic programming with state space of slice-wise polynomial size.

The contemporary research in parameterized algorithms features many more structural parameters of graphs, besides treewidth and pathwidth. In this work we explore the complexity of BinCSP parameterized by the following parameters of the Gaifman graph: (1) the vertex cover number, (2) the treedepth, and (3) a selection of related modulator-based parameters lying between the vertex cover number and the treedepth.

New completeness results for the W-hierarchy. The W-hierarchy was introduced around thirty years ago in the work by Downey and Fellows that founded the field of parameterized algorithms and complexity. In this hierarchy, we have a collection of classes, including $\mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \ldots \subseteq \mathrm{W}[\mathrm{SAT}] \subseteq \mathrm{W}[\mathrm{P}]$; see [20, 21, 30] for an overview and for bibliographic references. A large variety of problems are known to be complete (under fpt reductions) for W[1] and for W[2]. However, for classes W[t] with $t \geqslant 3$, there is so far only a handful of examples of natural problems known to be complete [1, 5, 6, 14, 15, 36]. Our first contribution is to give new examples of complete problems for W[t] for all odd $t \geqslant 3$.

Our first example concerns BinCSP parameterized by the *vertex cover number*: the minimum size of a vertex cover in the Gaifman graph.

▶ **Theorem 1.** *BinCSP parameterized by the vertex cover number of the Gaifman graph is complete for the class* W[3].

It was known that BinCSP parameterized by the vertex cover number is W[1]-hard [29, 44]. The W[3]-completeness is surprising, not only due to the small number of examples of natural W[3]-complete problems, but also because many problems appear to be fixed-parameter tractable or even have a kernel of polynomial size, when the vertex cover number is used as the parameter (e.g., [28, 29, 31, 34]).

For a graph G and a graph class $\mathcal{C}$, a *modulator* to $\mathcal{C}$ in G is a set of vertices W such that $G - W \in \mathcal{C}$. For instance, vertex covers are modulators to the class of edgeless graphs. A *feedback vertex set* is another type of a modulator, now to graphs without cycles, i.e., to forests. The *feedback vertex number* of a graph G is the minimum size of a feedback vertex set in G. We prove that the parameterization by the feedback vertex number yields a much harder problem.

▶ **Theorem 2.** *BinCSP parameterized by the feedback vertex number of the Gaifman graph is* W[SAT]*-hard and in* W[P]*.*

Finally, with similar techniques, we obtain the following completeness results for W[t] for all odd $t \geqslant 3$. Here, *treedepth* is a structural parameter measuring the "depth" of a graph, we will expand on it later on.

▶ **Theorem 3.** *For each integer $d \geqslant 1$, BinCSP is complete for* W[$2d+1$] *when parameterized by the minimum size of a modulator to a graph of treedepth at most d, and when parameterized by the minimum size of a modulator to a forest of depth at most d.*

Interestingly, each increase of the depth of the trees by one corresponds to an increase in the W-hierarchy by two levels: this is because one level of depth in the tree or forest corresponds to a conjunction (looking at all children of a node) with a disjunction (the choice of a value). Theorem 3 can be seen as an interpolation between Theorems 1 and 2: by allowing the forest to have larger and larger depth, we obtain harder and harder parameterized problems. This yields a family of natural complete problems for the odd levels of the W-hierarchy.

Theorem 1 is proved in Section 3, while Theorems 2 and 3 are proved in the full version [12].

Treedepth parameterization: class XSLP. As we argued, the classes XNLP and XALP can be seen as the "natural home" for BinCSP parameterized by pathwidth and treewidth respectively, and for many other problems on "path-like" or "tree-like" graphs. We introduce a new parameterized complexity class XSLP which is the "natural home" for the parameter treedepth instead, reflecting "shallow" graphs (this is what the letter S stands for).

The *treedepth* of a graph G is the minimum depth of a rooted forest F on the same vertex set as G such that every edge of G connects a vertex with its ancestor in F; thus, it is a measure of shallowness of a graph. While treedepth is never smaller than pathwidth, it can be arbitrarily large even on graphs of bounded pathwidth: a path on n vertices has pathwidth 1 and treedepth $\lceil \log_2(n+1) \rceil$. Despite being relatively lesser known than treewidth or pathwidth, treedepth appears naturally in many seemingly disconnected areas. For instance, it features a prominent role in the theory of Sparsity (see [43, Chapters 6 and 7] for an overview), has interesting combinatorics of its own (see e.g. [16, 19, 22, 39]), corresponds to important dividing lines in finite model theory (see e.g. [25, 40]), and governs the parameterized complexity of block-structured integer programming (see [24] for an overview). More importantly for us, a line of work [33, 37, 41, 42, 45, 46] uncovered that for many classic problems, on graphs of low treedepth one can design fixed-parameter algorithms that are both time- and space-efficient, which is conjectured not to be possible for the pathwidth or treewidth parameterizations [46]. This makes treedepth a prime candidate for a parameter that can be interesting from the point of view of BinCSP.

And so, we define two complexity classes: XSLP consists of all parameterized problems that can be reduced to BinCSP parameterized by treedepth in parameterized logspace (that is, in deterministic space $f(k) + \mathcal{O}(\log n)$ for a computable f), while XSLP$^+$ has the same definition, except we consider fpt reductions. This distinction is of technical nature: on one

hand we use parameterized logspace reductions to match the definitions of XALP and XNLP and retain the inclusion XSLP $\subseteq$ XNLP $\subseteq$ XALP, and on the other hand we would like to compare XSLP with the W-hierarchy, which requires closure under fpt reductions. In fact, XSLP$^+ \supseteq$ W[t] for every integer t (this will follows from Proposition 4).

We prove two alternative characterizations of XSLP. The first one is through a machine model: we prove that XSLP can be equivalently defined as problems that can be solved by an alternating Turing machine with the following resource bounds: (1) $f(k)\log n$ bits of nondeterminism, (2) $f(k) + \mathcal{O}(\log n)$ bits of conondeterminism, (3) alternation at most $f(k)$, and (4) working space $f(k) + \mathcal{O}(\log n)$ plus a read-once stack of size $f(k)\log n$ that can be only pushed upon and read only at the end of the computation. See Theorem 10 in Section 4.1 for a formal statement. This reflects the characterization of XALP through alternating Turing machines with different bounds on conondeterminism and the size of a computation tree, see [10, Theorem 1].

The main step in the proof of our machine characterization of XSLP is a regularization lemma for the considered machine model, allowing us to assume that the computation tree has always a very concrete shape. Interestingly, this step crucially uses the existence of fpt-sized *universal trees*, a tool fundamentally underlying the recent advances in the complexity of parity games. While universal trees can be seen only implicitly in the breakthrough work of Calude et al. [13], their central role in the approach was exposed in subsequent works [18, 38].

The second characterization is through model-checking first-order logic, and is inspired by the definition of the A-hierarchy; see [30, Chapter 8]. In essence, we provide a complete problem for XSLP, which amounts to model-checking first-order sentences in which universal quantification must follow a root-to-leaf path in a rooted forest present in the structure. Details and formal statements can be found in Section 4.2.

***d*-fold vertex cover and the S-hierarchy.** We "project" the class XSLP closer to lower levels of the W-hierarchy, thus obtaining a new hierarchy of parameterized classes sandwiched between the W-hierarchy and the A-hierarchy. For this, we introduce the following parameter.

The 1-*fold vertex cover number* of a graph G is simply the number of vertices of G. Inductively, for $d \geqslant 2$, the *d-fold vertex cover number* is the smallest integer k with the following property: there is a subset of vertices $U \subseteq V(G)$ with $|U| \leqslant k$ such that every connected component of $G - U$ has $(d-1)$-fold vertex cover number at most k. Alternatively, we can also define the parameter using a "fattened" variant of elimination trees (the decomposition notion underlying treedepth). Namely, G has d-fold vertex cover number at most k if and only if there is a rooted tree T of depth at most d, and a vertex partition $\{V_t \colon t \in V(T)\}$ of $V(G)$ such that $|V_t| \leqslant k$ for all $t \in V(T)$, and edges in G between vertices of V_s and V_t are only allowed when s and t are equal or are in an ancestor-descendant relationship in T.

We now define the parameterized complexity class[1] S[$2d-1$] as the fpt-closure of BinCSP parameterized by the d-fold vertex cover number, for all integers $d \geqslant 1$. The following result relates the introduced classes to the W-hierarchy, the A-hierarchy, and the class XSLP$^+$.

▶ **Proposition 4.** *For every integer $d \geqslant 1$, we have* W[$2d-1$] $\subseteq$ S[$2d-1$] $\subseteq$ A[$2d-1$] *and* S[$2d-1$] $\subseteq$ XSLP$^+$.

The proof is straightforward and is given in the full version [12].

[1] We remark that there is an already existing concept called the S-hierarchy, related to subexponential parameterized algorithms; see [30, Definition 16.9]. Since we are not aware of any subsequent work on the structure of this hierarchy, we took the liberty of using the same naming scheme for our classes.

While the definition of d-fold vertex cover seems not to have been discussed explicitly in the literature, the idea of alternating deletions of batches of vertices and splitting into connected components is not entirely new, as similar parameters that interpolate between vertex cover and treedepth have previously been studied. For example, 2-fold vertex cover is within a multiplicative factor of two of *vertex integrity*, a parameter that was introduced by Barefoot, Entringer and Swart [4] in 1987 (see [2] for a survey). In the context of block-structure integer programs, the *fracture number* [23] can be seen as an analogue of 2-fold vertex cover, while the concept of *topological height* [24] serves a role similar to that of d in the definition of d-fold vertex cover.

Comparison to List Coloring. The classic List Coloring problem can be interpreted as the special case of BinCSP where every constraint just stipulates that the values assigned to adjacent variables are different from each other. Therefore, a hardness result for List Coloring implies one for BinCSP. Vice versa, we can attempt to turn an instance of BinCSP on graph G into an instance of List Coloring by adding, for each edge uv in G and each forbidden pair of values (a, b), a vertex to G adjacent to u and v with color list $\{a, b\}$. This transformation does not significantly affect graph parameters such as treedepth, treewidth or pathwidth, so hardness and completeness results of BinCSP may also be inherited to List Coloring. However, the transformation may make dramatic changes to other parameters such as vertex cover and vertex modulator to a graph of treedepth at most d, where we can only easily deduce W$[2d-1]$-hardness from our W$[2d+1]$-hardness results. In fact, we separate the two problems with the following result, proved in the full version [12].

▶ **Theorem 5.** *List Coloring is in* W[2] *when parameterized by the vertex cover number and in* W[2d] *when parameterized by the size of a modulator to a treedepth-d graph.*

We believe that due to its robustness, BinCSP better suited to measure the complexity of parameters than List Coloring is. This is also witnessed by the (nearly) tight completeness results presented in Theorems 1, 2, and 3. Table 1 below presents a comparison of the parameterized complexity landscapes of BinCSP and of List Coloring under various structural parameterizations. We discuss this table in the full version [12].

2 Preliminaries

For integers $a \leqslant b$, we write $[a, b]$ for $\{a, a+1, \ldots, b\}$.

Graphs and their parameters. In this paper, we denote the *depth* of a rooted tree as the maximum number of vertices on a path from root to leaf. A *rooted forest* is a collection of rooted trees. The *depth* of a rooted forest is the maximum depth of the trees in the forest.[2]

We use standard graph notation. An *elimination forest* of a graph G, is a rooted forest F with the same vertex set as G, such that for each edge uv of G, u is an ancestor of v or v is an ancestor of u in F. (Note that the forest can contain edges that are not in G.) The *treedepth* of a graph G is the minimum depth of of rooted forest embedding of G.

Let $\mathcal{C}$ be a class of graphs. A *modulator* to $\mathcal{C}$ in a graph G is a set of vertices $W \subseteq V(G)$, such that the graph $G - W$ belongs to $\mathcal{C}$. A *vertex cover* of a graph G is a set of vertices $W \subseteq V(G)$, such that every edge of G has at least one endpoint in W. Note that a set of

[2] The definitions of depth of a tree used in the literature can differ by one. Here we count the number of vertices, e.g., a tree consisting of a single vertex has depth 1.

Table 1 Complexity of BinCSP and List Coloring. Results marked with $*$ are shown in this paper. Some results without a reference are easy to obtain.

Parameter	Binary CSP	List Coloring
number of vertices	W[1]-complete [27, 44]	poly-kernel
vertex cover	W[3]-complete $*$	W[1]-hard [29], in W[2] $*$
feedback vertex set	W[SAT]-hard, in W[P] $*$	W[3]-hard, in W[P] $*$
modulator to treedepth-d	W[$2d+1$]-complete $*$	W[$2d-1$]-hard, in W[$2d$] $*$
modulator to depth d-forest	W[$2d+1$]-complete $*$	W[$2d-1$]-hard, in W[$2d$] $*$
modulator to clique	para-NP-complete	FPT, poly-kernel [3, 35]
treedepth	XSLP-complete $*$	XSLP-complete $*$
tree partition width	XALP-complete [10]	W[1]-hard, in XL [8]
tree partition width + degree	XALP-complete [10]	FPT
pathwidth	XNLP-complete [11]	XNLP-complete [11]
bandwidth	XNLP-complete [11]	FPT
treewidth	XALP-complete [10]	XALP-complete [10]
treewidth + degree	XALP-complete [10]	FPT

vertices is a vertex cover if and only if it is a modulator to the class of edgeless graphs, or, equivalently, to the class of graphs with treedepth at most 1. A *feedback vertex set* in a graph G is a modulator to a forest, or, equivalently, a set of vertices that intersects each cycle in G.

Constraint satisfaction problems. We consider the BinCSP problem defined as follows. An instance of BinCSP is a triple

$$I = (G, \{D(u)\colon u \in V(G)\}, \{C(u,v)\colon uv \in E(G)\}),$$

where

- G is an undirected graph, called the *Gaifman graph* of the instance;
- for each $u \in V(G)$, $D(u)$ is a finite set called the *domain* of u; and
- for each $uv \in E(G)$, $C(u,v) \subseteq D(u) \times D(v)$ is a binary relation called the *constraint* at uv. Note that $C(u,v)$ is not necessarily symmetric; throughout this paper, we apply the convention that $C(v,u) = \{(b,a) \mid (a,b) \in C(u,v)\}$.

In the context of a BinCSP instance, we may sometimes call vertices *variables*. A *satisfying assignment* for an instance I is a function η that maps every variable u to a value $\eta(u) \in D(u)$ such that for every edge uv of G, we have $(\eta(u), \eta(v)) \in C(u,v)$. The BinCSP problem asks, for a given instance I, whether I is *satisfiable*, that is, there is a satisfying assignment for I.

The List Coloring problem is a special case of BinCSP defined as follows. An instance consists of a graph G and, for every vertex u of G, a set (*list*) of colors $L(u)$. The question is whether there is a mapping f of vertices to colors such that for every vertex u we have $f(u) \in L(u)$, and for each edge uv of G, we have $f(u) \neq f(v)$. Note that this is equivalent to a BinCSP instance where lists $L(u)$ are the domains, and all constraints are non-equalities: $C(u,v) = \{(a,b) \in L(u) \times L(v) \mid a \neq b\}$ for every edge uv.

Complexity theory. We assume the reader to be familiar with standard notions of the parameterized complexity theory, such as the W-hierarchy or parameterized reductions. For more background, see [17, 20, 21, 30]. Let us recall concepts directly used in this paper.

We say that a parameterized problem Q is in *parameterized logspace* if Q can be decided in (deterministic) space $f(k) + \mathcal{O}(\log n)$, for some computable function f. Note that every problem in parameterized logspace is fixed-parameter tractable, because a Turing machine working in space $f(k) + \mathcal{O}(\log n)$ has $2^{\mathcal{O}(f(k))} \cdot n^{\mathcal{O}(1)}$ configurations, and hence its acceptance can be decided in fixed-parameter time.

An *fpt-reduction* is a parameterized reduction that works in fixed-parameter time. A *pl-reduction* is a parameterized reduction that works in parameterized logspace, that is, can be computed in (deterministic) space $f(k) + \mathcal{O}(\log n)$, for some computable function f.

A Boolean formula is said to be *t-normalized* when it is the conjunction of disjunctions of conjunctions of ... of literals, with t levels of conjunctions or disjunctions. We only consider the case where $t \geqslant 2$, and assume that we start by conjunctions. Note that 2-normalized Boolean formulas are in Conjunctive Normal Form.

In the Weighted t-Normalized Satisfiability problem, we are given a t-normalized Boolean formula F on n variables, and an integer k, and ask if we can satisfy F by setting exactly k of the variables to true, and all other variables to false. This problem is complete for W[t], see e.g. [20, 21]. A t-normalized expression is said to be *anti-monotone* if each literal is the negation of a variable. We use the following result to simplify our proofs.

▶ **Theorem 6** (Downey and Fellows, see [20, 21]). *For every odd $t \geqslant 3$, Weighted Anti-Monotone t-Normalized Satisfiability is complete for W[t].*

We use the following result as starting point for membership proofs.

▶ **Theorem 7** (Downey and Fellows, see [20, 21]). *For every $t \geqslant 2$, Weighted t-Normalized Satisfiability is complete for* W[t].

3 W[3]-completeness for BinCSP parameterized by vertex cover

In this section, we prove Theorem 1. We prove the hardness below and refer to the full version [12] for the proof of membership.

▶ **Lemma 8.** *BinCSP with vertex cover as parameter is* W[3]*-hard.*

Proof. Take an instance of Weighted 3-Normalized Anti-Monotone Satisfiability, i.e., we have a Boolean formula F that is a conjunction of disjunctions of conjunctions of negative literals, and ask if we can satisfy it by setting exactly k variables to true. Suppose $x_1, \ldots, x_n$ are the variables used by F. Suppose F is the conjunction of r disjunctions of conjunctions of negative literals.

We build a graph G as follows. The vertex set $V(G)$ consists of a set $W = \{w_1, \ldots, w_k\}$ of size k, and a set $S = \{v_1, v_2, \ldots, v_r\}$ of size r. The set W will be the vertex cover of G, and S will form an independent set. We add edges from each vertex in W to each other vertex in the graph.

The domain of a vertex $w \in W$ is $D(w) = \{x_1, \ldots, x_n\}$. For distinct $w, w' \in W$, $w' \neq w'$, we set $C(w, w') = \{(x_i, x_j) \mid i \neq j\}$. This enforces that all vertices in W are assigned a different value – this corresponds to setting exactly k variables to true.

Now consider a vertex $v_i \in S$ for $i \in [1, r]$. We say that v_i represents the ith disjunction of conjunctions of literals in F, i.e., each of the disjunctions in the formula is represented by one vertex in the independent set. Suppose that this disjunction has t_i terms (each term is a conjunction of negative literals). We set $D(v_i) = [1, t_i]$, that is, each value for v_i is an integer in $[1, t_i]$.

The intuition is as follows. We set a variable x_i to true, if and only if exactly one vertex in W is assigned x_i. As all vertices in W will get a different value, we set in this way exactly k variables to true. The formula F is the conjunction of r disjunctions; each of these disjunctions is represented by one of the vertices $v_i \in S$. For each v_i, the disjunction represented by v_i must be satisfied, so one of its terms must be satisfied. The value of v_i tells a satisfied term, i.e., if the value of v_i is $j \in [1, t_i]$, then the jth term is satisfied. This is checked by looking at the edges from v_i to the vertices in W.

We now give the constraints that ensure the term is satisfied. Consider a vertex $v_i \in S$ and $w \in W$. Recall that the value of v_i is an integer in $[1, t_i]$ which represents one term in the ith disjunction of F, and that term is a conjunction of a number of negative literals. For $j \in [1, t_i]$ and $j' \in [1, n]$, we have $(j, x_{j'}) \in C(v_i, w)$ if and only if for each literal $\neg x_{j''}$ that appears in the jth term of the ith disjunction of F, $j'' \neq j'$.

We call the constructed graph G and write I for the corresponding instance of BinCSP.

▷ Claim 9. F can be satisfied by setting exactly k variables to true, if and only if I has a satisfying assignment.

Proof of Claim 9. Suppose F can be satisfied by making $x_{i_1}, \ldots, x_{i_k}$ true, and all other literals false. Then assign the vertices in W the values $x_{i_1}, \ldots, x_{i_k}$ successively. The constraints between vertices in W are thus satisfied.

Now consider a vertex $v_i \in S$. Consider the ith term F_i of the (upper level) conjunction of F. This term must be satisfied by the truth assignment. Suppose the term is $F_i = F_{i,1} \vee \cdots \vee F_{i,t_i}$. At least one of the F_{i_j}'s must be satisfied by the truth assignment, say $F_{i,j'}$. Then assign v_i the value j'.

We can verify that the constraints for edges between v_i and each w_j are fulfilled. By assumption, $F_{i,j'}$ holds. It thus cannot contain a negative literal $\neg x_\alpha$, where x_α is set to true. So w_j cannot be assigned x_α when $\neg x_\alpha$ is a literal in $F_{i,j'}$. Thus we found a satisfying assignment for I.

Now, suppose that I has a satisfying assignment. From the constraints between vertices in W, we see that all vertices in W have a different value. Set a variable x_i to true, if and only if a vertex in W has value x_i, and otherwise, set it to false. We have thus set exactly k variables to true.

Consider the ith term of the upper level conjunction of F. Suppose this term is $F_{i,1} \vee \ldots \vee F_{i,t_i}$. Suppose v_i is assigned value j. For each negative literal $\neg x_\alpha$ in the conjunction $F_{i,j}$, by the constraints, we cannot have a vertex in W that is assigned x_α, and thus x_α is set to false. Thus, the term $F_{i,j}$ is satisfied by the truth assignment, and thus F_i is satisfied. As this holds for all conjuncts of F, F is satisfied by the specified assignment. ◁

From Claim 9, we see that we have a parameterized reduction from Weighted Anti-Monotone 3-Normalized Satisfiability to BinCSP with vertex cover as parameter. The result now follows from the W[3]-hardness of Weighted Anti-Monotone 3-Normalized Satisfiability (Theorem 6). ◀

4 XSLP and treedepth

In this section we discuss the class XSLP and its various characterizations. As discussed in Section 1, we actually define two variants of this class, depending on the kind of reductions that we would like to speak about. Let BinCSP/td denote the following parameterized problem. We are given a BinCSP instance I and an elimination forest of the Gaifman graph of I

of depth at most k, which is the parameter. The task is to decide whether I is satisfiable. Then the two variants of XSLP are defined as the closures of this problem under pl- and fpt-reductions, respectively:

$$\text{XSLP} = [\text{BinCSP}/_{\text{td}}]^{\text{pl}} \qquad \text{and} \qquad \text{XSLP}^+ = [\text{BinCSP}/_{\text{td}}]^{\text{fpt}}.$$

That is, XSLP consists of all parameterized problems that are pl-reducible to $\text{BinCSP}/_{\text{td}}$, and XSLP^+ is defined similarly, but with fpt-reductions in mind.

Note that in the $\text{BinCSP}/_{\text{td}}$ problem we assume that a suitable elimination forest is provided on input. This is to abstract away the need of computing such an elimination forest; the complexity of this task is also an interesting question, but lies beyond the scope of this work.

4.1 A machine characterization

We first give a machine characterization of XSLP. We will use a model of an *alternating read-once stack machine*, or *AROSM* for brevity, which we now define. We assume familiarity with standard Turing machines, on which we build our model.

An alternating read-once stack machine M is a Turing machine that has access to three types of memory, each using $\{0, 1\}$ as the alphabet:

- a read-only input tape;
- a working tape; and
- a read-once stack.

The input tape and the working tape are accessed and manipulated as usual, by a head that may move, read, and (in the case of the working tape) write on the tape. The input to the machine is provided on the input tape. On the other hand, the stack is initially empty and the machine may, upon any transition, push a single symbol onto the stack. It cannot, however, read the stack until the final step of the computation. More precisely, the acceptance condition is as follows: The machine has a specified *final state*. Once it is reached, the computation finishes and the machine reads the ith bit of the stack, where i is the number whose binary encoding is the current content of the working tape. If this bit is 1, then M accepts, and otherwise it rejects.

A *configuration* of M is a 5-tuple consisting of the state, the content of the working tape, the content of the stack, and the positions of the heads on the input and the working tape.

Further, M is an *alternating machine*, which means that its states are partitioned into three types: *existential states*, *universal states*, and *deterministic states*. A configuration of a machine is existential/universal/deterministic if its state is so. When the state of the machine is deterministic, there is exactly one transition allowed. At existential and universal states, there are always two transitions allowed; these will be named the 0-*transition* and the 1-*transition*. The acceptance is defined as usual in alternating machines: when in an existential state, M may accept if at least one allowed transition leads to a configuration from which it may accept, and in a universal state we require that both transitions lead to configurations from which M may accept. The notion of a machine deciding a (parameterized) problem is as usual.

The $\forall$ *computation tree* of M for input x is defined as a tree of configurations with the following properties:

- the root is the initial configuration with input x;
- the leaves are configurations with the final state;
- every deterministic and every existential configuration has exactly one child, which is the unique, respectively any of the two configurations to which the machine may transit;

- every universal configuration has exactly two children, corresponding to the two configurations to which the machine may transit.

It follows that M accepts input x if there is a $\forall$ computation tree for input x where every leaf is a configuration in which M accepts. We call such $\forall$ computation trees *accepting*.

A *branch* of a (rooted) tree is a root-to-leaf path. For a $\forall$ computation tree T of machine M, we define the following quantities:

- The *working space* of T is the minimum number i such among configurations present in T, the head on the working tape is never beyond the ith cell.
- The *stack size* of T is the maximum size of the stack among all configurations in T.
- The *nondeterminism* of T is the maximum number of existential configurations on any branch of T.
- The *conondeterminism* of T is the maximum number of universal configurations on any branch of T.
- The *alternation* of a branch of T is the minimum number of blocks into which the branch can be partitioned so that each of the blocks does not simultaneously contain an existential and a universal configuration. The *alternation* of T is the maximum alternation on any branch of T.

We say that a machine M *decides* a parameterized problem Q using certain resources among those described above, if for any input (x, k), we have $(x, k) \in Q$ if and only if there is an accepting $\forall$ computation tree for (x, k) that has the resources bounded as prescribed.

Having all the necessary definitions in place, we can state the main result of this section.

▶ **Theorem 10.** *The following conditions are equivalent for a parameterized problem Q.*

(1) $Q \in \mathrm{XSLP}$*;*

(2) *Q can be decided by an alternating read-once stack machine that for input (x, k) with $|x| = n$, uses working space at most $f(k)+\mathcal{O}(\log n)$, stack size $f(k)\log n$, nondeterminism $f(k)\log n$, co-nondeterminism $f(k)+\mathcal{O}(\log n)$, and alternation $f(k)$, for some computable function f.*

Before we proceed to the proof of Theorem 10, let us discuss the necessity of different resource restrictions described in (2):

- Increasing the working space to $f(k)\log n$ (and thus rendering the stack, the nondeterminism and the co-nondeterminism unnecessary) would make the machine model at least as powerful (and in fact, equivalently powerful) as deterministic Turing machines with $f(k)\log n$ space; this corresponds to a class called XL. As XL^+ (the closure of XL under fpt reductions) contains AW[SAT] [30, Exercise 8.39], the supposition that the amended model is still equivalent to XSLP would imply the inclusion $\mathrm{AW}[\star] \subseteq \mathrm{AW}[\mathrm{SAT}] \subseteq \mathrm{XSLP}^+$. From the logic characterization that will be provided in Section 4.2 it follows that $\mathrm{AW}[\star] \supseteq \mathrm{XSLP}^+$, so in fact we would obtain a collapse $\mathrm{AW}[\star] = \mathrm{AW}[\mathrm{SAT}] = \mathrm{XSLP}^+$.
- If we increase the bound on allowed co-nondeterminism to $f(k)\log n$, thus matching the bound on the allowed nondeterminism, then it is not hard to see that the obtained machine model would be able to solve the model-checking problem for first-order logic on general relational structures, which is AW[$\star$]-complete. Consequently, if the amended machine model was still equivalent to XSLP, we would again obtain equality $\mathrm{AW}[\star] = \mathrm{XSLP}^+$, which we consider unlikely.
- If we let the machine use unbounded nondeterminism, then already for k constant and assuming no use of co-nondeterminism, our machines would be able to solve every problem in NL, including Directed Reachability. If the obtained machine model was still equivalent to XSLP, then Directed Reachability would be reducible (in L) to BinCSP on graphs of constant treedepth. But the latter problem is actually in L, so we would obtain L = NL.

- We believe that increasing the alternation from $f(k)$ to $f(k) + \mathcal{O}(\log n)$ yields a strictly more powerful machine model, though at this point we cannot pinpoint any concrete collapse that would be implied by the converse. However, it is not hard to check that an AROSM with resource bounds as in Theorem 10, but alternation $f(k) + \mathcal{O}(\log n)$, is able to solve BinCSP instances with Gaifman graphs of treedepth as large as $\log n$, but with all domains of size at most k. We do not see how to reduce this problem to BinCSP with domains of unbounded size, but treedepth bounded by $f(k)$.
- It is an interesting question whether the $f(k) \log n$ bound on the stack size can be lifted; that is, whether allowing unbounded stack size strictly increases the power of the considered machine model. On one hand, in all our proofs, the stack is essentially only used to store nondeterministic bits, and in any run there are at most $f(k) \log n$ of them anyway. So if the stack is used only for this purpose, then it is immaterial whether its size is bounded by $f(k) \log n$ or unbounded. On the other hand, the restriction on the stack size plays an important role in the proof of the implication (2) $\Rightarrow$ (1) of Theorem 10. We leave resolving this question open.

The remainder of this section is devoted to the proof of Theorem 10. Naturally, the argument is split into the forward and the backward implication.

We refer to the full version [12] for the proof of the simpler implication (1) $\Rightarrow$ (2), but briefly sketch it here. We use an AROSM to guess a satisfying assignment to the given ${}^{\text{BinCSP}}/_{\text{td}}$ instance, by going top-down through the associated forest. We use nondeterminism to guess the assignment for the next vertex u, and conondeterminism to verify whether the currently guessed partial assignment can be extended to all the subtrees rooted at the children of u.

We now proceed to the more difficult implication (2) $\Rightarrow$ (1) of Theorem 10. The main idea is that we introduce a restricted variant of a *regular* AROSM, which is an AROSM whose $\forall$ computation tree has a very specific shape, computable from k and the length of the input. We will then show two lemmas: (i) for every AROSM there is an equivalent regular one, and (ii) acceptance of a regular AROSM can be reduced to ${}^{\text{BinCSP}}/_{\text{td}}$. The main point in this strategy is that the assumption that the computation tree is fixed allows us to fix it as the elimination tree of the Gaifman graph of the constructed BinCSP instance.

More precisely, we will be working with the *contracted* $\forall$ *computation trees* defined as follows. Let T be a $\forall$ computation tree of an AROSM M, where without loss of generality we assume that the starting state of M is universal. A *universal block* of T is an inclusion-wise maximal subtree A of T such that the root of A is a universal configuration and A does not contain existential configurations. Note that removing all universal blocks from T breaks T into a collection of disjoint paths consisting only of deterministic and existential configurations; these will be called *existential blocks*. The *contraction* of T is the tree T' whose nodes are universal blocks of T, where the ancestor order is naturally inherited from T: one block is an ancestor of the other in T' if this holds for their roots in T. Note that a universal block B is a child of a universal block A in T' if and only if there is an existential block C that connects the root of B with a leaf of A. Thus, the edges of T' are in one-to-one correspondence with the existential blocks of T.

▶ **Definition 11.** *An AROSM M is* regular *if given $(1^n, k)$ one can in parameterized logspace compute a rooted tree $T_{n,k}$ with the following properties:*

- *$T_{n,k}$ has depth at most $f(k)$, for some computable function f; and*
- *for any input (x, k) with $|x| = n$, if M accepts (x, k), then M has a $\forall$ computation tree accepting (x, k) whose contraction is $T_{n,k}$.*

With this definition in place, we can state the two lemmas described before.

▶ **Lemma 12.** *If a parameterized problem Q can be decided by an AROSM M using the resource bounds stated in Theorem 10, then it can also be decided by a regular AROSM M' using such resource bounds.*

▶ **Lemma 13.** *If Q can be decided by a regular AROSM M using the resource bounds stated in Theorem 10, then $Q \in$ XSLP.*

The (2) ⇒ (1) implication of Theorem 10 follows directly by combining the two lemmas above. The proof of Lemma 13 is a conceptually straightforward, though technically a bit involved encoding of a ∀ computation tree of the machine through an instance of BinCSP whose elimination tree is (roughly) $T_{n,k}$. We give this proof in the full version [12]. The proof of Lemma 12 is the interesting part of the argument, as it involves the notion of *universal trees*.

Before we proceed, let us state a simple lemma that is used in our proofs several times. We included a proof in the full version [12] for completeness.

▶ **Lemma 14.** *Suppose T is a rooted tree with N leaves. Then there exists a labelling λ that maps every edge e of T to a binary string $\lambda(e) \in \{0,1\}^\star$ with the following properties:*
- *For every node u, the labels of edges connecting u with its children are pairwise different.*
- *For every leaf ℓ, the total length of labels on the root-to-ℓ path in T is at most $\lceil \log N \rceil$.*

Moreover, given T the labelling λ can be computed in deterministic logarithmic space.

4.1.1 Regularization

We now prove Lemma 12. We need the following definitions. An *ordered tree* is a rooted tree where for every vertex u there is a linear order $\preceq$ on the children of u. An *embedding* of an ordered tree S into an ordered tree T is an injective mapping $\phi\colon V(S) \to V(T)$ such that
- the root of S is mapped to the root of T, and
- for every node u of S, the children of u in S are mapped to distinct children of $\phi(u)$ in T in an order-preserving way: if $v \prec v'$ are distinct children of u in S, then $\phi(v) \prec \phi(v')$.

We will use the following result about the existence of *universal trees*.

▶ **Lemma 15** (follows from Jurdziński and Lazić [38], see also Theorem 2.2 of [18])**.** *For every pair of integers $n, k \in \mathbb{N}$ there exists an ordered tree $U_{n,k}$ such that*
- *$U_{n,k}$ has depth k;*
- *$U_{n,k}$ has at most $2n \cdot \binom{\lceil \log n \rceil + k + 1}{k}$ leaves; and*
- *for every ordered tree T of depth at most k and with at most n leaves, there is an embedding of T into $U_{n,k}$.*

Moreover, given $(1^n, k)$, the tree $U_{n,k}$ can be computed parameterized logspace.

We remark that the claim about the computability of $U_{n,k}$ in parameterized logspace is not present in [18, 38], but follows directly from the recursive construction presented there. In fact, we will also need the following property, which again follows directly from the construction, and which strengthens the embedding property stated in Lemma 15.

▶ **Lemma 16.** *For every node u of $U_{n,k}$, the subtree of $U_{n,k}$ rooted at u is isomorphic to $U_{n',k'}$ for some $n' \leqslant n$ and $k' \leqslant k$; the labeling of nodes of $U_{n,k}$ with suitable numbers n', k' can be computed along with $U_{n,k}$ within the algorithm of Lemma 15. Moreover, if $n_1, \ldots, n_p$ are nonnegative integers such that $n_1 + \ldots + n_p \leqslant n$, then there are distinct children $v_1 \prec v_2 \prec \ldots \prec v_p$ of the root of $U_{n,k}$ such that for every $i \in \{1, \ldots, p\}$, the subtree of $U_{n,k}$ rooted at v_i is isomorphic to $U_{n'_i,k-1}$ for some $n'_i \geqslant n_i$.*

Finally, observe that

$$2n \cdot \binom{\lceil \log n \rceil + k + 1}{k} \leqslant 2n \cdot 2^{\lceil \log n \rceil + k + 1} \leqslant \mathcal{O}(2^k \cdot n^2),$$

hence $U_{n,k}$ has $\mathcal{O}(2^k \cdot n^2)$ leaves.

We proceed to the proof of Lemma 12. Let us fix an AROSM M that on any input (x, k) with $|x| \leqslant n$, uses $f(k) \log n$ nondeterminism, $f(k) + d \log n$ conondeterminism, $f(k)$ alternation, $f(k) + d \log n$ working space, and $f(k) \log n$ stack size, where f is a computable function and $d \in \mathbb{N}$ is a constant. We may assume w.l.o.g. that the starting state of M is universal. Denote $K = f(k)$ and $N = 2^{f(k) + \lceil d \log n \rceil} \leqslant 2^{f(k)+1} \cdot n^d$; then K is an upper bound on the depth and N is an upper bound on the total number of leaves of any $\forall$ computation tree accepting (x, k) within the stipulated resources. By Lemma 15, we may compute the universal tree $U_{N,K}$ in deterministic space $h(k) + \mathcal{O}(\log n)$ for a computable function h. Note that $U_{N,K}$ has $N' = \mathcal{O}(2^K \cdot N^2) \leqslant \mathcal{O}(2^{3f(k)} \cdot n^{2d})$ leaves. The tree $U_{N,K}$ will serve the role of $T_{n,k}$ in the proof. Also, we use Lemma 14 to compute a suitable labeling λ of the edges of $U_{N,K}$ in which the total length of labels on every branch of $U_{N,K}$ is at most $\lceil \log N' \rceil \leqslant 3f(k) + 2d \log n + \mathcal{O}(1)$.

We are left with designing an AROSM M' that is equivalent to M, in the sense that M' accepts input (x, k) if and only if M does, and in such case the contracted $\forall$ computation tree of M' on (x, k) may be $U_{N,K}$. The idea is that machine M' will simulate M while inserting some dummy computation to "fill" the contracted $\forall$ computation tree of M to $U_{N,K}$. However, we will need to be very careful about how the conondeterminism of M is simulated.

A *stackless configuration* is a configuration of M, but without specifying the content of the stack; that is, it consists of the state of M, the content of the working tape, and the positions of the heads on the input and the working tapes. For a universal stackless configuration c of M, we define the *universal block* rooted at c, denoted $U(c)$, as a rooted tree of stackless configurations that is obtained just as the $\forall$ computation tree, except that M starts at c and we do not continue the simulation once the final state or any existential configuration is reached. Note here since M cannot read the stack except for the end of the computation, $U(c)$ is uniquely defined for every stackless configuration c. Thus, the leaves of $U(c)$ are existential or final (stackless) configurations, and whenever c is present in a $\forall$ computation tree T of M, T contains a copy of $U(c)$ rooted at c as a subtree.

The next claim shows that given a stackless configuration c, the universal block $U(c)$ can be computed within the allowed resources.

▷ Claim 17. Given a stackless configuration c of M, the universal block $U(c)$, together with a labelling of its edges with transitions taken, can be computed in deterministic space $h(k) + \mathcal{O}(\log n)$, for some computable h.

Proof. Let $Z = f(k) + \lfloor d \log n \rfloor$. Observe that for every binary string $r \in \{0, 1\}^Z$, we can compute the branch of $U(c)$ that takes the consecutive conondeterministic choices as prescribed by the consecutive bits of r. To do this, just simulate M starting from c and, whenever a conondeterministic choice needs to be made, use the next bit of r to determine how it is resolved. (This simulation stops when an existential or a final configuration is encountered.) Having this subprocedure, the whole $U(c)$ can be easily computed by iterating through consecutive strings $r \in \{0, 1\}^Z$ and outputting the branches of $U(c)$ one after the other. (Strictly speaking, from every next branch we output only the part after diverging from the previous branch.) Finally, note that r can be stored within the allowed space. ◁

With Claim 17 established, we proceed to the construction of M'. For the sake of the proof, suppose M has a $\forall$ computation tree T that is accepting and uses the allowed resources. Machine M' tries to verifies the existence of such T by traversing the universal tree $U_{N,K}$ and guessing, along the way, how the contraction T' of T embeds into $U_{N,K}$. By Lemma 15 we know that such an embedding always exists. The traversal of $U_{N,K}$ will be done in such a way that the contracted $\forall$ computation tree of M' will be always $U_{N,K}$.

At every point of computation, M' stores on its working tape a node u of $U_{N,K}$ and its contracted $\forall$ computation tree from this point on should be the subtree of $U_{N,K}$ rooted at u. Machine M' is always either in the *real mode* or in the *dummy mode*. In the real mode, M' is in the process of guessing a subtree of T. Therefore, then M' holds the following data:

- On the working tape, M' stores a stackless configuration c of M. The reader should think of c as of the configuration of M at the root of a universal block of T.
- On its own stack, M' holds the content of the stack of M.
- Additionally on the working tape, M' stores two integers a and b, denoting the total number of nondeterministic and conondeterministic bits used by M so far, respectively. (In other words, a and b are the total number of existential and universal configurations visited so far on a branch of T.) We maintain the following invariant: the subtree of $U_{N,K}$ rooted at u is $U_{N',K'}$ for some $K' \leqslant K$ and N' such that $N' \geqslant N/2^b$.

Then the task of M' is to verify the existence of a subtree S of a $\forall$ computation tree of M such that

- S has c supplied with the current content of the stack at its root;
- S embeds into the subtree of $U_{N,K}$ rooted at u;
- the nondeterminism and the conondeterminism of S together with a and b add to at most $f(k)\log n$ and $f(k) + d\log n$, respectively; and
- S is accepting, that is, every leaf of S is an accepting configuration.

In the dummy mode, M' is not guessing any part of T, so its task is to perform some meaningless computation in order to make its contracted $\forall$ computation tree equal to the subtree of $U_{N,K}$ rooted at u. So in this mode, M' holds on its working tape only the node u.

We now explain steps taken by M' in the real mode. Given c, M' applies the algorithm of Claim 17 to compute the universal block $U(c)$. (Formally speaking, $U(c)$ is not computed explicitly, as it would not fit within the working space, but at any point a bit from the description of $U(c)$ is needed, we run the algorithm of Claim 17 to compute this bit.) Let $\ell_1, \ldots, \ell_p$ be the leaves of $U(c)$, in the order as they appear in the description of $U(c)$. Informally, we wish to fit in $U(c)$ into the computation tree of M' while keeping enough "space" for the remaining computations M may wish to perform, without knowing how the computation will continue at the leaves. For every $i \in \{1, \ldots, p\}$, let b_i be total number of universal configurations on the branch of $U(c)$ finishing at ℓ_i. By assumption, the subtree of $U_{N,K}$ rooted at u is isomorphic to $U_{N',K'}$ for some $N' \geqslant N/2^b$ and $K' \leqslant K$. Similarly, we would like to find children $v_1 \prec v_2 \prec \ldots \prec v_p$ of u in $U_{N,K}$ such that the subtree rooted at each v_i is isomorphic to $U_{N_i,K'-1}$ where $N_i \geqslant N/2^{b+b_i}$. This follows from Lemma 16: we check that

$$\sum_{i=1}^{p} N/2^{b+b_i} \leqslant N' \sum_{i=1}^{p} 2^{-b_i} = N',$$

where the last equality follows since $U(c)$ is a binary tree. Note that given $b_1, \ldots, b_p$, we may compute the corresponding children $v_1, \ldots, v_p$ with sufficiently large subtrees in logarithmic space greedily: having found v_i, we can set v_{i+1} to be the $\prec$-smallest child v of u such that $v_i \prec v$ and the subtree rooted at v is isomorphic to $U_{N'',K'-1}$ for some $N'' \geqslant m_i$. Hence,

from now on we assume that the children $v_1, \ldots, v_p$ are given to us. (Again, formally, when we need any v_i, we run the logarithmic space algorithm computing $v_1, \ldots, v_p$ to retrieve the sought v_i.)

Machine M' conondeterministically guesses the label $\lambda(uv)$ of an edge uv connecting u with a child v; this can be done using conondeterministic $2|\lambda(uv)| + 1$ bits[3]. Noting that the pair $(u, \lambda(uv))$ uniquely determines v, we can now compute v. We have two cases:

- Suppose $v = v_i$ for some $i \in \{1, \ldots, p\}$. Then M' simulates all transitions of M on the path from c to the leaf ℓ_i in $U(c)$ (this may include some push operations). If ℓ_i is a final configuration, M' finishes the computation and verifies acceptance in the same way M would do. Otherwise, if ℓ_i is an existential configuration, M' further nondeterministically simulates M using its own nondeterminism, until a final or a universal configuration is encountered, or the bound of $f(k) \log n$ on the total number of nondeterministic steps is exceeded (together with a). In case of a final configuration, we do the same as before: machine M' concludes the computation and verifies whether M accepts. In case of a universal configuration, say c', M' moves the currently considered node of $U_{N,K}$ from u to v, and proceeds with working with c' at v. The counters a and b are updated by the total number of nondeterministic and conondeterminisitc bits used between ℓ and c' and between c and ℓ, respectively. Note here that the content of the stack has been appropriately updated while simulating the transitions of M from c to c'.
- Suppose $v \notin \{v_1, \ldots, v_p\}$. Then M' moves the currently considered node of $U_{N,K}$ from u to v, but enters v in the dummy mode.

This concludes the description of the behavior of M' in the real mode.

Finally, when in the dummy mode, machine M' does as follows:

- If u is a leaf, M' just accepts.
- If u is not a leaf, M' conondeterministically chooses a label $\lambda(uv)$ of an edge uv connecting u with a child v, using $2|\lambda(u, v)|+1$ conondeterminstic bits. Then M' computes v, performs a trivial nondeterministic transition, and enters v, again in the dummy mode.

This completes the construction of M'.

From the construction it follows that the contracted $\forall$ computation tree of M' on (x, k) is always $U_{N,K}$, hence M' is regular. Moreover, on every branch M' uses as much nondeterminism as M, that is, at most $f(k) \log n$, while the conondeterminism of M' is bounded by $2\lceil \log N' \rceil + k \leqslant 6f(k) + 4d \log n + k + \mathcal{O}(1)$, by the assumed properties of the labeling λ. The maximum stack length of M' is the same as that of M, while on its working tape, M' holds always at most one configuration of M plus $h(k) + \mathcal{O}(\log n)$ additional bits, for some computable function h. Finally, since every contracted $\forall$ computation tree of M accepting (x, k) within prescribed resources embeds into $U_{N,K}$, it is straightforward to see from the construction that M' accepts (x, k) within the prescribed resources if and only if M does. This concludes the proof of Lemma 12, so the proof of Theorem 10 is also complete.

4.2 A logic characterization

We now provide another characterization of XSLP, by providing a complete problem related to model-checking first-order logic. This reflects the definitions of classes AW[$\star$] and of the A-hierarchy, see [30, Chapter 8].

[3] For instance, the machine can guess consecutive bits of $\lambda(uv)$ interleaved with symbols 0 and 1, where 0 denotes "continue guessing" and 1 denotes "end of $\lambda(uv)$".

We use the standard terminology for relational structures. A (relational) signature is a set Σ consisting of *relation symbols*, where each relation symbol R has a prescribed arity $\mathrm{ar}(R) \in \mathbb{N}$. A Σ-*structure* $\mathbb{A}$ consists of a universe U and, for every relation symbol $R \in \Sigma$, its *interpretation* $R^{\mathbb{A}} \subseteq U^{\mathrm{ar}(R)}$ in $\mathbb{A}$. In this paper we only consider *binary signatures*, that is, signatures where every relation has arity at most 2.

For a signature Σ, we may consider standard first-order logic over Σ-structures. In this logic there are variables for the elements of the universe. Atomic formulas are of the form $x = y$ and $R(x_1, \ldots, x_k)$ for some $R \in \Sigma$ with $k = \mathrm{ar}(R)$, with the obvious semantics. These can be used to form larger formulas by using Boolean connectives, negation, and quantification (both existential and universal).

A Σ-structure $\mathbb{A}$ is called *forest-shaped* if Σ contains a binary relation parent such that $\mathsf{parent}^{\mathbb{A}}$ is the parent relation on a rooted forest with the vertex set being the universe of $\mathbb{A}$, and a unary relation root such that $\mathsf{root}^{\mathbb{A}}$ is the set of roots of this forest. We say that a first-order sentence φ over Σ is $\forall$-*guided* if it is of the form:

$$\varphi \quad = \quad \forall x_1 \exists y_1 \ \ldots \ \forall x_k \exists y_k \ (\mathsf{root}(x_1) \wedge \mathsf{parent}(x_1, x_2) \wedge \ldots \wedge \mathsf{parent}(x_{k-1}, x_k)) \Rightarrow \psi(x_1, y_1, \ldots, x_k, y_k)$$

where ψ is quantifier-free. In other words, φ is in a prenex form starting with a universal quantifier, and moreover we require that the first universally quantified variable is a root and every next universally quantified variable is a child of the previous one. Note that there are no restrictions on existential quantification.

For a binary signature Σ, we consider the problem of model-checking $\forall$-guided formulas on forest-shaped Σ-structures. In this problem we are given a forest-shaped Σ-structure $\mathbb{A}$ and a $\forall$-guided sentence φ, and the question is whether φ holds in $\mathbb{A}$. We consider this as a parameterized problem where $\|\varphi\|$ – the total length of an encoding of the sentence φ – is the parameter.

The following statement provides a characterization of XSLP in terms of the model-checking problem described above.

▶ **Theorem 18.** *There exists a binary signature Σ such that the following conditions are equivalent for a parameterized problem Q.*

(1) $Q \in$ XSLP*;*

(2) Q *can be* pl*-reduced to the problem of model-checking $\forall$-guided sentences on forest-shaped Σ-structures.*

The proof of Theorem 18 can be found in the full version [12], but we sketch it here.

For the (1) $\Rightarrow$ (2) implication, it suffices to pl-reduce $\textsc{BinCSP}/\mathrm{td}$ to the model-checking problem for $\forall$-guided sentences on forest-shaped structures. This is a fairly straightforward construction. Given an instance I of $\textsc{BinCSP}/\mathrm{td}$, we build a relational structure $\mathbb{A}$ consisting of the (given) elimination forest F of the Gaifman graph of I and the disjoint union of domains $D(u)$ of variables u of I. These domains are bound to respective variables using a binary predicate, and there is another binary predicate encoding the constraints. Then it is straightforward to write a $\forall$-guided sentence φ that checks the satisfiability of I in a top-down manner on F: existential variables are used to guess the evaluation of variables of I, while universal variables are used to verify the possibility of extending the current partial evaluation further down.

For the (2) $\Rightarrow$ (1) implication, by Theorem 10 it suffices to design an AROSM M that solves the model-checking problem for $\forall$-guided sentences on forest-shaped Σ-structures within the bounds stipulated in Theorem 10. Machine M uses its conondeterminism and nondeterminism to universally and existentially guess the evaluations of consecutive variables

$x_1, y_1, \ldots, x_k, y_k$, within $2k$ rounds of alternation. Here, the assumption that the input sentence is $\forall$-guided and the input structure is forest-shaped can be used in conjunction with Lemma 14 to bound the total conondeterminism used by $\mathcal{O}(k + \log n)$. Once all variables are evaluated, satisfaction of ψ can be checked within 4 additional rounds of alternation by assuming without loss of generality that ψ is in DNF.

5 Conclusion

In this paper we explored the parameterized complexity of BinCSP for a variety of relatively strong structural parameters, including the vertex cover number, treedepth, and several modulator-based parameters. We believe that together with the previous works on XALP and XNLP [7, 9, 10, 11, 26], our work uncovers a rich complexity structure within the class XP, which is worth further exploration. We selected concrete open questions below.

- In [10, 11], several problems such as Independent Set or Dominating Set, which are fixed-parameter tractable when parameterized by treewidth, were shown to be XALP- and XNLP-complete when parameterized by the *logarithmic* treewidth and pathwidth, which is at most k when the corresponding width measure is at most $k \log n$. Can one prove similar results for the class XSLP and parameterization by logarithmic treedepth?
- Theorem 3 provides natural complete problems only for the odd levels of the W-hierarchy. Similarly, we defined the S-hierarchy only for odd levels. It would be interesting to have a natural description of the situation also for the even levels.
- The characterizations of XSLP given by Theorems 10 and 18 can be "projected" to a rough characterizations of classes S[d] for odd d by stipulating that the alternation is at most d. Unfortunately, this projection turns out not to be completely faithful: the obtained problems do not precisely characterize the class S[d], but lie somewhere between S[$d - \mathcal{O}(1)$] and S[$d + \mathcal{O}(1)$]. Can we provide a compelling description of the levels of the S-hierarchy in terms of machine problems or in terms of model-checking first-order logic?
- What is the complexity of List Coloring parameterized by the vertex cover number? Currently, we know it is W[1]-hard and in W[2]. Similarly, what is the complexity of List Coloring and Precoloring Extension with the minimum size of a modulator to a treedepth-d graph as the parameter?
- Can one obtain a better understanding of the complexity of BinCSP and List Coloring parameterized by the feedback vertex number?

References

1 Faisal N. Abu-Khzam, Henning Fernau, Benjamin Gras, Mathieu Liedloff, and Kevin Mann. Enumerating minimal connected dominating sets. In *30th Annual European Symposium on Algorithms, ESA 2022*, volume 244 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1:1–1:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.1.

2 Kunwarjit S. Bagga, Lowell W. Beineke, Wayne D. Goddard, Marc J. Lipman, and Raymond E. Pippert. A survey of integrity. *Discrete Applied Mathematics*, 37:13–28, 1992. doi:10.1016/0166-218X(92)90122-Q.

3 Aritra Banik, Ashwin Jacob, Vijay Kumar Paliwal, and Venkatesh Raman. Fixed-parameter tractability of $(n - k)$ list coloring. *Theory of Computing Systems*, 64(7):1307–1316, 2020. doi:10.1007/s00224-020-10014-9.

4 Curtis A. Barefoot, Roger Entringer, and Henda Swart. Vulnerability in graphs—a comparative survey. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 1(38):13–22, 1987.

5 Thomas Bläsius, Tobias Friedrich, Julius Lischeid, Kitty Meeks, and Martin Schirneck. Efficiently enumerating hitting sets of hypergraphs arising in data profiling. *Journal of Computer and System Sciences*, 124:192–213, 2022. doi:10.1016/j.jcss.2021.10.002.

6 Thomas Bläsius, Tobias Friedrich, and Martin Schirneck. The complexity of dependency detection and discovery in relational databases. *Theoretical Computer Science*, 900:79–96, 2022. doi:10.1016/j.tcs.2021.11.020.

7 Hans L. Bodlaender, Gunther Cornelissen, and Marieke van der Wegen. Problems hard for treewidth but easy for stable gonality. In *48th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2022*, volume 13453 of *Lecture Notes in Computer Science*, pages 84–97. Springer, 2022. doi:10.1007/978-3-031-15914-5_7.

8 Hans L. Bodlaender, Carla Groenland, and Hugo Jacob. List Colouring Trees in Logarithmic Space. In *Proceedings 30th Annual European Symposium on Algorithms, ESA 2022*, volume 244 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24:1–24:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.24.

9 Hans L. Bodlaender, Carla Groenland, Hugo Jacob, Lars Jaffke, and Paloma T. Lima. XNLP-completeness for parameterized problems on graphs with a linear structure. In *Proceedings 17th International Symposium on Parameterized and Exact Computation, IPEC 2022*, volume 249 of *LIPIcs*, pages 8:1–8:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.IPEC.2022.8.

10 Hans L. Bodlaender, Carla Groenland, Hugo Jacob, Marcin Pilipczuk, and Michał Pilipczuk. On the complexity of problems on tree-structured graphs. In *17th International Symposium on Parameterized and Exact Computation, IPEC 2022*, volume 249 of *LIPIcs*, pages 6:1–6:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.IPEC.2022.6.

11 Hans L. Bodlaender, Carla Groenland, Jesper Nederlof, and Céline M. F. Swennenhuis. Parameterized problems complete for nondeterministic FPT time and logarithmic space. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*, pages 193–204. IEEE, 2021. doi:10.1109/FOCS52979.2021.00027.

12 Hans L. Bodlaender, Carla Groenland, and Michał Pilipczuk. Parameterized complexity of binary csp: Vertex cover, treedepth, and related parameters. *CoRR*, abs/2208.12543, 2022. arXiv:2208.12543.

13 Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasi-polynomial time. *SIAM J. Comput.*, 51(2):17–152, 2022. doi:10.1137/17m1145288.

14 Katrin Casel, Henning Fernau, Mehdi Khosravian Ghadikolaei, Jérôme Monnot, and Florian Sikora. On the complexity of solution extension of optimization problems. *Theoretical Computer Science*, 904:48–65, 2022. doi:10.1016/j.tcs.2021.10.017.

15 Jianer Chen and Fenghui Zhang. On product covering in 3-tier supply chain models: Natural complete problems for $W[3]$ and $W[4]$. *Theoretical Computer Science*, 363(3):278–288, 2006. doi:10.1016/j.tcs.2006.07.016.

16 Jiehua Chen, Wojciech Czerwiński, Yann Disser, Andreas Emil Feldmann, Danny Hermelin, Wojciech Nadara, Marcin Pilipczuk, Michał Pilipczuk, Manuel Sorge, Bartlomiej Wróblewski, and Anna Zych-Pawlewicz. Efficient fully dynamic elimination forests with applications to detecting long paths and cycles. In *2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 796–809. SIAM, 2021. doi:10.1137/1.9781611976465.50.

17 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

18 Wojciech Czerwiński, Laure Daviaud, Nathanaël Fijalkow, Marcin Jurdziński, Ranko Lazić, and Pawel Parys. Universal trees grow inside separating automata: Quasi-polynomial lower bounds for parity games. In *Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 2333–2349. SIAM, 2019. doi:10.1137/1.9781611975482.142.

19 Wojciech Czerwiński, Wojciech Nadara, and Marcin Pilipczuk. Improved bounds for the excluded-minor approximation of treedepth. *SIAM J. Discret. Math.*, 35(2):934–947, 2021. doi:10.1137/19M128819X.

20 Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1999. doi:10.1007/978-1-4612-0515-9.

21 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

22 Zdenek Dvořák, Archontia C. Giannopoulou, and Dimitrios M. Thilikos. Forbidden graphs for tree-depth. *Eur. J. Comb.*, 33(5):969–979, 2012. doi:10.1016/j.ejc.2011.09.014.

23 Pavel Dvořák, Eduard Eiben, Robert Ganian, Dušan Knop, and Sebastian Ordyniak. The complexity landscape of decompositional parameters for ILP: Programs with few global variables and constraints. *Artificial Intelligence*, 300:103561, 2021. doi:10.1016/j.artint.2021.103561.

24 Friedrich Eisenbrand, Christoph Hunkenschröder, Kim-Manuel Klein, Martin Koutecký, Asaf Levin, and Shmuel Onn. An algorithmic theory of integer programming. *CoRR*, abs/1904.01361, 2019. arXiv:1904.01361.

25 Michael Elberfeld, Martin Grohe, and Till Tantau. Where first-order and monadic second-order logic coincide. *ACM Trans. Comput. Log.*, 17(4):25, 2016. doi:10.1145/2946799.

26 Michael Elberfeld, Christoph Stockhusen, and Till Tantau. On the space and circuit complexity of parameterized problems: Classes and completeness. *Algorithmica*, 71(3):661–701, 2015. doi:10.1007/s00453-014-9944-y.

27 Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009. doi:10.1016/j.tcs.2008.09.065.

28 Michael R. Fellows, Daniel Lokshtanov, Neeldhara Misra, Frances A. Rosamond, and Saket Saurabh. Graph layout problems parameterized by vertex cover. In *19th International Symposium on Algorithms and Computation, ISAAC 2008*, volume 5369 of *Lecture Notes in Computer Science*, pages 294–305. Springer, 2008. doi:10.1007/978-3-540-92182-0_28.

29 Jirí Fiala, Petr A. Golovach, and Jan Kratochvíl. Parameterized complexity of coloring problems: Treewidth versus vertex cover. *Theoretical Computer Science*, 412(23):2513–2523, 2011. doi:10.1016/j.tcs.2010.10.043.

30 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. doi:10.1007/3-540-29953-X.

31 Fedor V. Fomin, Bart M. P. Jansen, and Michał Pilipczuk. Preprocessing subgraph and minor problems: When does a small vertex cover help? *Journal of Computer and System Sciences*, 80(2):468–495, 2014. doi:10.1016/j.jcss.2013.09.004.

32 Eugene C. Freuder. Complexity of K-tree structured Constraint Satisfaction Problems. In *8th National Conference on Artificial Intelligence, AAAI-90*, pages 4–9. AAAI Press / The MIT Press, 1990. URL: http://www.aaai.org/Library/AAAI/1990/aaai90-001.php.

33 Martin Fürer and Huiwen Yu. Space saving by dynamic algebraization. In *9th International Computer Science Symposium in Russia, CSR 2014*, volume 8476 of *Lecture Notes in Computer Science*, pages 375–388. Springer, 2014. doi:10.1007/978-3-319-06686-8_29.

34 Robert Ganian. Improving vertex cover as a graph parameter. *Discrete Mathematics and Theoretical Computer Science*, 17(2):77–100, 2015. doi:10.46298/dmtcs.2136.

35 Gregory Z. Gutin, Diptapriyo Majumdar, Sebastian Ordyniak, and Magnus Wahlström. Parameterized pre-coloring extension and list coloring problems. *SIAM Journal on Discrete Mathematics*, 35(1):575–596, 2021. doi:10.1137/20M1323369.

36 Miika Hannula, Bor-Kuan Song, and Sebastian Link. An algorithm for the discovery of independence from data. *arXiv*, abs/2101.02502, 2021. arXiv:2101.02502.

37 Falko Hegerfeld and Stefan Kratsch. Solving connectivity problems parameterized by treedepth in single-exponential time and polynomial space. In *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020*, volume 154 of *LIPIcs*, pages 29:1–29:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.STACS.2020.29.

38 Marcin Jurdziński and Ranko Lazić. Succinct progress measures for solving parity games. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017*, pages 1–9. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005092.

39 Ken-ichi Kawarabayashi and Benjamin Rossman. A polynomial excluded-minor approximation of treedepth. In *2018 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 234–246. SIAM, 2018. doi:10.1137/1.9781611975031.17.

40 Deepanshu Kush and Benjamin Rossman. Tree-depth and the formula complexity of subgraph isomorphism. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 31–42. IEEE, 2020. doi:10.1109/FOCS46700.2020.00012.

41 Wojciech Nadara, Michał Pilipczuk, and Marcin Smulewicz. Computing treedepth in polynomial space and linear FPT time. In *30th Annual European Symposium on Algorithms, ESA 2022*, volume 244 of *LIPIcs*, pages 79:1–79:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.79.

42 Jesper Nederlof, Michał Pilipczuk, Céline M. F. Swennenhuis, and Karol Węgrzycki. Hamiltonian Cycle parameterized by treedepth in single exponential time and polynomial space. In *46th International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 12301 of *Lecture Notes in Computer Science*, pages 27–39. Springer, 2020. doi:10.1007/978-3-030-60440-0_3.

43 Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity – Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-27875-4.

44 Christos H. Papadimitriou and Mihalis Yannakakis. On the complexity of database queries. *Journal of Computer and System Sciences*, 58(3):407–427, 1999. doi:10.1006/jcss.1999.1626.

45 Michał Pilipczuk and Sebastian Siebertz. Polynomial bounds for centered colorings on proper minor-closed graph classes. *J. Comb. Theory, Ser. B*, 151:111–147, 2021. doi:10.1016/j.jctb.2021.06.002.

46 Michał Pilipczuk and Marcin Wrochna. On space efficiency of algorithms working on structural decompositions of graphs. *ACM Trans. Comput. Theory*, 9(4):18:1–18:36, 2018. doi:10.1145/3154856.

Nondeterministic Interactive Refutations for Nearest Boolean Vector

Andrej Bogdanov
University of Ottawa, Canada

Alon Rosen
Bocconi University, Milano, Italy
Reichman University, Herzliya, Israel

Abstract

Most n-dimensional subspaces $\mathcal{A}$ of $\mathbb{R}^m$ are $\Omega(\sqrt{m})$-far from the Boolean cube $\{-1,1\}^m$ when $n < cm$ for some constant $c > 0$. How hard is it to certify that the Nearest Boolean Vector (NBV) is at least $\gamma\sqrt{m}$ far from a given random $\mathcal{A}$?

Certifying NBV instances is relevant to the computational complexity of approximating the Sherrington-Kirkpatrick Hamiltonian, i.e. maximizing $x^T Ax$ over the Boolean cube for a matrix A sampled from the Gaussian Orthogonal Ensemble. The connection was discovered by Mohanty, Raghavendra, and Xu (STOC 2020). Improving on their work, Ghosh, Jeronimo, Jones, Potechin, and Rajendran (FOCS 2020) showed that certification is not possible in the sum-of-squares framework when $m \ll n^{1.5}$, even with distance $\gamma = 0$.

We present a non-deterministic interactive certification algorithm for NBV when $m \gg n \log n$ and $\gamma \ll 1/mn^{1.5}$. The algorithm is obtained by adapting a public-key encryption scheme of Ajtai and Dwork.

2012 ACM Subject Classification Theory of computation → Interactive proof systems; Theory of computation → Problems, reductions and completeness; Theory of computation → Cryptographic primitives

Keywords and phrases average-case complexity, statistical zero-knowledge, nondeterministic refutation, Sherrington-Kirkpatrick Hamiltonian, complexity of statistical inference, lattice smoothing

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.28

Category Track A: Algorithms, Complexity and Games

Funding *Andrej Bogdanov*: Work supported by Hong Kong RGC GRF grant CUHK 14209920 and NSERC grant RGPIN-2023-05006.
Alon Rosen: Work supported by European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101019547) and Cariplo CRYPTONOMEX grant.

Acknowledgements We are grateful to Chris Jones for useful advice and feedback. Part of this work was done while the first author was affiliated with and the second author visited the Chinese University of Hong Kong.

1 Introduction

When can we expect to have a reduction from problem A to problem B? Complexity theory can be used not only to show existence of reductions but also to argue separations. For example, one reason an oracle for factoring is not considered an imminent threat to SAT is that the correctness of prime factorizations can be both proved and refuted, that is (the decision version of) factoring is in NP ∩ coNP.

In general, there cannot be a reduction (of sufficiently low complexity) from A to B if there is a complexity class that (conjecturally) separates the two. For worst-case problems in NP the separating class is often NP ∩ coNP or one of its close relatives (NP ∩ coAM or Statistical Zero-Knowledge).

It is natural to wonder whether analogous separations in average-case complexity can clarify the landscape of reductions within distributional NP; a class of particular importance to cryptography and learning theory. Reductions among non-NP-complete distributional problems do exist, but are few and far between. Notable examples include lattice problems [18, 24, 22, 17]. More recently, a web of reductions was developed to explain the hardness of various statistical inference problems [5].

A handful of average-case NP complete problems were found in the 1980-90s [16, 9]. All these problems are closely related to simulation of Turing Machines, perhaps necessarily so [29]. The conjectured hardness of combinatorial problems like random SAT or planted clique still lacks satisfactory explanation.

In the context of random SAT, Feige, Kim, and Ofek [7] showed that random 3CNF instances with n variables and $m \gg n^{1.4}$ equations admit efficient nondeterministic refutations of satisfiability, that is, belong to Avg-coNP.[1] Although most such instances are unsatisfiable, it is not known how to efficiently certify the lack of a satisfying assignment in the regime $n^{1.4} \ll m \ll n^{1.5}$. On the other hand, when $m \ll n^{1.4}$ not even nondeterministic refutations are known. Thus we do not expect a reduction from random 3SAT with clause-to-variable density $n^{0.41}$ to random 3SAT with density $n^{0.39}$ barring a major algorithmic advance.

Our contribution is an analogous result for the distributional Nearest Boolean Vector to a Subspace problem which was introduced by Mohanty, Raghavendra and Xu [19]. In Theorem 1 we show that for a certain parameter regime in which this problem may be intractable, the problem is in average-case statistical zero-knowledge (Avg-SZK) and therefore admits *interactive* nondeterministic refutations.

1.1 The Nearest Boolean Vector problem

We work with the following formulation of the Nearest Boolean Vector problem:

Nearest Boolean Vector (NBV):
Input: An n-dimensional subspace $\mathcal{A}$ of $\mathbb{R}^m$.
Yes instances: There exists a $v \in \{-1,1\}^m$ such that $\mathrm{dist}(v,\mathcal{A}) \leq \gamma\sqrt{m}$.
No instances: For all $v \in \{-1,1\}^m$, $\mathrm{dist}(v,\mathcal{A}) > \sqrt{m}/2$.

When $n < cm$ for a sufficiently small constant c, most subspaces $\mathcal{A}$ (chosen from the uniform Haar measure) are no instances [19]. We are interested in the errorless average-case complexity of NBV. An efficient average-case algorithm for distributional NBV can be viewed as an efficiently computable certificate that most subspaces are far from the Boolean cube.

When $\gamma < 1/2$, NBV is in NP. Several works [19, 8, 23] provide evidence that it is intractable on average in the regime $m \ll n^2$.

1.2 Our Result

We give a reduction from distributional NBV to the Statistical Distance to Uniform (SDU) problem. The input to SDU is a sampler of outputs in $\{0,1\}^n$, the YES instances are samplers whose outputs are $1-\delta$ far from uniform, and NO instances are samplers whose values are δ close to uniform. For $\delta = 1/3$ SDU is in the class Statistical Zero Knowledge (SZK) [26], which is a subclass of coAM.[2]

[1] Their result was recently extended to the semi-random model [12] in which the formula is arbitrary and only the literals are polarized randomly.

[2] When $\delta = 1/n$, SDU is in the more restricted class Non-Interactive Statistical Zero-Knowledge (NISZK) [10]. AvgNISZK membership can also be obtained for smaller γ.

▶ **Theorem 1.** *Let C be a sufficiently large constant and $\epsilon \geq 2^{-n/C}$. For all but an ϵ-fraction of instances, NBV with parameters $m = Cn \log n$ and $\gamma = 1/Cmn^{3/2} \log^{1/2}(n/\epsilon)$ is in* AvgSZK.

The proof is given in Section 3. In Section 5 we outline a tentative approach for improving the completeness error γ.

When ϵ is polynomial in n, SZK membership holds for all but a $n^{-O(1)}$ fraction of instances and the approximation factor γ has value $\tilde{\Theta}(1/mn^{3/2})$. When ϵ is $2^{-\Omega(n)}$ then the fraction of instances is exponential, but $\gamma = \Theta(1/mn^2)$.

2 Background and Overview

2.1 Average-case refutations

Refutations come up naturally in the study of combinatorial optimization. A worst-case approximation algorithm A for a minimization problem P is required to output a value within a factor of c of the optimum on all instances. Such an algorithm provides an efficient refutation of the claim

$$x \text{ has a solution of value at most } A(x)/c \tag{1}$$

for every instance x.

When efficient refutations are hard to obtain for all x it may be natural to relax the condition to hold for most x. An average-case refutation should still certify (1), but it is now allowed to fail on some small fraction of inputs x.

For many natural distributions, the optimum is tightly concentrated around its expectation. For example, the maximum number of satisfiable clauses in random 3SAT with sufficiently large clause-to-variable density is close to 7/8 on most instances. In particular, an average-case refutation must certify that most instances are not satisfiable, but it should be allowed to output "I don't know" on a small fraction of inputs. This motivates the following definition:

▶ **Definition 2.** *A refutation R with failure rate ϵ for distributional (promise) problem f is an algorithm that outputs "no" or "I don't know", is always correct ($R(x) = f(x)$ or "I don't know"), and outputs "I don't know" on at most an ϵ-fraction of inputs.*

While efficient deterministic or randomized refutations are needed for the design of approximation algorithms, in this work we are interested in the existence of nondeterministic (coNP-type) refutations. Such refutations yield efficiently *verifiable* certificates of (1) on most inputs. As a consequence of Theorem 1 we have

▶ **Corollary 3.** *There is a efficient nondeterministic interactive refutation for NBV with failure rate $\epsilon \geq 2^{-n/C}$ and parameters $m = Cn \log n$, $\gamma = 1/Cmn^{3/2} \log^{1/2}(n/\epsilon)$.*

2.2 Refutations in the Sum-of-Squares Framework

The sum-of-squares (SoS) framework is an incomplete but poweful framework for refuting optimization problems. It has been used to argue efficient refutations do not exist for problems such as clique [3]. The most notable incorrect prediction of SoS is on random 3LIN with perfect completeness [11, 27]. In that case not only do refutations exist but they can be found by Gaussian elimination.

In contrast, the nondeterministic refutations of Feige, Kim, and Ofek arise as solutions to the level-$O(n^{2\delta})$ SoS relaxation of random 3SAT with n variables and m constraints. This may be viewed as evidence that SoS correctly predicts refutability in problems that are immune to Gaussian elimination "attacks".

2.3 Sherrington-Kirkpatrick and Nearest Boolean Vector

The negative energy of the Sherrington-Kirkpatrick Hamiltonian at zero-temperature is the value

$$SK(M) = \min \frac{1}{\sqrt{n}} \cdot x^T M x \qquad \text{subject to } x \in \{\pm 1/\sqrt{n}\}^n$$

for a matrix M sampled from the Gaussian Orthogonal Ensemble. It can be efficiently certified that $SK(M) \leq 2 + \epsilon$ for every $\epsilon > 0$ and most matrices M via the relaxation

$$SK(M) \leq \min_{\|u\|=1} \frac{1}{\sqrt{n}} \cdot u^T M u = \lambda_1(M), \tag{2}$$

where $\lambda_1(M)$ is the largest eigenvalue of M, which is known to not exceed $2 + \epsilon$ for most matrices M.

Parisi [21] conjectured and Talagrand [28] proved that $SK(M)$ is in fact strictly smaller than 2 for most matrices M. The true value for most M is concentrated around Parisi's constant $P_* \approx 1.526$. More recently Montanari [20] found an algorithm that finds a solution x for which $x^T M x \leq P_* - \epsilon$ for most matrices M and proved its correctness under some plausible conjecture.

Mohanty, Raghavendra, and Xu [19] ask whether Montanari's algorithm can be matched with an efficient certificate that $SK(M) \leq P_* + \epsilon$ for most matrices M. Together with Montanari's algorithm, this would give an errorless heuristic for calculating $SK(M)$ up to lower-order terms. As a first step they show that SK reduces to the potentially more tractable Nearest Boolean Vector Problem.

Mohanty, Raghavendra, and Xu prove that for all $c, \gamma > 0$ there exists an $\epsilon > 0$ such that if NBV with parameters $m/n = c$ and γ admits efficient refutations than so does the claim $SK(M) \leq 2 - \epsilon$ for most M. Moreover, for sufficiently small c, most subspaces $\mathcal{A}$ are no-instances of NBV.

However, their main evidence for refutability of NBV is negative: They show that no refutations can be obtained from the natural degree-4 SoS relaxation of NBV for any constant c, even in case of perfect completeness $\gamma = 0$. A refutation algorithm for $\gamma = 0$ is merely required to certify that no Boolean vector belongs to the subspace $\mathcal{A}$. The SoS hardness regime was later extended to $m \ll n^{3/2}$ and to degree-$n^{\Omega(1)}$ SoS by Ghosh et al. [8]. It is believed that it can be further extended up to $m < n^2/4$, as (heuristically) suggested by calculations of the low-degree likelihood ratio (see Potechin et al. [23]).

Theorem 1 has no bearing on the complexity of certifying that $SK(M) \leq 2$ for most M. To obtain an improvement over the spectral certificate (2) the completeness error γ would have to be constant, or at least $m^{-\epsilon}$ for some small ϵ.

2.4 Algorithms for NBV

When $m \gg n^2$ and γ is a sufficiently small constant it is plausible that NBV can be efficiently solved by linearization. Represent $\mathcal{A}$ as the column span of B for some $m \times n$ matrix B. Consider the objective

$$\text{minimize } \sum_{i=1}^{m} \big(\langle B_i, x\rangle^2 - 1\big)^2 \quad \text{over } x \in \mathbb{R}^n, \tag{3}$$

where B_i is the i-th row of B. If $\mathcal{A}$ had a Boolean vector $\langle B_i, x\rangle = \pm 1$ the value of this objective would be zero. We suspect that for most matrices B it should be lower bounded by

$\Omega(m)$. If (3) were efficiently computable its value would be the required certificate. Although this is unlikely, the same argument can be applied to its linearization in which degree-2 monomials $x_i x_j$ are represented by variables y_{ij}:

$$\text{minimize} \sum_{i=1}^{m} \left(\sum_{j,k=1}^{n} B_{ij} B_{ik} y_{jk} - 1 \right)^2 \quad \text{over } y \in \mathbb{R}^{n(n+1)/2}, \tag{4}$$

which is a convex quadratic objective and therefore efficiently minimizable.

In the case of perfect completeness, $\gamma = 0$ NBV reduces to the Shortest Vector Problem (SVP) in lattices with approximation factor exponential in the dimension and can therefore be solved by the LLL algorithm [15] for any $m > n$. Here is an outline of the (standard) reduction R. Let the columns of $C \in \mathbb{R}^{m \times (m-n)}$ be a random orthonormal basis of the dual subspace $\mathcal{A}^{\perp}$. Consider the lattice $\mathcal{L}$ spanned by the rows of the $m \times (2m - n)$ matrix $C' = [\delta I_m | C]$ for $\delta = 2^{-2m^2}$. If $\mathcal{A}$ contained a Boolean vector x then $C'x$ would be a vector of length $\delta\sqrt{m}$ in $\mathcal{L}$. If not, by a union bound there is unlikely to exist a vector $x \in \{-2^m, \ldots, 2^m\}^m$ for which $\|Cx\| < 2^m \delta$ so the shortest vector in $\mathcal{L}$ has length at least $2^m \delta$.

2.5 Nondeterministic refutations for NBV

This reduction R extends to almost-perfect completeness $\gamma = 2^{-\Theta(m^2)}$. It is tempting to conjecture for $m \gg n \log n$ that there is a constant d such that R reduces NBV with parameter $\gamma = m^{-d}$ to SVP with approximation factor $\sqrt{m}$, which is a coNP problem [1]. Should such a reduction exist it would imply efficient nondeterministic refutations for NBV.

We were unable to prove the soundness of R in this parameter regime. Our preliminary calculations indicate that $\mathcal{L}$ may contain unusually short vectors for most instances $\mathcal{A}$ of NBV.

Instead, we prove Theorem 1 by adapting a public-key encryption scheme of Ajtai and Dwork [2] (see [4] for a "modern" description) into the desired reduction from NBV to SDU.

2.6 Refutations, SZK, and Public-key Encryption

The *chosen plaintext attack* security notion for one-bit encryption with public key PK and encryption algorithm Enc posits that the distributions $(PK, Enc(PK, 0))$ and $(PK, Enc(PK, 1))$ are computationally indistinguishable. In contrast, functionality requires that they be statistically distinguishable by the decryption algorithm.

The security of several public-key encryption candidates is argued using a model (fake) public-key distribution FK with the property that PK and FK are computationally indistinguishable while $(FK, Enc(FK, 0))$ and $(FK, Enc(FK, 1))$ are statistically indistinguishable. This proof strategy yields a reduction from distinguishing real and model public keys to SDU.

The security proof for the Ajtai-Dwork (AD) and Bogdanov et al.'s (BCHR) pancake encryptions are of this type. In BCHR, the model public key FK is a sequence m of independent standard n-dimensional Gaussians, while in the real public key PK an almost-periodic component is planted in a secret direction s of $\mathbb{R}^n$. If the almost-periodic component is concentrated around the values -1 and 1, the row-span of PK can be viewed as a yes-instance of NBV.

To turn this distinguisher between PK and FK into a refutation, we observe that the encryption remains functional even for a worst-case choice of PK that satisfies some efficiently verifiable conditions (the largest and smallest singular values of PK are pseudorandom). By

verifying these conditions the reduction from NBV to SDU ensures that *all* yes-instances of NBV map to yes-instances of SDU, while affecting only a small fraction of no-instances, thus providing interactive remoteness certificates for most instances of NBV.

The BCHR encryption and security proof suggest the following visualization of the remoteness certificates. If a random matrix FK is multiplied on the right by a random $x \sim \{\pm 1\}^m$ the output $FK \cdot x$ is close to a random Gaussian point in $\mathbb{R}^n$ (see Fact 17). On the other hand, $PK \cdot x$ is concentrated around "pancakes" perpendicular to the secret direction s. To certify remoteness, the verifier asks the prover to furnish an $x \in \{\pm 1\}^m$ close to a random Gaussian point g in $\mathbb{R}^n$. Unless g happens to land close to a pancake the prover will fail on an no- instance PK of NBV.

A fatal weakness of BCHR encryption is that it is insecure unless $m \gg n^2$, a setting of parameters in which NBV is tractable. In contrast, security of AD can be proved when $m = O(n \log n)$. This improvement is obtained by modifying the encryption from $\mathrm{round}(PK \cdot x)$ to $\mathrm{round}(\sigma A \cdot x) \bmod \mathcal{P}(B)$, where $PK = [A|B]$ with $A \in \mathbb{R}^{(m-n)\times n}$ and $B \in \mathbb{R}^{n\times n}$ is the public key matrix, $\mathcal{P}(B)$ is the parallelepiped spanned by the columns of B, and σ is a suitable scaling factor. Reduction 1 in Section 3 implements this security proof (in a different basis which is more suitable for analysis), again by imposing some efficiently verifiable conditions that hold for typical yes-instances but for none of the no-instances of NBV.

3 Refutation via Lattice Smoothing

We represent the random subspace $\mathcal{A}$ as the row space of a random $n \times m$ matrix $[A|B']$ of independent normal entries. It is sufficient to specify these entries up to $O(\log n)$ bits of precision. We carry out our analyses assuming infinite precision. It will be clear from the calculations that the additional effect of rounding the entries of A does not affect correctness.

For a real number x let $x = \lfloor x \rceil + \{x\}$ be its unique representation with $\lfloor x \rceil \in \mathbb{Z}$ and $\{x\} \in [-1/2, 1/2)$. Let $\{x\}_p$ be the multiple of $1/p$ in $[-1/2, 1/2)$ closest to $\{x\}$. The notation extends to vectors and matrices entrywise.

▷ Fact 4. (a) $|\{x\}| \le |x|$ and (b) $|\{x + y\}| \le |\{x\}| + |\{y\}|$.

We choose the modulus p to equal $Cn\sqrt{m}$ for a sufficiently large constant C. Let $\sigma = (1/\pi)\sqrt{n \ln(12mn/\epsilon + 2n)}$.

Reduction 1: On input $[A|B']$, $A \in \mathbb{R}^{n\times(m/2)}$, $B' \in \mathbb{R}^{n\times(m/2)}$,

1 Find a submatrix B of B' with smallest singular value at least $1/\sqrt{n}$.
2 If step 1 is unsuccessful, fail.
3 If any column of A has norm more than $2\sqrt{n}$, fail.
4 Otherwise, output the sampler S that maps $x \sim \{\pm 1\}^{m/2}$ to $\{\{\sigma B^{-1} A\}_p x\} \in \frac{1}{p}\mathbb{Z}_p^n$.

A naive implementation of step 1 would split B' into m/n candidate matrices B and attempt to find one with singular value $1/\sqrt{n}$, resulting in failure rate $\epsilon = 2^{-O(m/n)}$ which is $n^{-O(1)}$ when $m = O(n \log n)$. In Section 4 we design a greedy procedure for choosing B that improves the failure rate to $2^{-\Omega(n)}$.

Theorem 1 follows from Claims 5 and 8.

▷ Claim 5. Assume $\epsilon > 2^{-\Omega(n)}$. For all but an ϵ-fraction of instances $[A|B']$ the output of S is 1/3-close to a uniformly random element of $\frac{1}{p}\mathbb{Z}_p^n$.

▷ Fact 6 (Smoothing). [18, Lemmas 3.3 and 4.1] If all columns of $B \in \mathbb{R}^{n\times n}$ have norm at most b, g is standard normal in $\mathbb{R}^n$, and $\sigma \geq (b/2\pi)\sqrt{\ln(n/\epsilon + 2n)}$, then $\{\sigma B^{-1} g\}$ is ϵ-close to a uniform random point in $[-1/2, 1/2)^n$.

▷ Fact 7 (Leftover hash lemma). [13] If $C \sim \mathbb{Z}_p^{n\times m}$ is a random matrix and $x \in \mathbb{Z}_p^m$ be a random vector uniformly distributed on some set of size M then (C, Cx) is $\sqrt{p^n/M}$-close to uniformly random.

Proof of Claim 5. By Proposition 9 B can be found (efficiently) except with probability $\exp(-\Omega(m))$. By large deviation bounds all columns of B have norm at most $2\sqrt{n}$ except with probability $2^{-\Omega(n)}$. By our choice of parameters, both conditions are satisfied except with probability $2^{-\Omega(m)} + 2^{-\Omega(n)} \leq \epsilon/2$. Assuming this we argue the conclusion holds even when conditioning on B.

For each column a_i of A, $\sigma a_i \in \mathbb{R}^n$ is a normal vector of zero mean and covariance σI. By smoothing Fact 6, $\{\sigma B^{-1} a_i\}$ is $\epsilon/4m$-close to a uniform point in $[-1/2, 1/2)^n$. Therefore $C = \{\sigma B^{-1} A\}_p$ is $\epsilon/12$-close to a random matrix in $\frac{1}{p}\mathbb{Z}_p^{(m/2)\times n}$. By Fact 7, (C, Cx) is $\epsilon/12 + \sqrt{p^n/2^{m/2}}$-close to random. By our choice of parameters, $\epsilon/12 + \sqrt{p^n/2^{m/2}} \leq \epsilon/6$. By Markov's inequality the output of the sampler is 1/3-close to random except with probability $\epsilon/2$ over the choice of A, and therefore except with probability ϵ over the choice of A and B'. ◁

▷ Claim 8. If $[A|B']$ is a yes instance of NBV with parameters $m > Cn \log n$ and $\gamma < 1/Cmn^{3/2} \log^{1/2}(n/\epsilon)$, either the reduction fails, or the output of S is 2/3-far from random.

Proof. As $[A|B']$ is a yes instance of NBV there exists a witness $w \in \mathbb{R}^n$ such that $w[A|B'] = v + e$, where $v \in \{\pm 1\}^m$ and $\|e\| \leq \gamma\sqrt{m}$. Let D be the distinguisher that on input $y \in \frac{1}{p}\mathbb{Z}_p^n$ accepts if $|\{\langle wB, y\rangle\}| < 1/24$.

Assume y is uniform in $\frac{1}{p}\mathbb{Z}_p^n$. We show D accepts y with probability at most 1/6. We can write y as $\{u\}_p$ where u is uniform in $[0, 1)^n$. Let $e' = y - u$ and let v_B and e_B be the projections of v and e on the columns indexed by B. Then

$$\langle wB, y\rangle = \langle v_B + e_B, u + e'\rangle = \langle v_B, u\rangle + \langle v_B, e'\rangle + \langle e_B, y\rangle$$

The random variable $\{\langle v_B, u\rangle\}$ is uniform in $[-1/2, 1/2)$, so $|\{\langle v_B, u\rangle\}| > 1/12$ with probability 5/6. If this happens, by the triangle inequality,

$$\begin{aligned}
|\{\langle wB, y\rangle\}| &\geq |\{\langle v_B, u\rangle\}| - |\langle v_B, e'\rangle| - |\langle e_B, y\rangle| \\
&\geq 1/12 - \|v_B\|\|e'\| - \|e_B\|\|y\| \\
&\geq 1/12 - n/p - \gamma\sqrt{mn} \\
&> 1/24
\end{aligned}$$

and D rejects y.

Now assume the reduction does not fail so that $\|B^{-1}\| \leq \sqrt{n}$ and all columns of A and B have norm at most $2\sqrt{n}$. We will show that D accepts $y = \{\{\sigma B^{-1} A\}_p x\}$ with probability at least 5/6. Therefore D distinguishes this distribution from the uniform one, so the two must be 2/3-far.

Let $E = \{\sigma B^{-1} A\}_p - \{\sigma B^{-1} A\}$. Then

$$\{\sigma B^{-1} A\}_p = \{\sigma B^{-1} A\} + E = \sigma B^{-1} A - \lfloor \sigma B^{-1} A \rfloor + E.$$

Since x is integral,

$$y = \{\{\sigma B^{-1} A\}_p x\} = \{\sigma B^{-1} A x + E x\}.$$

Therefore

$$\langle wB, y\rangle = \langle wB, \sigma B^{-1}Ax\rangle + \langle wB, Ex\rangle - \langle wB, f\rangle,$$

where $f = \lfloor \sigma B^{-1}Ax + Ex \rceil$. The first term equals

$$\langle wB, \sigma B^{-1}Ax\rangle = \sigma\langle wA, x\rangle = \sigma\langle v_A, x\rangle + \sigma\langle e_A, x\rangle,$$

where v_A and e_A are the projections of v and e on the coordinates indexed by the columns of A. The third term equals

$$\langle wB, f\rangle = \langle v_B, f\rangle + \langle e_B, f\rangle.$$

As $\sigma\langle v_A, x\rangle$ and $\langle v_B, f\rangle$ are integers,

$$\begin{aligned}
|\{\langle wB, y\rangle\}| &\le |\sigma\langle e_A, x\rangle| + |\langle wB, Ex\rangle| + |\langle e_B, f\rangle| \\
&\le \sigma\|e_A\|\|x\| + \|wB\|\|Ex\| + \|e_B\|\|f\| \\
&\le \sigma\|e_A\|\|x\| + (\|v_B\| + \|e_B\|)\|Ex\| + \|e_B\|(\sigma\|B^{-1}\|\|Ax\| + \|Ex\| + \sqrt{n}) \\
&\le \sigma\gamma m + (\sqrt{n} + \gamma\sqrt{m})(\sqrt{mn}/p) + \gamma\sqrt{m}(\sigma\cdot\sqrt{n}\cdot\|Ax\| + \sqrt{mn}/p + \sqrt{n}).
\end{aligned}$$

As Ax is a random ± 1 sum of vectors of norm at most $2\sqrt{n}$, its expected squared norm is mn, so its norm is at most $3\sqrt{mn}$ with probability at least $5/6$. Since $\gamma < 1/Cmn^{3/2}\log^{1/2}(n/\epsilon)$, $p > Cn\sqrt{m}$, and so $p > C\gamma m\sqrt{n}$, each term on the right hand side is less than $1/72$ (if C is sufficiently large). Then the left hand side is less than $1/24$ and D accepts y. ◁

4 Well-conditioned submatrices of random matrices

We now present and analyze the simple greedy algorithm used in step 1 in Reduction 1.

▶ **Proposition 9.** *Let $B \in \mathbb{R}^{m\times n}$ be a random Gaussian matrix with $m > Cn$. The probability that B contains a square submatrix with smallest singular value at least $1/\sqrt{n}$ is $1 - \exp(-\Omega(m))$. Moreover this submatrix can be found efficiently.*

Think of the column vectors of B as a stream of random normal vector samples. The matrix A is constructed incrementally column by column, starting with the empty matrix. After $k-1$ columns of A have been chosen, the next sample from the stream is considered as a candidate for the k-th column. It is rejected unless

$$\rho = \sum_{i=1}^{k} \frac{1}{\sigma_k^2} \le \frac{k}{n-k+1}, \tag{5}$$

where $\sigma_1, \ldots, \sigma_k$ are the singular values of A.

Once all n columns of A have been chosen, (5) guarantees that the sum of inverse squares of the singular values is at most n, so the smallest singular value will be at least $\sqrt{n}$ as desired. It remains to argue that no more than $m-n$ rejections happen except with probability $\exp(-\Omega(m))$.

Evolution of ρ

We analyze the evolution of ρ as columns are being added to A. Let A_k be any non-singular $n \times k$ matrix. Then

$$\rho(A_k) = \frac{\sum_{i=1}^{k}\prod_{j\ne i}\sigma_j^2}{\prod_{i=1}^{k}\sigma_i^2} = -\frac{\chi_k'(0)}{\chi_k(0)},$$

where $\chi_k(\lambda) = \det(A_k^\top A_k - \lambda I)$. Given A_k, let A_{k+1} be the random matrix obtained by appending a random normal column x to A_k.

Let $L \in \mathbb{R}^{k\times k}$ be an orthogonal matrix such that $L^\top A_k^\top A_k L = \operatorname{diag}(\sigma_1^2, \ldots, \sigma_k^2)$. It can be obtained from the singular value decomposition of A_k. The matrix $L' \in \mathbb{R}^{(k+1)\times(k+1)}$ given by $L' = \operatorname{diag}(L, 1)$ is also orthogonal and

$$A_{k+1}L' = \begin{bmatrix} A_k & x \end{bmatrix} \cdot \begin{bmatrix} L & \\ & 1 \end{bmatrix} = \begin{bmatrix} A_k L & x \end{bmatrix}$$

Since the columns of $A_k L$ are orthogonal of length $\sigma_1, \ldots, \sigma_k$, the columns of

$$A_k L \operatorname{diag}(\sigma_1^{-1}, \ldots, \sigma_k^{-1})$$

can be completed to an orthonormal basis C. The change of variables

$$y^\top = x^\top C$$

is then an isometry, so $y_1, \ldots, y_n$ are independent standard normals, and $\|y\| = \|x\|$. Then

$$L'^\top A_{k+1}^\top A_{k+1} L' = \begin{bmatrix} \sigma_1^2 & & & & \sigma_1 y_1 \\ & \sigma_2^2 & & & \sigma_2 y_2 \\ & & \ddots & & \vdots \\ & & & \sigma_k^2 & \sigma_k y_k \\ \sigma_1 y_1 & \sigma_2 y_2 & \ldots & \sigma_k y_k & \|y\|^2 \end{bmatrix}$$

Therefore

$$\begin{aligned} \chi_{k+1}(\lambda) &= \det(A_{k+1}^\top A_{k+1} - \lambda I) \\ &= \det(L'^\top A_{k+1}^\top A_{k+1} L' - \lambda I) \\ &= (\|y\|^2 - \lambda)\prod_{i=1}^{k}(\sigma_i^2 - \lambda) - \sum_{i=1}^{k}\sigma_i^2 y_i^2 \prod_{j\neq i}(\sigma_j^2 - \lambda) \\ &= \chi_k(\lambda)\left(\|y\|^2 - \lambda - \sum_{i=1}^{k}\frac{\sigma_i^2 y_i^2}{\sigma_i^2 - \lambda}\right). \end{aligned}$$

We obtain the following recurrences:

$$\chi_{k+1}(0) = \chi_k(0)\|y^{\perp k}\|^2$$

$$\chi'_{k+1}(0) = \chi'_k(0)\|y^{\perp k}\|^2 - \chi_k(0)\left(1 + \sum_{i=1}^{k}\frac{y_i^2}{\sigma_i^2}\right),$$

where $y^{\perp k} = (y_{k+1}, \ldots, y_n)$.

▷ Claim 10. If $(n-k+1)\chi'_k(0) + k\chi_k(0) \geq 0$ then

$$\mathrm{E}\big[(n-k)\chi'_{k+1}(0) + (k+1)\chi_{k+1}(0)\big|A_k\big] \geq 0.$$

The claim follows from linearity of expectation using the facts $\mathrm{E}[y_i^2] = 1$ and $\mathrm{E}\|y^{\perp k}\|^2 = k$.

Proof of Proposition 9. We show that the number of samples required for each column of A is dominated by a geometric random variable whose success probability is some absolute constant $p_\star$. The expected number of samples required is then at most $n/p_\star$. By large deviation bounds for geometric random variables [14] the probability that more than Cn samples are required is then at most $\exp(-\Omega(Cnp_\star))$, assuming $C > 1/p_\star$.

For the first column of A to fulfill (5) its squared norm needs to be at least n. This is at least $p_\star$ by Corollary 12 (with $a_1 = \cdots = a_n = 1$ and $b = 0$).

Now suppose (5) holds after the k-th column was added. Fix A_k and let X be the random variable $(n-k)\chi'_{k+1}(0) + (k+1)\chi_{k+1}(0)$. By Claim 10 $\mathrm{E}[X] \geq 0$. The random variable X is of the form in Corollary 12 so $\Pr(X > \mathrm{E}[X]) \geq p_\star$. Once a column x has been picked so that $X \geq 0$, the invariant (5) will hold for the matrix $A_{k+1} = [A_k\ x]$. ◀

4.1 Anticoncentration

The concentration Q of a real-valued random variable X is $Q(X,h) = \sup_x \Pr(x \leq X \leq x+h)$.

▶ **Proposition 11.** *There exists an absolute constant C such that if $X_1, \ldots, X_n$ are independent mean zero, unit variance random variables such that $Q(X_i, h) \leq 3/4$ for all i and some $h \leq 1/4C$ then*

$$\Pr\big(a_1X_1 + \cdots + a_nX_n > 0\big) \geq \frac{h^2}{32 + 4h^2}.$$

for all $a_1, \ldots, a_n$.

▶ **Corollary 12.** *There is an absolute constant $p_\star$ so that for every n and $a_1, \ldots, a_n, b$,*

$$\Pr\big(a_1Z_1^2 + \cdots + a_nZ_n^2 + b \geq \mu\big) \geq p_\star,$$

where $Z_1, \ldots, Z_n$ are independent normals and $\mu = a_1 + \cdots + a_n + b$.

Proof. Apply Proposition 11 to the random variables $Y_i = (X_i^2 - 1)/\sqrt{2}$ which have mean zero and unit variance. The condition $Q(Y_i, h) \leq 3/4$ is satisfied for all $h \leq 0.2$. ◀

Proof of Proposition 11. Let $X = a_1X_1 + \cdots + a_nX_n$. We may assume X has unit variance. By Rogozin's inequality [25],

$$Q(X,H) \leq CH\Big(\sum a_i^2(1 - Q(a_iX_i, a_ih))\Big)^{-1/2} = 2CH \leq 2Ch,$$

where $H = h\max_i|a_i| \leq h$. Applying Claim 13 we get

$$\Pr[X > 0] \geq \frac{1}{t+h}(h(1-2Ch) - 2/t) = \frac{h/2 - 2/t}{t+h}.$$

Choosing $t = 8/h$ we get $\Pr(X > 0) \geq h^2/(32 + 4h^2)$. ◀

▷ Claim 13. For every zero-mean, unit-variance X, every $\lambda > 0$, and every $t \geq 1$

$$\Pr[X > 0] \geq \frac{1}{t+h}\big(h \cdot \Pr(-h < X \leq 0) - 2/t\big).$$

Proof. Let $p = \Pr(X \in (0,t])$ and $q = \Pr(X \in (-h, 0])$. Then

$$\begin{aligned}\mathrm{E}[X] &\leq -h\Pr(X \leq -h) + 0\Pr(-h < X \leq 0) + t\Pr(0 < X \leq t) + \mathrm{E}[X1(X > t)] \\ &\leq -h \cdot (1 - q - p) + t \cdot p + \mathrm{E}[X1(X > t)].\end{aligned}$$

As $\mathrm{E}[X] = 0$,

$$p \geq \frac{1}{t+h}\big(h(1-q) - \mathrm{E}[X1(X > t)]\big).$$

By Claim 14, $\mathrm{E}[X1(X > t)] \leq \mathrm{E}[|X|1(|X| > t)] \leq 2/t$. ◁

▷ Claim 14. For every zero-mean, unit-variance X and every $t \geq 1$,

$$\mathrm{E}\big[|X|1(|X| > t)\big] \leq 2/t.$$

Proof.

$$\begin{aligned}
\mathrm{E}\big[|X|1(|X| > t)\big] &= \int_0^\infty \Pr(|X|1(|X| > t) > x)dx \\
&= \int_0^t \Pr(|X| > t)dx + \int_t^\infty \Pr(|X| > x)dx \\
&\leq \int_0^t (1/t^2)dx + \int_t^\infty (1/x^2)dx \\
&= 2/t.
\end{aligned}$$

The inequality is Chebyshev's. ◁

5 Refutation via Boolean combinations

Theorem 1 was proved by adapting the Ajtai-Dwork encryption scheme into a refutation algorithm for NBV. In this Section we carry out an analogous analysis for the "pancake encryption" of Bogdanov, Cueto Noval, Hoffmann, and Rosen (BCHR).

Their public key is also computationally indistinguishable from a random subspace of $\mathbb{R}^m$. The dimension of this subspace is, however, only $o(\sqrt{m})$. As a consequence, the resulting refutation only applies to a regime of NBV that is efficiently tractable.

While BCHR becomes insecure when $n \gg \sqrt{m}$, we believe that a modification of it may be secure up to $n = m^{1-o(1)}$. The advantage of the BCHR-based reduction over Theorem 1 is that it applies to larger completeness error γ.

▶ **Theorem 15.** *For every constant ϵ there exists a constant C such that for all but an ϵ-fraction of instances, average-case NBV with parameters $m = C(n \log n)^2$ and $\gamma = 1/C\sqrt{m}$ is in SZK.*

Let Z be a normal random variable and let $\zeta_1 < \cdots < \zeta_r$ be the unique numbers such that $\Pr(Z \leq \zeta_i) = (2i+1)/2r$. The Gaussian rounding $\mathrm{round}_r \colon \mathbb{R} \to \{\zeta_1, \ldots, \zeta_r\}$ is the function $\mathrm{round}_r(z) = \zeta_i$ where i is the unique index for which $\lceil r \cdot \Pr(Z \leq z) \rceil = \lceil r \cdot \Pr(Z \leq \zeta_i) \rceil$ (see Figure 1). For $z \in \mathbb{R}^n$ let $\mathrm{round}_r \colon \mathbb{R}^n \to \{\zeta_1, \ldots, \zeta_r\}^n$ be given by $\mathrm{round}_r(z) = (\mathrm{round}_r(z_1), \ldots, \mathrm{round}_r(z_n))$. Ser $r = \max\{Cm, Cn^2/\gamma^2\}$.

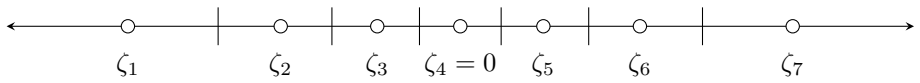


Figure 1 The function round_r for $r = 7$. All intervals have equal Gaussian measure. The values in the i-th interval round to ζ_i.

Reduction 2: On input $A \in \mathbb{R}^{m \times n}$,

1 If the largest singular value of A is more than $2\sqrt{m}$, fail.
2 If the smallest singular value of A is less than $\sqrt{m}/4$, fail.
3 Otherwise, output the sampler S that maps $x \sim \{\pm 1/\sqrt{m}\}^m$ to $\mathrm{round}_r(Ax)$.

Theorem 15 follows from Claims 16 and 19.

▷ Claim 16. For every ϵ there is a C so that for a $1-\epsilon$ fraction of instances $A \in \mathbb{R}^{m\times n}$, where $m = (Cn\log n)^2$, the output of S is 2/3-close to random.

▷ Fact 17. [4] The distribution $(A, \mathrm{round}_r(Ax))$ is $\sqrt{4en\ln r/\sqrt{m}}$-close to (A,ζ), where ζ is uniform over rounded values and independent of A.

▷ Fact 18. [6] Assume $m > 2n$. The largest and smallest singular values of A is at most $2\sqrt{n}$ and at least $\sqrt{n}/4$, except with probability $\exp(-\Omega(n))$.

Proof of Claim 16. By Fact 17, the joint distribution of A and the output of the sampler is $O(C^{-1/2})$-close to uniform. Therefore for all but $O(C^{-1/2})$ choices of A the output is 2/3-close to uniform. By a Chernoff bound and Fact 18 at most $2^{-\Omega(m)}$ other inputs A cause the reduction to fail. ◁

▷ Claim 19. If A is a yes instance of NBV with $\gamma < 1/C\sqrt{m}$, either Reduction 2 fails, or the output of S is 2/3-far from random.

▷ Fact 20. [4] For sufficiently large r, $\mathrm{round}_r(z)$, $z\in\mathbb{R}$ is $r^{-1/2}$-close to z unless $|z| > t$ for t such that $\Pr(|Z| > t) \le 3(r\ln r)^{-1/2}$, where Z is normal in $\mathbb{R}$.

Proof of Claim 19. Let $w\in\mathbb{R}^n$ be the witness for which $wA = v + e$ where $v\in\{\pm1\}^m$ and $\|e\| \le \gamma\sqrt{m}$. Let D be the distinguisher that, given ζ, accepts if $|\{\sqrt{m}\langle w,\zeta\rangle\}| \le 1/48$.

Assuming the reduction did not fail, by the assumption on singular values,

$$\frac{1}{4} \le \frac{\|v\| - \|e\|}{2\sqrt{m}} \le \|w\| \le \frac{\|v\| + \|e\|}{\sqrt{m}/4} \le 8.$$

If ζ is random, we argue that D rejects with probability at least 5/6. we can write $\zeta = \mathrm{round}_r(g)$ for a normal $g\in\mathbb{R}^n$. Let $e = \mathrm{round}_r(g) - g$. Then $\sqrt{m}\langle w,\zeta\rangle = \sqrt{m}\langle w,g\rangle + \sqrt{m}\langle w,e\rangle$. The random variable $\sqrt{m}\langle w,g\rangle$ is a univariate normal with standard deviation at least $\sqrt{m}\|w\| \ge \sqrt{m}/4$. By Fact 6, $\{\sqrt{m}\langle w,g\rangle\}$ is $2^{-\Omega(m)} < 1/24$ close to uniform in $[-1/2, 1/2)$. In particular, $|\{\sqrt{m}\langle w,g\rangle\}| > 1/24$ except with probability $11/12 - 1/24$. By Fact 20, $\|e\|_\infty \le r^{-1/2}$ except with probability $3n(r\ln r)^{-1/2} < 1/24$. Both events happen with probability at least 5/6. Assuming this,

$$|\{\sqrt{m}\langle w,\zeta\rangle\}| > 1/24 - |\{\sqrt{m}\langle w,e\rangle\}| \ge 1/24 - \sqrt{m}\|w\|\|e\| > 1/48$$

because $\sqrt{m}\|w\|\|e\| \le 8\sqrt{m}r^{-1/2}$ and D rejects.

If ζ is the output of the sampler we argue that the distinguisher accepts it with probability at least 8/9:

$$|\{\sqrt{m}\langle w,Ax\rangle\}| = |\{\sqrt{m}\langle v+e,x\rangle\}| = |\{\sqrt{m}\langle v,x\rangle + \sqrt{m}\langle e,x\rangle\}| = \sqrt{m}|\langle e,x\rangle| \tag{6}$$

because v and $\sqrt{m}x$ are integral. As x is random, $\mathrm{E}[\langle e,x\rangle^2] = \|e\|^2/m$. By Markov's inequality, $|\langle e,x\rangle| \le 3\|e\|/\sqrt{m}$ except with probability 1/9. If this holds (6) is at most $3\|e\| \le 3\gamma\sqrt{m}$.

As the largest singular value of A is at most $2\sqrt{m}$, all entries of Ax are between -2 and 2. By Fact 20, $\|\mathrm{round}_r(Ax) - Ax\|_\infty \le nr^{-1/2}$. Therefore

$$|\{\sqrt{m}\langle w, \mathrm{round}_r(Ax) - Ax\rangle\}| \le \sqrt{m}\|w\|\|\mathrm{round}_r(Ax) - Ax\| \le 8\sqrt{m}nr^{-1/2} \le \gamma\sqrt{m}.$$

Together with (6), $|\{\sqrt{m}\langle w,Ax\rangle\}| \le 4\gamma\sqrt{m} \le 1/48$. ◁

References

1. Dorit Aharonov and Oded Regev. Lattice problems in NP ∩ coNP. *J. ACM*, 52(5):749–765, September 2005. doi:10.1145/1089023.1089025.
2. Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, 1997.
3. Boaz Barak, Samuel Hopkins, Jonathan Kelner, Pravesh K. Kothari, Ankur Moitra, and Aaron Potechin. A nearly tight sum-of-squares lower bound for the planted clique problem. *SIAM Journal on Computing*, 48(2):687–735, 2019. doi:10.1137/17M1138236.
4. Andrej Bogdanov, Miguel Cueto Noval, Charlotte Hoffmann, and Alon Rosen. Public-key encryption from homogeneous clwe. Cryptology ePrint Archive, Paper 2022/093, 2022. URL: https://eprint.iacr.org/2022/093.
5. Matthew S. Brennan and Guy Bresler. Reducibility and statistical-computational gaps from secret leakage. In *Conference on Learning Theory, COLT 2020*, volume 125 of *Proceedings of Machine Learning Research*, pages 648–847. PMLR, 2020. URL: http://proceedings.mlr.press/v125/brennan20a.html.
6. Kenneth R. Davidson and Stanislaw J. Szarek. Chapter 8 - local operator theory, random matrices and banach spaces. In W.B. Johnson and J. Lindenstrauss, editors, *Handbook of the Geometry of Banach Spaces*, volume 1 of *Handbook of the Geometry of Banach Spaces*, pages 317–366. Elsevier Science B.V., 2001. doi:10.1016/S1874-5849(01)80010-3.
7. U. Feige, J. H. Kim, and E. Ofek. Witnesses for non-satisfiability of dense random 3CNF formulas. In *47th Annual IEEE Symposium on Foundations of Computer Science*, 2006. doi:10.1109/FOCS.2006.78.
8. Mrinalkanti Ghosh, Fernando Granha Jeronimo, Chris Jones, Aaron Potechin, and Goutham Rajendran. Sum-of-squares lower bounds for sherrington-kirkpatrick via planted affine planes. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 954–965. IEEE, 2020. doi:10.1109/FOCS46700.2020.00093.
9. Oded Goldreich. *Average Case Complexity, Revisited*, pages 422–450. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-22670-0_29.
10. Oded Goldreich, Amit Sahai, and Salil Vadhan. Can statistical zero knowledge be made non-interactive? or on the relationship of szk and niszk. In Michael Wiener, editor, *Advances in Cryptology – CRYPTO' 99*, pages 467–484, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.
11. Dima Grigoriev. Linear lower bound on degrees of positivstellensatz calculus proofs for the parity. *Theor. Comput. Sci.*, 259(1):613–622, May 2001.
12. Venkatesan Guruswami, Pravesh K. Kothari, and Peter Manohar. Algorithms and certificates for boolean CSP refutation: smoothed is no harder than random. In *54th Annual ACM SIGACT Symposium on Theory of Computing*, 2022. doi:10.1145/3519935.3519955.
13. R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, pages 12–24, New York, NY, USA, 1989. Association for Computing Machinery. doi:10.1145/73007.73009.
14. Svante Janson. Tail bounds for sums of geometric and exponential variables. *Statistics & Probability Letters*, 135:1–6, 2018. doi:10.1016/j.spl.2017.11.017.
15. H.W. Jr. Lenstra, A.K. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982. URL: http://eudml.org/doc/182903.
16. Leonid A. Levin. Average case complete problems. *SIAM J. Comput.*, 15(1):285–286, 1986. doi:10.1137/0215020.
17. Vadim Lyubashevsky and Daniele Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*, pages 577–594. Springer, 2009. doi:10.1007/978-3-642-03356-8_34.

18 Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007. doi:10.1137/S0097539705447360.

19 Sidhanth Mohanty, Prasad Raghavendra, and Jeff Xu. Lifting sum-of-squares lower bounds: Degree-2 to degree-4. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 840–853, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3357713.3384319.

20 Andrea Montanari. Optimization of the Sherrington-Kirkpatrick Hamiltonian. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, 2019. doi:10.1109/FOCS.2019.00087.

21 G. Parisi. Infinite number of order parameters for spin-glasses. *Phys. Rev. Lett.*, 43:1754–1756, December 1979. doi:10.1103/PhysRevLett.43.1754.

22 Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: Extended abstract. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, pages 333–342, New York, NY, USA, 2009. Association for Computing Machinery. doi:10.1145/1536414.1536461.

23 Aaron Potechin, Paxton Turner, Prayaag Venkat, and Alexander S. Wein. Near-optimal fitting of ellipsoids to random points, 2022. doi:10.48550/ARXIV.2208.09493.

24 Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), September 2009. doi:10.1145/1568318.1568324.

25 B. A. Rogozin. On the increase of dispersion of sums of independent random variables. *Theory of Probability & Its Applications*, 6(1):97–99, 1961. doi:10.1137/1106010.

26 Amit Sahai and Salil Vadhan. A complete problem for statistical zero knowledge. *J. ACM*, 50(2):196–249, March 2003. doi:10.1145/636865.636868.

27 Grant Schoenebeck. Linear level lasserre lower bounds for certain k-csps. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '08, pages 593–602, USA, 2008. IEEE Computer Society. doi:10.1109/FOCS.2008.74.

28 Michel Talagrand. *The Parisi Formula*, pages 349–474. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-22253-5_7.

29 Luca Trevisan. The program-enumeration bottleneck in average-case complexity theory. In *2010 IEEE 25th Annual Conference on Computational Complexity*, pages 88–95, 2010. doi:10.1109/CCC.2010.18.

A 4/3 Approximation for 2-Vertex-Connectivity

Miguel Bosch-Calvo
IDSIA, USI-SUPSI, Lugano, Switzerland

Fabrizio Grandoni
IDSIA, USI-SUPSI, Lugano, Switzerland

Afrouz Jabal Ameli
TU Eindhoven, The Netherlands

Abstract

The 2-Vertex-Connected Spanning Subgraph problem (2VCSS) is among the most basic NP-hard (Survivable) Network Design problems: we are given an (unweighted) undirected graph G. Our goal is to find a subgraph S of G with the minimum number of edges which is 2-vertex-connected, namely S remains connected after the deletion of an arbitrary node. 2VCSS is well-studied in terms of approximation algorithms, and the current best (polynomial-time) approximation factor is 10/7 by Heeger and Vygen [SIDMA'17] (improving on earlier results by Khuller and Vishkin [STOC'92] and Garg, Vempala and Singla [SODA'93]).

Here we present an improved 4/3 approximation. Our main technical ingredient is an approximation preserving reduction to a conveniently structured subset of instances which are "almost" 3-vertex-connected. The latter reduction might be helpful in future work.

2012 ACM Subject Classification Theory of computation → Routing and network design problems

Keywords and phrases Algorithm, Network Design, Vertex-Connectivity, Approximation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.29

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2305.02240` [3]

Funding The first 2 authors are partially supported by the SNF Grant 200021_200731 / 1.

1 Introduction

Real-world networks are prone to failures. For this reason it is important to design them so that they are still able to support a given traffic despite a few (typically temporary) failures of nodes or edges. The basic goal of survivable network design is to construct cheap networks which are resilient to such failures.

Most natural survivable network design problems are NP-hard, and a lot of work was dedicated to the design of approximation algorithms for them. One of the most basic survivable network design problems is the 2-Vertex-Connected Spanning Subgraph problem (2VCSS). Recall that an (undirected) graph $G = (V, E)$ is k-vertex-connected (kVC) if, after removing any subset W of at most $k - 1$ nodes (with all the edges incident to them), the residual graph $G[V \setminus W]$ is connected. In particular, in a 2VC graph G we can remove any single node while maintaining the connectivity of the remaining nodes (intuitively, we can tolerate a single node failure). In 2VCSS we are given a 2VC (unweighted) undirected graph $G = (V, E)$, and our goal is to compute a minimum cardinality subset of edges $S \subseteq E$ such that the (spanning) subgraph (V, S) is 2VC.

2VCSS is NP-hard: indeed an n-node graph G admits a Hamiltonian cycle iff it contains a 2VC spanning subgraph with n edges. Czumaj and Lingas [13] proved that the problem is APX-hard, hence most likely it does not admit a PTAS. A 2-approximation for 2VCSS can be obtained in different ways. For example one can compute an (open) ear decomposition of

the input graph and remove the trivial ears (containing a single edge). The resulting graph is 2VC and contains at most $2(n-1)$ edges (while the optimum solution must contain at least n edges). The first non-trivial 5/3 approximation was obtained by Khuller and Vishkin [28]. This was improved to 3/2 by Garg, Vempala and Singla [20] (see also an alternative 3/2 approximation by Cheriyan and Thurimella [11]). Finally Heeger and Vygen [24] presented the current-best 10/7 approximation[1]. Our main result is as follows (please see Section 2 for an overview of our approach):

▶ **Theorem 1.** *There is a polynomial-time $\frac{4}{3}$-approximation algorithm for 2VCSS.*

1.1 Related Work

An undirected graph G is k-edge-connected (kEC) if it remains connected after removing up to $k-1$ edges. The 2-Edge-Connected Spanning Subgraph problem (2ECSS) is the natural edge-connectivity variant of 2VCSS, where the goal is to compute a 2EC spanning subgraph with the minimum number of edges. Like 2VCSS, 2ECSS does not admit a PTAS unless $P = NP$ [13]. It is not hard to compute a 2 approximation for 2ECSS. For example it is sufficient to compute a DFS tree and augment it greedily. Khuller and Vishkin [27] found the first non-trivial 3/2-approximation algorithm. Cheriyan, Sebö and Szigeti [10] improved the approximation factor to 17/12. This was further improved to 4/3 in two independent and drastically different works by Hunkenschröder, Vempala and Vetta [25] and Sebö and Vygen [34]. The current best and very recent $\frac{118}{89} + \varepsilon < 1.326$ approximation is due to Garg, Grandoni and Jabal Ameli [19]. Our work exploits several ideas from the latter paper. The k-Edge Connected Spanning Subgraph problem (kECSS) is the natural generalization of 2ECSS to any connectivity $k \geq 2$ (see, e.g., [11, 17]).

A major open problem in the area is to find a better than 2 approximation for the weighted version of 2ECSS. This is known for the special case with 0-1 edge weights, a.k.a. the Forest Augmentation problem, by the recent work by Grandoni, Jabal-Ameli and Traub [21] (see also [2, 7, 6] for the related Matching Augmentation problem).

A problem related to kECSS is the k-Connectivity Augmentation problem (kCAP): given a k-edge-connected undirected graph G and a collection of extra edges L (*links*), find a minimum cardinality subset of links L' whose addition to G makes it $(k+1)$-edge-connected. It is known [14] that kCAP can be reduced to the case $k = 1$, a.k.a. the Tree Augmentation problem (TAP), for odd k and to the case $k = 2$, a.k.a. the Cactus Augmentation problem (CacAP), for even k. Several approximation algorithms better than 2 are known for TAP [1, 8, 9, 15, 16, 22, 29, 30, 31], culminating with the current best 1.393 approximation by Cecchetto, Traub and Zenklusen [5]. Till recently no better than 2 approximation was known for CacAP (excluding the special case where the cactus is a single cycle [18]): the first such algorithm was described by Byrka, Grandoni and Jabal Ameli [4], and later improved to 1.393 in [5]. In a recent breakthrough by Traub and Zenklusen, a better than 2 (namely 1.694) aproximation for the weighted version of TAP was achieved [35] (later improved to $1.5 + \varepsilon$ in [36]). Partial results in this direction where achieved earlier in [1, 12, 16, 22, 32].

[1] Before [24] a few other papers claimed even better approximation ratios [23, 26], however they have been shown to be buggy or incomplete, see the discussion in [24].

1.2 Preliminaries

We use standard graph notation. For a graph $G = (V, E)$, we let $V(G) = V$ and $E(G) = E$ denote its nodes and edges, resp. For $W \subseteq V$ and $F \subseteq E$, we use the shortcuts $G \setminus F := (V, E \setminus F)$ and $G \setminus W := G[V \setminus W]$. For a subgraph G', a node v and an edge e, we also use the shortcuts $v \in G'$ and $e \in G'$ meaning $v \in V(G')$ and $e \in E(G')$, resp. Throughout this paper we sometimes use interchangeably a subset of edges F and the corresponding subgraph (W, F), $W = \{v \in V : v \in f \in F\}$. The meaning will be clear from the context. For example, we might say that $F \subseteq E$ is 2VC or that F contains a connected component. In particular, we might say that $S \subseteq E$ is a 2VC spanning subgraph. Also, given two subgraphs G_1 and G_2, by $G' = G_1 \cup G_2$ we mean that G' is the subgraph induced by $E(G_1) \cup E(G_2)$. We sometimes represent paths and cycles as sequence of nodes. A k-vertex-cut of G is a subset W of k nodes such that $G[V \setminus W]$ has at least 2 connected components. A node defining a 1-vertex-cut is a cut vertex.

By $\mathsf{OPT}(G) \subseteq E(G)$ we denote an optimum solution to a 2VCSS instance G, and let $\mathsf{opt}(G) := |\mathsf{OPT}(G)|$ be its size. All the algorithms described in this paper are deterministic.

The proofs that are omitted here due to space constraints will appear in the journal version of the paper (see also [3]).

2 Overview of Our Approach

In this section we sketch the proof of our 4/3-approximation (Theorem 1). The details and proofs which are omitted here will be given in the following technical sections.

Our result relies on 3 main ingredients. The first one is an approximation-preserving (up to a small additive term) reduction of 2VCSS to instances of the same problem on properly *structured* graphs, which are "almost" 3VC in a sense described later (see Section 2.1).

At this point we compute a minimum-size 2-edge-cover H similarly to prior work: this provides a lower bound on the size of the optimal solution. For technical reasons, we transform H into a *canonical* form, without increasing its size (see Section 2.2).

The final step is to convert H into a feasible solution S. Starting from $S = H$, this is done by iteratively adding edges to and removing edges from S in a careful manner. In order to take the size of S under control, we assign 1/3 credits to each edge of the initial S, and use these credits to pay for any increase in the number of edges of S (see Section 2.3). We next describe the above ingredients in more detail.

2.1 A Reduction to Structured Graphs

Our first step is an approximation-preserving (up to a small additive factor) reduction of 2VCSS to instances of the same problem on properly *structured* graphs. This is similar in spirit to an analogous reduction for 2ECSS in [19]. In particular we exploit the notion of irrelevant edges and isolating cuts defined in that paper. We believe that our reduction might be helpful also in future work.

In more detail, we can get rid of the following *irrelevant* edges.

▶ **Lemma 2** (irrelevant edge). *Given a 2VC graph G, let $e = uv \in E(G)$ be such that $\{u, v\}$ is a 2-vertex-cut (we call e* irrelevant*). Then every optimal 2VCSS solution for G does not contain e.*

Proof. We will need the following observation:

▶ **Fact 3.** *Suppose that a minimal solution S to 2VCSS on a graph G contains a cycle C. Then S does not contain any chord f of C. Indeed, otherwise consider any open ear decomposition*[2] *of S which uses C as a first ear. Then f would be a trivial ear (consisting of a single edge) of the decomposition, and thus $S \setminus \{f\}$ would also be 2VC, contradicting the minimality of S.*

Let $H \subseteq E$ be any optimal (hence minimal) solution to 2VCSS on G. Assume by contradiction that H contains an irrelevant edge $e = uv$. Removing u and v splits H into different connected components $C_1, \ldots, C_k$, with $k \geq 2$. Each one of those components has edges u_iu, v_iv in H, where $u_i, v_i \in C_i$ for $i \in \{1, \ldots, k\}$, otherwise H would contain a cut vertex. Let P_1 be a path from u_1 to v_1 in C_1, and P_2 be a path from v_2 to u_2 in C_2. Then e is a chord of the cycle $P_1 \cup P_2 \cup \{uu_1, v_1v, vv_2, u_2u\}$, contradicting the minimality of H by Fact 3. ◀

We can enforce (see later) that our graph G is "almost" 3VC, in the sense that the only 2-vertex-cuts of G are a very specific type of *isolating* cuts defined as follows.

▶ **Definition 4** (isolating cut)**.** *Given a 2-vertex-cut $\{u, v\}$ of a graph G, we say that this cut is* isolating *if $G \setminus \{u, v\}$ has exactly two connected components, one of which consisting of* 1 *node. Otherwise the cut is* non-isolating*.*

Assuming that there are no non-isolating cuts, we can avoid the following local configuration: this will be helpful in the rest of our analysis.

▶ **Definition 5** (removable 5-cycle)**.** *We say that a 5-cycle C of a 2VC graph G is* removable *if it has at least two vertices of degree 2 in G.*

▶ **Lemma 6.** *Given a 2VC graph G without non-isolating cuts and with at least* 6 *nodes. Let C be a removable 5-cycle of G. Then in polynomial time one can find an edge e of C such that there exists an optimum solution to 2VCSS on G not containing e (we say that e is a* removable *edge).*

Proof. Assume $C = v_1v_2v_3v_4v_5$. If C has two vertices of degree 2 that are adjacent in C, namely v_1 and v_2, then $\{v_3, v_5\}$ is a non-isolating cut of G, a contradiction. Thus we can assume that C has exactly two non-adjacent vertices of degree 2, say v_1 and v_3 w.l.o.g.

We will show that the edge $e = v_4v_5$ is the desired removable edge. Let H be an optimal 2VCSS solution for G that uses the edge v_4v_5. Observe that in this case since v_1 and v_3 have degree 2, then H must contain all the edges of C.

To complete the argument we show that there exists an edge $f \in E(G) \setminus E(H)$, such that v_4v_5 is a chord of a cycle in $H' := H \cup \{f\}$: hence we can remove v_4v_5 from H' using Fact 3 to obtain an alternative optimum solution not containing v_4v_5.

Let $H'' = H \setminus \{v_4v_5\}$. There is no cycle C' in H'' that contains both v_4 and v_5, otherwise v_4v_5 is a chord of C' in H, contradicting the minimality of H by Fact 3. Therefore if we remove v_2 from H'', there must be no paths from v_4 to v_5. This means that there is a partition of $V(G) \setminus \{v_2\}$ into non-empty sets V_1 and V_2 such that, $\{v_3, v_4\} \in V_1$, $\{v_1, v_5\} \in V_2$ and there is no edge in H'' between V_1 and V_2. Since $|V(G)| \geq 6$, then we can assume w.l.o.g that $|V_1| \geq 3$.

[2] An ear-decomposition of an undirected graph G is a sequence of paths or cycles $P_0, \ldots, P_k$ (ears) spanning $E(G)$ such that P_0 is a cycle and P_i, $i \geq 1$, has its internal nodes disjoint from $V_{i-1} := V(P_0) \cup \ldots \cup V(P_{i-1})$ and its endpoints (or one node if P_i is a cycle) in V_{i-1}. We say that an ear-decomposition is open if P_i is a path, for $i \geq 1$. Every 2VC graph admits an open ear decomposition [33, Chapter 15].

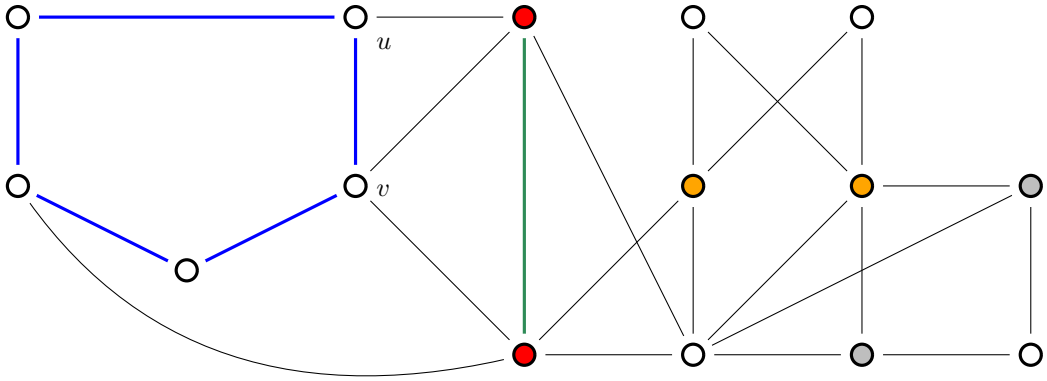


Figure 1 The cycle induced by the blue edges is a removable cycle, since it has two vertices of degree 2 in G. The edge uv is removable. The red and orange (resp. gray) pairs of vertices form a non-isolating (resp. isolating) cut. The green edge is irrelevant.

There must be an edge $f = u_1u_2 \in E(G)$ such that $u_1 \in V_1 \setminus \{v_3, v_4\}$ and $u_2 \in V_2$, otherwise $\{v_2, v_4\}$ is a non-isolating cut in G, a contradiction. Now we show that f is the desired edge. We claim that there exists a path P_1 in $H[V_1 \setminus \{v_3\}]$ between u_1 and v_4. Since H is 2VC, there exists a path P_1 between u_1 and v_4 not using v_2. Such path does not use v_3 either since this node is adjacent only to v_2 and v_4, and $u_1 \notin \{v_3, v_4\}$. If P_1 is not contained in $H[V_1]$, it would need to use at least two edges between V_1 and V_2 in H, however we argued before that H contains only one such edge, namely v_4v_5. Symmetrically, we claim that there exists a path P_2 in $H[V_2 \setminus \{v_1\}]$ between u_2 and v_5. Notice that $u_2 = v_5$ is possible, in which case the claim trivially holds. Hence next assume $u_2 \neq v_5$. Observe that $u_2 \neq v_1$ since u_2 is adjacent to $u_1 \notin \{v_2, v_5\}$. Thus, the claim about P_2 follows symmetrically to the case of P_1. Altogether, v_4v_5 is a chord of the cycle $P_1 \cup P_2 \cup \{f\} \cup C \setminus \{v_4v_5\}$ in $H' = H \cup \{f\}$, which implies the lemma. ◀

We are now ready to define a structured graph and to state our reduction to such graphs.

▶ **Definition 7** (structured graph). *A 2VC graph G is structured if it does not contain: (1) Irrelevant edges; (2) Non-isolating cuts; (3) Removable 5-cycles.*

▶ **Lemma 8.** *Given a constant $1 < \alpha \leq \frac{3}{2}$, if there exists a polynomial-time algorithm for 2VCSS on a structured graph G that returns a solution of cost at most $\max\{\mathsf{opt}(G), \alpha \cdot \mathsf{opt}(G) - 2\}$, then there exists a polynomial-time α-approximation algorithm for 2VCSS.*

We remark that any $\alpha - \varepsilon$ approximation of 2VCSS on structured graphs, for an arbitrarily small constant $\varepsilon > 0$, immediately implies an algorithm of the type needed in the claim of Lemma 8: indeed, instances with $\mathsf{opt}(G) \leq \max\{\frac{2}{\varepsilon}, \frac{2}{\alpha-1}\}$ can be solved exactly in constant time by brute force.

The algorithm at the heart of our reduction is algorithm RED given in Algorithm 1. Lines 1-2 solve by brute force instances with few nodes. Lines 3-4, 5-10, and 11-12 get rid recursively of irrelevant edges, non-isolating vertex cuts and removable 5-cycles, resp. When Line 13 is reached, the graph is structured and therefore we can apply a black-box algorithm ALG for structured instances of 2VCSS.

It is easy to see that the algorithm runs in polynomial time.

Algorithm 1 Reduction from arbitrary to structured instances of 2VCSS. Here G is 2VC and ALG is an algorithm for structured instances that returns a solution of cost at most $\max\{\mathsf{opt}(G), \alpha \cdot \mathsf{opt}(G) - 2\}$ for some $1 < \alpha \leq \frac{3}{2}$.

1: **if** $|V(G)| < \max\{6, \frac{2}{\alpha-1}\}$ **then**
2: Compute OPT(G) by brute force (in constant time) and **return** OPT(G)
3: **if** G contains an irrelevant edge **then**
4: **return** RED$(G \setminus \{e\})$
5: **if** G contains a non-isolating vertex cut $\{u, v\}$ **then**
6: let $(V_1, V_2), 2 \leq |V_1| \leq |V_2|$, be a partition of $V(G) \setminus \{u, v\}$ such that there are no edges between V_1 and V_2 in $G \setminus \{u, v\}$
7: let G_1 be the graph resulting from G by contracting V_2 into one node v_2 and G_2 the graph resulting from G by contracting V_1 into one node v_1 (keeping one copy of parallel edges in both cases)
8: let $H_1 =$ RED(G_1) and $H_2 =$ RED(G_2)
9: let E_1 (resp. E_2) be the two edges of H_1 (resp., H_2) with endpoints in v_2 (resp., v_1)
10: **return** $H := (H_1 \setminus E_1) \cup (H_2 \setminus E_2)$
11: **if** G contains a removable 5-cycle **then**
12: let e be the removable edge (found via Lemma 6) in that cycle and **return** RED$(G \setminus \{e\})$
13: **return** ALG(G)

▶ **Lemma 9.** RED(G) *runs in polynomial time in* $|V(G)|$ *if* ALG *does so.*

Proof. Let $n = |V(G)|$. First observe that each recursive call, excluding the corresponding subcalls, can be executed in polynomial time. In particular, we can find one irrelevant edge, if any, in polynomial time by enumerating all the possible 2-vertex-cuts. Furthermore, we can find some removable 5-cycle, if any, in polynomial time by enumerating all 5-cycles. Then, by Lemma 6, we can indentify a removable edge in such cycle. We also remark that in Lines 4 and 12 we remove one edge, and we never increase the number of edges. Hence the corresponding recursive calls increase the overall running time by a polynomial factor altogether.

It is then sufficient to bound the number $f(n)$ of recursive calls where we execute Lines 6-10 starting from a graph with n nodes. Consider one recursive call on a graph G with n nodes, where the corresponding graph G_1 has $5 \leq k \leq n/2 + 2$ nodes. Notice that G_2 has $n - k + 4$ nodes. Thus one has $f(n) \leq \max_{5 \leq k \leq n/2+2}\{f(k) + f(n - k + 4)\}$, which implies that $f(n)$ is polynomially bounded. ◀

Let us next show that RED produces a feasible solution.

▶ **Lemma 10.** *Given a 2VC graph* G, RED(G) *returns a feasible 2VCSS solution for* G.

Proof. Let us prove the claim by induction on $(|V(G)|, |E(G)|)$ in lexicographic order. The base cases are given when RED(G) executes Lines 2 or 13: in these cases RED clearly returns a feasible solution. Consider an instance G where RED(G) does not execute those lines (in the root call), and assume the claim holds for any instance G' where $(|V(G')|, |E(G')|)$ is strictly smaller than $(|V(G)|, |E(G)|)$ in lexicographic order. By Lemma 2, when RED recurses at Line 4, the graph $G \setminus \{e\}$ is 2VC, hence the recursive call returns a 2VC spanning subgraph by inductive hypothesis. A similar argument holds when Line 12 is executed, this time exploiting Lemma 6.

It remains to consider the case when Lines 6-10 are executed. Notice that both G_1 and G_2 are 2VC. In this case we can assume by inductive hypothesis that both H_1 and H_2 are 2VC. Consider any $w_1 \in V_1$. Since H_1 is 2VC, H_1 contains 2 vertex disjoint paths from w_1 to v_2. Notice that both u and v must be the second last node in exactly one such path, hence in particular there exist two (internally) vertex-disjoint paths P_{w_1u} and P_{w_1v} in H over the nodes $V_1 \cup \{u, v\}$ from w_1 to u and v, resp. Symmetrically, for each $w_2 \in V_2$ there exist two vertex disjoint paths P_{w_2u} and P_{w_2v} in H over the nodes $V_2 \cup \{u, v\}$ from w_2 to u and v, resp.

For any $w_1 \in V_1$ and $w_2 \in V_2$, the w_1-w_2 paths $P_{w_1u} \cup P_{w_2u}$ and $P_{w_1v} \cup P_{w_2v}$ in H are vertex disjoint. Similarly, for any $w_1 \in V_1$ and $w_2 \in V_2$, $P_{w_1u} \cup P_{w_1v}$ and $P_{w_2u} \cup P_{w_2v}$ are vertex disjoint u-v paths in H. Given $w_1 \in V_1$ and $w_1' \in V_1 \cup \{u, v\}$, consider the two vertex disjoint paths in H_1 between them. If these paths do not contain v_2, then they also belong to H. Otherwise exactly one of those paths contains the subpath $P' = uv_2v$: by replacing P' with $P_{w_2u} \cup P_{w_2v}$ for an arbitrary $w_2 \in V_2$, one obtains two vertex disjoint w_1-w_1' paths in H. A symmetric argument holds for $w_2 \in V_2$ and $w_2' \in V_2 \cup \{u, v\}$.

Assume to get a contradiction that H has a cut vertex w. If $w \in \{u, v\}$, then w is also a cut vertex in either H_1 or H_2. Thus we can assume w.l.o.g. $w \in V_1$. Consider the components resulting of removing the vertex w from H. If one of this components does not contain u nor v then w is also a cut vertex in H_1. Thus removing w from H yields two connected components C_u, C_v, with $u \in C_u, v \in C_v$. But since $w \in V_1$, no edge from H_2 present in H is removed by deleting w. In particular, there is a path from u to v in H, contradicting the fact that w is a cut vertex. ◀

It remains to analyze the approximation factor of RED.

▶ **Lemma 11.** $|\mathsf{RED}(G)| \leq \begin{cases} \mathsf{opt}(G), & \text{if } |V(G)| < \max\{6, \frac{2}{\alpha-1}\}; \\ \alpha \cdot \mathsf{opt}(G) - 2, & \text{if } |V(G)| \geq \max\{6, \frac{2}{\alpha-1}\}. \end{cases}$

Proof. We prove the claim by induction on $(|V(G)|, |E(G)|)$ in lexicographic order. The base cases correspond to the execution of Lines 2 and 13. Here the claim trivially holds. The claim holds by inductive hypothesis and by Lemmas 2 and 6 when Lines 4 and 12, resp., are executed. Notice that the 6 that appears in the max in the claim of the lemma is meant to guarantee that the conditions of Lemma 6 are satisfied.

It remains to consider the case when Lines 6-10 are executed. Let OPT be a minimum 2VC spanning subgraph of G, and OPT_i be an optimal 2VCSS solution for G_i, $i \in \{1, 2\}$. We will later show

$$|\mathsf{OPT}| = |\mathsf{OPT}_1| + |\mathsf{OPT}_2| - 4. \tag{1}$$

Notice that since $|H_i \cap E_i| = 2$ for $i \in \{1, 2\}$ and $H_1 \setminus E_1$ and $H_2 \setminus E_2$ are edge-disjoint, we have $|H| = |H_1| + |H_2| - 4$.

Notice that, for $|V_i| \geq \frac{2}{\alpha-1}$, one has $|\mathsf{OPT}_i| \leq \alpha|\mathsf{OPT}_i| - 2$. We now distinguish a few cases.

If $|V_2| < \max\{6, \frac{2}{\alpha-1}\}$, then $|H| = |H_1| + |H_2| - 4 = |\mathsf{OPT}_1| + |\mathsf{OPT}_2| - 4 = |\mathsf{OPT}|$.

If $|V_1| \geq \max\{6, \frac{2}{\alpha-1}\}$, then $|H| = |H_1| + |H_2| - 4 \leq \alpha|\mathsf{OPT}_1| - 2 + \alpha|\mathsf{OPT}_2| - 2 - 4 \leq \alpha(|\mathrm{OPT}_1| + |\mathrm{OPT}_2|) - 8 \leq \alpha|\mathsf{OPT}| + 4\alpha - 8 \leq \alpha|\mathsf{OPT}| - 2$. The last inequality uses the assumption $\alpha \leq 3/2$.

Finally, if $|V_1| < \max\{6, \frac{2}{\alpha-1}\}$ and $|V_2| \geq \max\{6, \frac{2}{\alpha-1}\}$, we have $|H| = |H_1| + |H_2| - 4 \leq |\mathsf{OPT}_1| + \alpha|\mathsf{OPT}_2| - 2 - 4 = (1-\alpha)|\mathsf{OPT}_1| + \alpha(|\mathsf{OPT}_1| + |\mathsf{OPT}_2|) - 6 \leq (1-\alpha)|\mathsf{OPT}_1| + 4\alpha + \alpha|\mathsf{OPT}| - 6 \leq \alpha|\mathsf{OPT}| - 2$. The last inequality holds since $|\mathsf{OPT}_1| \geq |V(G_1)| \geq 5$ and $\alpha > 1$.

It remains to prove (1). Let E_1 be the two edges of G_1 with endpoints in v_2 and E_2 be the two edges of G_2 with endpoints in v_1. Observe that E_i coincides with the E_i defined in Line 9. By the same argument as in the proof of Lemma 10, one has that $(\mathsf{OPT}_1 \setminus E_1) \cup (\mathsf{OPT}_2 \setminus E_2)$ is a 2VC spanning subgraph of G. Notice that $\mathsf{OPT}_1 \setminus E_1$ and $\mathsf{OPT}_2 \setminus E_2$ are edge-disjoint and that $|E_i \cap \mathsf{OPT}_i| = |E_i| = 2$ for $i \in \{1, 2\}$. Using this two facts we get that $|\mathsf{OPT}| \leq |(\mathsf{OPT}_1 \setminus E_1) \cup (\mathsf{OPT}_2 \setminus E_2)| = |\mathsf{OPT}_1| + |\mathsf{OPT}_2| - 4$.

For the other direction, assume by contradiction that $|\mathsf{OPT}| < |\mathsf{OPT}_1| + |\mathsf{OPT}_2| - 4$. Notice that $E(G) = (E(G_1) \setminus E_1) \dot\cup (E(G_2) \setminus E_2)$ and thus $\mathsf{OPT} = ((E(G_1) \setminus E_1) \cap \mathsf{OPT}) \dot\cup ((E(G_2) \setminus E_2) \cap \mathsf{OPT})$. Thus we have that either $|(E(G_1) \setminus E_1) \cap \mathsf{OPT}| < |\mathsf{OPT}_1| - 2$ or $|(E(G_2) \setminus E_2) \cap \mathsf{OPT}| < |\mathsf{OPT}_2| - 2$. Assume w.l.o.g. that $|(E(G_1) \setminus E_1) \cap \mathsf{OPT}| < |\mathsf{OPT}_1| - 2$. Then $((E(G_1) \setminus E_1) \cap \mathsf{OPT}) \cup \{uv_2, vv_2\}$ is a 2VC spanning subgraph of G_1 of cardinality less than $|\mathsf{OPT}_1|$, a contradiction. (1) follows. ◀

2.2 A Canonical 2-Edge-Cover

It remains to give a good enough approximation algorithm for structured graphs. The first step in our algorithm (similarly to prior work on related problems [6, 19, 25]) is to compute (in polynomial time [33, Chapter 30]) a minimum-cardinality 2-edge-cover[3] H of G. It is worth to remark that $|H| \leq \mathsf{opt}(G)$: indeed the degree of each node in any 2VC spanning subgraph of G must be at least 2.

For technical reasons, we transform H, without increasing its size, into another 2-edge-cover which is *canonical* in the following sense. We need some notation first. If a connected component of H has at least 6 edges we call it a *large component*, and otherwise a *small component*. Let C be a large component of H. We call every maximal 2VC subgraph of C a *block*, and every edge of C such that its removal splits that component into two connected components a *bridge*. Notice that every edge of C is either a bridge or belongs to some block in that component. Also, every edge of C belongs to at most one block, thus there is a unique partition of the edges of C into blocks and bridges (but a node of C might belong to multiple blocks and to multiple bridges). Observe that C is 2VC iff it has exactly one block. If C is large but not 2VC we call it a *complex component*. If a block B of a complex component C contains only one cut vertex of C, we say that B is a *leaf-block* of C. Notice that since H is a 2-edge-cover, C must have at least 2 leaf blocks.

▶ **Definition 12** (Canonical 2-Edge-Cover). *A 2-edge-cover S of a graph G is* canonical *if: (1) Every small component of S is a cycle; (2) For any complex component C of S, each leaf-block B of C has at least 5 nodes.*

▶ **Lemma 13.** *Given a minimum 2-edge-cover H of a structured graph G, in polynomial time one can compute a* canonical *2-edge-cover S of G with $|S| = |H|$.*

Proof. We start with $S := H$. At each step if there are edges $e \in E(G) \setminus E(S)$ and $e' \in E(S)$, such that $S' := S \cup \{e\} \setminus \{e'\}$ is a 2-edge-cover that has fewer connected components than S or it has the same number of connected components as S but has fewer bridges and blocks in total than S, then we replace S by S'. This process clearly terminates within a polynomial number of steps, returning a 2-edge-cover S of the same size as the initial H (hence in particular S must be minimal).

[3] A 2-edge-cover H of a graph G is a subset of edges such that each node v of G has at least 2 edges of H incident to it.

Let us show that the final S satisfies the remaining properties. Assume by contradiction that S has a connected component C with at most 5 edges that is not a cycle. By a simple case analysis C must be a 4-cycle plus one chord f. However this contradicts the minimality of S by Fact 3.

Finally assume by contradiction that S has a complex component C, with a leaf-block B such that B has at most 4 nodes. By the minimality of S, B must be a 3-cycle or a 4-cycle. Let $B = v_1 \dots v_k$, $k \in \{3, 4\}$, and assume w.l.o.g. that v_1 is the only cut-vertex of C that belongs to B. In this case we show that there must exist an edge $e = uz \in E(G)$ such that $u \in \{v_2, v_k\}$ and $z \notin B$. If this is not true then for $k = 3$, v_1 is a cut-vertex in G, and for $k = 4$, $\{v_1, v_3\}$ form a non-isolating cut, leading to a contradiction in both cases. Consider $S' := S \cup \{e\} \setminus \{uv_1\}$. Note that S' is a 2-edge-cover of the same size as S. Since uv_1 belongs to a cycle of S, then the number of connected components in S' is not more than in S. If $z \notin C$ the number of connected components of S' is less than in S, which is a contradiction. Otherwise the number of connected components of S and S' is the same. Now in S' all the bridges and the blocks of S that shared an edge with any path from u to z in $S \setminus \{uv_1\}$ become part of the same block and all the other bridges and blocks remain the same. This is a contradiction as the total number of bridges and blocks of S' is less than in S. ◀

2.3 A Credit-Based Argument

Next assume that we are given a minimum-cardinality canonical 2-edge-cover H of a structured graph G. Observe that, for $|H| \leq 5$, H is necessarily a cycle of length $|H|$ by the definition of canonical 2-edge-cover and a simple case analysis. In particular H is already a feasible (and optimal) solution. Therefore we next assume $|H| \geq 6$. Starting from $S = H$, we will gradually add edges to (and sometimes remove edges from) S, until S becomes 2VC. In order to keep the size of S under control, we use a credit-based argument similarly to prior work [6, 19, 21]. At high level, the idea is to assign a certain number of credits $\mathsf{cr}(S)$ to S. Let us define the cost of S as $\mathsf{cost}(S) = |S| + \mathsf{cr}(S)$. We guarantee that for the initial value of S, namely $S = H$, $\mathsf{cost}(S) \leq \frac{4}{3}|H|$. Furthermore, during the process $\mathsf{cost}(S)$ does not increase.

During the process we maintain the invariant that S is canonical. Hence the following credit assignment scheme is valid for any intermediate S:

1. To every small component C of S we assign $\mathsf{cr}(C) = |E(C)|/3$ credits.
2. Each large component C receives $\mathsf{cr}(C) = 1$ credits.
3. Each block B receives $\mathsf{cr}(B) = 1$ credits.
4. Each bridge b receives $\mathsf{cr}(b) = 1/4$ credits.

We remark that each large connected component C of S which is 2VC, receives one credit in the role of a component, and one additional credit in the role of a block of that component. Let $\mathsf{cr}(S) \geq 0$ the total number of credits assigned to the subgraphs of S. It is not hard to show that the initial cost of S is small enough.

▶ **Lemma 14.** $\mathsf{cost}(H) \leq \frac{4}{3}|H|$.

Proof. Let us initially assign $\frac{1}{4}$ credits to the bridges of H and $\frac{1}{3}$ credits to the remaining edges. Hence we assign at most $\frac{|H|}{3}$ credits in total. We next redistribute these credits so as to satisfy the credit assignment scheme.

Each small component C retains the credits of its edges. If C is large and 2VC then it has exactly one block B. Since $|E(C)| \geq 6$, its edges have at least 2 credits, so we can assign 1 credit to C and 1 to B.

Now consider a complex component C of H. The bridges keep their own credits. Since H is a 2-edge-cover and C is complex, then C has at least 2 leaf-blocks B_1 and B_2. By the definition of canonical, B_1 and B_2 have at least 5 nodes (hence edges) each. Therefore

together they have at least $\frac{10}{3} > 3$ credits, which is sufficient to assign one credit to C, B_1 and B_2. Any other block B of C (which has at least 3 edges) keeps the credits of its edges, hence at least 1 credit. Observe that $\mathsf{cost}(H) = |H| + \mathsf{cr}(H) \leq \frac{4}{3}|H|$ as desired. ◀

As mentioned before, starting from $S = H$, we will transform S without increasing its cost $\mathsf{cost}(S)$ until it becomes a single large component C that is 2VC (and thus it has exactly one block B) and therefore a 2VC spanning subgraph of G. Notice that at the end of the process $\mathsf{cr}(S) = \mathsf{cr}(C) + \mathsf{cr}(B) = 2$, hence $|S| = \mathsf{cost}(S) - 2 \leq \frac{4}{3}|H| - 2$. Combining this with the trivial case for $|H| \leq 5$, we obtain the following lemma.

▶ **Lemma 15.** *Given a canonical minimum 2-edge-cover H of a structured graph G, one can compute in polynomial time a 2VCSS solution S for G with $|S| \leq \max\{|H|, \frac{4}{3}|H| - 2\}$.*

Given the above results, it is easy to prove Theorem 1.

Proof of Theorem 1. By Lemma 8 it is sufficient to compute a solution of cost at most $\max\{\mathsf{opt}(G), \frac{4}{3} \cdot \mathsf{opt}(G) - 2\}$ on a structured graph G. We initially compute a canonical minimum 2-edge-cover H of G via Lemma 13. Then we apply Lemma 15 to obtain a 2VCSS solution S with $|S| \leq \max\{|H|, \frac{4}{3}|H| - 2\} \leq \max\{\mathsf{opt}(G), \frac{4}{3}\mathsf{opt}(G) - 2\}$. Clearly all steps can be performed in polynomial time. ◀

It remains to discuss the proof of Lemma 15 (assuming $|H| \geq 6$), which is the most technical part of our paper. The construction at the heart of the proof consists of a few stages. Recall that we start with a 2-edge-cover $S = H$, and then gradually transform S without increasing $\mathsf{cost}(S)$.

In the first stage of our construction we remove from S all the small components with the exception of the following type of 4-cycles that require a separate argument in the following.

▶ **Definition 16** (pendant 4-cycle)**.** *Let S be a 2-edge-cover of a graph G and C' be a large component of S. We say that a connected component C of S is a* pendant 4-cycle *(of C') if C is a 4-cycle and all the edges of G with exactly one endpoint in C have the other endpoint in C'.*

▶ **Lemma 17.** *Let G be a structured graph and H be a canonical minimum 2-edge cover of G, with $|H| \geq 6$. In polynomial time one can compute a canonical 2-edge-cover S of G such that the only small components of S are pendant 4-cycles and $\mathsf{cost}(S) \leq \mathsf{cost}(H)$.*

In the second stage of our construction we reduce to the case where S consists of large 2VC components only.

▶ **Lemma 18.** *Let G be a structured graph and S be a canonical 2-edge-cover of G such that the only small components of S are pendant 4-cycles. In polynomial time one can compute a canonical 2-edge-cover S' of G such that all the connected components of S' are 2VC and large, and $\mathsf{cost}(S') \leq \mathsf{cost}(S)$.*

At this point we can exploit the following definition and lemma from [19] to construct the desired 2VC spanning subgraph.

▶ **Definition 19** (Nice Cycle)**.** *Let $\Pi = (V_1, \ldots, V_k)$, $k \geq 2$, be a partition of the node-set of a graph G. A nice cycle N of G w.r.t. Π is a subset of edges with endpoints in distinct subsets of Π such that: (1) N induces one cycle of length at least 2 in the graph obtained from G by collapsing each V_i into a single node; (2) given the two edges of N incident to some V_i, these edges are incident to distinct nodes of V_i unless $|V_i| = 1$.*

▶ **Lemma 20** ([19]). *Let $\Pi = (V_1, \dots, V_k)$, $k \geq 2$, be a partition of the node-set of a 2VC graph G. In polynomial time one can compute a nice cycle N of G w.r.t. Π.*

▶ **Lemma 21.** *Let G be a structured graph and S be a 2-edge-cover of G such that all the connected components of S are 2VC and large. In polynomial time one can compute a 2VCSS solution S' for G with $\mathsf{cost}(S') \leq \mathsf{cost}(S)$.*

Proof. Initially set $S' = S$. Consider the partition $\Pi = (V_1, \dots, V_k)$ of $V(G)$ where V_i is the set of vertices of the 2VC component C_i of S'. If $k = 1$, S' already satisfies the claim. Otherwise, using Lemma 20 we can compute a nice cycle N of G w.r.t. Π. Let us replace S' with $S'' := S' \cup N$. W.l.o.g assume N is incident to $V_1, ..., V_r$ for some $2 \leq r \leq k$. Then in S'' the nodes $V_1 \cup \ldots \cup V_r$ belong to a unique (large) 2VC connected component C'. Furthermore $\mathsf{cost}(S') - \mathsf{cost}(S'') = \sum_{i=1}^{r}(\mathsf{cr}(C_i) + \mathsf{cr}(B_i)) - \mathsf{cr}(C') - \mathsf{cr}(B') - r = 2r - 2 - r \geq 0$, where B_i is the only block of the component C_i and B' the only block of C'. By iterating the process for a polynomial number of times one obtains a single 2VC component, hence the claim. ◀

The proof of Lemma 15 follows by chaining Lemmas 17, 18, and 21, and by the previous simple observations.

References

1. David Adjiashvili. Beating approximation factor two for weighted tree augmentation with bounded costs. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2384–2399. SIAM, 2017. doi:10.1137/1.9781611974782.157.
2. Étienne Bamas, Marina Drygala, and Ola Svensson. A simple LP-based approximation algorithm for the matching augmentation problem. In Karen Aardal and Laura Sanità, editors, *Integer Programming and Combinatorial Optimization - 23rd International Conference, IPCO 2022, Eindhoven, The Netherlands, June 27-29, 2022, Proceedings*, volume 13265 of *Lecture Notes in Computer Science*, pages 57–69. Springer, 2022. doi:10.1007/978-3-031-06901-7_5.
3. Miguel Bosch-Calvo, Fabrizio Grandoni, and Afrouz Jabal Ameli. A 4/3 approximation for 2-vertex-connectivity, 2023. arXiv:2305.02240.
4. Jaroslaw Byrka, Fabrizio Grandoni, and Afrouz Jabal Ameli. Breaching the 2-approximation barrier for connectivity augmentation: a reduction to steiner tree. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 815–825. ACM, 2020. doi:10.1145/3357713.3384301.
5. Federica Cecchetto, Vera Traub, and Rico Zenklusen. Bridging the gap between tree and connectivity augmentation: unified and stronger approaches. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 370–383. ACM, 2021. doi:10.1145/3406325.3451086.
6. Joe Cheriyan, Jack Dippel, Fabrizio Grandoni, Arindam Khan, and Vishnu V. Narayan. The matching augmentation problem: a 7/4-approximation algorithm. *Math. Program.*, 182(1):315–354, 2020. doi:10.1007/s10107-019-01394-z.
7. Joseph Cheriyan, Robert Cummings, Jack Dippel, and J. Zhu. An improved approximation algorithm for the matching augmentation problem. *CoRR*, abs/2007.11559, 2020. arXiv:2007.11559.
8. Joseph Cheriyan and Zhihan Gao. Approximating (unweighted) tree augmentation via lift-and-project, part I: stemless TAP. *Algorithmica*, 80(2):530–559, 2018. doi:10.1007/s00453-016-0270-4.

9 Joseph Cheriyan and Zhihan Gao. Approximating (unweighted) tree augmentation via lift-and-project, part II. *Algorithmica*, 80(2):608–651, 2018. doi:10.1007/s00453-017-0275-7.

10 Joseph Cheriyan, András Sebö, and Zoltán Szigeti. Improving on the 1.5-approximation of a smallest 2-edge connected spanning subgraph. *SIAM J. Discret. Math.*, 14(2):170–180, 2001. doi:10.1137/S0895480199362071.

11 Joseph Cheriyan and Ramakrishna Thurimella. Approximating minimum-size k-connected spanning subgraphs via matching. *SIAM J. Comput.*, 30(2):528–560, 2000. doi:10.1137/S009753979833920X.

12 Nachshon Cohen and Zeev Nutov. A (1+ln2)-approximation algorithm for minimum-cost 2-edge-connectivity augmentation of trees with constant radius. *Theor. Comput. Sci.*, 489-490:67–74, 2013. doi:10.1016/j.tcs.2013.04.004.

13 Artur Czumaj and Andrzej Lingas. On approximability of the minimum-cost k-connected spanning subgraph problem. In Robert Endre Tarjan and Tandy J. Warnow, editors, *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, 17-19 January 1999, Baltimore, Maryland, USA*, pages 281–290. ACM/SIAM, 1999. URL: http://dl.acm.org/citation.cfm?id=314500.314573.

14 E. A. Dinits, A. V. Karzanov, and M. V. Lomonosov. On the structure of a family of minimal weighted cuts in a graph. *Studies in Discrete Optimization*, pages 290–306, 1976.

15 Guy Even, Jon Feldman, Guy Kortsarz, and Zeev Nutov. A 1.8 approximation algorithm for augmenting edge-connectivity of a graph from 1 to 2. *ACM Trans. Algorithms*, 5(2):21:1–21:17, 2009. doi:10.1145/1497290.1497297.

16 Samuel Fiorini, Martin Groß, Jochen Könemann, and Laura Sanità. Approximating weighted tree augmentation via chvátal-gomory cuts. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 817–831. SIAM, 2018. doi:10.1137/1.9781611975031.53.

17 Harold N. Gabow and Suzanne Gallagher. Iterated rounding algorithms for the smallest k-edge connected spanning subgraph. *SIAM J. Comput.*, 41(1):61–103, 2012. doi:10.1137/080732572.

18 Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, and Krzysztof Sornat. On the cycle augmentation problem: Hardness and approximation algorithms. In Evripidis Bampis and Nicole Megow, editors, *Approximation and Online Algorithms - 17th International Workshop, WAOA 2019, Munich, Germany, September 12-13, 2019, Revised Selected Papers*, volume 11926 of *Lecture Notes in Computer Science*, pages 138–153. Springer, 2019. doi:10.1007/978-3-030-39479-0_10.

19 Mohit Garg, Fabrizio Grandoni, and Afrouz Jabal Ameli. Improved approximation for two-edge-connectivity. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 2368–2410. SIAM, 2023. doi:10.1137/1.9781611977554.ch92.

20 Naveen Garg, Santosh S. Vempala, and Aman Singla. Improved approximation algorithms for biconnected subgraphs via better lower bounding techniques. In Vijaya Ramachandran, editor, *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas, USA*, pages 103–111. ACM/SIAM, 1993. URL: http://dl.acm.org/citation.cfm?id=313559.313618.

21 Fabrizio Grandoni, Afrouz Jabal Ameli, and Vera Traub. Breaching the 2-approximation barrier for the forest augmentation problem. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1598–1611. ACM, 2022. doi:10.1145/3519935.3520035.

22 Fabrizio Grandoni, Christos Kalaitzis, and Rico Zenklusen. Improved approximation for tree augmentation: saving by rewiring. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 632–645, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3188745.3188898.

23 Prabhakar Gubbala and Balaji Raghavachari. Approximation algorithms for the minimum cardinality two-connected spanning subgraph problem. In Michael Jünger and Volker Kaibel, editors, *Integer Programming and Combinatorial Optimization, 11th International IPCO Conference, Berlin, Germany, June 8-10, 2005, Proceedings*, volume 3509 of *Lecture Notes in Computer Science*, pages 422–436. Springer, 2005. doi:10.1007/11496915_31.

24 Klaus Heeger and Jens Vygen. Two-connected spanning subgraphs with at most 10/7 opt edges. *SIAM J. Discret. Math.*, 31(3):1820–1835, 2017. doi:10.1137/16M1091587.

25 Christoph Hunkenschröder, Santosh S. Vempala, and Adrian Vetta. A 4/3-approximation algorithm for the minimum 2-edge connected subgraph problem. *ACM Trans. Algorithms*, 15(4):55:1–55:28, 2019. doi:10.1145/3341599.

26 Raja Jothi, Balaji Raghavachari, and Subramanian Varadarajan. A 5/4-approximation algorithm for minimum 2-edge-connectivity. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA*, pages 725–734. ACM/SIAM, 2003. URL: http://dl.acm.org/citation.cfm?id=644108.644227.

27 Samir Khuller and Uzi Vishkin. Biconnectivity approximations and graph carvings. In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 759–770. ACM, 1992. doi:10.1145/129712.129786.

28 Samir Khuller and Uzi Vishkin. Biconnectivity approximations and graph carvings. *J. ACM*, 41(2):214–235, 1994. doi:10.1145/174652.174654.

29 Guy Kortsarz and Zeev Nutov. Lp-relaxations for tree augmentation. In Klaus Jansen, Claire Mathieu, José D. P. Rolim, and Chris Umans, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France*, volume 60 of *LIPIcs*, pages 13:1–13:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.APPROX-RANDOM.2016.13.

30 Guy Kortsarz and Zeev Nutov. A simplified 1.5-approximation algorithm for augmenting edge-connectivity of a graph from 1 to 2. *ACM Trans. Algorithms*, 12(2):23:1–23:20, 2016. doi:10.1145/2786981.

31 Hiroshi Nagamochi. An approximation for finding a smallest 2-edge-connected subgraph containing a specified spanning tree. *Discret. Appl. Math.*, 126(1):83–113, 2003. doi:10.1016/S0166-218X(02)00218-4.

32 Zeev Nutov. On the tree augmentation problem. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, pages 61:1–61:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ESA.2017.61.

33 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Springer Science & Business Media, 2003.

34 András Sebö and Jens Vygen. Shorter tours by nicer ears: 7/5-approximation for the graph-tsp, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. *Comb.*, 34(5):597–629, 2014. doi:10.1007/s00493-014-2960-3.

35 Vera Traub and Rico Zenklusen. A better-than-2 approximation for weighted tree augmentation. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1–12. IEEE, 2021. doi:10.1109/FOCS52979.2021.00010.

36 Vera Traub and Rico Zenklusen. A (1.5+ϵ)-approximation algorithm for weighted connectivity augmentation. *CoRR*, abs/2209.07860, 2022. doi:10.48550/arXiv.2209.07860.

Lower Bounds for Pseudo-Deterministic Counting in a Stream

Vladimir Braverman
Rice University, Houston, TX, USA

Robert Krauthgamer
Weizmann Institute of Science, Rehovot, Israel

Aditya Krishnan
Pinecone, San Francisco, CA, USA

Shay Sapir
Weizmann Institute of Science, Rehovot, Israel

Abstract

Many streaming algorithms provide only a high-probability relative approximation. These two relaxations, of allowing approximation and randomization, seem necessary – for many streaming problems, both relaxations must be employed simultaneously, to avoid an exponentially larger (and often trivial) space complexity. A common drawback of these randomized approximate algorithms is that independent executions on the same input have different outputs, that depend on their random coins. *Pseudo-deterministic* algorithms combat this issue, and for every input, they output with high probability the same "canonical" solution.

We consider perhaps the most basic problem in data streams, of counting the number of items in a stream of length at most n. Morris's counter [CACM, 1978] is a randomized approximation algorithm for this problem that uses $O(\log \log n)$ bits of space, for every fixed approximation factor (greater than 1). Goldwasser, Grossman, Mohanty and Woodruff [ITCS 2020] asked whether pseudo-deterministic approximation algorithms can match this space complexity. Our main result answers their question negatively, and shows that such algorithms must use $\Omega(\sqrt{\log n / \log \log n})$ bits of space.

Our approach is based on a problem that we call *Shift Finding*, and may be of independent interest. In this problem, one has query access to a shifted version of a known string $F \in \{0,1\}^{3n}$, which is guaranteed to start with n zeros and end with n ones, and the goal is to find the unknown shift using a small number of queries. We provide for this problem an algorithm that uses $O(\sqrt{n})$ queries. It remains open whether $\text{poly}(\log n)$ queries suffice; if true, then our techniques immediately imply a nearly-tight $\Omega(\log n / \log \log n)$ space bound for pseudo-deterministic approximate counting.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Lower bounds and information complexity; Theory of computation → Pseudorandomness and derandomization

Keywords and phrases streaming algorithms, pseudo-deterministic, approximate counting

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.30

Category Track A: Algorithms, Complexity and Games

Related Version *arXiv Version*: https://arxiv.org/abs/2303.16287

Funding *Vladimir Braverman*: Work partially supported by ONR Award N00014-18-1-2364 and NSF awards 1652257, 1813487 and 2107239.
Robert Krauthgamer: Work partially supported by ONR Award N00014-18-1-2364, by a Weizmann-UK Making Connections Grant, by a Minerva Foundation grant, and the Weizmann Data Science Research Center.
Aditya Krishnan: Work partially done while the author was at Johns Hopkins University and supported by the MINDS Data Science Fellowship.

Shay Sapir: This research was partially supported by the Israeli Council for Higher Education (CHE) via the Weizmann Data Science Research Center.

1 Introduction

Computing over data streams is a rich algorithmic area that has developed enormously, and actually started with the simple-looking problem of approximate counting [28]. Let us first recall the streaming model: The input is a stream, i.e., a sequence of items, and the goal is to compute a pre-defined function of these items, such as the number of items (or number of the distinct items), while making one sequential pass over the stream (or sometimes a few passes). Many useful functions actually depend on the items as a multiset, i.e., ignoring their order, or even only on their frequencies (like the famous ℓ_p-norm of the frequency vector). Another possible goal is to produce a sample, rather than computing a function, e.g., to produce a uniformly random item.

The primary measure of efficiency for streaming algorithms is their space complexity, and for many problems, researchers have designed space-efficient algorithms, often with space complexity that is even polylogarithmic in the input size. However, this comes at a price – these algorithms are usually randomized (and not deterministic) and/or compute an approximate solution (rather than exact one). In fact, oftentimes both relaxations are needed in order to achieve low space complexity. For example, to count the number of items in a stream of length at most n, there is a randomized approximation algorithm using $O(\log\log n)$ bits of space, but algorithms that are exact or deterministic must use $\Omega(\log n)$ bits [28]. Another example is the ℓ_2-norm of the frequency vector of items from a ground set $[d]$ (or equivalently, of a d-dimensional vector under a sequence of additive updates) – there is a randomized approximation algorithm that uses $O(\log d)$ bits of space, but algorithms that are exact or deterministic must use $\Omega(d)$ bits of space [1].

Gat and Goldwasser [9] initiated the study of *pseudo-deterministic* algorithms, which informally means that when run (again) on the same input, with high probability they produce exactly the same output. This notion combats a potential issue with randomized algorithms, that independent executions on the same input might return different outputs, depending on the algorithm's coin tosses. Many known streaming algorithms suffer from this issue, which is a serious concern for some users and applications. Pseudo-deterministic algorithms were later considered in the streaming model by Goldwasser, Grossman, Mohanty and Woodruff [16], and these are formally defined as follows.

▶ **Definition 1.1.** *A streaming algorithm A is* pseudo-deterministic *(PD) if there is a function $F(\cdot)$ defined on inputs of A (streams), such that for every stream σ,*

$$\Pr[A(\sigma) = F(\sigma)] \geq 9/10,$$

where the probability is over the random choices of the algorithm. We shall refer to F as the canonical function *of algorithm A.*[1]

We focus on *estimation problems*, which ask to approximate a numerical value, and are very popular in the streaming model. For such problems, the notion of PD relaxes the exact setting and the deterministic one, since exact algorithms have one canonical output (the

[1] The canonical function F depends on the order arrival of the stream items. In an alternative definition, the canonical function depends on the items only as a multiset, i.e., ignoring their order in the stream. These two definitions are equivalent in the setting of approximate counting, which is the focus of our work.

Table 1 Known space bounds (in bits) for 2-approximate counting in a stream of length at most n. Folklore bounds are stated without a reference.

Algorithms	Upper bound	Lower bound
Exact or deterministic	$O(\log n)$	$\Omega(\log n)$
Randomized and approximate	$O(\text{loglog}\, n)$ [28]	$\Omega(\text{loglog}\, n)$ [29]
Pseudo-deterministic	$O(\log n)$	$\Omega(\sqrt{\log n/\text{loglog}\, n})$ [Thm. 1.2]

exact numerical value), and hence they are PD. Thus the known lower bounds for these settings do not apply for PD algorithms, and a central question, identified in [16], remains open:

Are there efficient PD streaming algorithms for estimation problems?

Currently, no lower bounds are known for natural estimation problems, although for several search problems, like reporting an element from a stream with deletions (equivalently, an index from the support of the frequency vector), it is known that lower bounds for deterministic algorithms extend to PD algorithms [16].

1.1 Main Result: Approximate Counting

Perhaps the most basic problem in the streaming model is to count the number of stream items. Exact counting, i.e., computing the number of items exactly, requires $\Theta(\log n)$ bits of space when the stream has length at most n, even for randomized algorithms with some error probability. Work by Morris [28], later refined in [8, 18, 29], showed that the number of stream items can be $(1+\epsilon)$-approximated with probability $9/10$ using $O_\epsilon(\text{loglog}\, n)$ bits of space, where $\epsilon > 0$ is arbitrary but fixed. Throughout, we refer to multiplicative approximation, and use the notations $O_c(\cdot)$ and $\Omega_c(\cdot)$ to hide factors that are polynomial in c. Morris's algorithm has found many applications, both in theory and in practice [27, 29]. An open question stated explicitly by Goldwasser, Grossman, Mohanty and Woodruff [16] is whether there is a PD algorithm for this problem using $O(\text{loglog}\, n)$ bits of space. We answer their question negatively, by proving the following lower bound.

▶ **Theorem 1.2** (Main Result). *For every $c, n > 1$, every PD streaming algorithm that c-approximates the number of items in a stream of length at most $(c+1)n$ must use $\Omega_c(\sqrt{\log n/\text{loglog}\, n})$ bits of space.*

To be more precise, our lower bound is actually $\Omega\big(\frac{\log n}{\sqrt{\log n \,\text{loglog}(cn)}+\log c}\big)$, which is still $\Omega(\sqrt{\frac{\log n}{\text{loglog}\, n}})$ as long as $c < 2^{\sqrt{\log n \,\text{loglog}\, n}}$. Previously, there was a large gap for this problem, between $O(\log n)$ bits (by a deterministic algorithm) and $\Omega(\text{loglog}\, n)$ bits (from the randomized setting) [29]. See Table 1 for a summary of the known bounds.

Our proof analyzes the promise variant of c-approximate counting for streams of length at most $(c+1)n$, which we denote by $\Pi^{AC}_{c,n}$; this variant asks to distinguish whether the number of stream items is $\le n$ or $> cn$ (see Definition 2.1). A crucial property of PD algorithms is that they have to be PD also for inputs in the range $[n+1, cn]$ (i.e., outside the promise). We rely on this property of PD algorithms to prove the following result, which immediately yields Theorem 1.2 as a corollary.

▶ **Theorem 1.3** (Main Result). *For every $c, n > 1$, every PD streaming algorithm for problem $\Pi^{AC}_{c,n}$ must use $\Omega_c(\sqrt{\log n / \log\log n})$ bits of space.*

Our proof of Theorem 1.3 appears in Section 4. It is based on a problem that we call Shift Finding, which may be of independent interest, as it is very natural and likely to find connections to other problems. In addition, it can potentially lead to a near-tight $\Omega(\log n/ \log\log n)$ lower bound for PD streaming, by simply improving our algorithmic result for Shift Finding. A very recent independent work by Grossman, Gupta and Sellke [20] shows a tight $\Omega(\log n)$ bound for $\Pi^{AC}_{c,n}$, using a very different technique, which views the PD streaming algorithm as a Markov chain with a limited number of states.

1.2 Main Technique: The Shift Finding Problem

Our main result relies on *algorithms* for the shift Finding problem $\Pi^{SF}_{c,n}$, which is defined below. Let us first introduce some basic terminology. A function $F : [m] \to \{0, 1\}$ can also be viewed as a string $F \in \{0, 1\}^m$, and vice versa, and we sometimes use these interchangeably. Given $s \in [0, n]$, let the shifted version of this F be the function $F_s : x \mapsto F(s + x)$, with a properly restricted domain, see Section 2.

▶ **Definition 1.4** (Shift Finding). *Let $c, n > 1$. In problem $\Pi^{SF}_{c,n}$, the input is a string $P \in \{0, 1\}^{(c-1)n}$, and one has query access to a string F_{s^*} that is the concatenation of $n - s^*$ zeros, then P, and finally s^* ones, for an unknown $s^* \in [0, n]$. Thus, a query for $x \in [0, cn]$ returns $F_{s^*}(x)$. The goal is to output s^*.*

The measure of complexity of an algorithm for this problem is the number of queries that it makes to F_{s^*}. A randomized algorithm is required to be correct (in its output s^*) with probability 9/10.

This problem may be also of independent interest. In a different variant of shift finding, the input is a random string $c \in \{0, 1\}^n$ and a vector x that is obtained from the string c by a cyclic shift τ and some noise (random bit flips), and the goal is to compute the shift τ with high probability. This problem is related to GPS synchronization, see [23, 2] for more details. There is a sublinear time algorithm for this problem, running in time roughly $O(n^{0.641})$ [2]. One main difference is that in our Definition 1.4, one string is completely known to the algorithm, and the only concern is the number of queries to the second string.

1.2.1 Connection to PD Counting

We show that an algorithm for Shift Finding ($\Pi^{SF}_{c,n}$) implies a space lower bound for PD streaming algorithm for counting ($\Pi^{AC}_{c,n}$).

▶ **Theorem 1.5.** *Let $c, n > 1$, and suppose that the Shift Finding problem $\Pi^{SF}_{c,n}$ admits a randomized algorithm that makes at most $q = q(c, n)$ queries (possibly adaptive). Then, every PD streaming algorithm for the approximate counting problem $\Pi^{AC}_{c,n}$ must use $\Omega(\frac{\log n}{\log q})$ bits of space.*

It immediately follows that if the Shift Finding problem $\Pi^{SF}_{c,n}$ can be solved using polylog(n) queries (for fixed $c > 1$), then PD approximate counting requires $\Omega(\frac{\log n}{\log\log n})$ bits of space. However, our current upper bound for Shift Finding is $q = O(\sqrt{cn})$ queries (Theorem 1.8) and is not strong enough to yield a nontrivial lower bound for PD approximate counting.

Therefore, to prove our main lower bound (Theorem 1.3), we revert to a generalization of Theorem 1.5 where the Shift Finding algorithm is still given an instance of problem $\Pi^{SF}_{c,n}$ (namely, a string F and query access to F_{s^*}), but reports a small set $R \subset [0, n]$ (say of size

$|R| \leq t$) that contains the unknown shift (i.e., $s^* \in R$). This algorithm may be randomized provided that it is PD, and its canonical function maps each instance of problem $\Pi^{SF}_{c,n}$ to a set R of size t that contains s^*.

▶ **Theorem 1.6.** *Let $c, n > 1$, and suppose there is a PD algorithm Q that, given an instance of problem $\Pi^{SF}_{c,n}$, makes at most $q = q(c,n)$ queries (possibly adaptive) to F_{s^*} and its canonical function M maps the input to a set $R \subset [0,n]$ of size $t = t_c(n)$ that contains s^*. Then every PD streaming algorithm for problem $\Pi^{AC}_{c,n}$ must use $\Omega(\frac{\log(n/t)}{\log q})$ bits of space.*

We use Theorem 1.6, (more precisely its proof arguments rather than its statement) to prove our main result (Theorem 1.3), see Section 4. At a high level, the proof of Theorem 1.3 proceeds by splitting into two cases, depending on the canonical function F. Roughly speaking, in one case we show a Shift Finding algorithm that returns a set of size $t = n/2^{\sqrt{\log n}}$ using $q = O(\log n)$ queries by binary search, and in the other case an algorithm to find the shift (i.e., $t = 1$) with probability $9/10$ using $q = 2^{\sqrt{\log n}}$ uniformly random queries.

As a corollary of Theorem 1.5, we get that the *tracking* version of approximate counting must use $\Omega(\log n)$ bits of space, which is tight with a straightforward deterministic counting. Tracking means that the algorithm produces an output after every stream item rather than at the end of the stream, and with probability $9/10$, all the outputs are simultaneously correct (i.e., approximate the number of items seen so far).

▶ **Corollary 1.7** (Tracking). *For every $c, n > 1$, every PD tracking algorithm that c-approximates the number of items in a stream of length $(c+1)n$ must use $\Omega(\log n)$ bits of space.*

In contrast, for standard randomized algorithms, there is a tracking algorithm for $(1+\epsilon)$-approximate counting that uses $O_\epsilon(\log\log n)$ bits of space, for any fixed $\epsilon > 0$ [29]. Corollary 1.7 follows by an easy modification of the proof of Theorem 1.5. That proof uses $O(\log q)$ repetitions of a PD streaming algorithm, and then employs a union bound on q input streams, which is not necessary for tracking algorithms and thus the bound follows.

A more direct argument is essentially by equivalence to exact counting. For a stream with $s < n$ items, the state of a PD tracking algorithm with canonical function F can be used to compute s, as follows. Simulate insertion of more items to the stream until the output of the algorithm changes to 1 (which corresponds to the first 1 in F_s), from which s can be computed.

1.2.2 An Algorithm for Shift Finding

Consider a special case of the Shift Finding problem $\Pi^{SF}_{c,n}$, where the input string P is a run of zeros followed by a run of ones (viewed as a function, it is a step function); then the algorithm can perform a binary search using $O(\log(cn))$ queries, and find the unique location where F_{s^*} switches from value 0 to 1, and hence recover s^*. At the other extreme, suppose the input string P is random; then with high probability every set of $O(\log n)$ queries from P (and thus from F_{s^*}) will be answered differently (viewed as a string in $\{0,1\}^{O(\log n)}$). Based on these observations, one may hope that problem $\Pi^{SF}_{c,n}$ admits an algorithm that makes polylog(cn) queries. We leave this as an open question and prove a weaker bound of $O(\sqrt{cn})$ queries.

▶ **Theorem 1.8** (Shift Finding Algorithm). *There is a deterministic algorithm for problem $\Pi^{SF}_{c,n}$ that makes $O(\sqrt{cn})$ queries.*

A key observation in our result, that may be useful in future work, is that for every shift s^* there is a "short witness" that uses exactly 2 queries. We formalize this as verifying a given guess s for the shift s^*.

▶ **Lemma 1.9** (Short Witness). *There is a deterministic algorithm that, given as input an instance of problem $\Pi^{SF}_{c,n}$ and $s < n$, makes 2 queries to F_{s^*} and returns "yes" if $s = s^*$ and "no" otherwise.*

The proofs of Theorem 1.8 and Lemma 1.9 appear in Section 5. At a high level, the Shift Finding algorithm in Theorem 1.8 queries the set $\{F_{s^*}(0), F_{s^*}(\sqrt{cn}), F_{s^*}(2\sqrt{cn}), ..., F_{s^*}(cn)\}$, and then uses the short witness (Lemma 1.9) to check every feasible $s \in [n]$ (i.e., that agrees with the query answers). Following an observation by Peter Kiss, we are able to improve our Shift Finding algorithm to use only $O((cn)^{1/3} \log n)$ queries; details omitted.

1.3 Related Work

Pseudo-deterministic algorithms

The notion of pseudo-deterministic algorithms was introduced by [9] (they originally called them Bellagio algorithms), followed by a long sequence of works that studied it in different models [13, 19, 14, 30, 24, 15, 5, 31, 11, 21, 12, 16, 26, 6, 17, 10, 7]. In the streaming and sketching models, [16] proved strong lower bounds for finding a non-zero entry in a vector (given in a stream with deletions), and for sketching ℓ_2-norms. Another related setting is that of sublinear time computation. Under certain assumptions, PD algorithms (in the sublinear time region) were shown to admit the following relation with deterministic algorithms – if for a certain problem there is a PD algorithm using q queries, then there is a deterministic algorithm using $O(q^4)$ queries [13]. The techniques of [13] do not seem to extend to streaming algorithms.

Adaptive adversarial streams

In this setting, the stream items are chosen adversarially and depend on past outputs of the streaming algorithm (i.e., the stream is adaptive) [3]. This model is considered to be between PD algorithms and the standard randomized setting, in the sense that for streams of length m, amplifying a PD algorithm to success probability $1 - \frac{1}{10m}$ (by $O(\log m)$ repetitions and taking the median) guarantees (by a union bound) that the algorithm outputs the canonical solution after every stream item with probability 9/10, thus the adversary acts as an oblivious one (the adversary knows in advance the output of the streaming algorithm, which is the canonical function). For approximate counting, adaptive streams and standard (oblivious) streams are equivalent (since the stream items are identical) and thus admit an algorithm using $O(\log\log n)$ bits of space.

There is a vast body of work designing algorithms for adaptive streams, but not much is known in terms of lower bounds. Lower bounds are known for some search problems, like finding a spanning forest in a graph undergoing edge insertions and deletions, but also for graph coloring [4]. Regarding estimation problems, the only lower bound we are aware of is for some artificial problem [25]. Recently, Stoeckl [32] showed a lower bound on streaming algorithms that use a bounded amount of randomness, conditioned on a lower bound for PD algorithms. In the related model of linear sketching, Hardt and Woodruff [22] showed lower bounds on the dimensions of sketching algorithms, which applies to many classical problems, like ℓ_p-norm estimation and heavy hitters.

2 Preliminaries

▶ **Definition 2.1** (Approximate counting). *Let $c, n > 1$. In problem $\Pi^{AC}_{c,n}$, the input is a stream of $l \leq (c+1)n$ identical items. The goal is to output 0 if $l \leq n$ and 1 if $l > cn$ (and otherwise the output can be either 0 or 1).*

Let A be a PD algorithm for problem $\Pi^{AC}_{c,n}$, and let $F : [0, (c+1)n] \to \{0,1\}$ be the canonical function of A. Thus, there is a fixed string $P \in \{0,1\}^{(c-1)n}$ such that

$$F(x) = \begin{cases} 0 & \text{if } x \in [0, n]; \\ 1 & \text{if } x \in [cn+1, (c+1)n]; \\ P(x-n) & \text{otherwise.} \end{cases}$$

For $s^* \in [0, n]$, let $F_{s^*} : [0, (c+1)n - s^*] \to \{0,1\}$ be a shifted version of F, namely the function $F_{s^*} : x \mapsto F(s^* + x)$. We use these notations throughout the paper.

Our proofs are based on a reduction from a simple one-way communication problem, called MESSAGE and denoted Π^{MSG}_{Σ}, where Alice's input x is from an alphabet Σ that is fixed in advance, Bob has no input, and the goal is that Bob outputs x with probability at least 2/3. It is well known that this problem requires $\Omega(\log|\Sigma|)$ bits of communication, even for randomized protocols using shared randomness. We provide a proof for completeness.

▶ **Lemma 2.2.** *For every alphabet Σ, every one-way communication protocol (even with shared randomness) for problem Π^{MSG}_{Σ} must use $\Omega(\log|\Sigma|)$ bits of communication.*

Proof. Let $\mathcal{A}$ be a protocol for problem Π^{MSG}_{Σ}. For a random string r representing the randomness of $\mathcal{A}$, let $\Sigma_r \subset \Sigma$ be the set of all $s \in \Sigma$ for which Bob correctly recovers s. Let r^* be a string maximizing $|\Sigma_r|$, then by averaging, $|\Sigma_{r^*}| \geq \frac{2}{3}|\Sigma|$. Consider an instance of $\mathcal{A}$ that uses r^* as its random string. Assume by contradiction that the number of communication bits is less than $\log|\Sigma_{r^*}|$, then by the pigeonhole principle there are two distinct inputs $s, s' \in \Sigma_{r^*}$ such that $\mathcal{A}(s)$ and $\mathcal{A}(s')$ result in the same message. Bob then cannot distinguish between (i.e., has the same output distribution for) s and s', a contradiction. Hence, the number of bits of communication is at least $\log|\Sigma_{r^*}| = \Omega(\log|\Sigma|)$. ◀

3 Lower Bounds for PD Approximate Counting via Shift Finding

In this section, we prove Theorem 1.5. The proof involves three problems from different settings: (a) PD approximate counting in the streaming model; (b) Shift Finding in the query-access model; and (c) MESSAGE in one-way communication with shared randomness. The proof essentially shows that if there is an algorithm for Shift Finding that makes only q queries and also a streaming algorithm for PD approximate counting that uses b bits of space, then MESSAGE can be solved using $O(b \log q)$ bits of communication. Combining this bound with the well-known lower bound for MESSAGE in Lemma 2.2 yields a lower bound for b.

A core idea in the proof is that an execution of a PD streaming algorithm A for the approximate counting problem $\Pi^{AC}_{c,n}$ on a stream with s^* insertions, can be used (even without knowing s^*, by making additional insertions and then querying the streaming algorithm A) to provide query access to the shifted function $F_{s^*} : x \mapsto F(s^* + x)$. This query access, along with a query-efficient algorithm for the Shift Finding problem $\Pi^{SF}_{c,n}$, is then used to solve an instance of the MESSAGE problem Π^{MSG}_{Σ}.

In fact, we prove the following theorem, which holds for each string F separately (rather than a bound that depends on the worst-case F), and yields Theorem 1.5 as an immediate corollary.

▶ **Theorem 3.1.** *Let A be a PD streaming algorithm for problem $\Pi^{AC}_{c,n}$, where $c, n > 1$, and let $F : [0, (c+1)n] \to \{0,1\}$ be the canonical function of A. Suppose that Shift Finding with respect to this specific F (the problem of finding an unknown shift $s^* \in [n]$ with probability at least $9/10$ given query access to F_{s^*}) admits a randomized algorithm that makes at most $q = q(F)$ (possibly adaptive) queries. Then the streaming algorithm A must use $\Omega(\frac{\log n}{\log q})$ bits of space.*

Proof. Define algorithm A' to be an amplification of A to success probability $1 - 1/(10q)$, by running $O(\log q)$ independent repetitions and reporting their majority. Assume there exists an algorithm Q that for every $s^* \in [n]$, makes at most $q = q(F)$ queries to F_{s^*} (possibly adaptive) and outputs s^* with probability at least $9/10$.

Consider an instance of problem Π^{MSG}_{Σ} with alphabet $\Sigma = [0, n]$, and consider the following protocol for it. Alice starts an execution of the streaming algorithm A' using the shared randomness, then takes her input $s^* \in \Sigma$ and makes s^* stream insertions to algorithm A', and finally sends the state (memory contents) of A' to Bob.

Bob continues the execution of the streaming algorithm A' (using the shared randomness), and uses it to provide query access to F_{s^*}, as follows. In order to query F_{s^*} at any index x, Bob makes a fresh copy A_0 of the streaming algorithm A', insert x stream items to algorithm A_0 and then reads its output. With probability at least $1 - 1/(10q)$, the answer that Bob gets is indeed $F_{s^*}(x)$ (because the number of items inserted to this instance of the algorithm is $x + s^*$). Bob uses this query access and his knowledge of F to simulate algorithm Q (with the goal of recovering s^*).

Consider Bob's simulation of algorithm Q. If Q was executed with true query access to F_{s^*}, then it would have had success probability $9/10$, and would have made a sequence of queries X_Q to F_{s^*}. This sequence X_Q depends only on F_{s^*} and the coin tosses of algorithm Q. In particular, revealing X_Q (i.e., conditioned on X_Q) does not affect the coins of the streaming algorithm A', and it still succeeds with probability at least $1 - 1/(10q)$. We can thus apply a union bound to conclude that algorithm A' succeeds on all queries $x \in X_Q$ (i.e., outputs the corresponding $F_{s^*}(x)$) with probability at least $1 - q \cdot \frac{1}{10q} = 9/10$. Hence, when Bob simulates algorithm Q using the streaming algorithm A', with probability $9/10$ (over the coins of A') the execution is identical to running algorithm Q with true access to F_{s^*}, which itself succeeds with probability $9/10$. By a union bound, with probability $8/10$ both algorithm Q and the streaming algorithm A' succeed, in which case Bob recovers s^*, and therefore this communication protocol solves problem Π^{MSG}_{Σ} with alphabet $\Sigma = [0, n]$.

By Lemma 2.2, the message Alice sends must contain $\Omega(\log n)$ bits, and thus the streaming algorithm A' must use $\Omega(\log n)$ bits of space. Recall that algorithm A' consists of $O(\log q)$ copies of the streaming algorithm A and thus algorithm A must use $\Omega(\frac{\log n}{\log q})$ bits of space. ◀

4 Lower Bound for PD Approximate Counting

In this section, we prove Theorem 1.3, i.e., for every $c, n > 1$, we prove that every PD streaming algorithm for the approximate counting problem $\Pi^{AC}_{c,n}$ must use $\Omega_c(\sqrt{\frac{\log n}{\log\log n}})$ bits of space.

Let F be the canonical function of a PD streaming algorithm for problem $\Pi^{AC}_{c,n}$. Our analysis is split into two cases depending on F, which informally correspond to whether a fixed pattern (like "01") appears in the string F at most t times or not. These cases are analyzed using Theorems 1.6 and 3.1. The overall bound will be derived by optimizing the threshold t between the two cases to roughly $t = n/2^{\sqrt{\log n}}$.

4.1 Scenario One

In this scenario, there is a specific pattern in F that appears at most t times, where $t = t_c(n)$ will be set at the end of our proof. We first consider the pattern "01" in F, which corresponds to $x \in [0, (c+1)n-1]$ such that $F(x) = 0$ and $F(x+1) = 1$, and later generalize this pattern to a broader family.

▶ **Lemma 4.1.** *If the pattern "01" appears at most t times in F, then every PD streaming algorithm for problem $\Pi^{AC}_{c,n}$ whose canonical function is F must use $\Omega(\frac{\log(n/t)}{\log\log(cn)})$ bits of space.*

Proof. The proof is by a reduction from problem MESSAGE, similarly to the proof of Theorem 3.1. Perhaps the most delicate part is the definition of an alphabet Σ for the MESSAGE problem Π^{MSG}_{Σ}, and it proceeds as follows.

Given $s \in [n]$, consider the following execution of Binary Search on the function F_s. Initialize $l = 0$ and $r = cn + 1$, and at every iteration query $F_s(\lfloor \frac{l+r}{2} \rfloor)$; if $F_s(\lfloor \frac{l+r}{2} \rfloor) = 0$, then $l \leftarrow \lfloor \frac{l+r}{2} \rfloor$, otherwise $r \leftarrow \lfloor \frac{l+r}{2} \rfloor$. These iterations maintain the invariant that $F_s(l) = 0$ and $F_s(r) = 1$, and after at most $\log(cn)$ iterations arrive at $r = l + 1$ with the pattern "01". Define a mapping $M : [n] \to [cn]$ such that $M(s)$ is the location where the binary search finds a "01" in F_s, i.e., the final index l; thus $F(s + M(s)) = 0$ and $F(s + M(s) + 1) = 1$.

In order to define an alphabet Σ, consider a partitioning of $[n]$ to buckets, defined such that items s, s' are from the same bucket B if and only if they are mapped to the same value $M(s) = M(s')$. For every bucket B and every $s, s' \in B$, we know from above that $F(s' + M(s)) = 0$ and $F(s' + M(s) + 1) = 1$, so there are at most t possibilities for s' (one of which is $s' = s$), and thus the size of the bucket $|B| \leq t$. Define $\Sigma \subset [n]$ by taking one representative from each bucket. Thus, every $s_1 \neq s_2 \in \Sigma$ satisfy $M(s_1) \neq M(s_2)$ and $|\Sigma| \geq n/t$.

Let A be a streaming algorithm whose canonical function is F and let algorithm A' be an amplification of algorithm A that succeeds with probability $1 - 1/(10\log(cn))$ (by making $O(\log\log(cn))$ repetitions and taking the majority). Consider an instance of the MESSAGE problem Π^{MSG}_{Σ}, and proceed similarly to the proof of Theorem 3.1. We provide a self-contained analysis for completeness. Alice and Bob perform the following protocol. Alice starts an execution of algorithm A' using the shared randomness. For input $s^* \in \Sigma$, she inserts s^* stream items to algorithm A' and sends the state (memory contents) of this algorithm A' to Bob. In order to get query access to F_{s^*} at index x, Bob makes a fresh copy A_0 of algorithm A', continues the algorithm's execution (using the shared randomness), inserts x stream items to algorithm A_0 and finally reads its output. Bob uses this query access to simulate the Binary Search algorithm on F_{s^*} (with the goal of recovering $M(s^*)$). He then infers which bucket corresponds to his result, and outputs the representative of that bucket (which is s^* if he recovers $M(s^*)$).

If the Binary Search algorithm were executed with true query access to F_{s^*}, then it would have output $M(s^*)$ and would have made a sequence of queries X_{BS} to F_{s^*}. This sequence depends only on F_{s^*}, and in particular independent of the random coins of algorithm A'. Thus by a union bound, algorithm A' succeeds on all queries $x \in X_{BS}$ (i.e. outputs the corresponding $F_{s^*}(x)$) with probability at least $1 - \log(cn) \cdot 1/(10\log(cn)) = 9/10$. Hence,

when Bob simulates the Binary Search algorithm using the streaming algorithm A', then with probability 9/10 the execution is identical to running the Binary Search algorithm with true query access to F_{s^*}. Thus with this probability 9/10, Bob recovers $M(s^*)$, and hence outputs s^*, which concludes the correctness analysis of the communication protocol.

By Lemma 2.2, the message Alice sends must contain $\Omega(\log|\Sigma|) \geq \Omega(\log(n/t))$ bits, and thus algorithm A' must use $\Omega(\log(n/t))$ bits of space. Recall that algorithm A' is made of $O(\text{loglog}(cn))$ copies of algorithm A and thus algorithm A must use $\Omega(\frac{\log(n/t)}{\text{loglog}(cn)})$ bits of space. ◀

▶ Remark 4.2. This proof can be easily generalized to prove Theorem 1.6. The first extension is by replacing the Binary Search algorithm and the corresponding buckets with any deterministic algorithm Q that returns a subset containing s^*. In order to generalize Q to any PD algorithm Y, consider the canonical function of Y instead of the mapping M, and apply the same proof. It holds because the crucial property of the Binary Search algorithm was the existence of the mapping M. Then by an additional union bound, both algorithms Q and A' succeed with probability 8/10 (as in the proof of Theorem 3.1).

We now generalize Lemma 4.1 to a larger family of patterns in F, where each pattern is characterized by a parameter $k \in [n]$, and appears at index $x \in [0, (c+1)n - k]$ such that $F(x) = 0$ and $F(x+k) = 1$. These patterns are allowed to overlap with each other (for different values of k). Denote such a pattern by "$0?^{k-1}1$", where each question mark can represent either 0 or 1, and the number of question marks is $k-1 < n$. A copy of this pattern can be found in $O(\log \frac{n}{k})$ queries to F_{s^*} by a binary search on the grid $(0, k, ..., \lceil \frac{cn}{k} \rceil k)$, since $F_{s^*}(0) = 0$ and $F_{s^*}(\lceil \frac{cn}{k} \rceil k) = 1$. Hence, if there exists k for which this pattern appears at most t times in F, then the communication protocol above can be adjusted to imply that algorithm A must use at least $\Omega(\frac{\log(n/t)}{\text{loglog}(cn/k)}) \geq \Omega(\frac{\log(n/t)}{\text{loglog}(cn)})$ bits of space. The only change in the proof is in the number of queries that Bob makes, which affects the number of repetitions in algorithm A', and thus only affects the loglog term.

▶ **Corollary 4.3.** *If for some $k \leq n$ the pattern "$0?^{k-1}1$" appears at most t times in F, then every PD streaming algorithm for problem $\Pi^{AC}_{c,n}$ whose canonical function is F, must use $\Omega(\frac{\log(n/t)}{\text{loglog}(cn)})$ bits of space.*

4.2 Scenario Two

In this scenario, for every $k \leq n$ the pattern "$0?^{k-1}1$" appears at least t times in F.

▶ **Lemma 4.4.** *If for all $k \in [n]$, the pattern "$0?^{k-1}1$" appear at least t times in F, then every PD streaming algorithm for problem $\Pi^{AC}_{c,n}$ whose canonical function is F, must use $\Omega(\frac{\log n}{\log(cn/t)+\text{loglog}\, n})$ bits of space.*

Proof. In this case, there is an algorithm for the Shift Finding problem $\Pi^{SF}_{c,n}$ using $q = O(\frac{cn \log n}{t})$ queries to F_{s^*}, as follows.

1. let $S = [0, n]$
2. repeat the following $\frac{10cn \log n}{t}$ times:
   a. pick $r \in [cn]$ uniformly at random and query $F_{s^*}(r)$
   b. let $S \leftarrow \{s \in S : \ F(s+r) = F_{s^*}(r)\}$
3. if $|S| = 1$, return $s \in S$; else return FAIL

The final set S clearly contains the shift s^*. It remains to show that all $s \neq s^*$ are removed from the set S with high probability.

Fix $s \in [n], s \neq s^*$. There are t values for $r \in [cn]$ for which $F(s^*+r) \neq F(s+r)$, as follows. Assume without loss of generality that $s^* < s$ and denote $k = s - s^* \in [n]$. Let l be a location that corresponds to the pattern "$0?^{k-1}1$" in F, i.e. $F(l) = 0$ and $F(l+k) = 1$. If $l \in [s^*+1, s^*+cn]$, then there is $r \in [cn]$ such that $s^* + r = l$, for which $F(s^*+r) = 0 \neq F(l+k) = F(s+r)$. There are at least t locations for this pattern (i.e. possible values for l), thus it remains to show that indeed $l \in [s^*+1, s^*+cn]$. It must be that $l + k > n$ since $F(x) = 0$ for all $x \leq n$, and similarly $l \leq cn$ since $F(x) = 1$ for all $x > cn$. Hence $l \in [n-k+1, cn] \subset [s^*+1, s^*+cn]$, and thus there are t values for $r \in [cn]$ for which $F(s^*+r) \neq F(s+r)$ (each value for r corresponds to a possible value for l).

Thus, in each repetition, s is removed from the set S with probability at least $\frac{t}{cn}$. The probability s is not removed after $\frac{10cn\log n}{t}$ repetitions is $(1-\frac{t}{cn})^{(10cn\log n)/t} < \frac{1}{n^2}$. By a union bound, all $s \neq s^*$ are removed with probability $1 - \frac{1}{n}$, which concludes the correctness analysis of the algorithm for problem $\Pi^{SF}_{c,n}$.

By Theorem 3.1, every PD streaming algorithm for the approximate counting problem $\Pi^{AC}_{c,n}$ with a canonical function F must use $\Omega(\frac{\log n}{\log((cn\log n)/t)})$ bits of space. ◀

4.3 Concluding the Proof of Theorem 1.3

Concluding the two scenarios, set $t = n/2^{\sqrt{\log n \cdot \log\log(cn)}}$ and get by Corollary 4.3 and Lemma 4.4 that every PD streaming algorithm for the approximate counting problem $\Pi^{AC}_{c,n}$ must use

$$\Omega(\min\{\tfrac{\log(n/t)}{\log\log(cn)}, \tfrac{\log n}{\log((cn/t)\log n)}\}) = \Omega(\tfrac{\log n}{\sqrt{\log n \log\log(cn)} + \log c})$$

bits of space, which boils down to $\Omega(\sqrt{\frac{\log n}{\log\log n}})$ for $c < 2^{\sqrt{\log n \log\log n}}$.

5 Shift Finding Algorithm

One can hope to prove tighter lower bounds for PD streaming algorithms for the approximate counting problem $\Pi^{AC}_{c,n}$, and a possible approach is by solving the Shift Finding problem $\Pi^{SF}_{c,n}$ using polylog n queries. Recall that in problem $\Pi^{SF}_{c,n}$, the input is a string $P \in \{0,1\}^{(c-1)n}$, which can be represented by a string F which is a concatenation of n zeros, P and then n ones; and query access to a shifted version of F with shift s^*, denoted F_{s^*}. As stated in Theorem 1.8, we show a deterministic algorithm for problem $\Pi^{SF}_{c,n}$ using $O(\sqrt{cn})$ queries (Algorithm 1), and we leave open the question whether it is the right bound. The proof relies on an efficient verification algorithm that for input s, uses 2 queries and returns "yes" if and only if $s = s^*$, as stated in Lemma 1.9 and described next.

Proof of Lemma 1.9. Denote by $l \in [n+1, cn+1]$ the smallest number such that $F(l) = 1$, and by $r \in [n, cn]$ the largest number such that $F(r) = 0$. For input $s \in [0, n]$, the verification algorithm returns "no" if $F_{s^*}(l-s) = 0$ or $F_{s^*}(r-s) = 1$, and otherwise returns "yes".

If $s = s^*$, then $F_{s^*}(x-s) = F(x)$ and the verification algorithm outputs "yes". If $s > s^*$, then $s^* - s + l < l$ and thus $F_{s^*}(l-s) = F(s^*-s+l) = 0$ and the verification algorithm outputs "no". Similarly, if $s < s^*$ then $F_{s^*}(r-s) = 1$ and the verification algorithm outputs "no". ◀

▶ Remark 5.1. There is a randomized algorithm for problem $\Pi^{SF}_{c,n}$ using $\tilde{O}_c(\sqrt{n})$ queries that is similar to the proof of Theorem 1.3 in Section 4. It proceeds by considering those two scenarios. In scenario one, instead of constructing the set Σ, query witnesses for all the t

possible shifts using $2t$ queries and hence recover the unknown shift s^*. In scenario two, the proof of Theorem 1.3 shows how to find the unknown shift s^* in $O(\frac{cn}{t}\log n)$ queries with high probability. Hence, by setting $t = \sqrt{cn\log n}$, this algorithm finds the unknown shift in $O(\max\{t+\log(cn), \frac{cn}{t}\log n\}) \leq O(\sqrt{cn\log n})$ queries with high probability.

Next is a slight improvement, a deterministic algorithm in $O(\sqrt{cn})$ queries, proving Theorem 1.8.

Algorithm 1 Deterministic Shift Finding in $O(\sqrt{cn})$ queries.

Input: n, c, F and query access to F_{s^*}
Output: s^*

1: $Q \leftarrow (F_{s^*}(0), F_{s^*}(\sqrt{cn}), F_{s^*}(2\sqrt{cn}), ..., F_{s^*}(cn))$
2: let $S \leftarrow \left\{s \in [0,n] : \forall i \in [0,\sqrt{cn}], F_s(i\sqrt{cn}) = Q(i)\right\}$ ▷ i.e. the set of all shifts that could have produced Q
3: **for** $s \in S$ **do**
4: check the witness of s
5: **if** $s = s^*$ **then** return s

▶ **Lemma 5.2.** *The set S in Algorithm 1 is of size $O(\sqrt{cn})$.*

Proof. Assume by contradiction that $|S| \geq \sqrt{cn}+1$. Hence by the pigeonhole principle, there exists $s_1 < s_2 \in S$ such that $s_1 = s_2 \mod \sqrt{cn}$. Hence for all $i \in [0, \sqrt{cn} - \frac{s_2-s_1}{\sqrt{cn}}]$,

$$Q(i) = F_{s_2}(i\sqrt{cn}) = F_{s_1}(s_2 - s_1 + i\sqrt{cn}) = Q(\tfrac{s_2-s_1}{\sqrt{cn}} + i),$$

where the first and last transitions hold since $s_1, s_2 \in S$ and $\frac{s_2-s_1}{\sqrt{cn}}$ is an integer number, and the second transition is by definition. Thus Q has a period of length $\frac{s_2-s_1}{\sqrt{cn}} \leq \lfloor\frac{s_2}{\sqrt{cn}}\rfloor$. However, for $i \in [\sqrt{cn} - \lfloor\frac{s_2}{\sqrt{cn}}\rfloor + 1, \sqrt{cn}]$ the values that Q get are $Q(i) = F_{s_2}(i\sqrt{cn}) = 1$ since $s_2 + i\sqrt{cn} \geq cn$; thus all entries in Q are equal 1, which contradicts the fact that $Q(0) = 0$, and thus completes the proof. ◀

Algorithm 1 returns the shift s^* since $s^* \in S$ and by the correctness of the verifier in Lemma 1.9. The number of queries Algorithm 1 makes is $O(|S| + |Q|) = O(\sqrt{cn})$, which proves Theorem 1.8.

References

1 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 20–29, 1996. doi:10.1145/237814.237823.

2 Alexandr Andoni, Piotr Indyk, Dina Katabi, and Haitham Hassanieh. Shift finding in sub-linear time. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 457–465, 2013. doi:10.1137/1.9781611973105.33.

3 Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. *J. ACM*, 69(2):17:1–17:33, 2022. doi:10.1145/3498334.

4 Amit Chakrabarti, Prantar Ghosh, and Manuel Stoeckl. Adversarially robust coloring for graph streams. In *13th Innovations in Theoretical Computer Science Conference, ITCS*, volume 215 of *LIPIcs*, pages 37:1–37:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.37.

5 Peter Dixon, A. Pavan, and N. V. Vinodchandran. On pseudodeterministic approximation algorithms. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS*, volume 117 of *LIPIcs*, pages 61:1–61:11. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.MFCS.2018.61.

6 Peter Dixon, A. Pavan, and N. V. Vinodchandran. Complete problems for multi-pseudodeterministic computations. In *12th Innovations in Theoretical Computer Science Conference, ITCS*, volume 185 of *LIPIcs*, pages 66:1–66:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ITCS.2021.66.

7 Peter Dixon, A. Pavan, Jason Vander Woude, and N. V. Vinodchandran. Pseudodeterminism: promises and lowerbounds. In *STOC '22: 54th Annual ACM Symposium on Theory of Computing*, pages 1552–1565, 2022. doi:10.1145/3519935.3520043.

8 Philippe Flajolet. Approximate counting: A detailed analysis. *BIT*, 25(1):113–134, 1985. doi:10.1007/BF01934993.

9 Eran Gat and Shafi Goldwasser. Probabilistic search algorithms with unique answers and their cryptographic applications. *Electron. Colloquium Comput. Complex.*, TR11-136, 2011. URL: https://eccc.weizmann.ac.il/report/2011/136, arXiv:TR11-136.

10 Sumanta Ghosh and Rohit Gurjar. Matroid intersection: A pseudo-deterministic parallel reduction from search to weighted-decision. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, volume 207 of *LIPIcs*, pages 41:1–41:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.APPROX/RANDOM.2021.41.

11 Michel X. Goemans, Shafi Goldwasser, and Dhiraj Holden. Doubly-efficient pseudo-deterministic proofs. *CoRR*, abs/1910.00994, 2019. arXiv:1910.00994.

12 Oded Goldreich. Multi-pseudodeterministic algorithms. *Electron. Colloquium Comput. Complex.*, TR19-012, 2019. URL: https://eccc.weizmann.ac.il/report/2019/012, arXiv:TR19-012.

13 Oded Goldreich, Shafi Goldwasser, and Dana Ron. On the possibilities and limitations of pseudodeterministic algorithms. In *Innovations in Theoretical Computer Science, ITCS*, pages 127–138. ACM, 2013. doi:10.1145/2422436.2422453.

14 Shafi Goldwasser and Ofer Grossman. Perfect bipartite matching in pseudo-deterministic RNC. *Electron. Colloquium Comput. Complex.*, TR15-208, 2015. URL: https://eccc.weizmann.ac.il/report/2015/208, arXiv:TR15-208.

15 Shafi Goldwasser, Ofer Grossman, and Dhiraj Holden. Pseudo-deterministic proofs. In *9th Innovations in Theoretical Computer Science Conference, ITCS*, volume 94 of *LIPIcs*, pages 17:1–17:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ITCS.2018.17.

16 Shafi Goldwasser, Ofer Grossman, Sidhanth Mohanty, and David P. Woodruff. Pseudo-deterministic streaming. In *11th Innovations in Theoretical Computer Science Conference, ITCS*, volume 151 of *LIPIcs*, pages 79:1–79:25, 2020. doi:10.4230/LIPIcs.ITCS.2020.79.

17 Shafi Goldwasser, Russell Impagliazzo, Toniann Pitassi, and Rahul Santhanam. On the pseudo-deterministic query complexity of NP search problems. In *36th Computational Complexity Conference, CCC*, volume 200 of *LIPIcs*, pages 36:1–36:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CCC.2021.36.

18 André Gronemeier and Martin Sauerhoff. Applying approximate counting for computing the frequency moments of long data streams. *Theory Comput. Syst.*, 44(3):332–348, 2009. doi:10.1007/s00224-007-9048-z.

19 Ofer Grossman. Finding primitive roots pseudo-deterministically. *Electron. Colloquium Comput. Complex.*, TR15-207, 2015. URL: https://eccc.weizmann.ac.il/report/2015/207, arXiv:TR15-207.

20 Ofer Grossman, Meghal Gupta, and Mark Sellke. Tight space lower bound for pseudo-deterministic approximate counting. arXiv preprint, 2023. arXiv:2304.01438.

21 Ofer Grossman and Yang P. Liu. Reproducibility and pseudo-determinism in log-space. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 606–620. SIAM, 2019. doi:10.1137/1.9781611975482.38.

22 Moritz Hardt and David P. Woodruff. How robust are linear sketches to adaptive inputs? In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, pages 121–130, 2013. doi:10.1145/2488608.2488624.

23 Haitham Hassanieh, Fadel Adib, Dina Katabi, and Piotr Indyk. Faster GPS via the sparse fourier transform. In *The 18th Annual International Conference on Mobile Computing and Networking, Mobicom*, pages 353–364. ACM, 2012. doi:10.1145/2348543.2348587.

24 Dhiraj Holden. A note on unconditional subexponential-time pseudo-deterministic algorithms for BPP search problems. *CoRR*, abs/1707.05808, 2017. arXiv:1707.05808.

25 Haim Kaplan, Yishay Mansour, Kobbi Nissim, and Uri Stemmer. Separating adaptive streaming from oblivious streaming using the bounded storage model. In *Advances in Cryptology – CRYPTO*, volume 12827 of *Lecture Notes in Computer Science*, pages 94–121. Springer, 2021. doi:10.1007/978-3-030-84252-9_4.

26 Zhenjian Lu, Igor Carboni Oliveira, and Rahul Santhanam. Pseudodeterministic algorithms and the structure of probabilistic time. In *STOC '21: 53rd Annual ACM Symposium on Theory of Computing*, pages 303–316, 2021. doi:10.1145/3406325.3451085.

27 Jérémie O. Lumbroso. How Flajolet processed streams with coin flips. *CoRR*, abs/1805.00612, 2018. arXiv:1805.00612.

28 Robert Morris. Counting large numbers of events in small registers. *Commun. ACM*, 21(10):840–842, 1978. doi:10.1145/359619.359627.

29 Jelani Nelson and Huacheng Yu. Optimal bounds for approximate counting. In *Proceedings of the 41st ACM Symposium on Principles of Database Systems*, PODS, pages 119–127, 2022. doi:10.1145/3517804.3526225.

30 Igor Carboni Oliveira and Rahul Santhanam. Pseudodeterministic constructions in subexponential time. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing, STOC*, pages 665–677, 2017. doi:10.1145/3055399.3055500.

31 Igor Carboni Oliveira and Rahul Santhanam. Pseudo-derandomizing learning and approximation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, volume 116 of *LIPIcs*, pages 55:1–55:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.55.

32 Manuel Stoeckl. Streaming algorithms for the missing item finding problem. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 793–818, 2023. doi:10.1137/1.9781611977554.ch32.

Minimum Chain Cover in Almost Linear Time

Manuel Cáceres
Department of Computer Science, University of Helsinki, Finland

Abstract

A minimum chain cover (MCC) of a k-width directed acyclic graph (DAG) $G = (V, E)$ is a set of k chains (paths in the transitive closure) of G such that every vertex appears in at least one chain in the cover. The state-of-the-art solutions for MCC run in time $\tilde{O}(k(|V| + |E|))$ [Mäkinen et at., TALG], $O(T_{MF}(|E|) + k|V|)$, $O(k^2|V| + |E|)$ [Cáceres et al., SODA 2022], $\tilde{O}(|V|^{3/2} + |E|)$ [Kogan and Parter, ICALP 2022] and $\tilde{O}(T_{MCF}(|E|) + \sqrt{k}|V|)$ [Kogan and Parter, SODA 2023], where $T_{MF}(|E|)$ and $T_{MCF}(|E|)$ are the running times for solving maximum flow (MF) and minimum-cost flow (MCF), respectively.

In this work we present an algorithm running in time $O(T_{MF}(|E|) + (|V| + |E|) \log k)$. By considering the recent result for solving MF [Chen et al., FOCS 2022] our algorithm is the first running in almost linear time. Moreover, our techniques are deterministic and derive a deterministic near-linear time algorithm for MCC if the same is provided for MF. At the core of our solution we use a modified version of the mergeable dictionaries [Farach and Thorup, Algorithmica], [Iacono and Özkan, ICALP 2010] data structure boosted with the SIZE-SPLIT operation and answering queries in amortized logarithmic time, which can be of independent interest.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Network flows; Theory of computation → Sorting and searching

Keywords and phrases Minimum chain cover, directed acyclic graph, minimum flow, flow decomposition, mergeable dictionaries, amortized running time

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.31

Category Track A: Algorithms, Complexity and Games

Related Version *Previous Version*: `https://arxiv.org/abs/2305.02166`

Funding This work was funded by the Academy of Finland (grants No. 352821, 328877).

Acknowledgements I am very grateful to Alexandru I. Tomescu and Brendan Mumey for initial discussions on flow decomposition of a minimum flow representing an MPC. I am also very grateful to the anonymous reviewers for their useful suggestions.

1 Introduction

Computing a minimum-sized set of chains covering all vertices of a DAG $G = (V, E)$ is a well known poly-time solvable problem [15, 18], with many applications in widespread research fields such as bioinformatics [34, 7, 13, 4, 8, 32]. Here we call such an object a *minimum chain cover* (an MCC) $\mathcal{C}$ containing k chains $\mathcal{C} = \{C_1, \ldots, C_k\}$, which are paths in the transitive closure of G. The *size* k of an MCC is known as the *width* of G and equals the maximum number of pairwise unreachable vertices (antichain) of G, by Dilworth's theorem [15] on partially ordered sets (posets).

The history of MCC. It was Fulkerson [18] in the 1950s the first to show a poly-time algorithm for posets (transitive DAGs). His algorithm reduces the problem to finding a maximum matching in a bipartite graph with $2|V|$ vertices and $|E|$ edges, and thus can be

solved in $O(|E|\sqrt{|V|})$ time by using the Hopcroft-Karp algorithm [21][1]. Improvements on these ideas were derived in the $O(|V|^2 + k\sqrt{k}|V|)$ and $O(\sqrt{|V|}|E| + k\sqrt{k}|V|)$ time algorithms of Chen and Chen [10, 11], and the $O(k|V|^2)$ time algorithm of Felsner et al. [17]. In the same article, Felsner et al. showed a combinatorial approach to compute a *maximum antichain* (MA) in near-linear time for the cases $k = 2, 3, 4$, which was latter generalized to $O(f(k)(|V| + |E|))$ [5], $f(k)$ being an exponential function. State-of-the-art approaches improve exponentially on its running time dependency on k. These approaches solve the strongly related problem of *minimum path cover* (MPC). An MPC $\mathcal{P}$ is a minimum-sized set of *paths* covering the vertices of G, and thus it is also a valid chain cover. Moreover, since every MCC can be transformed into an MPC (by connecting consecutive vertices in the chains), the *size* of an MPC also equals the width k.

The state-of-the-art for MCC. Mäkinen et al. [29] provided an algorithm for MPC, running in time $O(k(|V| + |E|) \log |V|) = \widetilde{O}(k(|V| + |E|))$ while Cáceres et al. [6] presented the first $O(k^2|V| + |E|)$ parameterized linear time algorithm. Both algorithms are based on a classical reduction to *minimum flow* [30], which we will revisit later. In the same work, Cáceres et al. [6] showed how to compute an MPC in time $O(T_{MF}(|E|) + ||\mathcal{P}||)$ and a MA in time $O(T_{MF}(|E|))$, where $T_{MF}(|E|)$ is the running time for solving *maximum flow* (MF) and $||\mathcal{P}||$ is the total length of the reported MPC. As such, by using the recent result for MF of Chen et al. [9] we can solve MPC and MA in (almost) optimal (input+output size) time. However, the same does not apply to MCC as the total length of an MCC can be exactly $|V|$ (e.g. by removing repeated vertices) while the total length of an MPC can be $\Omega(k|V|)$ in the worst case, as shown in Figure 1.

The $k|V|$ barrier was recently overcame by Kogan and Parter [26, 27] by reducing the total length of an MPC. They obtain this improvement by using *reachability shortcuts* and by devising a more involved reduction to *minimum cost flow* (MCF). Their algorithms run in time $\widetilde{O}(|E| + |V|^{3/2})$ [26] and $\widetilde{O}(\sqrt{k}|V| + |E|^{1+o(1)})$ [27] using the MCF algorithms of Bran et al. [35] and Chen et al. [9], respectively.

In this paper we present an algorithm for MCC improving its running time dependency on k exponentially w.r.t. the state-of-the-art.

▶ **Theorem 1.** *Given a k-width DAG $G = (V, E)$ we can compute a minimum chain cover in time $O(T_{MF}(|E|) + (|V| + |E|) \log k)$, where $T_{MF}(|E|)$ is the time for solving maximum flow.*

Thus by applying the flow algorithm of Chen et al. [9] we solve the problem in almost-linear time for the first time.

▶ **Corollary 2.** *Given a k-width DAG $G = (V, E)$ we can compute a minimum chain cover in time $O(|E|^{1+o(1)})$ w.h.p.*

Moreover, our solution in Theorem 1 uses a MF solver as a black box and it is deterministic otherwise. Therefore, we provide a deterministic MCC solution in *near-linear time* if one is found for MF. At the core of our solution we use mergeable dictionaries boosted with the SIZE-SPLIT operation to efficiently transform the flow outputted by the MF solver into an MCC. *Mergeable dictionaries* is a data structure maintaining a dynamic partition $\mathcal{K}$ of the natural numbers $\{1, \ldots, k\}$ (the reuse of k is intentional as this is how our approach will use the data structure), starting from the partition $\mathcal{K} = \{\{1, \ldots, k\}\}$ and supporting the following operations, for $K, K_1, K_2 \in \mathcal{K}$:

[1] Recent fast solutions for maximum matching do not speed up this approach as one needs to compute the transitive closure of the DAG first.

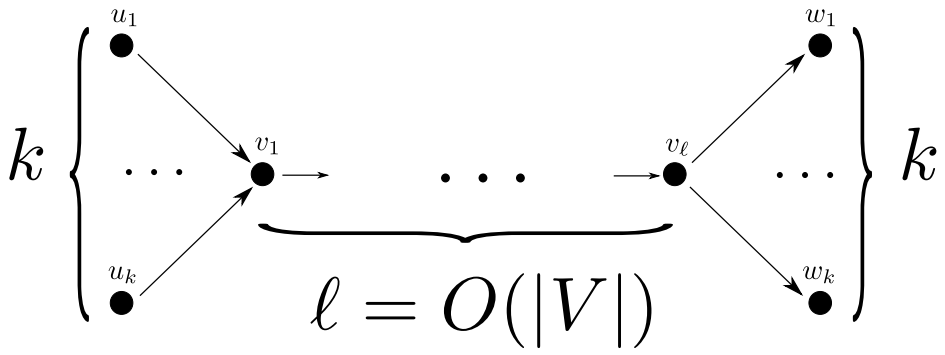


Figure 1 Example k-width DAG where every MPC $\mathcal{P}$ has total length $||\mathcal{P}|| = \Omega(k|V|)$. Indeed, every path in an MPC must start from some u_i and traverse the middle path $v_1, \ldots, v_\ell$ until reaching some w_j, since otherwise it is not possible to cover the rest of the graph minimally. Moreover, if $k = \ell = |V|/3$, then $||\mathcal{P}|| = \Omega(|V|^2)$. On the other hand, there is always an MCC $\mathcal{C}$ of total size $||\mathcal{C}|| = |V|$ (in this case only one of the chains cover the vertices of the middle path).

- SEARCH(K, j): returns $\max_{i \in K, i \leq j} i$ if any such element exists[2].
- MERGE(K_1, K_2): replaces K_1 and K_2 by $K_1 \cup K_2$ in $\mathcal{K}$.
- SPLIT(K, j): replaces K by $K' = \{i \in K \mid i \leq j\}$ (only if $K' \neq \emptyset$) and $K \setminus K'$ (only if $K \neq K'$) in $\mathcal{K}$.

Note that the MERGE operation does not assume that $\max_{i \in K_1} i < \min_{i \in K_2} i$ (sets are non-overlapping) as generated by the SPLIT operation. If the previous condition (or the analogous $\max_{i \in K_2} i < \min_{i \in K_1} i$) is assumed, the operation is known as JOIN. Mergeable dictionaries have applications in Lempel-Ziv decompression [16, 3, 31], mergeable trees [19] and generalizations of union-find-split [28]. In this paper we show how to modify them to obtain a fast MCC algorithm. Next, we review the different approaches used to implement mergeable dictionaries in the literature.

Mergeable dictionaries. The first efficient implementation of mergeable dictionaries can be derived[3] from the *segment merge* strategy proposed by Farach and Thorup [16] to efficiently MERGE two self-balanced binary search trees, assuming that operations SEARCH, SPLIT and JOIN are implemented in logarithmic time. The authors showed how to implement the MERGE operation by minimally SPLITing both sets into non-overlapping sets and pairwise JOINing the resulting parts. They proved that after $|V|+|E|$ operations (the abuse of notation is again intentional) their strategy works in $O(\log k \cdot \log(|V| + |E|))$ amortized time per operation. Later, Iacono and Özkan [22] presented a mergeable dictionaries implementation running in $O(\log k)$ amortized time per operation, based on biased skip lists [1]. The same amortized running time was later achieved by Karczmarz [24] with a very simple approach using tries [14] to represent the sets.

[2] This query is also known as *predecessor* query in the literature.
[3] The authors of [16] do not define mergeable dictionaries formally.

Our algorithm for MCC computes a minimum flow f^* encoding an MPC and then extracts an MCC from f^* by processing G in topological order and querying mergeable dictionaries boosted with the SIZE-SPLIT operation. Formally, we require the following operations, for $K, K_1, K_2 \in \mathcal{K}, s \in \{1, \ldots, |K| - 1\}$:

- SOME(K): returns some element $i \in K$.
- MERGE(K_1, K_2): replaces K_1 and K_2 by $K_1 \cup K_2$ in $\mathcal{K}$.
- SIZE-SPLIT(K, s): replaces K by $K' \subseteq K, |K'| = s$ and $K \setminus K'$ in $\mathcal{K}$.

In Section 3 we show how to implement these operations in $O(\log k)$ amortized time each. Then, in Section 4 we show our MCC algorithm using the previously described data structure. Besides the theoretical improvement already explained, we highlight the simplicity of our solutions, both in our proposal for boosted mergeable dictionaries as well as in our algorithm for MCC.

2 Notation and preliminaries

Graphs. For a vertex $v \in V$ we denote $N^-(v)$ ($N^+(v)$) to be the set of *in(out)-neighbors* of v that is $N^-(v) = \{u \in V \mid (u, v) \in E\}$ ($N^+(v) = \{w \in V \mid (v, w) \in E\}$). A v_1v_ℓ-path is a sequence of vertices $P = v_1, \ldots, v_\ell$ such that $(v_i, v_{i+1}) \in E$ for $i \in \{1, \ldots, \ell - 1\}$, in this case we say that v_1 *reaches* v_ℓ. We say that P is *proper* if $\ell \geq 2$ and that P is a cycle if $v_1 = v_\ell$. A *directed acyclic graph* (DAG) is a graph without proper cycles. In a DAG we can compute, in linear time [23, 33], a *topological order* $v_1, \ldots, v_{|V|}$ of its vertices such that for every $i < j$, $(v_j, v_i) \notin E$. In this paper we assume that G is a DAG and since our algorithms run in time $\Omega(|V| + |E|)$, we assume that an input *topological order* is given. A chain is a sequence of vertices $C = v_1, \ldots, v_{\ell'}$ such that for each $i \in \{1, \ldots, \ell' - 1\}$ v_i reaches v_{i+1}. We denote $|C| = \ell'$ to the length of the chain. A *chain cover* $\mathcal{C}$ is a set of chains such that every vertex appears in some chain of $\mathcal{C}$. We say that it is a *chain decomposition* if every vertex appears in exactly one chain of $\mathcal{C}$ and a *path cover* if every chain of $\mathcal{C}$ is a path, in this case we denote it $\mathcal{P}$ instead. We denote $||\mathcal{C}||$ to the total length of a chain cover that is $||\mathcal{C}|| = \sum_{C \in \mathcal{C}} |C|$. An antichain is a subset of vertices $A \subseteq V$ such that for $u, v \in A, u \neq v$, u does not reach v. The minimum size of a chain cover is known as the *width* of G and we denote it k.

Flows. Given a function of *demands* $d : E \to \mathbb{N}_0$ and $s, t \in V$, an *st-flow* is a function $f : E \to \mathbb{N}_0$ satisfying *flow conservation* that is $inFlow_v := \sum_{u \in N^-(v)} f(u, v) = \sum_{w \in N^+(v)} f(v, w) := outFlow_v$ for each $v \in V \setminus \{s, t\}$, and *satisfying the demands* that is $f(e) \geq d(e)$ for each $e \in E$. A *flow decomposition* of k (the reuse of notation is again intentional) paths of f is a collection $\mathcal{D} = P_1, \ldots, P_k$ of st-paths such that for each edge $e \in E$, $f(e) = |\{P_i \in \mathcal{D} \mid e \in P_i\}|$.[4] The *size* $|f|$ of f is defined as the net flow entering t (equivalently exiting s by flow conservation) that is $|f| = inFlow_t - outFlow_t$. The problem of *minimum flow* looks for a feasible st-flow of minimum size. Finding an MPC can be reduced to decompose a specific minimum flow [30], we will revisit this reduction in Section 4. The same techniques applied for the problem of *maximum flow* can be used in the context of the minimum flow problem [12, 2]. In fact, for MPC one can directly apply a maximum flow algorithm with *capacities* at most $|V|$ [6, Theorem 2.2 (full version)].

[4] This is a simplified definition of flow decomposition which suffices for our purposes.

Data structures. A *self-balancing binary search tree* such as an AVL-tree or a red-black tree [20] is a binary search tree supporting operations SEARCH, SPLIT and JOIN in logarithmic time each (in the worst case). A (binary) *trie* [14] is a binary tree representing a set of integers by storing their *binary representation* as *root-to-leaf* paths of the trie. In our tries all root-to-leaf paths have the same length $\lfloor \log k \rfloor + 1$. For a data structure supporting a set of operations, and a potential function ϕ capturing the state of the data structure, we say that the *amortized* time of an operation equals to its (worst-case) running time plus the change $\Delta\phi$ in the potential triggered by the operation. If we apply a sequence of $O(|V| + |E|)$ operations whose amortized time is $O(\log k)$ the total (worst-case) running time is $O((|V| + |E|) \log k)$.

3 Mergeable dictionaries with SIZE-SPLIT

We show how to implement the *boosted* mergeable dictionaries supporting operations SOME, MERGE and SIZE-SPLIT in $O(\log k)$ amortized time each. To achieve this result we modify an existing solution of mergeable dictionaries implementing operations SEARCH, MERGE and SPLIT by adding the SELECT operation. Formally for $K \in \mathcal{K}, s \in 1, \ldots, |K|$,

- SELECT(K, s): returns the s-th smallest element in K.

With the SELECT operation we can use (normal) mergeable dictionaries to implement SOME and SIZE-SPLIT as follows:

- SOME$(K) \leftarrow$ SEARCH(K, k).
- SIZE-SPLIT$(K, s) \leftarrow$ SPLIT$(K$,SELECT$(K, s))$.

While for SOME it suffices to do a SEARCH with a known upper bound (recall that k is the maximum element in the universe considered), in the case of SIZE-SPLIT we can first SELECT the corresponding pivot and use this pivot to SPLIT the set by its value, obtaining the desired sizes for the split.

This reduction allows us to obtain the boosted mergeable dictionaries by simply implementing the SELECT operation with logarithmic amortized cost (SIZE-SPLIT can be seen as one call to SELECT followed by a separate call to SPLIT). Moreover, if the implementation of SELECT does not modify the data structure (and thus the potential ϕ), its amortized time equals its (worst-case) running time (as ϕ does not change). We show that this is indeed the case in both the segment merge strategy of Farach and Thorup [16], and the trie implementation of Karczmarz [24], the latter achieving the desired running time.

The mergeable dictionaries based on segment merge represent each set as a self-balancing binary search tree. As such, the SELECT operation can be implemented in $O(\log k)$ time by storing the sub-tree sizes at every node of the tree, which can be maintained (updated when the tree changes) in the same (worst-case) running time as the normal operations of the tree (see e.g. [25]).

Similarly, the implementation of Karczmarz [24] represents every set as a trie of its elements. As discussed in Section 2, a trie stores its elements as their binary representation encoded as (equal length) root-to-leaf paths of the trie. For example, if $k = 6$ the corresponding binary representation of 3 is 011, which is represented as the root-to-leaf path following the left child, then its right child and then its right child. Analogous to binary search trees, the nodes of a trie can be augmented to store the number of leaves in their respective sub-trees. If such augmentation of a trie is performed, then the operation SELECT can be implemented in $O(\log k)$ (worst-case) time similar to the implementation on binary search trees, since the leaves in the trie follow the same order as the elements they represent:

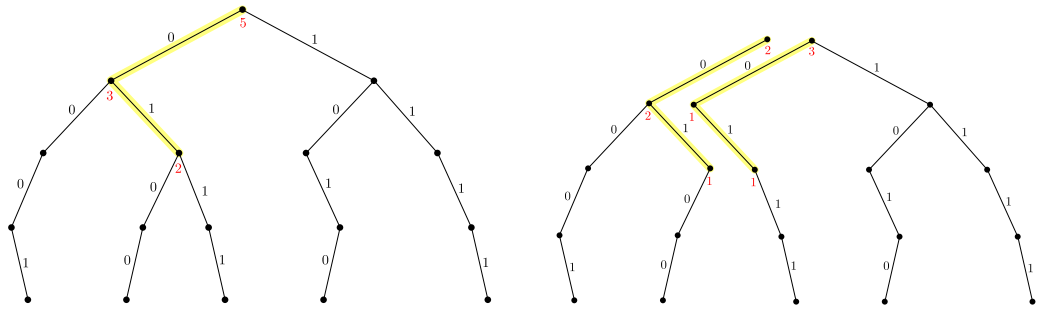


Figure 2 Result of calling SIZE-SPLIT($K = \{1, 4, 7, 10, 15\}, 2$) on a trie representation of boosted mergeable dictionaries. The binary representation of numbers in the set are spelled as root-to-leaf paths of the trie. Some number of leaves' counters are written in red under their respective nodes. Those counters are used to answer $4 \leftarrow \text{SELECT}(K, 2)$. After this, $\text{SPLIT}(K, 4)$ is performed. The path P representing the longest prefix (01) between the binary representations of 4 (0100) and 7 (0111) is highlighted in yellow. The right figure shows the end result, note that only the counters in the nodes of P change.

for a query SELECT(s) on a trie node, it suffices to look at the number of leaves l (elements) under the left child, if $l \geq s$ we continue to the left child answering SELECT(s), otherwise we continue to the right child answering SELECT($s - l$).

Karczmarz [24] showed that SPLIT(K, j) can be performed by finding the root-to-node path P corresponding to the longest common prefix between the binary representations of SEARCH(K, j) and of the smallest value greater than j in K. It then splits the trie by removing the right children of the nodes of P (to form K') and then joining those nodes as right children of a copy of P (to create $K \setminus K'$). Note that this procedure can be easily augmented to maintain the number of leaves under each node: only the nodes in P (and in its copy) decrease their value by the number of leaves of their lost children. Figure 2 shows an example of the SPLIT operation. MERGE(K_1, K_2) is implemented as a simple recursive algorithm that at every step keeps one of the (common) nodes and then merges the corresponding left and right sub-trees (if one of those sub-trees is empty then it just keeps the other sub-tree, we refer to the original work [24] for details). In this case the maintenance of the number of leaves can be performed when returning from the recursive calls: simply recompute the number of leaves as the sum of the number of leaves of their two children. Each of these computations is a constant time operation, and thus they do not change the asymptotic running time of MERGE. We then obtain the following lemma.

▶ **Lemma 3.** *There exists a data structure maintaining a dynamic partition $\mathcal{K}$ of $\{1, \ldots, k\}$ starting from $\mathcal{K} = \{\{1, \ldots, k\}\}$ and answering operations SOME, MERGE and SIZE-SPLIT such that for a sequence of $n = \Omega(k)$ operations[5] it answers in total $O(n \log k)$ time.*

Proof. We first note that operations SOME, MERGE and SIZE-SPLIT can be implemented in the same asymptotic (worst-case) running time as operations SEARCH, MERGE and SPLIT in the data structure of Karczmarz [24], respectively. Indeed, as previously discussed, SOME is implemented as one call to SEARCH, MERGE is implemented in the same way as in [24] but taking care of the number-of-leaves counters' updates which does not affect the asymptotic running time, and SIZE-SPLIT is implemented as one call to SELECT (in

[5] This requirement comes from the fact that mergeable dictionaries actually start from an empty collection of sets, however, one can create the singleton sets and merge then in total $O(k \log k)$ time.

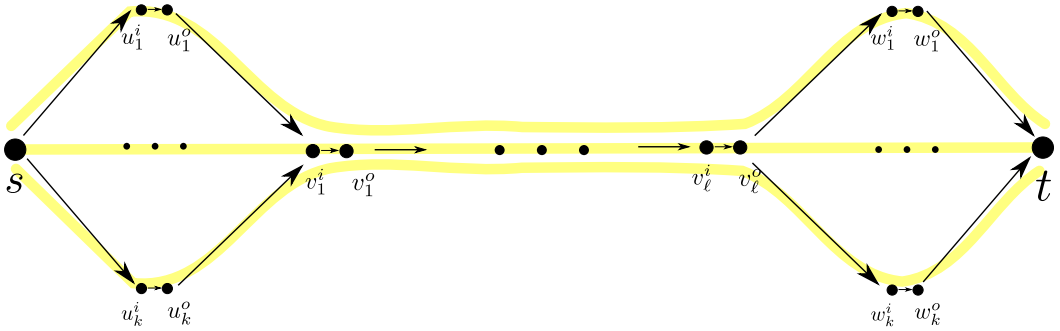


Figure 3 Flow reduction (demands are not shown) of the graph of Figure 1 and a minimum flow on it. Only edges with positive flow are shown. The flow is implicitly presented as a flow decomposition showing every path highlighted in yellow, this flow decomposition corresponds to an MPC of the original DAG.

$O(\log k)$ time) followed by one call to SPLIT (also in $O(\log k)$ time). For the amortized analysis we reuse the potential function used by Karczmarz [24], namely, the number of nodes on all tries. Operations SOME and MERGE follow the same potential change as in [24] and thus have an $O(\log k)$ amortized running time. Finally, since SELECT does not change the total number of nodes (nor any of the tries) the potential change of a SIZE-SPLIT is the same as the one of a SPLIT, that is $O(\log k)$ as in [24]. The lemma follows by using the amortized running times of the data structure's operations. ◀

4 An almost linear time algorithm for MCC

We show how to use Lemma 3 to obtain a fast MCC algorithm. Our algorithm computes a minimum flow f^* encoding an MPC of G and then uses the data structure from Lemma 3 to efficiently extract an MCC from f^*. Next, we describe the well known [30] reduction from MPC to MF by following the notation of [6, Section 2.3 (full version)].

Given the DAG G we build its *flow reduction* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as the graph obtained by adding a global source s, a global sink t and splitting every vertex $v \in V$ into two copies connected by an edge. Additionally, the first copy, v^i, is connected from the in-neighbors of v and the second copy, v^o, is connected to the out-neighbors of v. Formally, $\mathcal{V} = \{s, t\} \cup \{v^i \mid v \in V\} \cup \{v^o \mid v \in V\}$, and $\mathcal{E} = \{(s, v^i) \mid v \in V\} \cup \{(v^o, t) \mid v \in V\} \cup \{(v^i, v^o) \mid v \in V\} \cup \{(u^o, v^i) \mid (u, v) \in E\}$. Note that $|\mathcal{E}| = O(|V| + |E|)$, and that $\mathcal{G}$ is also a DAG. We also define demands on the edges, $d : \mathcal{E} :\to \mathbb{N}_0$, as 1 if the edge is of the form (v^i, v^o), and 0 otherwise. Intuitively, the demands *require* that at least one unit of flow goes through every vertex (of G), which directly translates into to the path cover condition of covering each vertex with at least one path. In fact, every flow decomposition (recall Section 2) of a feasible st-flow f of $\mathcal{G}, d$ corresponds to a path cover of G of size $|f|$. Moreover, every decomposition of a minimum flow f^* of $\mathcal{G}, d$ corresponds to an MPC of G, and thus $k = |f^*|$ [6, Section 2.3 (full version)]. Figure 3 illustrates these ideas with an example.

Since every vertex cannot belong to more than $|V|$ paths in an MPC, the problem can be reduced to maximum flow [2, Theorem 3.9.1]. We summarize these results in the following lemma.

Algorithm 1 Non-optimized pseudocode for our MCC algorithm. A naive implementation of this algorithm obtains an $O(||\mathcal{P}||)$ running time as explained in this manuscript.

Input: A directed acyclic graph $G = (V, E)$.
Output: A minimum chain decomposition $\mathcal{C} = C_1, \ldots, C_k$ of G.

```
(G, d) ← Build the flow reduction of G detailed in Section 4
f*, k = |f*| ← Use Lemma 4 to obtain a minimum flow of (G, d)
Initialize C_i as an empty list for i ∈ {1, ..., k}
I_s ← {1, ..., k}
for v ∈ V in topological order do
    I_v ← Take f*(s, v^i) elements from I_s
    for u^o ∈ N^-(v^i) do
        I_uv ← Take f*(u^o, v^i) elements from I_u
        I_v ← I_v ∪ I_uv
    i ← Choose an element from I_v
    C_i.append(v)
return C_1, ..., C_k
```

▶ **Lemma 4** (Adaptation of [6, Theorem 2.2 (full version)]). *We can compute a flow f^* of $\mathcal{G}, d$ such that every flow decomposition of f^* corresponds to an MPC of G, in time $O(T_{MF}(|E|))$, where $T_{MF}(|E|)$ is the time for solving maximum flow.*

A decomposition algorithm is simple in this case: start from s and follow a path P of positive flow edges until arriving at t, and then update f^* decreasing the flow on the edges of P by one. Repeat this process until no flow remains. The st-paths obtained during the decomposition can be easily transformed into an MPC $\mathcal{P}$ of G (trim s and t and replace v^i, v^o by v on each path, see e.g. [27]).

If implemented carefully (see e.g. [26, Lemma 1.11 (full version)]), the previous algorithm runs in time $O(||\mathcal{P}||)$ and it outputs a valid MCC $\mathcal{P}$ (recall that every MPC is an MCC). However, as shown in Figure 1, $||\mathcal{P}||$ can be $\Omega(k|V|)$ in the worst case. Our algorithm overcomes this barrier by instead directly extracting (from f^*) a minimum chain decomposition (MCD, recall Section 2) and thus its total length is exactly $|V|$.

The main idea to extract an MCD $\mathcal{C} = C_1, \ldots, C_k$ from f^* is to compute, for each vertex $v \in \mathcal{V}$, the set $I_v \subseteq \{1, \ldots, k\}$ of indices such that v would belong to paths $\mathcal{P}_v = \{P_i \mid i \in I_v\}$ in a flow decomposition $\mathcal{D} = P_1, \ldots, P_k$ of f^*. Note that $I_s = I_t = \{1, \ldots, k\}$ by construction of $\mathcal{G}, d$. To efficiently compute these sets, we process the vertices in a topological order of $\mathcal{G}$, for example $s, v_1^i, v_1^o, \ldots, v_{|V|}^i, v_{|V|}^o, t$ (recall that a topological order $v_1, \ldots, v_{|V|}$ of G is assumed as input). When processing vertex v we compute I_v as follows: for every $u \in N^-(v)$ we *take* (exactly) $f^*(u, v)$ elements from I_u, let us denote I_{uv} to these elements. Then, we compute I_v as the *union* $\bigcup_{u \in N^-(v)} I_{uv}$. And finally, we *take an arbitrary element* $i \in I_v$ and append v to C_i. Algorithm 1 shows a pseudocode for this algorithm.

Note that vertices are added to their respective chains in the correct order since they are processed in topological order.

Moreover, since indices are moved from one vertex to the other only if there is an edge between them, consecutive vertices are always connected by a path in G, and thus the lists C_i correspond to proper chains. Finally, adding each vertex to only one such chain ensures that the end result is indeed an MCD (every vertex in exactly one chain).

An important aspect of the algorithm is that exactly $f^*(u^i, v^i)$ ($f^*(s, v^i)$) elements are *taken out* of I_u (I_s). This step is always possible thanks to flow conservation of f^*.

Algorithm 2 Our MCC algorithm running in time $O(T_{MF}(|E|) + (|V| + |E|) \log k)$, where $T_{MF}(|E|)$ is the time for solving maximum flow. The algorithm uses the boosted mergeable dictionaries from Section 3.

Input: A directed acyclic graph $G = (V, E)$.
Output: A minimum chain decomposition $\mathcal{C} = C_1, \ldots, C_k$ of G.

```
(G, d) ← Build the flow reduction of G detailed in Section 4
f*, k = |f*| ← Use Lemma 4 to obtain a minimum flow of (G, d)
Initialize C_i as an empty list for i ∈ {1, ..., k}
Initialize mergeable dictionaries maintaining a partition of {1, ..., k}
I_s ← {1, ..., k}
for v ∈ V in topological order do
    I_v, I_s ← SIZE-SPLIT(I_s, f*(s, v^i))
    for u^o ∈ N^-(v^i) do
        I_uv, I_u ← SIZE-SPLIT(I_u, f*(u^o, v^i))
        I_v ← MERGE(I_v, I_uv)
    i ← SOME(I_v)
    C_i.append(v)
return C_1, ..., C_k
```

▶ **Lemma 5.** *Given a k-width DAG $G = (V, E)$ as input, Algorithm 1 computes a minimum chain decomposition $\mathcal{C} = C_1, \ldots, C_k$ of G.*

Proof. By Lemma 4, every flow decomposition of f^* corresponds to an MPC of G. We prove that each C_i is a chain of V, since every vertex is added to exactly one C_i we conclude that $\mathcal{C} = C_1, \ldots, C_k$ is a minimum chain decomposition. Inductively, if v is added after v' in chain C_i, then v' reaches v in G. Indeed, $i \in I_v$ and in particular $i \in I_u$ for some $u \in N^-(v)$, and inductively v' reaches u in G and thus also reaches v. ◀

Moreover, since only splits and unions are performed, the sets I_v's and I_{uv}'s form a partition of $\{1, \ldots, k\}$ at any point during the algorithm's execution.

A simple implementation of sets I_v's and I_{uv}'s as linked lists, allows us to perform the unions and element picks in constant time, but the splits in $O(f^*(u^o, v^i))$ (and $O(f^*(s, v^i))$) time each, and thus in $O\left(\sum_{v \in V} \left(f^*(s, v^i) + f^*(v^o, t)\right) + \sum_{(u,v) \in E} f^*(u^o, v^i)\right) = O(||\mathcal{P}||)$ time in total. However, we can implement the sets I_v's and I_{uv}'s as boosted mergeable dictionaries from Section 3 to speed up the total running time. Algorithm 2 shows the final result.

▶ **Theorem 1.** *Given a k-width DAG $G = (V, E)$ we can compute a minimum chain cover in time $O(T_{MF}(|E|) + (|V| + |E|) \log k)$, where $T_{MF}(|E|)$ is the time for solving maximum flow.*

Proof. The correctness of the algorithm follows from the previous discussion and Lemma 5 since Algorithm 2 is an implementation of Algorithm 1. Building the flow reduction takes $O(|V| + |E|)$ time and obtaining the minimum flow f^* takes $O(T_{MF}(|E|))$ time by Lemma 4. The rest of the running time is derived from the calls to the mergeable dictionaries' operations. SOME is called $O(|V|)$ times while SIZE-SPLIT and MERGE are called once per edge in the flow reduction that is $O(|\mathcal{E}|) = O(|V| + |E|)$ times. By applying Lemma 3 the total time of the $O(|V| + |E|)$ operation calls is $O((|V| + |E|) \log k)$. ◀

We finish our paper by encapsulating our result into a tool that can be used to efficiently extract a set of vertex-disjoint chains $\mathcal{C}$, encoding a set of paths $\mathcal{P}$, from a flow f that encodes $\mathcal{P}$ as a flow decomposition. If the problem can be modeled as a maximum flow/minimum cost flow problem, the result of Chen et al. [9] allows us to solve such problems in almost linear time. As a simple example, we could solve the ℓ-cover problem (find a set of ℓ vertex-disjoint chains covering the most vertices) in almost linear time.

▶ **Corollary 6.** *Let $G = (V, E)$ be a DAG, and $f : E \to \mathbb{N}_0$ a flow encoding a set of $|f|$ paths $\mathcal{P}$ of G as a flow decomposition into weight-1 paths. In $O((|V| + |E|) \log |f|)$ time, we can compute a set of $|f|$ vertex-disjoint chains $\mathcal{C}$ of G, which can (alternatively) be obtained by removing repeated vertices from $\mathcal{P}$.*

References

1 Amitabha Bagchi, Adam L Buchsbaum, and Michael T Goodrich. Biased skip lists. *Algorithmica*, 42:31–48, 2005.

2 Jørgen Bang-Jensen and Gregory Z Gutin. *Digraphs: theory, algorithms and applications*. Springer Science & Business Media, 2008.

3 Philip Bille, Mikko Berggren Ettienne, Travis Gagie, Inge Li Gørtz, and Nicola Prezza. Decompressing Lempel-Ziv compressed text. In *Proceedings of the 30th Data Compression Conference (DCC 2020)*, pages 143–152. IEEE, 2020.

4 Manuel Cáceres, Massimo Cairo, Andreas Grigorjew, Shahbaz Khan, Brendan Mumey, Romeo Rizzi, Alexandru I Tomescu, and Lucia Williams. Width helps and hinders splitting flows. In *Proceedings of the 30th Annual European Symposium on Algorithms (ESA 2022)*, pages 31:1–31:14, 2022.

5 Manuel Cáceres, Massimo Cairo, Brendan Mumey, Romeo Rizzi, and Alexandru I Tomescu. A linear-time parameterized algorithm for computing the width of a DAG. In *Proceedings of the 47th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2021)*, pages 257–269. Springer, 2021.

6 Manuel Cáceres, Massimo Cairo, Brendan Mumey, Romeo Rizzi, and Alexandru I Tomescu. Sparsifying, shrinking and splicing for minimum path cover in parameterized linear time. In *Proceedings of the 33rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2022)*, pages 359–376. SIAM, 2022.

7 Manuel Cáceres, Brendan Mumey, Edin Husić, Romeo Rizzi, Massimo Cairo, Kristoffer Sahlin, and Alexandru I Tomescu. Safety in multi-assembly via paths appearing in all path covers of a DAG. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 19(6):3673–3684, 2021.

8 Ghanshyam Chandra and Chirag Jain. Sequence to graph alignment using gap-sensitive co-linear chaining. In *Proceedings of the 27th Annual International Conference on Research in Computational Molecular Biology (RECOMB 2023)*, pages 58–73. Springer, 2023.

9 Li Chen, Rasmus Kyng, Yang P Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *Proceedings of the 63rd IEEE Annual Symposium on Foundations of Computer Science (FOCS 2022)*, pages 612–623. IEEE, 2022.

10 Yangjun Chen and Yibin Chen. An efficient algorithm for answering graph reachability queries. In *Proceedings of the 24th International Conference on Data Engineering (ICDE 2008)*, pages 893–902. IEEE, 2008.

11 Yangjun Chen and Yibin Chen. On the graph decomposition. In *Proceedings of the 4th IEEE Fourth International Conference on Big Data and Cloud Computing (BDCloud 2014)*, pages 777–784. IEEE, 2014.

12 Eleonor Ciurea and Laura Ciupala. Sequential and parallel algorithms for minimum flows. *Journal of Applied Mathematics and Computing*, 15(1):53–75, 2004.

13 Nicola Cotumaccio and Nicola Prezza. On indexing and compressing finite automata. In *Proceedings of the 32nd ACM-SIAM Symposium on Discrete Algorithms (SODA 2021)*, pages 2585–2599. SIAM, 2021.

14 Rene De La Briandais. File searching using variable length keys. In *Papers presented at the the March 3-5, 1959, western joint computer conference*, pages 295–298, 1959.

15 Robert P Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51(1):161–166, 1950.

16 Martin Farach and Mikkel Thorup. String matching in Lempel—Ziv compressed strings. *Algorithmica*, 20(4):388–404, 1998.

17 Stefan Felsner, Vijay Raghavan, and Jeremy Spinrad. Recognition algorithms for orders of small width and graphs of small Dilworth number. *Order*, 20(4):351–364, 2003.

18 Delbert R Fulkerson. Note on Dilworth's decomposition theorem for partially ordered sets. *Proceedings of the American Mathematical Society*, 7(4):701–702, 1956.

19 Loukas Georgiadis, Haim Kaplan, Nira Shafrir, Robert E Tarjan, and Renato F Werneck. Data structures for mergeable trees. *ACM Transactions on Algorithms*, 7(2):1–30, 2011.

20 Leo J Guibas and Robert Sedgewick. A dichromatic framework for balanced trees. In *Proceedings of the 19th Annual Symposium on Foundations of Computer Science (FOCS 1978)*, pages 8–21. IEEE, 1978.

21 John E Hopcroft and Richard M Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

22 John Iacono and Özgür Özkan. Mergeable dictionaries. In *Proceedings of the 37th International Colloquium on Automata, Languages, and Programming (ICALP 2010)*, pages 164–175. Springer, 2010.

23 Arthur B Kahn. Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562, 1962.

24 Adam Karczmarz. A simple mergeable dictionary. In *Proceedings of the 15th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2016)*, pages 7:1–7:13, 2016.

25 Donald E Knuth. *The art of computer programming: Volume 3: Sorting and Searching*. Addison-Wesley Professional, 1998.

26 Shimon Kogan and Merav Parter. Beating matrix multiplication for $n^{1/3}$-directed shortcuts. In *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. Full version available at `https://www.weizmann.ac.il/math/parter/sites/math.parter/files/uploads/main-lipics-full-version_3.pdf`.

27 Shimon Kogan and Merav Parter. Faster and unified algorithms for diameter reducing shortcuts and minimum chain covers. In *Proceedings of the 34th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2023)*, pages 212–239. SIAM, 2023.

28 Katherine Jane Lai. Complexity of union-split-find problems. Master's thesis, Massachusetts Institute of Technology, 2008.

29 Veli Mäkinen, Alexandru I Tomescu, Anna Kuosmanen, Topi Paavilainen, Travis Gagie, and Rayan Chikhi. Sparse Dynamic Programming on DAGs with Small Width. *ACM Transactions on Algorithms*, 15(2):1–21, 2019.

30 Simeon C Ntafos and S Louis Hakimi. On path cover problems in digraphs and applications to program testing. *IEEE Transactions on Software Engineering*, 5(5):520–529, 1979.

31 Simon J Puglisi and Massimiliano Rossi. On Lempel-Ziv decompression in small space. In *Proceedings of the 29th Data Compression Conference (DCC 2019)*, pages 221–230. IEEE, 2019.

32 Nicola Rizzo, Manuel Caceres, and Veli Mäkinen. Chaining of maximal exact matches in graphs. *arXiv preprint*, 2023. `arXiv:2302.01748`.

33 Robert E Tarjan. Edge-disjoint spanning trees and depth-first search. *Acta Informatica*, 6(2):171–185, 1976.

34 Cole Trapnell, Brian A Williams, Geo Pertea, Ali Mortazavi, Gordon Kwan, Marijke J Van Baren, Steven L Salzberg, Barbara J Wold, and Lior Pachter. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 28(5):511, 2010.

35 Jan van den Brand, Yin Tat Lee, Yang P Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Minimum cost flows, MDPs, and ℓ_1-regression in nearly linear time for dense instances. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2021)*, pages 859–869, 2021.

Improved Hardness Results for the Guided Local Hamiltonian Problem

Chris Cade
QuSoft and University of Amsterdam (UvA), The Netherlands

Marten Folkertsma
QuSoft and CWI, Amsterdam, The Netherlands

Sevag Gharibian
Paderborn Universität, Germany

Ryu Hayakawa
Kyoto University, Japan

François Le Gall
Nagoya University, Japan

Tomoyuki Morimae
Kyoto University, Japan

Jordi Weggemans
QuSoft and CWI, Amsterdam, The Netherlands

Abstract

Estimating the ground state energy of a local Hamiltonian is a central problem in quantum chemistry. In order to further investigate its complexity and the potential of quantum algorithms for quantum chemistry, Gharibian and Le Gall (STOC 2022) recently introduced the *guided local Hamiltonian problem (GLH)*, which is a variant of the local Hamiltonian problem where an approximation of a ground state (which is called a guiding state) is given as an additional input. Gharibian and Le Gall showed quantum advantage (more precisely, BQP-completeness) for GLH with 6-local Hamiltonians when the guiding state has fidelity (inverse-polynomially) close to 1/2 with a ground state.

In this paper, we optimally improve both the locality and the fidelity parameter: we show that the BQP-completeness persists even with 2-local Hamiltonians, and even when the guiding state has fidelity (inverse-polynomially) close to 1 with a ground state. Moreover, we show that the BQP-completeness also holds for 2-local physically motivated Hamiltonians on a 2D square lattice or a 2D triangular lattice. Beyond the hardness of estimating the ground state energy, we also show BQP-hardness persists when considering estimating energies of *excited* states of these Hamiltonians instead. Those make further steps towards establishing *practical quantum advantage* in quantum chemistry.

2012 ACM Subject Classification Theory of computation → Quantum complexity theory

Keywords and phrases Quantum computing, Quantum advantage, Quantum Chemistry, Guided Local Hamiltonian Problem

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.32

Category Track A: Algorithms, Complexity and Games

Related Version This paper is based on our following two technical reports
Previous Version: `https://arxiv.org/abs/2207.10097`
Previous Version: `https://arxiv.org/abs/2207.10250`

Funding CC acknowledges support from QuSoft and CWI, as well as the University of Amsterdam (UvA) under a POC (proof of concept) fund. MF and JW were supported by the Dutch Ministry of Economic Affairs and Climate Policy (EZK), as part of the Quantum Delta NL programme. RH, FLG, and TM were supported by MEXT Quantum Leap Flagship Program (MEXT Q-LEAP) grant No. JPMXS0120319794. RH was also supported by JSPS KAKENHI Grant Number JP22J11727. FLG was also supported by JSPS KAKENHI grants Nos. JP19H04066, JP20H05966, JP20H00579, JP20H04139, JP21H04879. SG was supported by DFG grants 450041824 and 432788384. TM was supported by JST Moonshot JPMJMS2061-5-1-1, JST FOREST, the Grant-in-Aid for Scientific Research (B) No.JP19H04066, the Grant-in Aid for Transformative Research Areas (A) 21H05183, and the Grant-in-Aid for Scientific Research (A) No.22H00522.

Acknowledgements We thank Jonas Helsen for feedback on an earlier draft, and Ronald de Wolf for helpful comments.

1 Introduction

Simulation of physical systems is one of the originally envisioned applications of quantum computing [12, 13]. Quantum chemistry, in particular, has seen much activity on this front in recent years, e.g. [1, 4, 5, 24, 29, 31]. There, a central goal is to estimate the ground state energy of a given k-local Hamiltonian H, denoted the *k-local Hamiltonian problem (k-LH)*. Roughly, for this problem, a k-local Hamiltonian $H = \sum_i H_i$ on n qubits is a $2^n \times 2^n$ Hermitian matrix, specified succinctly via "local quantum clauses" H_i acting on $k \in O(1)$ qubits each. The eigenvalues of H are the discrete energy levels of the corresponding quantum system. In particular, the smallest eigenvalue, which we denote $\lambda_0(H)$, is called the ground state energy. An eigenvector corresponding to $\lambda_0(H)$ is called a ground state, and describes a state of the quantum system in the energy configuration $\lambda_0(H)$. Note that k-LH strictly generalizes classical k-SAT, in that any instance of the latter can be embedded into the former.

Unfortunately, it is nowadays well-known that estimating ground state energies of local Hamiltonians is QMA-complete [23]. This hardness persists, moreover, even in the bosonic [32] and fermionic settings [30]. Thus, assuming BQP $\neq$ QMA, one cannot hope for an efficient algorithm for k-LH on *all* k-local Hamiltonians.

What actually happens in practice

In an attempt to bypass worst-case hardness results, in practice the quantum chemistry community often adopts the following two-step procedure:

- (Step 1: Ground state approximation) A classical heuristic algorithm is applied to obtain a "guiding state" $|\psi\rangle$, which is hoped to have "good" fidelity with a ground state.
- (Step 2: Ground state energy approximation) The guiding state $|\psi\rangle$ is used in Quantum Phase Estimation (QPE) [22] to efficiently compute the corresponding ground state energy [2, 4]. (A more recent approach is based on variational quantum algorithms, aimed more at near-term hardware (see [9] for a survey), but which is heuristic in nature (unlike QPE).)

Two comments: (1) There is something special about Step 2 – it is a unique strength of quantum computers to be able to resolve an eigenvalue (within additive $1/\mathrm{poly}(n)$ precision) of a (sparse) Hermitian matrix given just an *approximation* $|\psi\rangle$ to the corresponding eigenvector (via QPE)![1] Indeed, the closely related task of (sparse) matrix inversion, which can be solved efficiently on a quantum computer coherently by diagonalizing the matrix and "manually" inverting its eigenvalues via postselection, is BQP-complete [19]. (2) In general, one does *not* expect a good[2] guiding state for arbitrary local Hamiltonian H to exist, as this would imply QCMA = QMA. And even if such a guiding state *did* exist, finding it can still be hard. For example, minimizing $\mathrm{tr}(H\rho)$ over the "simplest" quantum ansatz of tensor product states, i.e. $\rho = \rho_1 \otimes \rho_2 \otimes \cdots \otimes \rho_n$ for $\rho_i \in \mathcal{L}(\mathbb{C}^2)$, remains NP-hard (seen by letting H be a diagonal Hamiltonian encoding a classical 3-SAT instance).

---

[1] Actually, quantum computers can efficiently prepare a ground state with fidelity $1-1/\exp(n)$ given access to a guiding state $|u\rangle$ that has inverse polynomial fidelity with a ground state $|g\rangle$ (i.e. $|\langle u|g\rangle| \geq 1/\mathrm{poly}(n)$) using quantum amplitude amplification for local Hamiltonians that have inverse-polynomial spectral gaps [26].

[2] "Good" here meaning a state $|\psi\rangle$ with inverse polynomial fidelity with a ground state, and with a succinct classical description allowing $|\psi\rangle$ to be prepared efficiently.

Directions for study

With Steps 1 and 2 above in mind, in order to practically obtain a quantum advantage for quantum chemistry problems, there are two branches of study necessary:

- (Step 1: Ground state approximation) Here, the best one can hope for is fast algorithms tailored to physically motivated *special cases* of Hamiltonians H (either heuristic or worst-case poly-time complexity). This is arguably the bottleneck for fast quantum algorithms outperforming classical techniques [25].
- (Step 2: Ground state energy approximation) A thorough complexity theoretic understanding of which Hamiltonian families provably permit quantum computers to outperform classical ones, assuming a good guiding state has been found (in Step 1).

In [15], the formal study of the second step above was initiated. Specifically, the *Guided k-local Hamiltonian problem (k-GLH)* was introduced, which is stated roughly as follows (formally given in Definition 6): Given a k-local Hamiltonian H, an appropriate "representation" of a guiding state $|\psi\rangle$ with δ-fidelity with the ground space of H, and real thresholds $\beta > \alpha$, estimate the ground state energy of H. Then, two results were shown:

- For any constant k, k-GLH can be efficiently solved *classically* within *constant* precision, i.e. for $\beta - \alpha \in \Theta(1)$ and $\delta \in \Theta(1)$.
- In contrast, 6-GLH is BQP-hard for *inverse polynomial* precision, i.e. $\beta - \alpha \geq 1/\text{poly}(n)$, and $\delta = 1/\sqrt{2} - 1/\text{poly}(n)$.

The latter regime of inverse-polynomial precision turns out to be the relevant one for solving quantum chemistry problems in practice – the desired "chemical accuracy" is about 1.6 millihartree (which is constant relative to an unnormalized Hamiltonian), which upon renormalization of the Hamiltonian (as done here) yields the claimed inverse polynomial precision. This BQP-hardness result thus gives theoretical evidence for the superiority of quantum algorithms for chemistry.

Four important problems were left open in [15]: Is k-GLH still BQP-hard with larger δ, and in particular for δ arbitrarily close to 1? Is k-GLH still BQP-hard for $k < 6$? Is k-GLH still BQP-hard for estimating the excited state energies? Is k-GLH still BQP-hard for physically motivated Hamiltonians?

This work

In this work, we continue the agenda toward Step 2 above by resolving these four open questions. Here are our main contributions:

- First, we show that BQP-hardness continues to hold even for $\delta = 1 - 1/\text{poly}(n)$, i.e. even when we are promised the guiding state $|\psi\rangle$ is a remarkably good approximation to the ground state.
- Second, we show that BQP-hardness continues to hold even for $k = 2$. (Note that for $k = 1$, the problem can be solved efficiently classically, even without a guiding state.)
- Third, we extend the BQP-hardness results to the case when one is interested in estimating energies of excited states, rather than just the groundstate. Interestingly, we are only able to show BQP-completeness in this setting by showing that the first point holds, i.e. the BQP-hardness in the regime $\delta \in [\frac{1}{2} + \Omega(1/poly(n)), 1 - \Omega(1/poly(n))]$.
- Fourth, we prove hardness results for *physically motivated* Hamiltonians. They include XY model (constraints of the form $XX + YY$), Heisenberg model (constraints of the form $XX + YY + ZZ$), the antiferromagnetic XY model and the antiferromagnetic Heisenberg model (i.e. "Quantum Max Cut" [16]). In contrast, the BQP-hardness construction of [15] is arguably artificial, because they used the circuit-to-Hamiltonian construction of [23] and query Hamiltonian construction of [3].

To formalize the third direction, we introduce the *Guided k-Local Hamiltonian Low Energy*-problem (k-GLHLE) in which the guiding state has δ-fidelity with the c'th excited state of H and the problem is to estimate the c'th excited state energy of H (for a formal definition, see Definition 6). Then, the four contributions above are summarized in the following theorem.

▶ **Theorem 1** (Main result). *For any $\delta \in (0, 1 - \Omega(1/\mathrm{poly}(n)))$, constant $k \geq 2$ and some integer $0 \leq c \leq \mathcal{O}(poly(n))$, there exist $a, b \in [-1, 1]$ with $b - a \in \Omega(1/\mathrm{poly}(n))$ such that k-GLHLE is* BQP*-hard. Moreover, it is still* BQP*-hard if the 2-local Hamiltonian is restricted to any of the following families of Hamiltonians:*

- *non-2SLD Hamiltonian on a 2D square lattice*
- *antiferromagnetic Heisenberg model*
- *antiferromagnetic XY model on a 2D triangular lattice.*

Here, the "non-2SLD" Hamiltonians are, roughly, 2-local Hamiltonians that cannot be diagonalized via single-qubit unitaries (see Definition 5 for the formal definition). (The term 2SLD is short for "the 2-local parts of all interactions in the set are simultaneously locally diagonalizable".) It was originally introduced in the Hamiltonian complexity classification of Cubitt and Montanaro [10]. The XY model and the Heisenberg model are examples of non-2SLD Hamiltonians.

Techniques

Now let us explain our technical contributions. Our first result is the improvement of the fidelity δ (Proposition 7 in Section 3). The construction of [15] cannot exceed $\delta = 1/2$, but we achieve the fidelity $\delta = 1 - 1/\mathrm{poly}(n)$. Let us explain why the construction of [15] cannot exceed the fidelity $\delta = 1/2$. Their construction for the BQP-hardness result is the following local Hamiltonian

$$H = \frac{\alpha + \beta}{2} I \otimes |0\rangle\langle 0| + H' \otimes |1\rangle\langle 1|,$$

where $\beta - \alpha > 1/\mathrm{poly}(n)$ and H' is a certain local Hamiltonian whose lowest eigenvalue is $\leq \alpha$ in the YES case and is $\geq \beta$ in the NO case. It is clear that a ground state of H is $|\psi\rangle \otimes |1\rangle$ in the YES case, where $|\psi\rangle$ is a ground state of H'. For the NO case, a ground state is $|0...0\rangle \otimes |0\rangle$. It can then be easily observed that the optimal guiding state (i.e. the guiding state that has the maximum fidelity with ground states *in both the YES and the NO cases*) is written as $|\phi\rangle \otimes |+\rangle$ for a certain choice of $|\phi\rangle$, which shows that the fidelity cannot exceed $1/2$ in this construction.

To overcome the problem, we use the perturbation theory approaches of [21, 7]. In particular, we use first-order perturbation theory, either using the general Schrieffer-Wolf transform framework of [7] or a more first-principles approach via the Projection Lemma. The main idea is to use a large energy penalty term to rule out all low-energy states which do not look like "history states". We then show that the corresponding guiding state can be chosen as the semi-classical subset state introduced in [15] (see Definition 2 in Section 2). To obtain this, we notice that the ground state of our Hamiltonian is gapped and unique. This is because we are doing a reduction from BQP (as opposed to QMA). In other words, there is no QMA "proof" to be plugged into the history state construction, and therefore there is a unique low-energy history state. In sum, via perturbation theory, we are able to *directly* approximate the ground state with a guiding state in *both* YES and NO cases, as opposed to the block encoding approach of [15], which used equally weighted orthogonal subspaces to *separately* encode the YES and NO cases, respectively.

Our second result is BQP-hardness of k-GLH for $k = 2$ (Propositions 9 in Section 3). Here, the universal simulation setup of [11, 33] cannot be directly applied, because although their results can approximately preserve the ground space of the input Hamiltonian, it was not known whether semi-classical subset states can be mapped to semi-classical subset states under such simulation frameworks, and the latter is essential for guiding states used in GLH. We show that this is indeed the case. In particular, we show that the original semi-classical subset state of the input 5-GLH instance is mapped to a state with polynomially many ancilla qubits in the low-energy subspace of the simulating 2-local Hamiltonian.

Our third result is the BQP-hardness for physically motivated 2-local Hamiltonians (Proposition 10 and Proposition 12). The main obstacle here is that ground states of physically motivated 2-local Hamiltonians are not known to be guided by semi-classical subset states. To solve the problem, we introduce another class of semi-classical states which we call *semi-classical encoded states* (see Definition 3 in Section 2). Intuitively, semi-classical encoded states are states constructed from semi-classical subset states by applying a local isometry on each qubit. Although semi-classical encoded states are more general than semi-classical subset states, they still allow succinct descriptions and efficient classical sampling algorithms (Lemma 4). For us, it is essential that semi-classical encoded states are closed under the applications of the local encoding of states during the perturbative simulations. We show that semi-classical encoded states indeed satisfy this property, and therefore can guide ground states of physically motivated 2-local Hamiltonians. The semi-classical encoded states newly introduced in this paper are of independent interest, and seem to have many other interesting applications.

Finally, our fourth result is to extend the k-GLH problem to the question of excited state energy estimation, we call this the *Guided k-Local Hamiltonian Low Energy* (k-GLHLE) problem. In Ref. [20], the authors show that determining the cth excited state energy of a k-local Hamiltonian ($k \geq 3$), where $c = \text{poly}(n)$, is QMA-complete – even if all the $c - 1$ energy eigenstates and corresponding energies are known. In their construction, they embed a k-local Hamiltonian H, encoding the QMA computation, in a Hamiltonian H' living on a larger Hilbert space. This allows them to add up to polynomial number of artificial eigenstates to H' below the groundstate of H. Finding the c'th eigenvalue of H' is then just as hard as finding the groundstate of H. We show that this construction translates to the setting with guiding states. As a bonus, we also show that the unguided problem is QMA-hard for $k = 2$, which was left open in [20].

Open questions

There are many open questions surrounding GLH, as well as the more general important goal of solving quantum chemistry problems on quantum computers. For example, we have shown BQP-hardness of GLH for physically motivated Hamiltonians such as those with Heisenberg interactions. An important next step would be to show BQP-hardness for the specific types of fermionic Hamiltonians which are currently being studied in the quantum chemistry literature. Another subtle but important point is that, technically, the level of precision required for GLH in quantum chemistry scales as $1/n$, while the hardness promise gap scales as $o(1/n)$ in [15] and the present paper. Can this be improved to $\Theta(1/n)$? A positive resolution to the quantum PCP conjecture would presumably, in turn, allows one to obtain hardness for gap $\Theta(1)$. Absent this, we are unaware of any circuit-to-Hamiltonian construction which is able to achieve $O(1/n)$ promise gap. Moreover, as mentioned earlier, the main bottleneck for quantum chemistry on quantum computers is the arduous task of *finding* a good guiding state (if it even exists!). Can good heuristics be designed for this?

Efforts to date suggest the answer so far is negative [25]. Finally, more interestingly (but more challengingly), can one show *rigorous* poly-time guiding-state computation algorithms for the specific families of Hamiltonians considered in the quantum chemistry literature?

2 Preliminaries

Notation

We denote by $[M]$ the set $\{1, \ldots, M\}$. We write $\lambda_i(A)$ to denote the ith eigenvalue of a Hermitian matrix A, ordered in non-decreasing order, with $\lambda_0(A)$ denoting the smallest eigenvalue (ground energy). We denote $\mathrm{eig}(A) = \{\lambda_0(A), \ldots, \lambda_{\dim(A)-1}(A)\}$ for the (ordered) set of all eigenvalues of A.

2.1 Semi-classical states

In this section, we formally introduce the guided local Hamiltonian problem. We first define two classes of semi-classical states. The term "semi-classical" is motivated by the requirement for such states that they should be efficiently described (as an input of the problem) and efficiently samplable.[3]

▶ **Definition 2** (Semi-classical subset state). *We say that a normalized state $|u\rangle \in \mathbb{C}^{2^n}$ is a semi-classical subset state if there is a subset $S \subseteq \{0,1\}^n$ with $|S| = \mathrm{poly}(n)$ such that*

$$|u\rangle = \frac{1}{\sqrt{|S|}} \sum_{x \in S} |x\rangle \,.$$

A semi-classical subset state can be efficiently described by the description of S. It is clear that we can efficiently sample from the probability distribution that outputs $x \in \{0,1\}^n$ with probability $|\langle x|u\rangle|^2$, i.e. according to the uniform distribution over S.

We next introduce a generalized version of a semi-classical subset state.

▶ **Definition 3** (Semi-classical encoded state). *We say that a normalized state $|u\rangle \in \mathbb{C}^{2^m}$, for $n < m \in \mathcal{O}(n)$, is a semi-classical encoded state if there is a subset $S \subseteq \{0,1\}^n$ with $|S| = \mathrm{poly}(n)$ and a set of isometries $V_1, V_2, ..., V_n$, where each of V_i maps a 1-qubit state to an $\mathcal{O}(1)$-qubit state, such that*

$$|u\rangle = \frac{1}{\sqrt{|S|}} \sum_{x \in S} V_1(|x_1\rangle) \otimes V_2(|x_2\rangle) \otimes \cdots \otimes V_n(|x_n\rangle).$$

A semi-classical encoded state is indeed a semi-classical subset state if the encoding is trivial (i.e. $V_1 = V_2 = \cdots = V_n = I$). A semi-classical encoded state can be described by the description of S and isometries $V_1, V_2 ..., V_n$. We can also efficiently sample from the semi-classical encoded state as we show in the following lemma.

▶ **Lemma 4.** *Given the description of an m-qubit semi-classical encoded state $|u\rangle$, we can classically efficiently sample from the probability distribution that outputs $x \in \{0,1\}^m$ with probability $|\langle x|u\rangle|^2$.*

[3] The requirement of sampling access for a guiding state is motivated by the existence of an efficient classical algorithm for the GLH problem with constant precision, given a guiding state with sampling access, as shown in [15]. One type of a semi-classical state we use in this paper is a polynomial-size variant of the notion of subset states, first introduced in [18].

Proof. Assume we are given the description, $S \subseteq \{0,1\}^n$ and $V_1, V_2, ..., V_n$, of the semi-classical encoded state

$$|u\rangle = \frac{1}{\sqrt{|S|}} \sum_{x \in S} V_1(|x_1\rangle) \otimes V_2(|x_2\rangle) \otimes \cdots \otimes V_n(|x_n\rangle).$$

Let $P(y_0, y_1, ..., y_{i-1}) = |(\langle y_0, y_1, ..., y_{i-1}| \otimes I)\,|u\rangle\,|^2$ be the probability that the measurement outcome of the first i qubits of $|u\rangle$ in the computational basis is $y_0, y_1, ..., y_{i-1}$. For each $i \in [m]$, we can efficiently calculate $P(y_0, y_1, ..., y_{i-1})$ because $|S| = \text{poly}(n)$ and $V_1(|x_1\rangle) \otimes V_2(|x_2\rangle) \otimes \cdots \otimes V_n(|x_n\rangle)$ is a product state of $\mathcal{O}(1)$-qubit states. Then, we can also efficiently calculate the conditional probability

$$P(z|y_0, y_1, ..., y_{i-1}) = \frac{P(y_0, y_1, ..., y_{i-1}, z)}{P(y_0, y_1, ..., y_{i-1})}.$$

If the bits $y_0, y_1, ..., y_{i-1}$ have already been sampled, we compute $P(z|y_0, y_1, ..., y_{i-1})$ and sample the next bit by tossing the coin with bias $P(0|y_0, y_1, ..., y_{i-1})$. In this way, we can classically efficiently sample from the probability distribution that outputs x with probability $|\langle x|u\rangle|^2$. ◀

2.2 Non-2SLD Hamiltonian and geometry of interaction

To state the result, we introduce some families of Hamiltonians. Given a set of (at most) two-body interactions $\mathcal{S} = \{h_\alpha\}$, $\mathcal{S}$-Hamiltonian refers to the family of Hamiltonians that can be written in the form

$$H = \sum_{\langle i,j \rangle \in E} J_{i,j} h^{(i,j)}_{\alpha_{i,j}}, \tag{1}$$

where $J_{i,j} \in \mathbb{R}$, $h^{(i,j)}_{\alpha_{i,j}}$ is two-local interaction chosen from $\mathcal{S}$ and E is the set of edges that represents the connectivity of interaction [10]. If the connectivity of two-body interaction is restricted to a 2D square lattice, we call such a family $\mathcal{S}$-Hamiltonian on a 2D square lattice. We also introduce the notion of 2SLD and non-2SLD:

▶ **Definition 5** (2SLD interaction [10]). *Suppose $\mathcal{S}$ is a set of interactions at most 2 qubits. We say that $\mathcal{S}$ is 2SLD if there exists $U \in \text{SU}(2)$, such that for all $h_i \in \mathcal{S}$,*

$$U^{\otimes 2} h_i (U^\dagger)^{\otimes 2} = \alpha_i Z \otimes Z + A_i \otimes I + I \otimes B_i,$$

where $\alpha_i \in \mathbb{R}$ and A_i, B_i are arbitrary single-qubit Hamiltonians.

A set $\mathcal{S}$ is non-2SLD if it is not 2SLD. In particular, such non-2SLD $\mathcal{S}$ includes the following physically motivated[4] Hamiltonians:

- $\{Z, X, ZZ, XX\}$ ($ZZXX$ interaction [6])
- $\{Z, X, ZX, XZ\}$ (ZX interaction [6])
- $\{XX + YY\}$ (general XY interaction)
- $\{XX + YY + ZZ\}$ (general Heisenberg interaction).

If there is only a single type of interaction (like $\mathcal{S} = \{XX + YY + ZZ\}$), the Hamiltonian is called *semi-translationally-invariant*. (Interaction strength can differ in each term.)

[4] For clarity, in [10] and here, all hardness results require *non-uniform* weights on constraints. It is an open question whether one can obtain (say) QMA-hardness results with uniform (i.e. unit weight) constraints for such models. This remains an interesting open question, as many-body physicists typically utilize unit weights to model physical systems.

Restriction on the sign of the interaction

We also introduce a further restricted class of $\mathcal{S}$-Hamiltonian in which all the signs of the coefficients are promised to be non-negative (i.e. all of $J_{i,j}$ in eq. (1) must satisfy $J_{i,j} \geq 0$). We call such a family of Hamiltonians as $\mathcal{S}^+$-Hamiltonian following [28]. In [28], the following results are shown:

- $\{\alpha XX + \beta YY + \gamma ZZ\}^+$-Hamiltonian is QMA-complete if $\alpha + \beta > 0$, $\alpha + \gamma > 0$ and $\beta + \gamma > 0$ hold.
- $\{\alpha XX + \beta YY + \gamma ZZ\}^+$-Hamiltonian is QMA-complete if the interactions are restricted to the edges of a 2D triangular lattice if $\alpha XX + \beta YY + \gamma ZZ$ is not proportional to $XX + YY + ZZ$ in addition to the condition that $\alpha + \beta > 0$, $\alpha + \gamma > 0$ and $\beta + \gamma > 0$ hold.

The first type of $\mathcal{S}^+$-Hamiltonian includes the antiferromagnetic Heisenberg model ($\{XX + YY + ZZ\}^+$-Hamiltonian) and the antiferromagnetic XY model ($\{XX + YY\}^+$-Hamiltonian) as important special cases. The antiferromagnetic XY model (unlike the antiferromagnetic Heisenberg model) remains QMA-complete if its geometric interaction is restricted to a 2D triangular lattice as it is included in the second type of $\mathcal{S}^+$-Hamiltonian above.

3 GLHLE hardness constructions

We next define the guided local Hamiltonian low energy (GLHLE) problem, which can be viewed as a generalization of GLH by considering arbitrary eigenstates of Hamiltonians[5]. For an n-qubit Hamiltonian H, we denote Π_c the projector onto the space spanned by the states of H that have energy $\lambda_c(H)$.

▶ **Definition 6** (Guided Local Hamiltonian Low Energy). *GLHLE*(k, c, a, b, δ)

Input: *A k-local Hamiltonian H on n qubits such that $\|H\| \leq 1$ and the description of a semi-classical encoded state $|u\rangle \in \mathbb{C}^{2^n}$, a constant $c \in \mathbb{N}_{\geq}0$.*

Promise: $\|\Pi_c |u\rangle\|^2 \geq \delta$, *where Π_c denotes the projection on the subspace spanned by the cth eigenstates, ordered in order of non-decreasing energy, of H, and either $\lambda_c(H) \leq a$ or $\lambda_c(H) \geq b$ holds.*

Goal: *Decide whether $\lambda_c(H) \leq a$ or $\lambda_c(H) \geq b$.*

The proof of Theorem 1 consists of five parts: first, we show that 5-local GLH with $\delta = 1 - \Omega(1/\text{poly}(n))$ fidelity is BQP-hard. Then, we show how to extend this result to the BQP-hardness of the 6-local GLHLE problem. Next, we improve the locality parameter and show a reduction from 6-local GLHLE to 2-local GLHLE. Simultaneously we show that this also holds when we restict the Hamiltonians to be non$-2SLD$ $\mathcal{S}$-Hamiltonian on a $2D$ square lattice. Finally, we show that BQP-hardness persists if we restrict the family of Hamiltonians to be $\{XX + YY + ZZ\}^+$-Hamiltonians, or $\{XX + YY\}^+$-Hamiltonians on a $2D$ triangular lattice.

We state these five parts as propositions and prove them one by one, from this our main result (Theorem 1) follows.

[5] This definition of GLH is very similar to the definition of GLH$^*(k, a, b, \delta)$ in [15]. The difference is that while the guiding states used in [15] are restricted to semi-classical subset states (Definition 2), in our definition we use the more general concept of semi-classical encoded states (Definition 3). Note that our BQP-hardness result for general 2-local Hamiltonians (Proposition 9) actually holds even when the guiding state is a semi-classical subset state. Proposition 9, which optimally improves both the locality and fidelity parameters of [15], therefore holds in exactly the same setting as [15]. We use semi-classical encoded states only to show BQP-hardness for further restricted families of Hamiltonians (Propositions 10 and 12).

3.1 Increasing the allowed fidelity

The first proposition focuses on increasing the allowed fidelity of the guiding state with the ground state of the Hamiltonian of interest (and hence $c = 0$).

▶ **Proposition 7.** *For any $\delta \in (0, 1 - \Omega(1/\mathrm{poly}(n)))$, there exist $a, b \in [0,1]$ with $b - a \in \Omega(1/\mathrm{poly}(n))$ such that the problem GLHLE$(5, 0, a, b, \delta)$ is* BQP*-hard. Moreover, it is still* BQP*-hard with the additional two promises that*

1. *H has a non-degenerate ground state separated from the first excited state by a spectral gap $\gamma \in \Omega(1/\mathrm{poly}(n))$ in both the cases $\lambda_0(H) \leq a$ and $\lambda_0(H) \geq b$.(We call such instances γ-gapped* GLH(k, a, b, δ)*.)*
2. *The guiding state is restricted to be a semi-classical subset state.*

Proof. Let $\Pi = (\Pi_{\mathrm{yes}}, \Pi_{\mathrm{no}})$ be a promise problem in BQP, and $x \in \{0,1\}^n$ be an input. Let $U_x = U_m U_{m-1} ... U_1$ be a quantum circuit that decides x consisting of $m = \mathrm{poly}(n)$ gates. U_x acts on $|x\rangle_A \otimes |0...0\rangle_B$ where A denotes the n-qubit input register and B denotes the poly-size ancilla register. By measuring the output register of $U_x |x\rangle_A \otimes |0...0\rangle_B$, the quantum verifier outputs 1 with probability at least α if $x \in \Pi_{\mathrm{YES}}$ (at most β if $x \in \Pi_{\mathrm{NO}}$, respectively). We may assume $\alpha = 1 - 2^{-n}$ and $\beta = 2^{-n}$ via the standard error reduction for BQP.

Consider a pre-idled quantum verifier $\tilde{U}_x \coloneqq U_x I \cdots I$, where I is the identity gate. The $\tilde{U}_x$ consists of $M \coloneqq m + N$ gates, where N is the number of idling steps. ($N = \mathrm{poly}(n)$ is taken properly later.) Consider Kitaev's [23] 5-local circuit-to-Hamiltonian construction with an additional scaling factor:

$$H \coloneqq \Delta(H_{\mathrm{in}} + H_{\mathrm{prop}} + H_{\mathrm{stab}}) + H_{\mathrm{out}}. \tag{2}$$

Here,

$$H_{\mathrm{in}} \coloneqq (I - |x\rangle\langle x|)_A \otimes (I - |0...0\rangle\langle 0...0|)_B \otimes (|0\rangle\langle 0|)_C \tag{3}$$

$$H_{\mathrm{out}} \coloneqq |0\rangle\langle 0|_{\mathrm{out}} \otimes |M\rangle\langle M|_C \tag{4}$$

$$H_{\mathrm{stab}} \coloneqq \sum_{j=1}^{M-1} |0\rangle\langle 0|_{C_j} \otimes |0\rangle\langle 0|_{C_{j+1}} \tag{5}$$

$$H_{\mathrm{prop}} \coloneqq \sum_{t=1}^{M} H_t, \text{where} \tag{6}$$

$$H_t \coloneqq -\frac{1}{2} U_t \otimes |t\rangle\langle t-1|_C - \frac{1}{2} U_t^\dagger \otimes |t-1\rangle\langle t|_C + \frac{1}{2} I \otimes (|t\rangle\langle t|_C + |t-1\rangle\langle t-1|_C). \tag{7}$$

It is known that the non-degenerate and zero-energy ground space of $H_0 \coloneqq H_{\mathrm{in}} + H_{\mathrm{prop}} + H_{\mathrm{stab}}$ is spanned by $|\psi_{\mathrm{hist}}\rangle$, where

$$|\psi_{\mathrm{hist}}\rangle \coloneqq \frac{1}{\sqrt{M+1}} \sum_{t=0}^{M} \tilde{U}_t \tilde{U}_{t-1} \cdots \tilde{U}_1 |x\rangle_A \otimes |0...0\rangle_B \otimes |t\rangle_C .$$

It is also known that the smallest non-zero eigenvalue of H_0 is larger than $\pi^2/(64M^2)$ [17, Lemma 2.2] (based on [14, Lemma 3]).

We apply the Schrieffer-Wolf transformation for this H by taking sufficiently large Δ. Note that $H_{\mathrm{out}} = |0\rangle\langle 0| \otimes I \otimes |M\rangle\langle M|$ and $\|H_{\mathrm{out}}\| = 1$. We would take

$$\Delta \geq 16 \cdot 64 M^2 / \pi^2.$$

Then, H has a one-dimensional ground space spanned by a ground state $|g\rangle$. In the following, we analyze the fidelity between $|g\rangle$ and $|\psi_{\mathrm{hist}}\rangle$, and the eigenvalue of $|g\rangle$ in the YES and NO cases.

Analysis of fidelity

Using Equation (12) of Appendix B , the bound

$$\| |g\rangle - |\psi_{\text{hist}}\rangle \| \in \mathcal{O}\left((\Delta/M^2)^{-1}\right)$$

holds. Let us introduce the following state:

$$|u\rangle := \frac{1}{\sqrt{N}} \sum_{t=1}^{N} |x\rangle_A \otimes |0...0\rangle_B \otimes |t\rangle_C .$$

This is a semi-classical subset state. This state satisfies

$$|\langle u|\psi_{\text{hist}}\rangle|^2 = \frac{N}{m+N+1}.$$

Therefore, for any positive polynomial r, we can take sufficiently large $N, \Delta \in \mathcal{O}(\text{poly}(n))$ so that $|\langle u|g\rangle|^2 \geq 1 - 1/r(n)$.

Analysis of eigenvalue

Next, we see the ground state energy of H in both the YES case and the NO case. The first-order effective Hamiltonian is given by

$$H_{\text{eff},1} = |\psi_{\text{hist}}\rangle \langle\psi_{\text{hist}}| H_{\text{out}} |\psi_{\text{hist}}\rangle \langle\psi_{\text{hist}}| .$$

The history state is defined as

$$|\psi_{\text{hist}}\rangle = \frac{1}{\sqrt{M+1}} \sum_{t=1}^{M} \tilde{U}_t \cdots \tilde{U}_1 |x\rangle_A \otimes |0...0\rangle_B \otimes |t\rangle_C$$

and

$$\langle\psi_{\text{hist}}| H_{\text{out}} |\psi_{\text{hist}}\rangle = \frac{1}{M+1} \langle x,0| U_x^\dagger (|0\rangle\langle 0|_{\text{out}} \otimes I) U_x |x,0\rangle .$$

The eigenvalue of $H_{\text{eff},1}$ is given by $\langle\psi_{\text{hist}}| H_{\text{out}} |\psi_{\text{hist}}\rangle$ and this is $\mathcal{O}((\Delta/M^2)^{-1}) = \mathcal{O}(1/\text{poly}(n))$-close to the ground state energy of H using Equation (11).

It can be verified that $\langle\psi_{\text{hist}}| H_{\text{out}} |\psi_{\text{hist}}\rangle \leq (1-\alpha)/(M+1)$ if U_x accepts x with probability at least α and $\langle\psi_{\text{hist}}| H_{\text{out}} |\psi_{\text{hist}}\rangle \geq (1-\beta)/(M+1)$ if U_x accepts x with probability at most β. As we have mentioned earlier, we can assume $\alpha = 1 - 2^{-n}$ and $\beta = 2^{-n}$. Therefore, the ground state energy a of H lies in the range of $0 \pm \mathcal{O}((\Delta/M^2)^{-1})$ if $x \in \Pi_{\text{yes}}$ and the ground state energy b of H lies in the range of $1/(M+1) \pm \mathcal{O}((\Delta/M^2)^{-1})$ if $x \in \Pi_{\text{no}}$.

We also see the spectral gap between the ground state and any excited state in both the YES and NO cases. We first see the NO case. As we have shown, the ground state energy lies in $\frac{1-2^n}{M+1} \pm \mathcal{O}((M^2/\Delta))$. In $H = \Delta(H_{\text{in}} + H_{\text{prop}} + H_{\text{stab}}) + H_{\text{out}}$, the eigenvalues of $\Delta(H_{\text{in}} + H_{\text{prop}} + H_{\text{stab}})$ is perturbed at most $\|H_{\text{out}}\| = 1$. Therefore, the smallest non-zero eigenvalue of H is larger than $(\Delta\pi^2)/(64M^2) - 1$. The spectral gap in the NO case is therefore

$$\mathcal{O}\left(\frac{\Delta}{M^2}\right) - 1 - \left(\frac{1-2^{-n}}{M+1} + \mathcal{O}\left(\frac{M^2}{\Delta}\right)\right).$$

The ground state energy in the YES case is smaller than that in the NO case. Therefore, we can take sufficiently large $\Delta \in \text{poly}(n)$ so that H has inverse-polynomial spectral gap and $b - a \in \Omega(1/\text{poly}(n))$. Finally, we can normalize H by a polynomially large factor, which concludes the proof. ◀

3.2 Extending to excited states

The next proposition extends the result to excited states, at the cost of increasing the locality of the construction by one.

▶ **Proposition 8.** *For any $\delta \in \Omega(0, 1 - 1/poly(n))$ there exist $a, b \in [-1, 1]$ with $b - a \in \Omega(1/poly(n))$ and some number $0 \leq c \leq poly(n)$ such that GLHLE$(6, c, a, b, \delta)$ is* BQP*-hard even when,*

1. *the c'th eigenvalue of H, $\lambda_c(H)$, is non-degenerate and is separated by a gap $\gamma \in \Omega(1/poly(n))$ from both $\lambda_{c-1}(H)$ and $\lambda_{c+1}(H)$. (We call such instances γ-gapped GLHLE(k, c, a, b, δ).)*
2. *The guiding state is restricted to be a semi-classical subset state.*

Proof. We will reduce directly from the BQP-complete Hamiltonian H as defined in Eq. (2). Again, let $|u\rangle$ be a semi-classical guiding state such that $|\langle u|\psi_0\rangle| \geq \zeta$. Consider the following 6-local Hamiltonian $H^{(c)}$ on $n+1$ qubits[6]:

$$H^{(c)} = H^{(z)} \otimes |0\rangle\langle 0| + H^{(s)} \otimes |1\rangle\langle 1|, \tag{8}$$

where

$$H^{(z)} = \sum_{i=0}^{d} 2^i |1\rangle\langle 1|_i + \sum_{i=d+1}^{n} 2^{d+1} |1\rangle\langle 1|_i - \left(c - \frac{1}{2}\right) I,$$
$$H^{(s)} = \frac{1}{2} \frac{H + I/4}{\|H\| + 1/4} - \frac{1}{4} I,$$

where we have that $d = \lceil \log_2(c) \rceil$. $H^{(z)}$ has exactly c states with negative energy, with the smallest eigenvalue being $-c + \frac{1}{2}$ and the largest eigenvalue value at $\sum_{i=0}^{d} 2^i + \sum_{i=d+1}^{n} 2^{d+1} - \left(c - \frac{1}{2}\right) = 2^{d+1} + 2^{d+1}(n-d) - \frac{1}{2} - c$. The spectrum jumps in integer steps of 1, and has as largest negative (resp. smallest non-negative) energy value $-\frac{1}{2}$ (resp. $\frac{1}{2}$). Since $\text{eig}(H^{(s)}) \in [-1/4, 1/4]$, we must have that $H^{(s)}$ sits precisely at the c'th excited state level (or $c+1$'th eigenstate level) in $H^{(c)}$. Therefore, given a guiding state $|u\rangle$ for H such that $|\langle u|\psi_0\rangle| \geq \delta$, one has that the guiding state $|u^{(c)}\rangle = |u\rangle \otimes |1\rangle$ is also semi-classical and must have $|\langle u^{(c)}|\psi_c^{(c)}\rangle| \geq \delta$, where $|\psi_c^{(c)}\rangle$ denotes the cth excited state of $H^{(c)}$. Since this construction of $H^{(c)}$ and $|u^{(c)}\rangle$ provides a polynomial time reduction from an instance of GLH(k, a, b, δ) to one of GLHLE(k, c, a, b, δ), whenever $c = \mathcal{O}(\text{poly}(n))$, we must have that GLHLE(k, c, a, b, δ) is BQP-hard whenever $k \geq 6$. The gap between $\lambda_c(H^{(c)}) - \lambda_{c-1}(H^{(c)}) = \frac{1}{4}$ and the gap between $\lambda_{c+1}(H^{(c)}) - \lambda_c(H^{(c)}) = \gamma$ as before. The norm of the new Hamiltonian is bounded by $|H^{(c)}| = \mathcal{O}(\text{poly}(n))$, hence after normalisation we retain $\lambda_c(H^{(c)}) - \lambda_{c-1}(H^{(c)}) \geq \lambda_{c+1}(H^{(c)}) - \lambda_c(H^{(c)}) = \Omega(1/\text{poly}(n))$. ◀

3.3 Locality reduction and reduction to physically motivated Hamiltonians via strong Hamiltonian simulation

The next two propositions bring (i) the locality k down to 2 and (ii) extend the result to any of non-2SLD $\mathcal{S}$-Hamiltonian on a 2D square lattice.

[6] Note that this gadget can be trivially changed such that estimating the n highest energy states is BQP-hard.

▶ **Proposition 9.** *Any γ-gapped GLHLE(k, c, a, b, δ) with $k \in \mathcal{O}(1)$, $b - a \in \Omega(1/\mathrm{poly}(n))$, $\delta \in (0, 1-\Omega(1/\mathrm{poly}(n)))$, $0 \leq c \leq$ poly(n), and $\gamma \in \Omega(1/\mathrm{poly}(n))$ with a guiding semi-classical subset state can be reduced to γ'-gapped GLHLE$(2, c, a', b', \delta')$ with $b' - a' \in \Omega(1/\mathrm{poly}(n))$, $\delta' \in (0, 1 - \Omega(1/\mathrm{poly}(n)))$ and $\gamma' \in \Omega(\mathrm{poly}(n))$, and with a guiding semi-classical subset state in polynomial time.*

▶ **Proposition 10.** *Any γ-gapped GLHLE(k, c, a, b, δ) with $k \in \mathcal{O}(1)$, $b - a \in \Omega(1/\mathrm{poly}(n))$, $\delta \in (0, 1 - \Omega(1/\mathrm{poly}(n)))$, $0 \leq c \leq$ poly(n) and $\gamma \in \Omega(1/\mathrm{poly}(n))$, and with a guiding semi-classical subset state can be reduced to γ'-gapped GLHLE$(2, c, a', b', \delta')$ with $b' - a' \in \Omega(1/\mathrm{poly}(n))$, $\delta' \in (0, 1 - \Omega(1/\mathrm{poly}(n)))$ and $\gamma' \in \Omega(\mathrm{poly}(n))$ in polynomial time whose Hamiltonian is restricted to any of non-2SLD $\mathcal{S}$-Hamiltonian on a 2D square lattice.*

Proof of Propositions 9 and 10. Let H and $|u\rangle$ be arbitrary inputs of GLHLE(k, c, a, b, δ) with $k \in \mathcal{O}(1)$, $b - a \in \Omega(1/\mathrm{poly}(n))$, $\delta \in (0, 1 - \Omega(1/\mathrm{poly}(n)))$. From Theorem 15 (in Appendix A.1), we can efficiently find a non-2SLD $\mathcal{S}$-Hamiltonian H' on a 2D square lattice that is a strong (Δ, η, ϵ)-simulation of H given the description of H. We take $\epsilon < (b - a)/2$, $b' = b - \epsilon$, $a' = a + \epsilon$ and $\Delta = \mathcal{O}(\epsilon^{-1}\|H\|^2 + \eta^{-1}\|H\|)$ so that $\lambda_c(H') \leq a'$ if $\lambda_c(H) \leq a$, and $\lambda_c(H') \geq b'$ if $\lambda_c(H) \geq b'$ while $b' - a' \in \Omega(1/\mathrm{poly}(n))$.

We have shown the *existence* of desirable eigenvectors in the simulated Hamiltonian. What remains to show is that (i) the encoded state of $|u\rangle$ still has $1 - 1/\mathrm{poly}(n)$ fidelity with c'th excited state of H' and (ii) the encoded state is still a semi-classical subset state after the simulation by a 2-local Hamiltonian (for concluding Proposition 9) , and (iii) the encoded state is still a semi-classical encoded state after the simulation by an arbitrary non-2SLD $\mathcal{S}$-Hamiltonian on a 2D square lattice (for concluding Proposition 10).

(i) Verification of the fidelity. The fidelity can be analyzed by the following lemma:

▶ **Lemma 11** (Simulation of the gapped excited state)**.** *Suppose the c'th excited state $|g\rangle$ of H is non-degenerate and separated from both the $c-1$'th excited state and $c+1$'th excited state by a gap γ. Suppose H' is a (Δ, η, ϵ)-simulation of H such that $2\epsilon < \gamma$. Then H' has a non-degenerate c'th excited state $|g'\rangle$ and*

$$\|\mathcal{E}_{\mathrm{state}}(|g\rangle) - |g'\rangle\| \leq \eta + \mathcal{O}(\gamma^{-1}\epsilon).$$

Proof. This is a slight modification of Lemma 2 of [8]. First, the non-degeneracy of the c'th excited state of H' follows because the i'th smallest eigenvalues of H and H' differs at most ϵ for all $0 \leq i \leq \dim(H) - 1$, and ϵ satisfies $2\epsilon < \gamma$. Consider H as an unperturbed Hamiltonian and $V \coloneqq \tilde{\mathcal{E}}^\dagger H' \tilde{\mathcal{E}} - H$ as a perturbation. Then, the perturbed Hamiltonian $H + V = \tilde{\mathcal{E}}^\dagger H' \tilde{\mathcal{E}}$ has a non-degenerate c'th excited state $\tilde{\mathcal{E}}_{\mathrm{state}}(|g'\rangle)$. The first-order perturbation theory for eigenvectors gives $\| |g\rangle - \tilde{\mathcal{E}}^\dagger_{\mathrm{state}}(|g'\rangle)\| \in \mathcal{O}(\gamma^{-1}\epsilon)$. Therefore, it follows that $\|\tilde{\mathcal{E}}_{\mathrm{state}}(|g\rangle) - |g'\rangle\| = \|\tilde{\mathcal{E}}_{\mathrm{state}}(|g\rangle) - \tilde{\mathcal{E}}_{\mathrm{state}}(\tilde{\mathcal{E}}^\dagger_{\mathrm{state}}(|g'\rangle))\| \in \mathcal{O}(\gamma^{-1}\epsilon)$ using that $\tilde{\mathcal{E}}_{state}$ is an isometry and $|g'\rangle \in \mathrm{Im}(\tilde{\mathcal{E}}_{state})$. Finally, by using $\|\mathcal{E}_{state} - \tilde{\mathcal{E}}_{state}\| \leq \eta$, $\|\mathcal{E}_{\mathrm{state}}(|g\rangle) - |g'\rangle\| \leq \eta + \mathcal{O}(\gamma^{-1}\epsilon)$ follows. ◀

Using Lemma 11, we can take sufficiently small ϵ and η to ensure $\|\mathcal{E}_{\mathrm{state}}(|u\rangle) - |g'\rangle\| \leq \delta' = \delta - 1/\mathrm{poly}(n)$. Because the Hamiltonian simulation is efficient, the operator norm $\|H'\|$ and the number of qubits of H' is in poly(n).

(ii) Verification of the semi-classical property for Proposition 9. We start from a semi-classical subset state $|u\rangle = 1/\sqrt{|S|}\sum_{x\in S}|x\rangle$. We show that after the simulation of the original k-local Hamiltonian H where $k \in \mathcal{O}(1)$ by an 2-local Hamiltonian, the corresponding encoding $\mathcal{E}_{\mathrm{state}}(|u\rangle)$ is still a semi-classical subset state.

In order to simulate the k-local Hamiltonian by a 2-local Hamiltonian (that has no restriction on the family of Hamiltonian), it is enough to use mediator qubit gadgets that attach $|0\rangle$ states for mediator qubits (called subdivision and 3-to-2 gadgets [27]). A k-local term can be simulated by $(\lceil k/2 \rceil + 1)$-local terms using the subdivision gadget. Moreover, subdivision gadgets can be applied to each of the terms of the Hamiltonian in parallel [28, 11]. Therefore, we can reduce a k-local Hamiltonian to a 3-local Hamiltonian by $\mathcal{O}(\log k)$ rounds of applications of the subdivision gadgets. Then we can use the 3-to-2 gadgets in parallel to reduce to a 2-local Hamiltonian. In the corresponding encoding of states of this procedure, polynomially many $|0\rangle$ states are attached to the original state. Clearly, by attaching polynomially many $|0\rangle$ states, a polynomial-size subset state is mapped to another polynomial-size subset state:

$$\frac{1}{\sqrt{|S|}} \sum_{x \in S} |x\rangle \to \frac{1}{\sqrt{|S|}} \sum_{x \in S} |x\rangle |0\rangle^{\otimes \mathrm{poly}(n)} = \frac{1}{\sqrt{|S|}} \sum_{x \in S \times \{0 \ldots 0\}} |x\rangle .$$

This concludes the proof of Proposition 9.

(iii) Verification of the semi-classical property for Proposition 10. We proceed to show that starting from a semi-classical subset state $|u\rangle$, the resulting state is a semi-classical encoded state when we simulate the original Hamiltonian by a non-2SLD $\mathcal{S}$-Hamiltonian on a 2D square lattice. There are three types of encodings used in the simulation:

- **Mediator qubits.** In this encoding, some simple ancilla states are attached to the original state.
- **Subspace encoding.** In this encoding, a local isometry is applied to the original state.
- **Local Unitaries.** In this encoding, local unitary $U \otimes U \otimes \cdots \otimes U$, where each of U acts on one qubit, is applied to the original state.

We restate the chain of Hamiltonian simulations of Appendix C:

Arbitrary k-local Hamiltonian
↓ (1) Mediator qubits. (Attach a semi-classical subset state $|\alpha\rangle$.)
Spatially sparse 5-local Hamiltonian
↓ (2) Mediator qubits. (Attach polynomially many $|+_y\rangle$ states.)
Spatially sparse 10-local real Hamiltonian
↓ (3) Mediator qubits. (Attach polynomially many $|0\rangle$ or $|1\rangle$ states.)
Spatially sparse 2-local Pauli interactions with no Y-terms
↓ (4) Subspace encoding.
Spatially sparse $\mathcal{S}_0 = \{XX + YY + ZZ\}$ or $\{XX + YY\}$ Hamiltonian
↓ (5) Mediator qubits. (Attach polynomially many $|0\rangle$ or $|1\rangle$ states.)
$\mathcal{S}_0$-Hamiltonians on a 2D square lattice
↓ (6) Mediator qubits, Subspace encoding, and local unitary.
Arbitrary non-2SLD $\mathcal{S}$-Hamiltonian on a 2D square lattice

In step (1), a semi-classical subset state is attached to a semi-classical subset state $|u\rangle$. The resulting state is also a semi-classical subset state:

$$\begin{aligned} |u\rangle = \frac{1}{\sqrt{|S|}} \sum_{x \in S} |x\rangle \to & \frac{1}{\sqrt{|S|}} \sum_{x \in S} |x\rangle \otimes \frac{1}{\sqrt{|S'|}} \sum_{x' \in S'} |x'\rangle \\ & = \frac{1}{\sqrt{|S||S'|}} \sum_{x \in S \times S'} |x\rangle . \end{aligned} \tag{9}$$

The resulting state after the encodings of steps (2)∼(4) is a semi-classical encoded state because in these steps, a tensor product of single-qubit states is attached to a semi-classical subset state and then the state is encoded by a local isometry. By further performing a local encoding to the semi-classical encoded state, the resulting state is also a semi-classical encoded state. This concludes the proof of Proposition 10. ◀

Finally, we show a BQP-hardness result for the antiferromagnetic Hamiltonian.

▶ **Proposition 12.** *For any $\delta \in (0, 1 - \Omega(1/\mathrm{poly}(n)))$, there exist $a, b \in [0, 1]$ with $b - a \in \Omega(1/\mathrm{poly}(n))$ and $0 \leq c \leq \mathcal{O}(poly(n))$ such that the problem GLHLE$(2, c, a, b, \delta)$ with $b - a \in \Omega(1/\mathrm{poly}(n))$ is* BQP*-hard for Hamiltonians that are restricted to either $\{XX+YY+ZZ\}^+$-Hamiltonian, or $\{XX+YY\}^+$-Hamiltonian on a 2D triangular lattice.*

Proof. We first prove the case of $\{XX+YY+ZZ\}^+$-Hamiltonian. This can be reduced from the GLHLE problem of $\{XX+YY+ZZ\}$-Hamiltonian with a semi-classical encoded state as a guiding state, which is shown to be BQP-hard in Proposition 10. The $\{XX+YY+ZZ\}$-Hamiltonian can be simulated by $\{XX+YY+ZZ\}^+$-Hamiltonian using the "basic gadget" (this is a type of a mediator qubit gadget) of [28]. In the corresponding encoding of the state, a tensor product of two-qubit states is attached to the original state. This encodes a semi-classical encoded state to another semi-classical encoded state. The reason is as follows. Let us denote the attached tensor product of polynomially many two-qubit states as

$$|\phi_1\rangle \otimes |\phi_2\rangle \otimes \cdots |\phi_m\rangle = V_1' |0\rangle \otimes V_2' |0\rangle \otimes \cdots V_m' |0\rangle ,$$

where $|\phi_1\rangle , ..., |\phi_m\rangle$ are two-qubit states and $V_1', ..., V_m'$ are isometries such that $V_i' |0\rangle = |\phi_i\rangle$ for each $i \in [m]$. Then, the original semi-classical encoded state represented by a polynomial-size subset S and a local isometry $V_1 \otimes V_2 \otimes \cdots \otimes V_n$ is mapped to a semi-classical encoded state represented by a subset $S \times \{0...0\}$ and a local isometry $V_1 \otimes \cdots \otimes V_n \otimes V_1' \otimes \cdots \otimes V_m'$. This concludes the case of $\{XX+YY+ZZ\}^+$-Hamiltonian.

We next show the BQP-hardness of the GLHLE problem of $\{XX+YY\}^+$-Hamiltonian on a 2D triangular lattice with a semi-classical encoded state. We show a reduction from the GLHLE problem of $\{XX+YY\}$-Hamiltonian on a 2D square lattice with a semi-classical encoded state as a guiding state, which is shown to be BQP-hard in Proposition 10. It is shown in [28] how to simulate $\{XX+YY\}$-Hamiltonian on a 2D square lattice by $\{XX+YY\}^+$-Hamiltonian on a 2D triangular lattice by using mediator qubit gadgets. The corresponding encoding is just attaching a product state of polynomially many $\mathcal{O}(1)$-qubit states to the original guiding state. Therefore, the original semi-classical encoded state is mapped to another semi-classical encoded state (by a similar reason as in the case of $\{XX+YY+ZZ\}^+$-Hamiltonian). ◀

References

1 Scott Aaronson. Why quantum chemistry is hard. *Nature Physics*, 5:707–708, 2009. doi:10.1038/nphys1415.

2 Daniel S. Abrams and Seth Lloyd. Quantum algorithm providing exponential speed increase for finding eigenvalues and eigenvectors. *Physical Review Letters*, 83:5162–5165, 1999. doi:10.1103/PhysRevLett.83.5162.

3 Andris Ambainis. On physical problems that are slightly more difficult than QMA. In *29th IEEE Conference on Computational Complexity (CCC)*, pages 32–43, 2014.

4 Alán Aspuru-Guzik, Anthony D. Dutoi, Peter J. Love, and Martin Head-Gordon. Simulated quantum computation of molecular energies. *Science*, 309(5741):1704–1707, 2005. doi:10.1126/science.1113479.

5 Bela Bauer, Sergey Bravyi, Mario Motta, and Garnet Kin-Lic Chan. Quantum algorithms for quantum chemistry and quantum materials science. *Chemical Reviews*, 120(22):12685–12717, 2020. doi:10.1021/acs.chemrev.9b00829.

6 Jacob D Biamonte and Peter J Love. Realizable hamiltonians for universal adiabatic quantum computers. *Physical Review A*, 78(1):012352, 2008.

7 Sergey Bravyi, David DiVincenzo, and Daniel Loss. Schrieffer–Wolff transformation for quantum many-body systems. *Annals of physics*, 326(10):2793–2826, 2011.

8 Sergey Bravyi and Matthew Hastings. On complexity of the quantum ising model. *Communications in Mathematical Physics*, 349(1):1–45, 2017.

9 Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3:625–644, 2021. doi:s42254-021-00348-9.

10 Toby Cubitt and Ashley Montanaro. Complexity classification of local hamiltonian problems. *SIAM Journal on Computing*, 45(2):268–316, 2016.

11 Toby Cubitt, Ashley Montanaro, and Stephen Piddock. Universal quantum hamiltonians. *Proceedings of the National Academy of Sciences*, 115(38):9497–9502, 2018.

12 Richard Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6–7):467–488, 1982.

13 Richard Feynman. Quantum mechanical computers. *Optics News*, 11:11, 1985.

14 Sevag Gharibian and Julia Kempe. Hardness of approximation for quantum problems. In *Proceedings of the 39th International Colloquium on Automata, Languages and Programming (ICALP 2012)*, pages 387–398, 2012.

15 Sevag Gharibian and François Le Gall. Dequantizing the quantum singular value transformation: hardness and applications to quantum chemistry and the quantum PCP conjecture. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 19–32, 2022. Full version available as arXiv:2111.09079. doi:10.1145/3519935.3519991.

16 Sevag Gharibian and Ojas Parekh. Almost Optimal Classical Approximation Algorithms for a Quantum Generalization of Max-Cut. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*, volume 145, pages 31:1–31:17, 2019.

17 Sevag Gharibian and Justin Yirka. The complexity of simulating local measurements on quantum systems. *Quantum*, 3:189, 2019. doi:10.22331/q-2019-09-30-189.

18 Alex Bredariol Grilo, Iordanis Kerenidis, and Jamie Sikora. QMA with subset state witnesses. In *International Symposium on Mathematical Foundations of Computer Science*, pages 163–174. Springer, 2015.

19 Aram W. Harrow, Avinatan Hassadim, and Seth Lloyd. Quantum algorithm for solving linear systems of equations. *Physical Review Letters*, 15(103):150502, 2009.

20 Stephen P. Jordan, David Gosset, and Peter J. Love. Quantum-Merlin-Arthur–complete problems for stoquastic hamiltonians and markov matrices. *Phys. Rev. A*, 81:032331, March 2010. arXiv:0905.4755. doi:10.1103/PhysRevA.81.032331.

21 Julia Kempe, Alexei Yu. Kitaev, and Oded Regev. The complexity of the local Hamiltonian problem. *SIAM journal on Computing*, 35(5):1070–1097, 2006.

22 Alexei Yu. Kitaev. Quantum measurements and the Abelian Stabilizer Problem, 1995. arXiv:quant-ph/9511026.

23 Alexei Yu. Kitaev, Alexander H. Shen, and Mikhail N. Vyalyi. *Classical and Quantum Computation*. American Mathematical Society, 2002.

24 Joonho Lee, Dominic W. Berry, Craig Gidney, William J. Huggins, Jarrod R. McClean, Nathan Wiebe, and Ryan Babbush. Even more efficient quantum computations of chemistry through tensor hypercontraction. *PRX Quantum*, 2:030305, 2021. doi:10.1103/PRXQuantum.2.030305.

25 Seunghoon Lee, Joonho Lee, Huanchen Zhai, Yu Tong, Alexander M Dalzell, Ashutosh Kumar, Phillip Helms, Johnnie Gray, Zhi-Hao Cui, Wenyuan Liu, et al. Is there evidence for exponential quantum advantage in quantum chemistry? *arXiv preprint*, 2022. arXiv:2208.02199.

26 Lin Lin and Yu Tong. Near-optimal ground state preparation. *Quantum*, 4:372, 2020.

27 Roberto Oliveira and Barbara M. Terhal. The complexity of quantum spin systems on a two-dimensional square lattice. *Quantum Information and Computation*, 8(10):0900–0924, 2008.

28 Stephen Piddock and Ashley Montanaro. The complexity of antiferromagnetic interactions and 2d lattices. *Quantum Information & Computation*, 17(7-8):636–672, 2017.

29 Markus Reiher, Nathan Wiebe, Krysta M. Svore, Dave Wecker, and Matthias Troyer. Elucidating reaction mechanisms on quantum computers. *Proceedings of the National Academy of Sciences*, 114(29):7555–7560, 2017. doi:10.1073/pnas.1619152114.

30 Norbert Schuch and Frank Verstraete. Computational complexity of interacting electrons and fundamental limitations of density functional theory. *Nature Physics*, 5:732–735, 2009.

31 Yuan Su, Dominic W. Berry, Nathan Wiebe, Nicholas Rubin, and Ryan Babbush. Fault-tolerant quantum simulations of chemistry in first quantization. *PRX Quantum*, 2:040332, November 2021. doi:10.1103/PRXQuantum.2.040332.

32 Tzu-Chieh Wei, Michele Mosca, and Ashwin Nayak. Interacting boson problems can be QMA hard. *Physical Review Letters*, 104:040501, 2010.

33 Leo Zhou and Dorit Aharonov. Strongly universal Hamiltonian simulators. *arXiv preprint*, 2021. arXiv:2102.02991.

A Approximate Hamiltonian simulation

A.1 Introduction of approximate Hamiltonian simulation

While in the QMA-hardness reduction it suffices to focus only on the eigenvalues in the simulation, in the reduction of GLH it is also important to know how the eigenvectors change in the perturbative simulation. It is convenient to introduce the notion of *approximate Hamiltonian simulation* to show the reduction of GLH.

▶ **Definition 13** (Approximate Hamiltonian simulation [11], [33]). *We say that an m-qubit Hamiltonian H' is a (Δ, η, ϵ)-simulation of an n-qubit Hamiltonian H if there exists a local encoding $\mathcal{E}(M) = V(M \otimes P + \bar{M} \otimes Q)V^\dagger$ such that*

1. *There exists an encoding $\tilde{\mathcal{E}}(M) = \tilde{V}(M \otimes P + \bar{M} \otimes Q)\tilde{V}^\dagger$ such that $\tilde{\mathcal{E}}(\mathbb{1}) = P_{\leq\Delta(H')}$ and $\|\tilde{V} - V\| \leq \eta$, where $P_{\leq\Delta(H')}$ is the projector onto the subspace spanned by eigenvectors of H' with eigenvalue below Δ,*
2. *$\|H'_{\leq\Delta} - \tilde{\mathcal{E}}(H)\| \leq \epsilon$, where $H'_{\leq\Delta} \coloneqq P_{\leq\Delta(H')}H'$.*

Here, V is a local isometry that can be written as $V = \bigotimes_i V_i$ where each V_i is an isometry acting on at most 1 qubit, and P and Q are locally orthogonal projectors (i.e. for all i there exist orthogonal projectors P_i and Q_i acting on the same subsystem as V_i such that $P_iQ_i = 0$, $P_iP = P$ and $Q_iQ = Q$) such that $P + Q = I$, and $\bar{M}$ is the complex conjugate of M. Moreover, we say that the simulation is efficient if m and $\|H'\|$ are at most $\mathcal{O}(\mathrm{poly}(n, \eta^{-1}, \epsilon^{-1}, \Delta))$, and the description of H' can be computable in $\mathrm{poly}(n)$ time given the description of H.

We approximately simulate the original Hamiltonian H in the low-energy subspace of H'. There is a corresponding encoding of a state which can be taken as

$$\mathcal{E}_{\mathrm{state}}(\rho) = V(\rho \otimes \sigma)V^\dagger$$

for σ such that $P\sigma = \sigma$ (if $P \neq 0$). If ρ is the eigenvector of H with eigenvalue α, then $\mathcal{E}_{\mathrm{state}}(\rho)$ is approximately the eigenvector of H' with eigenvalue $\alpha' \in [\alpha - \epsilon, \alpha + \epsilon]$.

In [33], it is shown that there exist families of Hamiltonians that can efficiently simulate any $\mathcal{O}(1)$-local Hamiltonians. They call such families of Hamiltonians *strongly universal Hamiltonians.*[7] We use the construction of strongly universal Hamiltonians of [33] to show Proposition 9. Formally, the strong (and weak) universality is defined as follows:

▶ **Definition 14** (Strong and weak universality [33]). *A family of Hamiltonians $\mathcal{H} = \{H_m\}$ is weakly universal if given any $\Delta, \eta, \epsilon > 0$, any $\mathcal{O}(1)$-local, n-qubit Hamiltonian can be (Δ, η, ϵ)-simulated. Such a family is strongly universal if the simulation is always efficient.*

The following result is shown in [33]:

▶ **Theorem 15** ([33]). *Any non-2SLD $\mathcal{S}$-Hamiltonian on a 2D-square lattice is strongly universal.*

B Schrieffer-Wolf transformation for 1-dimensional gapped ground space

Let us introduce the Schrieffer-Wolf transformation and its approximation [7] which we use in the proof. We only consider the case when the unperturbed Hamiltonian has 1-dimensional ground space.

Let H_0 be a Hamiltonian that has 1-dimensional ground space spanned by $|g_0\rangle$ whose energy is 0. Let us assume that the smallest non-zero eigenvalue of H_0 is larger than one. Consider the following (perturbed) Hamiltonian: $H = \Delta H_0 + V$. We shall always assume that $\|V\| \leq \Delta/2$ in the following. Then, there is only one eigenvector (which we denote $|g\rangle$) of H with eigenvalue lying in the interval of $[-\Delta/2, \Delta/2]$ (Lemma 3.1 of [7]).

Then, the Schrieffer-Wolf (SW) transformation is defined as a unitary U_{SW} that maps the ground space of H to that of H_0. That is, $U_{\mathrm{SW}}|g\rangle = |g_0\rangle$. The Hamiltonian

$$H_{\mathrm{eff}} = \Pi_0 U_{\mathrm{SW}}(\Delta H_0 + V)U_{\mathrm{SW}}^\dagger \Pi_0$$

is called the effective low-energy Hamiltonian. Here, Π_0 is the projector onto the ground space of H_0. The eigenvector of H_{eff} is $|g_0\rangle$ and the eigenvalue is the same as the eigenvalue of $|g\rangle$ with respect to H.

Next, we show how to approximate U_{SW} and H_{eff}. We only need the simplest first-order approximation in the proof of Proposition 7. In the following, we further assume $\|V\| \leq \Delta/16$. Then, it is known that

$$\|I - U_{\mathrm{SW}}\| \in \mathcal{O}(\Delta^{-1}\|V\|) \tag{10}$$

and

$$\|H_{\mathrm{eff}} - \Pi_0 V \Pi_0\| \in \mathcal{O}(\Delta^{-1}\|V\|^2) \tag{11}$$

hold (Lemma 3.4 [7], Lemma 4 [8]). This means that I and $\Pi_0 V \Pi_0$ work as the first-order approximation of U_{SW} and H_{eff}, respectively. The derivation and the forms of the higher-order terms can be found in [7]. From eq. (10), it follows that

$$\| \, |g\rangle - |g_0\rangle \, \| = \left\| (I - U_{\mathrm{SW}}^\dagger)\, |g_0\rangle \right\| \in \mathcal{O}(\Delta^{-1}\|V\|). \tag{12}$$

It follows from eq. (11) that the ground state energy of H differs at most $\mathcal{O}(\Delta^{-1}\|V\|^2)$ from the eigenvalue of $H_{\mathrm{eff},1} \coloneqq \Pi_0 V \Pi_0$ (restricted to the space spanned by $|g_0\rangle$).

[7] It would be possible to show Theorem 1 by modifying the verifier circuit $\tilde{U}_x$ following [27] to make the constructed Hamiltonian spatially sparse. We believe Proposition 9 is interesting because the reduction holds for arbitrary $\mathcal{O}(1)$-local Hamiltonian even if it is not originally spatially sparse.

C Encoding of states for strong Hamiltonian simulation

We sketch the construction of the strong Hamiltonian simulation introduced in [33]. The simulation mainly consists of two parts. First, they construct spatially sparse 5-local Hamiltonian [27] using a quantum phase estimation circuit and its modification. This procedure may be thought of as a "Hamiltonian-to-circuit" (then goes back to Hamiltonian by circuit-to-Hamiltonian) construction. Then, they perturbatively simulate the spatially sparse Hamiltonian with known techniques in the literature [27, 11, 28]. In the following, we overview their construction.

(1) Arbitrary k-local Hamiltonian $\rightarrow$ spatially sparse 5-local Hamiltonian ([33])

Let H be a target $\mathcal{O}(1)$-local Hamiltonian. Assume that H can be written as $H = \sum_i E_i |\psi_i\rangle \langle\psi_i|$ where $\{E_i\}$ and $\{|\psi_i\rangle\}$ are the eigenvalues and eigenvectors of H. In [33], they showed that there is a spatially sparse quantum circuit $U_{\mathrm{PE}}^{\mathrm{sparse}}$ that approximately estimates the energy of H, i.e.

$$U_{\mathrm{PE}}^{\mathrm{sparse}} \sum_i c_i |\psi_i\rangle |0^m\rangle \approx \sum_i c_i |\psi_i\rangle |\tilde{E}_i\rangle |other\rangle ,$$

where $\{c_i\}$ are arbitrary coefficients and $\{|\tilde{E}_i\rangle\}$ are approximations of $\{E_i\}$.

The circuit $U_{\mathrm{PE}}^{\mathrm{sparse}}$ is implemented first by constructing $U_{\mathrm{NN}}^{\mathrm{sparse}}$ that consists of 1D nearest-neighborhood interaction. Then, $U_{\mathrm{NN}}^{\mathrm{sparse}}$ is converted into a spatially sparse circuit using ancilla qubits and swap gates.

Then they combine uncomputation and idling to construct

$$U = (\mathrm{Idling})(U_{\mathrm{PE}}^{\mathrm{sparse}})^\dagger (\mathrm{Idling}) U_{\mathrm{PE}}^{\mathrm{sparse}} .$$

They apply circuit-to-Hamiltonian construction for this U to construct spatially sparse 5-local Hamiltonian H_{circuit}. They use first-order perturbation theory to show that H_{circuit} simulates H in its low-energy subspace. The encoding of H_{circuit} to the low energy subspace of H is approximated by the map: $H \rightarrow H \otimes |\alpha\rangle \langle\alpha|$. Here, $|\alpha\rangle$ is a subset state with poly(n)-size subset S' that is related to the history state of the idling steps after uncomputation. For detail, see the proof of Proposition 2 of [33]. Then, the corresponding encoding of the state is

$$|u\rangle \rightarrow |u\rangle \otimes |\alpha\rangle .$$

The encoded state is also a semi-classical subset state if $|u\rangle$ is a semi-classical subset state.

(2) Spatially sparse 5-local Hamiltonian $\rightarrow$ Spatially sparse 10-local real Hamiltonian (Lemma 22 of [11])

In this simulation, the state is encoded by attaching polynomially many $|+_y\rangle$ where $|+_y\rangle$ is the $+1$ eigenvector of Pauli Y matrix:

$$|u\rangle \rightarrow |u\rangle \otimes |+_y\rangle \otimes \cdots \otimes |+_y\rangle . \tag{13}$$

This encoding does not map a semi-classical subset state into a semi-classical state but maps into a semi-classical encoded state. The reason is as follows. Let V_y be a unitary such that $|+_y\rangle = V_y |0\rangle$, and $|u\rangle = 1/\sqrt{|S|} \sum_{x\in S} |x\rangle$. Then, the right side of eq. (13) can be written as

$$|u\rangle \otimes |+_y\rangle \otimes \cdots \otimes |+_y\rangle = \frac{1}{\sqrt{|S|}} \sum_{x\in S\times\{0\ldots0\}} I \otimes \cdots \otimes I \otimes V_y \otimes \cdots \otimes V_y |x\rangle .$$

This is a semi-classical encoded state with a subset $S \times \{0...0\}$ and a local isometry (this is indeed a local unitary) $I \otimes \cdots \otimes I \otimes V_y \otimes \cdots \otimes V_y$.

(3) Spatially sparse 10-local real Hamiltonian → Spatially sparse 2-local Pauli interactions with no Y-terms ([27, 10])

This can be done first by simulating the 10-local real Hamiltonian with 11-local Hamiltonian whose Pauli decomposition does not contain any Pauli Y terms [11, Lemma 40]. In the corresponding encoding, $|1\rangle$ states are attached for the polynomially many mediator qubits introduced in the simulation. Then, we can use subdivision gadgets and 3-to-2 gadgets [27]. In this simulation, polynomially many mediator qubits are introduced, and the encoding of states is just to add $|0\rangle$ states for each of the mediator qubits. The resulting Hamiltonian can be written in the form $\sum_{i<j}\alpha_{ij}A_{ij}+\sum_k(\beta_k X_k+\gamma_k Z_k)$, where A_{ij} is one of the interactions of X_iX_j, X_iZ_j, Z_iX_j or Z_iZ_j.

(4) Subspace encoding for spatially sparse $\mathcal{S}_0=\{XX+YY+ZZ\}$ or $\{XX+YY\}$ Hamiltonian (Theorem 42 of [11])

We have already obtained 2-local Hamiltonian in the form $\sum_{i<j}\alpha_{ij}A_{ij}+\sum_k(\beta_k X_k+\gamma_k Z_k)$. Then we show how to simulate this Hamiltonian with arbitrary non-2SLD $\mathcal{S}$-Hamiltonians. We first consider $\mathcal{S}_0$ Hamiltonian, where $\mathcal{S}_0=\{XX+YY+ZZ\}$ or $\mathcal{S}_0=\{XX+YY\}$. In this simulation, we use subspace encoding in which the *logical qubit* of the original Hamiltonian is encoded into four *physical qubits*. Consider the simulation by Heisenberg interaction $\{XX+YY+ZZ\}$ for example. Each logical qubit is encoded into 4 qubit state by an isometry that is defined as

$$V|0\rangle=|0_L\rangle=|\Psi_-\rangle_{13}|\Psi_-\rangle_{24} \tag{14}$$

$$V|1\rangle=|1_L\rangle=\frac{2}{\sqrt{3}}|\Psi_-\rangle_{12}|\Psi_-\rangle_{34}-\frac{1}{\sqrt{3}}|\Psi_-\rangle_{13}|\Psi_-\rangle_{24}, \tag{15}$$

where $|\Psi_-\rangle=(|01\rangle-|10\rangle)/\sqrt{2}$. For details, see [11, Theorem 42]. The encoding of states for XX+YY interaction is the same. A semi-classical encoded state is clearly mapped to a semi-classical encoded state by applying a local isometry of the corresponding subspace encoding.

(5) Spatially sparse $\mathcal{S}_0$-Hamiltonian → $\mathcal{S}_0$-Hamiltonians on a 2D square lattice (Lemma 47 of [11])

This simulation can be done using three perturbative gadgets called subdivision, fork, and crossing gadgets. All of these gadgets attach a mediator qubit for each use of the gadgets. $\mathcal{O}(1)$ rounds of parallel use of perturbative gadgets are sufficient to simulate a spatially sparse $\mathcal{S}_0$-Hamiltonian by a $\mathcal{S}_0$-Hamiltonians on a 2D square lattice, which prevents the interaction strength to grow exponentially. (For general interaction graphs, $\mathcal{O}(\log n)$ rounds of perturbative simulations are necessary.)

(6) $\mathcal{S}_0$-Hamiltonian on 2D square lattice → Arbitrary non-SLD $\mathcal{S}$-Hamiltonian on a 2D square lattice (Theorem 43 of [11])

Finally, this simulation is similarly done by using variants of mediator qubit gadgets or subspace encoding gadgets as well as applying local unitaries.[8]

[8] Applying local unitaries means to simulate H by $U^{\otimes n}H(U^\dagger)^{\otimes n}$ where U acts on one qubit. The corresponding encoding of state is $\mathcal{E}_{state}(|\psi\rangle)=U^{\otimes n}|\psi\rangle$.

Planar #CSP Equality Corresponds to Quantum Isomorphism – A Holant Viewpoint

Jin-Yi Cai
Department of Computer Sciences, University of Wisconsin-Madison, WI, USA

Ben Young[1]
Department of Computer Sciences, University of Wisconsin-Madison, WI, USA

Abstract

Recently, Mančinska and Roberson proved [11] that two graphs G and G' are *quantum isomorphic* if and only if they admit the same number of homomorphisms from all *planar* graphs. We extend this result to planar #CSP with any pair of sets $\mathcal{F}$ and $\mathcal{F}'$ of real-valued, arbitrary-arity constraint functions. Graph homomorphism is the special case where each of $\mathcal{F}$ and $\mathcal{F}'$ contains a single symmetric 0-1-valued binary constraint function. Our treatment uses the framework of planar Holant problems. To prove that quantum isomorphic constraint function sets give the same value on any planar #CSP instance, we apply a novel form of *holographic transformation* of Valiant [13], using the quantum permutation matrix $\mathcal{U}$ defining the quantum isomorphism. Due to the noncommutativity of $\mathcal{U}$'s entries, it turns out that this form of holographic transformation is only applicable to planar Holant. To prove the converse, we introduce the quantum automorphism group $\mathrm{Qut}(\mathcal{F})$ of a set of constraint functions/tensors $\mathcal{F}$, and characterize the intertwiners of $\mathrm{Qut}(\mathcal{F})$ as the signature matrices of planar Holant$(\mathcal{F} \mid \mathcal{EQ})$ quantum gadgets. Then we define a new notion of (projective) connectivity for constraint functions and reduce arity while preserving the quantum automorphism group. Finally, to address the challenges posed by generalizing from 0-1 valued to real-valued constraint functions, we adapt a technique of Lovász [9] in the classical setting for isomorphisms of real-weighted graphs to the setting of quantum isomorphisms.

2012 ACM Subject Classification Mathematics of computing → Graph theory; Theory of computation → Problems, reductions and completeness

Keywords and phrases #CSP, Quantum isomorphism, Holant, Gadget, Intertwiners, Planar graphs

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.33

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2212.03335`

Acknowledgements The authors thank David Roberson for his insightful comments and suggestions, and Austen Fan for helpful discussions.

1 Introduction

Graph Homomorphism and #CSP

A homomorphism from graph K to graph X is an edge-preserving map from the vertex set $V(K)$ of K to the vertex set $V(X)$ of X. A well-studied problem in complexity theory is to count the number of distinct homomorphisms from K to X, which can be expressed as

$$\sum_{\sigma: V(K) \to V(X)} \prod_{(u,v) \in E(K)} (A_X)_{\sigma(u),\sigma(v)},$$

[1] Corresponding author

the value of the *partition function* of X evaluated on K, where A_X is the adjacency matrix of X. From this perspective, graph homomorphism naturally generalizes to a counting constraint satisfaction problem (#CSP) by replacing $\{A_X\}$ with a set $\mathcal{F}$ of $\mathbb{R}$ or $\mathbb{C}$-valued *constraint functions* on one or more inputs from a finite domain $V(\mathcal{F})$, and replacing K with a set of constraints and variables, where each constraint applies a constraint function to a sequence of variables. The problem is to compute the partition function, which is the sum over all variable assignments of the product of the constraint function evaluations. Letting $V(\mathcal{F}) = V(X)$ and the constraint and variable sets be $E(K)$ and $V(K)$, respectively, with each edge/constraint applying A_X to its two endpoints, we recover the special case of counting homomorphisms from K to X.

Bulatov [2] proved that every problem #CSP($\mathcal{F}$), parameterized by a finite set $\mathcal{F}$ of 0-1-valued constraint functions, is either (1) solvable in polynomial-time or (2) #P-complete. Dyer and Richerby [6] proved that this *complexity dichotomy* has a decidable criterion. This dichotomy was further extended to nonnegative real-valued, and then to all complex-valued constraint functions [5, 4]. When we restrict to planar #CSP instances (for which the bipartite constraint-variable incidence graph is planar), a further *complexity trichotomy* is known for the Boolean domain (where $V(\mathcal{F}) = \{0, 1\}$) [7], that there are exactly three classes: (1) polynomial-time solvable; (2) #P-hard for general instances but solvable in polynomial-time over planar structures; and (3) #P-hard over planar structures. Furthermore, Valiant's holographic algorithm with matchgates [13] is universal for all problems in class (2): *Every* #P-hard #CSP problem that is solvable in polynomial-time in the planar setting is solvable by this one algorithmic strategy. However, for planar #CSP on domains of size greater than 2, a full complexity classification is open.

Holant Problems

We carry out much of our work in the planar Holant framework from counting complexity, which we find natural to this theory, and of which planar #CSP itself is a special case. Like a #CSP problem, a Holant problem is parameterized by a set $\mathcal{F}$ of constraint functions. The input to a planar Holant problem is a *signature grid*, a planar graph where each edge represents a variable and every vertex is assigned a constraint function from $\mathcal{F}$. A vertex's constraint function is applied to its incident edges. This is dual to the #CSP view of graph homomorphism, where each edge is a (necessarily binary) constraint and each vertex is a variable. As with #CSP, the computational problem is to compute the Holant value – the sum over all variable (edge) assignments, of the product of the evaluations of the constraint functions. A (planar) *gadget* is a (planar) Holant signature grid with a number of *dangling* edges, representing external variables. Each gadget has an associated *signature matrix*, which stores the Holant value for each fixed assignment to the dangling edges. The study of Holant problems is motivated by Valiant's *holographic transformations* [13], which are certain Holant value-preserving transformations of the constraint functions by invertible matrices.

Classical and Quantum Isomorphism

As suggested above, one can view a q-vertex real-weighted graph X, via its adjacency matrix $A_X \in \mathbb{R}^{q\times q}$, as an $\mathbb{R}$-valued binary (i.e. two input variables) constraint function. Two q-vertex graphs X and Y are isomorphic if one can apply a permutation to the rows and columns of A_X to obtain A_Y. Equivalently, if we convert A_X and A_Y to vectors $a_X, a_Y \in \mathbb{R}^{q^2}$, there is a permutation matrix P satisfying $P^{\otimes 2}a_X = a_Y$. For n-ary constraint functions $F, G \in \mathbb{R}^{[q]^n}$, where $[q] = \{1, 2, \ldots, q\}$, a natural generalization applies. F and G are isomorphic if there is a permutation matrix P satisfying $P^{\otimes n}f = g$, where $f, g \in \mathbb{R}^{q^n}$ are the vector versions of F and G.

Quantum isomorphism of (undirected, unweighted) graphs, introduced in [1], is a relaxation of classical isomorphism. Graphs X and Y are *quantum isomorphic* if there is a perfect winning strategy in a two-player *graph isomorphism game* in which the players share and can perform measurements on an entangled quantum state. This condition is equivalent to the existence of a *quantum permutation matrix* matrix $\mathcal{U}$ – a relaxation of a permutation matrix whose entries do not necessarily commute – satisfying $\mathcal{U}^{\otimes 2}a_X = a_Y$ [10]. Analogously to classical isomorphism, in this work we define n-ary constraint functions F and G to be *quantum isomorphic* if there is a quantum permutation matrix $\mathcal{U}$ satisfying $\mathcal{U}^{\otimes n}f = g$. Sets $\mathcal{F}$ and $\mathcal{G}$ of constraint functions of equal cardinality are quantum isomorphic if there is a single quantum permutation matrix defining a quantum isomorphism between every pair of corresponding functions in $\mathcal{F}$ and $\mathcal{G}$.

In [8], Lovász proved that two graphs are isomorphic if and only if they admit the same number of homomorphisms from every graph. Fifty years later, Mančinska and Roberson [11] proved that two graphs are *quantum* isomorphic if and only if they admit the same number of homomorphisms from all *planar* graphs. We generalize this result to #CSP and sets of constraint functions. We achieve this via graph combinatorics, results from quantum group theory, and a novel form of holographic transformation, establishing new connections between planar Holant, #CSP, quantum permutation matrices, and quantum isomorphism.

While quantum permutation matrices, quantum isomorphism, and other quantum constructions in this paper are somewhat abstract and technical, we believe it is precisely these concepts' abstractness that makes the connections we develop between them and the very concrete, combinatorial concept of planarity so fascinating and potentially fruitful. Our result that quantum isomorphism exactly captures planarity could lead to entirely novel, algebraic methods of studying the complexity of planar #CSP and Holant.

Our Results

Our main result is the following theorem, a broad extension of the main result of Mančinska and Roberson [11], recast into the well-studied Holant and #CSP frameworks.

▶ **Theorem** (Theorem 9, informal). *Sets $\mathcal{F}$ and $\mathcal{G}$ of $\mathbb{R}$-valued constraint functions are quantum isomorphic iff the partition function of every planar #CSP($\mathcal{F}$) instance is preserved upon replacing every constraint function in $\mathcal{F}$ with the corresponding function in $\mathcal{G}$.*

Our general constraint functions add significant complexity relative to the graph homomorphism special case in [11], since, unlike unweighted graph adjacency matrices, they can be (1) asymmetric (i.e. permuting the argument order affects their value), (2) n-ary, for $n > 2$, and (3) arbitrary real-valued. Each of these three extensions adds intricacies and challenges not present in [11], which we address with novel approaches that reveal new, deeper connections between quantum permutation matrices and planar graphs.

First, in Subsection 3.1 we give a procedure for decomposing any planar Holant signature grid corresponding to a planar #CSP instance into a small set of simple gadgets. Here arise the first new complications associated with higher-arity signatures. The dangling edges of simple gadgets extracted from the signature grid may not be oriented correctly, so we must use certain other gadgets to pivot them to the correct orientation, respecting planarity.

With some preparation in Subsection 3.2, we prove the quantum Holant theorem in Subsection 3.3. The forward direction of Theorem 9 is a direct corollary, giving a more graphical and more intuitive proof than that of the graph homomorphism special case in [11]. The gadget decomposition gives an expression for the Holant value as a product of the component gadgets' signature matrices. So, assuming $\mathcal{F}$ and $\mathcal{G}$ are quantum isomorphic, we

use the quantum permutation matrix $\mathcal{U}$ defining the quantum isomorphism as a *quantum holographic transformation*, inserting tensor powers of $\mathcal{U}$ and its inverse between every pair of signature matrices in the product without changing the Holant value. Then a sequence of these holographic transformations converts every signature in $\mathcal{F}$ to the corresponding signature in $\mathcal{G}$. The quantum holographic transformation does not work on general signature grids, since viewing $\mathcal{U}$ itself as a constraint function in the signature grid is not in general well-defined, as $\mathcal{U}$'s entries do not commute and the partition function does not specify an order to multiply the constraint function evaluations. However, the planarity of the signature grid and the resulting gadget decomposition and matrix product expression for the Holant value implicitly provide a multiplication order. Quantum holographic transformations apply to the planar version of the general Holant problem parameterized by a set $\mathcal{F}$ of constraint functions (not just the special case of #CSP), and should be of independent interest.

The success of the quantum holographic transformation for asymmetric signatures is also dependent on the fact that the holographic transformation action of a quantum permutation matrix is invariant under gadget rotations and reflections. The asymmetry and rotation and reflection issues are only relevant in the context of planar signature grids, since in nonplanar grids, one can simply cross and twist the incident edges to achieve the desired input order. Hence this is another interesting connection between quantum permutation matrices and the structural properties of planar graphs.

In Section 4, to prove the reverse direction of Theorem 9, we turn to the theory of quantum groups [15, 14]. We introduce the quantum automorphism group $\mathrm{Qut}(\mathcal{F})$ of a set $\mathcal{F}$ of signatures, an abstraction of the classical automorphism group satisfying many of the same properties. Using the planar gadget decomposition, we prove that the signature matrices of planar Holant gadgets in the context of $\#\mathrm{CSP}(\mathcal{F})$, a very concrete, combinatorial concept, exactly capture the abstract *intertwiner space* of $\mathrm{Qut}(\mathcal{F})$. A natural approach to the rest of the proof breaks down for constraint functions of arity > 2. Hence we introduce a method to reduce a constraint function's arity while maintaining its inclusion in the original intertwiner space. Then we say a constraint function is *projectively connected* if this procedure yields a connected graph upon reaching arity 2. Finally, we show that if $\mathcal{F}$ and $\mathcal{G}$ are projectively connected and the quantum automorphism group of the disjoint union of $\mathcal{F}$ and $\mathcal{G}$ maps a "vertex" of $\mathcal{F}$ to a "vertex" of $\mathcal{G}$, then $\mathcal{F}$ and $\mathcal{G}$ are quantum isomorphic (analogous to the familiar classical fact for graphs). For 0-1 valued functions, Mančinska and Roberson [11] ensured connectivity by taking complements. However, for real-valued functions F and G this method does not work: we cannot take the complement to assume they are projectively connected. Instead, we adapt to the quantum setting a technique of Lovász [9] in the classical setting for real-weighted graphs, and extract a quantum isomorphism to complete the proof.

All of the above results extend to sets of constraint functions over $\mathbb{C}$ that are closed under conjugation and for which the quantum isomorphism respects conjugation (both properties are trivially satisfied by constraint functions over $\mathbb{R}$). In the full version, our proof is carried out in this setting. In this extended abstract, we specialize to constraint functions over $\mathbb{R}$.

In Appendix A, we give an alternate approach for enforcing constraint function connectivity due to Roberson [12], which adds new binary connected constraint functions to $\mathcal{F}$ and $\mathcal{G}$ rather than modify the existing constraint functions to be projectively connected. We explore two further topics in the full version. First, we extend the connection between quantum isomorphism and nonlocal games. We define graph isomorphism games for real-weighted directed graphs and prove the following generalization of a result in [11]: real-weighted graphs F and G admit the same number of homomorphisms from all planar graphs if and only if there is a perfect quantum commuting strategy for the (F, G)-isomorphism game. Second,

we discuss how pivoting dangling edges around a gadget and horizontally reflecting gadgets, graphical manipulations that arise naturally throughout our work, correspond to the dual and adjoint operations in the pivotal dagger category of gadgets.

We hope that our results, in particular the quantum holographic transformation technique in Theorem 18, will lead to further applications of quantum group theory in the study of planar #CSP and Holant complexity.

2 Preliminaries

Constraint functions and #CSP

▶ **Definition 1** (Constraint function, $V(F)$, $V(\mathcal{F})$). *A tensor $F \in \mathbb{R}^{[q]^n}$, for $q, n \geq 1$, is a* constraint function *of domain size q and arity n. For $\mathbf{x} \in [q]^n$, we write $F_{\mathbf{x}} = F_{x_1,\ldots,x_n} = F(x_1, \ldots, x_n) \in \mathbb{R}$. We write $V(F)$ for $[q]$, thus $F \in \mathbb{R}^{V(F)^n}$. Whenever we specify a set $\mathcal{F}$ of constraint functions, it is assumed that all $F \in \mathcal{F}$ have the same domain, which we call $V(\mathcal{F})$, with $|V(\mathcal{F})| = q$.*

▶ **Definition 2** (#CSP, Z). *A* #CSP problem *#CSP($\mathcal{F}$) is parameterized by a set $\mathcal{F}$ of constraint functions. A* #CSP($\mathcal{F}$) instance *K is defined by a pair (V, C), where V is a set of variables and C is a multiset of* constraints*. Each constraint $c = (F^c, v_{c_1}, \ldots, v_{c_{n_F}})$ consists of a constraint function $F^c \in \mathcal{F}$ and an ordered tuple of variables to which F is applied. The* partition function *Z, on input #CSP($\mathcal{F}$) instance K, outputs*

$$Z(K) = \sum_{\sigma: V \to V(\mathcal{F})} \prod_{(F^c, v_{c_1}, \ldots, v_{c_{n_F}}) \in C} F^c(\sigma(v_{c_1}), \ldots, \sigma(v_{c_{n_F}})).$$

▶ **Definition 3** (Compatible constraint function sets, $K_{\mathcal{F}\to\mathcal{G}}$). *Let $\mathcal{F} = \{F_i\}_{i\in[t]}$, $\mathcal{G} = \{G_i\}_{i\in[t]}$ be two sets of constraint functions on the same domain $[q]$. $\mathcal{F}$ and $\mathcal{G}$ are* compatible *if, for all $i \in [t]$, F_i and G_i have common arity n_i. Call F_i and G_i* corresponding *constraint functions.*

For compatible $\mathcal{F}$ and $\mathcal{G}$ and any #CSP($\mathcal{F}$) instance K, define a #CSP($\mathcal{G}$) instance $K_{\mathcal{F}\to\mathcal{G}}$ by replacing every constraint $(F_i, v_{i_1}, \ldots, v_{i_{n_i}})$ of K with the corresponding constraint $(G_i, v_{i_1}, \ldots, v_{i_{n_i}})$.

Often it will be useful to "flatten" a constraint function F into a matrix:

▶ **Definition 4** ($F^{m,d}$, f). *For $F \in \mathbb{R}^{[q]^n}$ and any $m, d \geq 0$, $m + d = n$, let $F^{m,d} \in \mathbb{R}^{[q^m]\times[q^d]}$ be the $q^m \times q^d$ matrix defined by $F^{m,d}_{x_1\ldots x_m, x_n\ldots x_{m+1}} = F(x_1, \ldots, x_n)$, where $x_1 \ldots x_m \in \mathbb{N}$ is the base-q integer with the most significant digit x_1, and similarly for $x_n \ldots x_{m+1}$ (in decreasing index). We write $f = F^{n,0} \in \mathbb{R}^{q^n}$; it is called the* signature vector *of F.*

Quantum permutation matrices and quantum isomorphism

A core construction in this work is the *quantum permutation matrix*, a generalization of classical permutation matrix, whose entries come from an arbitrary C^*-algebra rather than $\{0, 1\}$. For the purposes of this work, one can view a C^*-algebra as simply an abstraction of $\mathbb{C}$, equipped with an involution $*$ analogous to conjugation, and whose elements, critically, do not necessarily commute. More generally, one can think of a C^*-algebra as the algebra of bounded operators on a Hilbert space.

▶ **Definition 5** (Quantum permutation matrix). *A matrix $\mathcal{U} = (u_{ij})$ with entries from a C^*-algebra with unit element $\mathbf{1}$ is called a* quantum permutation matrix *if it satisfies the following conditions for all i, j:*

- $u_{ij}^2 = u_{ij} = u_{ij}^*$;
- $\sum_j u_{ij} = \sum_i u_{ij} = \mathbf{1}$.

If the C^*-algebra in question is $\mathbb{C}$, then the first condition implies $\mathcal{U}$ is a 0-1 matrix, and then the second condition implies $\mathcal{U}$ is a classical permutation matrix. Hence the abstraction of $\mathbb{C}$ to an arbitrary C^*-algebra is one of the many abstractions from "classical" to "quantum" constructions throughout this work.

Recall that graphs X and Y with adjacency matrices $A_X, A_Y \in \{0,1\}^{[q]\times[q]}$ are classically isomorphic if and only if $PA_X = A_Y P$ for some classical permutation matrix P. Hence we say X and Y are *quantum isomorphic* ($X \cong_{qc} Y$) [1, 10] if there is a quantum permutation matrix $\mathcal{U}$ satisfying $\mathcal{U}A_X = A_Y\mathcal{U}$. Equivalently, $\mathcal{U}^{\otimes 2}a_X = a_Y$, where $a_X, a_Y \in \{0,1\}^{q^2}$ are the signature vectors of A_X and A_Y. Hence the following definition is a generalization of quantum graph isomorphism to higher-arity constraint functions over $\mathbb{R}$.

▶ **Definition 6** ($\cong_{qc}$). *$F, G \in \mathbb{R}^{[q]^n}$ are* quantum isomorphic *($F \cong_{qc} G$) if there is a $q \times q$ quantum permutation matrix $\mathcal{U}$ satisfying $\mathcal{U}^{\otimes n} f = g$. Compatible sets $\mathcal{F}$ and $\mathcal{G}$ of constraint functions are* quantum isomorphic *($\mathcal{F} \cong_{qc} \mathcal{G}$) if there is a $q \times q$ quantum permutation matrix $\mathcal{U}$ satisfying $\mathcal{U}^{\otimes \operatorname{arity}(F_i)} f_i = g_i$ for every i.*

Holant, gadgets and signature matrices

A Holant problem Holant($\mathcal{F}$), like a #CSP problem, is parameterized by a set $\mathcal{F}$ of constraint functions, usually called *signatures*. The input to Holant($\mathcal{F}$) is a *signature grid* Ω, which consists of an underlying multigraph X with vertex set V and edge set E. Each vertex $v \in V$ is assigned a signature $F_v \in \mathcal{F}$ of arity $\deg(v)$. The incident edges $E(v) = (e_1^v, \ldots, e_{\deg(v)}^v)$ of v are input variables to F_v taking values in $V(\mathcal{F})$. We use Pl-Holant($\mathcal{F}$) to specify that input signature grids must have planar underlying multigraphs. For planar Holant, the input variables of F_v are labeled in cyclic order starting with one particular edge, labeled with a diamond. The output on input Ω is

$$\operatorname{Holant}_\Omega(\mathcal{F}) = \sum_{\sigma: E \to V(\mathcal{F})} \prod_{v \in V} F_v(\sigma|_{E(v)}), \tag{1}$$

where $F_v(\sigma|_{E(v)}) = (F_v)(\sigma(e_1^v), \ldots, \sigma(e_{\deg(v)}^v))$. For sets $\mathcal{F}$ and $\mathcal{G}$ of signatures, define the problem Holant($\mathcal{F} \mid \mathcal{G}$) as follows. A signature grid in the context of Holant($\mathcal{F} \mid \mathcal{G}$) has a bipartite underlying multigraph with bipartition $V = V_1 \sqcup V_2$ such that the vertices in V_1 and V_2 are assigned signatures from $\mathcal{F}$ and $\mathcal{G}$, respectively.

The Holant problems in this work always include the following set of signatures.

▶ **Definition 7** (E_n, $\mathcal{EQ}$). *For fixed q, define the 0-1-valued* equality *constraint function $E_n \in \mathbb{R}^{[q]^n}$ by $E_n(x_1, \ldots, x_n) = 1$ iff $x_1 = \ldots = x_n$. Define $\mathcal{EQ} = \bigcup_n E_n$.*

To each #CSP($\mathcal{F}$) instance $K = (V, C)$ we associate a signature grid Ω_K in the context of Holant($\mathcal{F} \mid \mathcal{EQ}$) defined as follows: For every constraint $c \in C$, if c applies function F of arity n, create a degree-n vertex assigned F, called a *constraint vertex*. For each variable $v \in V$, if v appears in the mulitset of constraints $C_v \subseteq C$, create a degree-$|C_v|$ vertex assigned $E_{|C_v|} \in \mathcal{EQ}$, called an *equality vertex*, and edges (v, c) for every $c \in C_v$ such that the cyclic order of edges incident to each constraint vertex matches the order of variables in the constraint. Any edge assignment σ must assign all edges incident to an equality vertex the same value (or else the term corresponding to σ is 0), so we can view σ as #CSP variable assignent. Hence $Z(K) = \operatorname{Holant}_{\Omega_K}(\mathcal{F} \mid \mathcal{EQ})$.

For example, to compute the number of homomorphisms $K \to X$, consider a $\#\mathrm{CSP}(A_X)$ instance where the vertices of K are variables and each edge of K is a constraint applying function $A_X \in \mathbb{R}^{V(X)^2}$ (X's adjacency matrix) to the edge's two endpoints. The corresponding Holant signature grid Ω_K starts with underlying graph K, with K's vertices assigned the appropriate equality signature from $\mathcal{EQ}$, and we subdivide each of K's edges by placing degree-2 constraint vertices, assigned signature A_X, connected to the labeled equality vertices. See Figure 1. We always depict equality and constraint vertices as circles and squares, respectively.

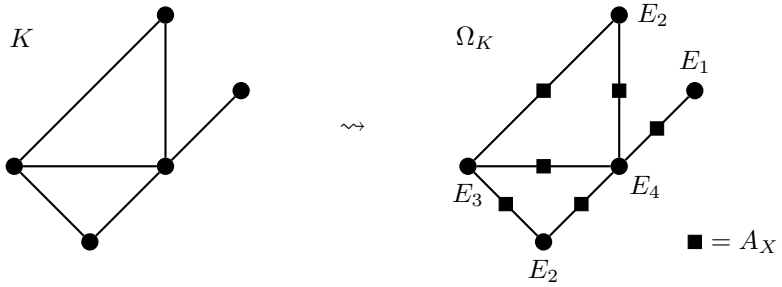


Figure 1 A graph K and the corresponding Holant$(A_X \mid \mathcal{EQ})$ signature grid Ω_K for computing the number of homomorphisms from K to X. Square vertices are assigned signature A_X.

Generalizing graph homomorphism to #CSP entails replacing A_X with an arbitrary set $\mathcal{F}$ of constraint functions, and replacing the degree-2 vertices assigned A_X with arbitrary-degree vertices assigned signatures from $\mathcal{F}$.

▶ **Definition 8** (Planar #CSP instance). *A #CSP instance K is* planar *if the underlying multigraph of the corresponding Holant signature grid Ω_K is planar.*

We now have the notation to state our main theorem.

▶ **Theorem 9** (Main result). *Let $\mathcal{F}$, $\mathcal{G}$ be compatible sets of constraint functions. Then $\mathcal{F} \cong_{qc} \mathcal{G}$ if and only if $Z(K) = Z(K_{\mathcal{F}\to\mathcal{G}})$ for every planar $\#CSP(\mathcal{F})$ instance K.*

If $\mathcal{F} = \{A_X\}$ and $\mathcal{G} = \{A_Y\}$, then Theorem 9 specializes to the result of [11]: graphs X and Y are quantum isomorphic iff they admit the same number of homomorphisms from every planar graph K.

▶ **Definition 10** (Gadget). *A* gadget *is a Holant signature grid equipped with an ordered set of dangling edges (edges with only one endpoint), defining external variables.*

Our gadgets will be in the context of Pl-Holant$(\mathcal{F} \mid \mathcal{EQ})$*, the Holant problem equivalent to $\#CSP(\mathcal{F})$. In this case, we specify that all dangling edges must be attached to equality vertices (vertices assigned signatures in $\mathcal{EQ}$).*

See Figures 2 and 3 for examples of gadgets in the context of Pl-Holant$(\mathcal{F} \mid \mathcal{EQ})$. We draw dangling edges lighter and thinner than internal edges.

▶ **Definition 11** ($M(\mathbf{K})$). *Let $\mathbf{K}$ be a gadget with n dangling edges and containing signatures of domain size q. For any $m, d \geq 0$, $m + d = n$, define $\mathbf{K}$'s (m, d)-*signature matrix *$M(\mathbf{K}) \in \mathbb{R}^{q^m \times q^d}$ by letting $M(\mathbf{K})_{\mathbf{x},\mathbf{y}}$ be the Holant value when the first m dangling edges (called* output *dangling edges) are assigned $x_1, \ldots, x_m$ and the last d dangling edges (called* input *dangling edges) are assigned $y_d, \ldots, y_1$. We draw the output/input dangling edges to the left/right of the gadget.*

▶ **Definition 12.** *Gadget $\mathbf{K}$ is* planar *if the underlying multigraph has an embedding with no edges (dangling or not) crossing, and the dangling edges are in cyclic order in the outer face.*

For a plane embedding of a planar gadget, we draw its output and input dangling edges on the left in order from top to bottom and on the right in order from bottom to top, respectively. For a gadget's signature matrix, observe that we consider the input dangling edges in reverse order, so from top to bottom. This definition preserves planarity of $\circ$:

▶ **Definition 13** (Gadget $\circ, \otimes, \dagger$). *For a gadget* $\mathbf{K}$ *with* $m+d$ *dangling edges, write* $\mathbf{K} \in \mathfrak{G}(m,d)$ *to mean we consider* $\mathbf{K}$ *with* m *output and* d *input dangling edges.*

- *For* $\mathbf{K} \in \mathfrak{G}(m,d), \mathbf{L} \in \mathfrak{G}(d,w)$*, define the composition* $\mathbf{K} \circ \mathbf{L} \in \mathfrak{G}(m,w)$ *by connecting each input dangling edge of* $\mathbf{K}$ *with the corresponding output dangling edge of* $\mathbf{L}$*. If* $\circ$ *creates adjacent vertices assigned* $E_a, E_b \in \mathcal{EQ}$*, we contract the edge between them, merging them into a single vertex assigned* E_{a+b-2}*. This does not change the Holant value.*
- *For* $\mathbf{K} \in \mathfrak{G}(m_1,d_1), \mathbf{L} \in \mathfrak{G}(m_2,d_2)$*, define the tensor product* $\mathbf{K} \otimes \mathbf{L} \in \mathfrak{G}(m_1+m_2, d_1+d_2)$ *by placing* $\mathbf{K}$ *above* $\mathbf{L}$*.*
- *For* $\mathbf{K} \in \mathfrak{G}(m,d)$*, define the (conjugate) transpose* $\mathbf{K}^\dagger \in \mathfrak{G}(d,m)$ *by reflecting* $\mathbf{K}$*'s underlying multigraph horizontally.*

See Figure 2. It is well known that applying the $\circ, \otimes, \dagger$ operations to gadgets corresponds to applying these operations to their signature matrices. See e.g. [3].

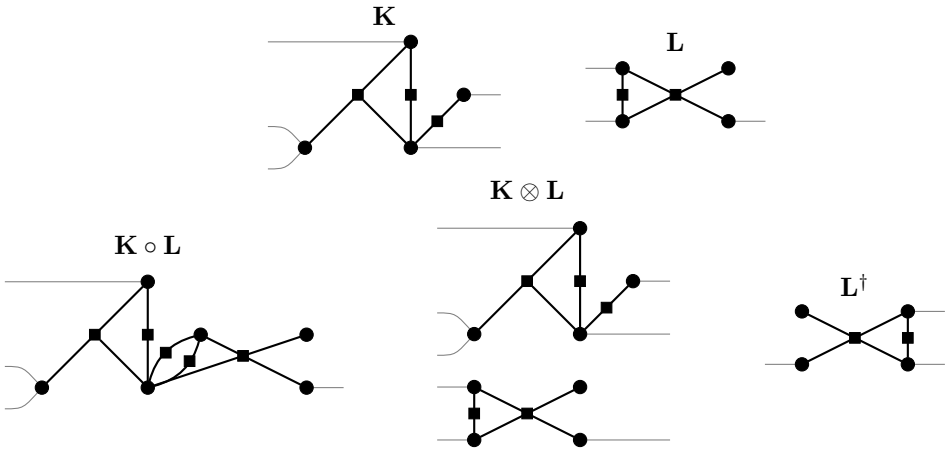


Figure 2 Operations on gadgets $\mathbf{K}$ and $\mathbf{L}$ in the context of Pl-Holant($\mathcal{F} \mid \mathcal{EQ}$).

3 Quantum Isomorphism Implies Planar #CSP Equivalence

3.1 The Planar Gadget Decomposition

Throughout, let $F \in \mathbb{R}^{[q]^n}$ denote a constraint function in $\mathcal{F}$, a set of constraint functions.

▶ **Definition 14** ($\mathcal{P}_\mathcal{F}$, $\mathcal{P}_\mathcal{F}(m,d)$). *Let* $\mathcal{P}_\mathcal{F}$ *be the collection of all planar gadgets in the context of* Holant($\mathcal{F} \mid \mathcal{EQ}$)*. Recall that all dangling edges of such a gadget are attached to vertices assigned signatures in* $\mathcal{EQ}$*.*

Let $\mathcal{P}_\mathcal{F}(m,d) \subseteq \mathcal{P}_\mathcal{F}$ *be the subset of gadgets with* m *output and* d *input dangling edges.*

The discussion after Definition 7 constructs a Pl-Holant($\mathcal{F} \mid \mathcal{EQ}$) instance modelling any given planar #CSP($\mathcal{F}$) instance. One can easily invert this construction to produce a planar #CSP($\mathcal{F}$) instance modelling any given Pl-Holant($\mathcal{F} \mid \mathcal{EQ}$) instance. Hence the signature grids underlying $\mathcal{P}_\mathcal{F}(m,d)$ are exactly the set of signature grids Ω_K corresponding to planar #CSP($\mathcal{F}$) instances.

We next introduce two families of fundamental gadgets in $\mathcal{P}_\mathcal{F}$. See Figure 3.

▶ **Definition 15** ($\mathbf{E}^{m,d}$, $\mathbf{F}^{m,d}$). *For $m, d \geq 0$, let $\mathbf{E}^{m,d}$ be the gadget consisting of a single vertex, assigned E_{m+d}, with m output and d input dangling edges.*

For $m, d \geq 0$ and $(m+d)$-ary signature function F, let $\mathbf{F}^{m,d}$ be the gadget consisting of a central degree-$(m+d)$ vertex assigned F, and m left and d right "arms", each with a vertex assigned E_2 with an output or input dangling edge, respectively. Define $\mathbf{I} = \mathbf{E}^{1,1}$.

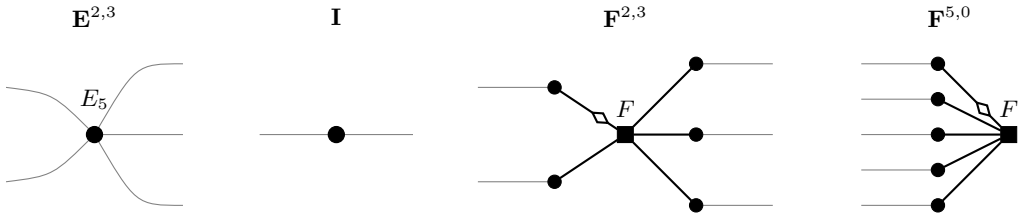


Figure 3 Examples of the fundamental gadgets $\mathbf{E}^{m,d}$ and $\mathbf{F}^{m,d}$. The diamond indicates the first input to asymmetric F.

Observe that $M(\mathbf{E}^{m,d}) = E^{m,d}$ and $M(\mathbf{F}^{m,d}) = F^{m,d}$; in particular $M(\mathbf{F}^{n,0}) = f$.

The next lemma addresses a key new issue raised by viewing our higher-arity constraint functions as explicit vertices in the signature grid. Section 4 requires all $\mathbf{F}$ gadgets to be in the form $\mathbf{F}^{m+d,0}$, but the decomposition procedure below produces gadgets $\mathbf{F}^{m,d}$ for arbitrary m and d (see Figure 5). Hence we must pivot $\mathbf{F}^{m,d}$'s dangling edges between input and output. $\mathbf{E}^{2,0}, \mathbf{E}^{0,2}, \mathbf{I} \in \langle \mathbf{E}^{1,0}, \mathbf{E}^{1,2} \rangle_{\circ,\otimes,\dagger}$ (the closure of $\{\mathbf{E}^{1,0}, \mathbf{E}^{1,2}\}$ under $\circ, \otimes, \dagger$), so we apply a procedure like the one in Figure 4.

▶ **Lemma 16.** *Let F be an n-ary constraint function. Then $\mathbf{F}^{m,d} \in \langle \mathbf{E}^{1,0}, \mathbf{E}^{1,2}, \mathbf{F}^{n,0} \rangle_{\circ,\otimes,\dagger}$ for all $m + d = n$.*

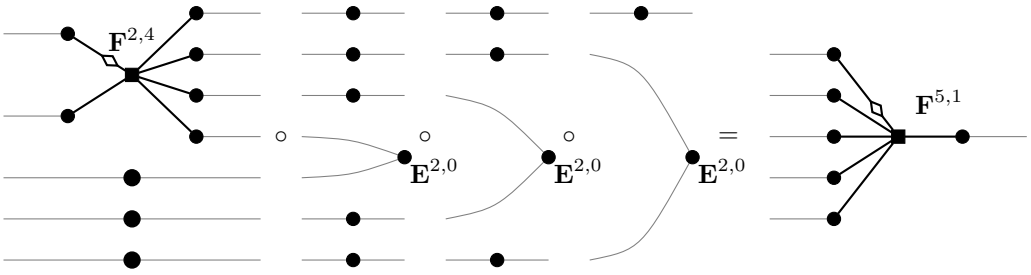


Figure 4 $(\mathbf{F}^{2,4} \otimes \mathbf{I}^{\otimes 3}) \circ (\mathbf{I}^{\otimes 3} \otimes \mathbf{E}^{2,0} \otimes \mathbf{I}^{\otimes 2}) \circ (\mathbf{I}^{\otimes 2} \otimes \mathbf{E}^{2,0} \otimes \mathbf{I}) \circ (\mathbf{I} \otimes \mathbf{E}^{2,0}) = \mathbf{F}^{5,1}$.

Next we show that any planar Holant($\mathcal{F} \mid \mathcal{EQ}$) gadget can be decomposed into the $\mathbf{F}^{\text{arity}(F),0}$ gadgets containing the signatures in $\mathcal{F}$, and two small equality gadgets.

▶ **Theorem 17.** $\mathcal{P}_{\mathcal{F}} = \langle \mathbf{E}^{1,0}, \mathbf{E}^{1,2}, \{\mathbf{F}^{\text{arity}(F),0} \mid F \in \mathcal{F}\} \rangle_{\circ,\otimes,\dagger}$.

The reverse inclusion follows from the fact that $\circ, \otimes, \dagger$ preserve planarity. The idea for the forward inclusion is to decompose an arbitrary $\mathbf{K} \in \mathcal{P}_{\mathcal{F}}$ into a composition of copies of $\mathbf{E}^{m,d}$ and appropriate $\mathbf{F}^{m,d}$, tensored with copies of $\mathbf{I}$. We use Lemma 16 to convert each $\mathbf{F}^{m,d}$ to $\mathbf{F}^{n,0}$. To extract an equality or constraint vertex, we apply one of the two extraction procedures shown in Figure 5, or their horizontal reflections. The extraction procedures guarantee that remaining gadget is still planar, bipartite, and has all dangling edges incident to equality vertices (i.e. is in $\mathcal{P}_{\mathcal{F}}$), so we apply induction.

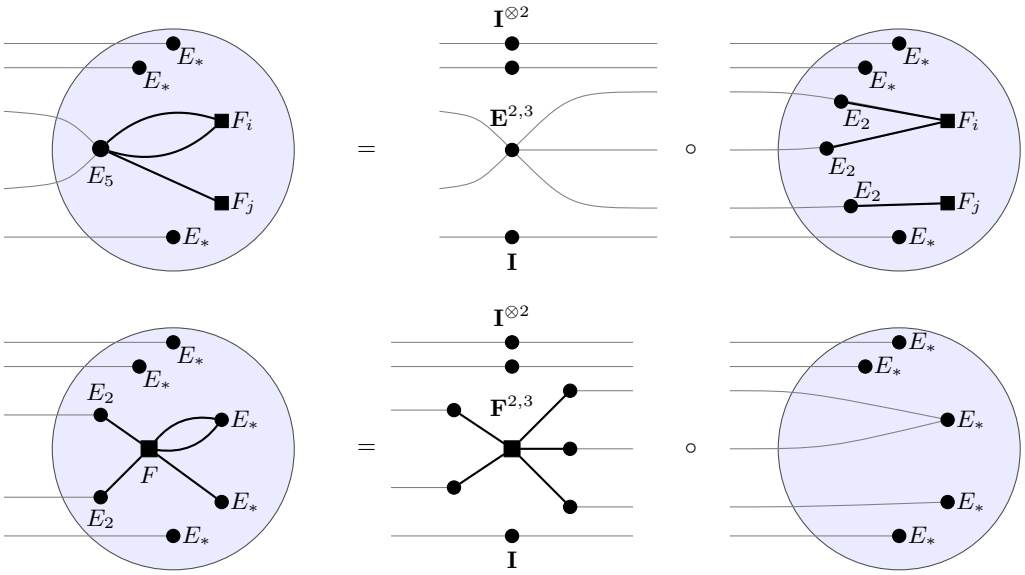


Figure 5 Extracting an equality (top) or constraint vertex from a planar Holant($\mathcal{F} \mid \mathcal{EQ}$) gadget.

3.2 Gadgets and quantum permutation matrices

The quantum Holant theorem, proved in Subsection 3.3, stems from viewing the quantum permutation matrix $\mathcal{U}$ itself as a signature in a Holant signature grid, indicated by a triangle vertex ▲. An immediate corollary of this theorem is one half of our main result Theorem 9: Planar #CSP instances with quantum isomorphic signature sets have the same partition function value. The proof of this result via the quantum Holant theorem is graphical and more intuitive than the proof in [11] of the graph homomorphism special case, and ties quantum isomorphism into the well-studied Holant framework. Furthermore, the graphical calculus of the quantum Holant theorem nicely highlights one of the key new difficulties of our generalization: unlike constraint functions derived from homomorphisms to undirected graphs (the case considered in [11]), general constraint functions F can be *asymmetric*: F's value is not necessarily preserved under permutation of its inputs. By planarity, permutations that cross F's input edges are not allowed, but the dihedral group actions – rotations and reflection – are allowed. Define the rotated constraint function $F^{(r)}$ for $r \in [\text{arity}(F)]$ by $F^{(r)}(x_1, \ldots, x_n) = F(x_{r+1}, \ldots, x_n, x_1, \ldots, x_r)$ and the reflected constraint function $F^\dagger$ by $F^\dagger(x_1, \ldots, x_n) = F(x_n, \ldots, x_1)$. Assuming F and G are quantum isomorphic, that is, $\mathcal{U}^{\otimes n} f = g$, it is not *a priori* obvious (due to noncommutativity), but true, that

$$\mathcal{U}^{\otimes n} f = g \iff \mathcal{U}^{\otimes n} f^{(r)} = g^{(r)} \text{ and } \mathcal{U}^{\otimes n} f = g \iff \mathcal{U}^{\otimes n} f^\dagger = g^\dagger. \tag{2}$$

The quantum Holant theorem uses $\mathcal{U}$ to transform each F in the Holant signature grid into G (via the assumption $\mathcal{U}^{\otimes n} f = g$), and since the signatures $F^{(r)}$ and $F^\dagger$ may appear in the signature grid in place of F, the theorem's success is dependent on the identities (2).

The bottom two panes of Figure 6 below illustrate a case of (2) for 5-ary f and g. The fact that Figure 6 uses $\mathbf{F}^{2,3}$ and $\mathbf{G}^{2,3}$ (with signature matrices $F^{2,3}$ and $G^{2,3}$) in place of $\mathbf{F}^{5,0}$ and $\mathbf{G}^{5,0}$ (with signature vectors f and g) is due to the additional useful identity

$$\mathcal{U}^{\otimes n} f = g \iff \mathcal{U}^{\otimes m} F^{m,d} (\mathcal{U}^{\otimes d})^\dagger = G^{m,d} \text{ for any } m + d = n, \tag{3}$$

illustrated in the upper right pane of Figure 6.

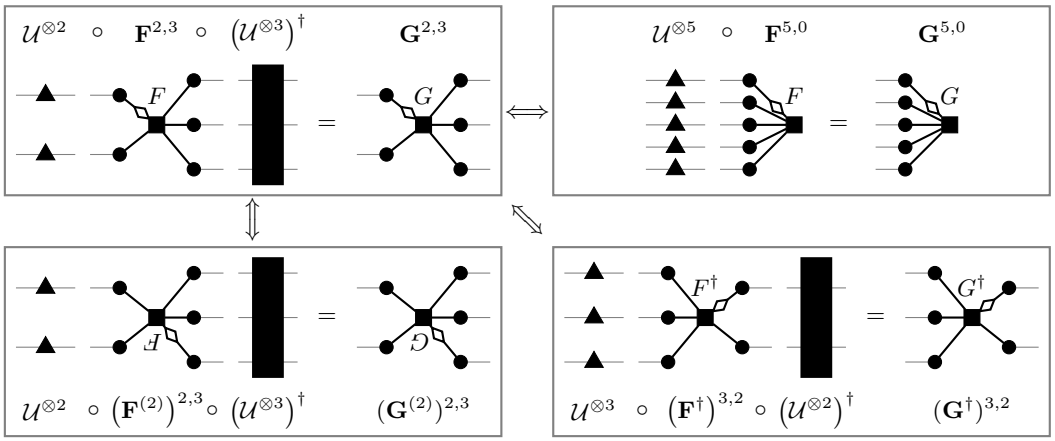


Figure 6 The "action" of $\mathcal{U}$ transforming F to G is preserved under (clockwise from top right) edge pivoting, signature reflection, and signature rotation. $(\mathcal{U}^{\otimes n})^\dagger$ is drawn as a black box, since, due to noncommutativity, $(\mathcal{U}^{\otimes n})^\dagger \neq (\mathcal{U}^\dagger)^{\otimes n}$ in general.

3.3 The Quantum Holant Theorem

The quantum Holant theorem gives a quantum version of *holographic transformation*. A holographic transformation [13] transforms one Holant signature grid Ω into another Ω' resulting in the same Holant value. For a set $\mathcal{F}$ of signatures and an invertible $T \in \mathbb{C}^{q \times q}$, write $T\mathcal{F} = \{T^{\otimes k} f \mid F \in \mathcal{F} \text{ has arity } k\}$. Define $\mathcal{F}T$ similarly. Valiant's Holant Theorem in [13] states that $\text{Holant}_\Omega(\mathcal{F} \mid \mathcal{G}) = \text{Holant}_{\Omega'}(T\mathcal{F} \mid \mathcal{G}T^{-1})$, for any Ω and sets of signatures $\mathcal{F}, \mathcal{G}$, where Ω' is constructed from Ω by replacing every signature in $\mathcal{F}$ or $\mathcal{G}$ with the corresponding transformed signature.

The Holant signature grids Ω_K and $\Omega_{K_{\mathcal{F}\to\mathcal{G}}}$ satisfying $Z(K) = \text{Pl-Holant}_{\Omega_K}(\mathcal{F} \mid \mathcal{EQ})$ and $Z(K_{\mathcal{F}\to\mathcal{G}}) = \text{Pl-Holant}_{\Omega_{K_{\mathcal{F}\to\mathcal{G}}}}(\mathcal{G} \mid \mathcal{EQ})$ are the same, up to replacement of every signature $F \in \mathcal{F}$ by the corresponding signature $G \in \mathcal{G}$. Assuming $\mathcal{F} \cong_{qc} \mathcal{G}$, there is a quantum permutation matrix $\mathcal{U}$ satisfying $\mathcal{U}^{\otimes n} f = g$ for every $F \in \mathcal{F}$ and corresponding $G \in \mathcal{G}$. This suggests we perform a *quantum* holographic transformation using $\mathcal{U}$. A calculation shows that $E^{0,n} = E^{0,n}\mathcal{U}^{\otimes n}$ for every n and any quantum permutation matrix $\mathcal{U}$. As $\mathcal{U}^{-1} = \mathcal{U}^\dagger$ is a quantum permutation matrix, $(\mathcal{EQ})\mathcal{U}^{-1} = \mathcal{EQ}$. Then the Holant theorem setting T to $\mathcal{U}$ seems to give

$$Z(K) = \text{Holant}_{\Omega_K}(\mathcal{F} \mid \mathcal{EQ}) = \text{Holant}_{\Omega_{K_{\mathcal{F}\to\mathcal{G}}}}(\mathcal{G} \mid \mathcal{EQ}) = Z(K_{\mathcal{F}\to\mathcal{G}}) \tag{4}$$

for any, not necessarily planar, #CSP($\mathcal{F}$) instance K. However, this cannot be true. If $\mathcal{F} = \{F\}$ and $\mathcal{G} = \{G\}$, where F and G are symmetric, binary, and 0-1 valued, (4) implies that the graphs with adjacency matrices F and G admit the same number of homomorphisms from any graph, giving $F \cong G$, a classical result of Lovász [8]. In other words, any quantum isomorphic graphs are classically isomorphic. But this is known to be false – see e.g. [1]. The main reason for this failure is the noncommutativity of $\mathcal{U}$'s entries.

When Ω_K is planar, however, this can be rescued by using the decomposition procedure for Ω_K in the proof of Theorem 17, which produces a sequence of gadgets whose signature matrices multiply to the Holant value. This defines an order of Ω_K's vertices, giving a globally consistent way in which $\mathcal{U}$'s entries are multiplied, for the partition function (sum of product expression). We will use $\mathcal{U}$ as a "quantum holographic transformation" by inserting $\mathcal{U}^{\otimes k}$ and its inverse $(\mathcal{U}^{\otimes k})^\dagger$ between every pair of these gadgets, converting every $F \in \mathcal{F}$ to the corresponding $G \in \mathcal{G}$ and preserving $\mathcal{EQ}$.

▶ **Theorem 18** (Quantum Holant Theorem). *Let $\mathcal{U}$ be a $q \times q$ quantum permutation matrix, and let $\mathcal{F}$ and $\mathcal{UF}$ be compatible sets of domain-q real-valued signatures. Then for every* Pl-Holant($\mathcal{F}$) *signature grid Ω,*

$$\text{Pl-Holant}_{\Omega}(\mathcal{F}) = \text{Pl-Holant}_{\Omega'}(\mathcal{UF}),$$

where Ω' is constructed from Ω by replacing every signature in $\mathcal{F}$ with the corresponding signature in $\mathcal{UF}$.

We omit its proof in this extended abstract. The main idea is to perform successive quantum holographic transformations using $\mathcal{U}$; the "pushing through" a set of $\mathcal{U}$'s as tensor powers is illustrated in Figure 7. Since the central gadget may be $\left(\mathbf{F}_i^{(r)}\right)^{m_i,d_i}$ or $\left(\mathbf{F}_i^{\dagger}\right)^{m_i,d_i}$ (depending on the orientation of F in the signature grid) rather than simply $\mathbf{F}^{m_i+d_i,0}$, the identities (2) and (3) illustrated in Figure 6 are necessary.

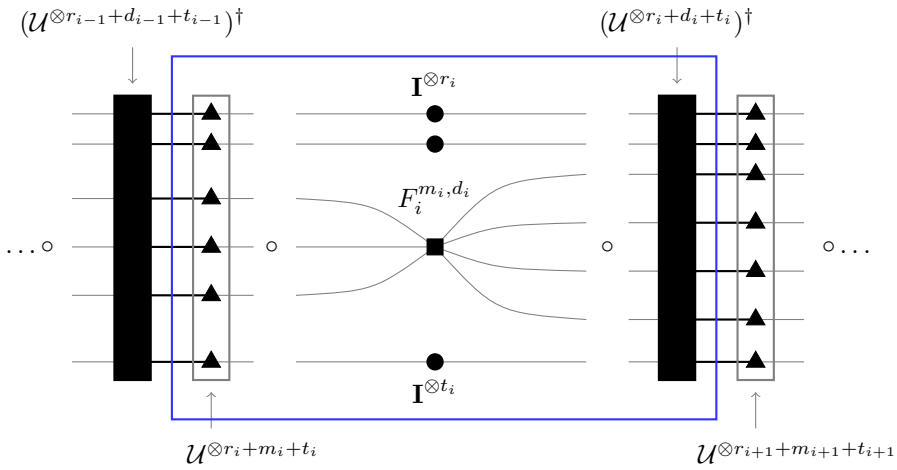


Figure 7 A visualization of a step in the proof of the quantum Holant theorem. We insert factors of the form $I = (\mathcal{U}^{\otimes k})^{\dagger}\mathcal{U}^{\otimes k}$ between every pair of gadgets $\mathbf{I}^{\otimes r_i} \otimes \mathbf{F}_i^{m_i,d_i} \otimes \mathbf{I}^{\otimes t_i}$, then apply gadget composition associativity. The resulting gadget in the blue box is $\mathbf{I}^{\otimes r_i} \otimes \mathbf{G}_i^{m_i,d_i} \otimes \mathbf{I}^{\otimes t_i}$.

Observe that, unlike other results in this work, the quantum Holant theorem is not only applicable to Pl-Holant($\mathcal{F} \mid \mathcal{EQ}$), but more generally to Pl-Holant($\mathcal{F}$) for *any* signature set $\mathcal{F}$. Since holographic transformations are an indespensable tool for the study of Holant problems, and planar Holant is the subject of much active research, the quantum Holant theorem should be of independent interest.

▶ **Corollary 19.** *Let $\mathcal{F}$ and $\mathcal{G}$ be compatible sets of constraint functions. If $\mathcal{F} \cong_{qc} \mathcal{G}$, then $Z(K) = Z(K_{\mathcal{F}\to\mathcal{G}})$ for every planar #CSP($\mathcal{F}$) instance K.*

4 Planar #CSP Equivalence Implies Quantum Isomorphism

The Quantum Automorphism Group

The most abstract construction in this work is the quantum automorphism group Qut($\mathcal{F}$) of a constraint function set $\mathcal{F}$. We omit the full formal definition here, as understanding it is not relevant to the rest of the work.

▶ **Definition 20** ($\mathrm{Qut}(\mathcal{F})$). *For a set $\mathcal{F}$ of constraint functions with $|V(\mathcal{F})| = q$, the* quantum automorphism group $\mathrm{Qut}(\mathcal{F})$ *of $\mathcal{F}$ is defined by the universal*[2] *C^*-algebra $C(\mathrm{Qut}(\mathcal{F}))$ generated by the entries of a $q \times q$ matrix $\mathcal{U} = (u_{ij})$ subject to the relations specifying that $\mathcal{U}$ is a quantum permutation matrix and $\mathcal{U}^{\otimes \mathrm{arity}(F)} f = f$ for every $F \in \mathcal{F}$.*

Observe that, just as $\mathcal{U}^{\otimes \mathrm{arity}(F)} f = g$ defines a quantum isomorphism between F and G, $\mathcal{U}^{\otimes \mathrm{arity}(F)} f = f$ defines quantum automorphisms of F. It is helpful, and sufficient for our purposes, to identify $\mathrm{Qut}(\mathcal{F})$ with $\mathcal{U}$ and $C(\mathrm{Qut}(\mathcal{F}))$, thought of as the algebra of continuous functionals on $\mathrm{Qut}(\mathcal{F})$. Indeed, if $\mathcal{U}$'s entries commute, then $C(\mathrm{Qut}(\mathcal{F}))$ concretizes as the algebra of continuous functionals on the classical automorphism group $\mathrm{Aut}(\mathcal{F})$. Thus the relationship between $\mathrm{Qut}(\mathcal{F})$ and $\mathrm{Aut}(\mathcal{F})$ is analogous to the relationship between quantum and classical permutation matrices: the former is an abstraction of the latter, sharing many familiar properties and constructions.

One such construction is the *orbits* of $\mathrm{Qut}(\mathcal{F})$ on the domain $V(\mathcal{F})$ [10]. If $\mathcal{U}$ is the quantum permutation matrix defining $\mathrm{Qut}(\mathcal{F})$, then $x, y \in V(\mathcal{F})$ are in the same orbit of $\mathrm{Qut}(\mathcal{F})$ if and only if $u_{xy} \neq 0$ (one can draw an analogy with the orbits of $\mathrm{Aut}(\mathcal{F})$ on $V(\mathcal{F})$ by viewing u_{xy} as corresponding to the automorphisms mapping x to y).

Another such construction is the *intertwiner space* of $\mathrm{Qut}(\mathcal{F})$.

▶ **Definition 21** ($C_{\mathrm{Qut}(\mathcal{F})}(m,d)$, $C_{\mathrm{Qut}(\mathcal{F})}$). $C_{\mathrm{Qut}(\mathcal{F})}(m,d) = \{M \in \mathbb{C}^{q^m \times q^d} \mid \mathcal{U}^{\otimes m} M = M \mathcal{U}^{\otimes d}\}$ *is the space of* (m,d)-intertwiners *of* $\mathrm{Qut}(\mathcal{F})$, *and* $C_{\mathrm{Qut}(\mathcal{F})} = \bigcup_{m,d} C_{\mathrm{Qut}(\mathcal{F})}(m,d)$.

Observe that the equation $\mathcal{U}^{\otimes m} M = M \mathcal{U}^{\otimes d}$ resembles, after multiplying by $(\mathcal{U}^{\otimes d})^{-1} = (\mathcal{U}^{\otimes d})^\dagger$, the equation $\mathcal{U}^{\otimes m} F^{m,d} \left(\mathcal{U}^{\otimes d}\right)^\dagger = G^{m,d}$ characterizing quantum isomorphism of F and G (recall (3)). Hence it is reasonable to assume that $C_{\mathrm{Qut}(\mathcal{F})}$ consists of gadget signature matrices. The next lemma, proved in the full version using techniques from quantum group theory, is a step towards this conclusion.

▶ **Lemma 22.** $C_{\mathrm{Qut}(\mathcal{F})} = \langle E^{1,0}, E^{1,2}, \{f \mid F \in \mathcal{F}\} \rangle_{+,\circ,\otimes,\dagger}$.

Note that the RHS of Lemma 22 is the linear span of the signature matrices of the gadgets in the RHS of Theorem 17. This observation motivates the following definition.

▶ **Definition 23** ($\mathfrak{Q}^{\mathcal{P}}_{\mathcal{F}}(m,d)$). *A* planar (m,d)-quantum $\mathcal{F}$-gadget *is a finite linear combination of gadgets in $\mathcal{P}_{\mathcal{F}}(m,d)$ with coefficients in $\mathbb{C}$. Let $\mathfrak{Q}^{\mathcal{P}}_{\mathcal{F}}(m,d)$ be the collection of all planar (m,d)-quantum $\mathcal{F}$-gadgets. Extend the signature matrix function M linearly to $\mathfrak{Q}^{\mathcal{P}}_{\mathcal{F}}(m,d)$.*

Applying M to quantum $\mathcal{F}$-gadgets composed of gadgets on the RHS of Theorem 17 yields the RHS of Lemma 22, so we have the following key connection between the planar gadget decomposition and the intertwiners of $\mathrm{Qut}(\mathcal{F})$.

▶ **Theorem 24.** $C_{\mathrm{Qut}(\mathcal{F})}(m,d) = \{M(\mathbf{Q}) \mid \mathbf{Q} \in \mathfrak{Q}^{\mathcal{P}}_{\mathcal{F}}(m,d)\}$ *for every $m, d \in \mathbb{N}$.*

Say a #CSP instance $K = (V, C)$ is 1-*labeled* if there is a distinguished *labeled* variable $v \in V$. For $x \in V(\mathcal{F})$, let Z^x be the partition function on 1-labeled instances, defined identically to Z, except only summing over those $\sigma : V \to V(\mathcal{F})$ such that $\sigma(v) = x$. A 1-labeled #CSP($\mathcal{F}$) instance K with labeled variable v corresponds to a Holant($\mathcal{F} \mid \mathcal{EQ}$) gadget $\mathbf{K}$ with a single dangling edge incident to the equality vertex constructed from v. For $x \in V(\mathcal{F})$, $M(\mathbf{K})_x = Z^x(K)$. Then the connection between intertwiners and gadget signature matrices established in Theorem 24 yields the following lemma, a quantum analogue of several similar classical results for graph homomorphism, including [9, Lemma 2.4].

▶ **Lemma 25.** *$x, y \in V(\mathcal{F})$ are in the same orbit of $\mathrm{Qut}(\mathcal{F})$ if and only if $Z^x(K) = Z^y(K)$ for all 1-labeled planar #CSP($\mathcal{F}$) instances K.*

[2] A "universal" C^*-algebra is roughly analogous to the generator-and-relation presentation of a group.

Arity Reduction and Projective Connectivity

For n-ary constraint functions, the remaining results require n-dimensional generalizations of orbits. However, such higher-dimensional orbits are not known to exist for $n > 2$. This is another new difficulty posed by our extension of binary graph homomorphism to higher-arity functions. We overcome it by the following arity-reduction technique.

▶ **Lemma 26.** *Let F be an n-ary constraint function with $n > 2$ and let $\mathcal{U}$ define* $\mathrm{Qut}(\{F\})$*, so $\mathcal{U}^{\otimes n} f = f$. Define an arity-$(n-1)$ constraint function F' by $F'_{x_2,\ldots,x_n} = \sum_{x_1} F_{x_1,x_2,\ldots,x_n}$. Then $\mathcal{U}^{\otimes n-1} f' = f'$ (where f' is the vectorization of F').*

After $n-2$ applications of Lemma 26, the resulting binary constraint function is the adjacency matrix of a $\mathbb{R}$-weighted graph. We would like this $\mathbb{R}$-weighted graph to be *connected*, meaning the transitive closure of the relation $\sim$ on $V(X)$ defined by $x \sim y \iff X_{xy} \neq 0$ has only a single equivalence class. We define connnectivity for higher-arity tensors motivated by Lemma 26. For $n \geq 2$, an n-ary constraint function F is *projectively connected* if the $\mathbb{R}$-weighted graph X defined by $X_{uv} = \sum_{x_1,\ldots,x_{n-2}} F_{x_1,\ldots,x_{n-2},u,v}$ is connected.

▶ **Definition 27** ($\oplus$)**.** *Let $F \in \mathbb{R}^{V(F)^n}, G \in \mathbb{R}^{V(G)^n}$ be constraint functions of the same arity n. The* direct sum *$F \oplus G \in \mathbb{R}^{(V(F) \sqcup V(G))^n}$ of F and G is defined for all $\mathbf{x} \in (V(F) \sqcup V(G))^n$ by setting $(F \oplus G)_{\mathbf{x}}$ to be $F_{\mathbf{x}}$ or $G_{\mathbf{x}}$ if $\mathbf{x} \in V(F)^n$ or $\mathbf{x} \in V(G)^n$, respectively, and 0 otherwise. For constraint function sets $\mathcal{F}$ and $\mathcal{G}$ of size s, define $\mathcal{F} \oplus \mathcal{G} = \{F_i \oplus G_i \mid i \in [s]\}$.*

For $n = 2$ and 0-1 valued F and G, the direct sum is the disjoint union of the graphs whose adjacency matrices are F and G. Projective connectivity is desirable due to the following lemma, an extension of [10, Theorem 4.5] to higher arity, and an analogue of the fact that, for connected graphs X and Y, if there exist vertices $x \in V(X)$ and $y \in V(Y)$ in the same orbit of the automorphism group of the disjoint union of X and Y, then $X \cong Y$.

▶ **Lemma 28.** *Let $\mathcal{F}$ and $\mathcal{G}$ be constraint function sets with domains $V(\mathcal{F})$ and $V(\mathcal{G})$, respectively, and further assume that $\mathcal{F}$ and $\mathcal{G}$ contain a pair of corresponding projectively connected constraint functions. If there are some $\hat{x} \in V(\mathcal{F})$, $\hat{y} \in V(\mathcal{G})$ in the same orbit of* $\mathrm{Qut}(\mathcal{F} \oplus \mathcal{G})$*, then $\mathcal{F} \cong_{qc} \mathcal{G}$.*

The proof of Lemma 28 proceeds roughly as follows. Let $\mathcal{U}$ define $\mathrm{Qut}(\mathcal{F} \oplus \mathcal{G})$, and let F and G be the corresponding projectively connected constraint functions in $\mathcal{F}$ and $\mathcal{G}$, respectively. Summing out all but two indices of $F \oplus G$ (as in Lemma 26), we obtain a $\mathbb{R}$-weighted graph Z whose subgraphs induced by $V(F)$ and $V(G)$ are connected, and such that $\mathcal{U}^{\otimes 2} z = z$. Then, following the proof of [10, Theorem 4.5], we extract from Z enough information about $\mathcal{U}$ to show that its quarter submatrix $(u_{xy})_{x \in V(X), y \in V(Y)}$ is itself a quantum permutation matrix, defining a quantum isomorphism between $\mathcal{F}$ and $\mathcal{G}$.

Finally, we come to the proof of the reverse implication of Theorem 9. Lemma 28 assumes $\mathcal{F}$ contains a projectively constraint function, which in general is not true. For (unweighted) graphs, one can take the complement to assume the graphs are connected, but this trick does not apply to our $\mathbb{R}$-weighted constraint functions. Instead, we adapt an idea from Lovász's proof of [9, Corollary 2.6], which is roughly the *classical* case of Theorem 9 restricted to positive-real-weighted graphs. The idea is to add a new vertex to each graph, each adjacent to all other vertices by edges of the same nonzero weight. The new vertices are the targets of a result analogous to Lemma 25, each graph is now connected, and by symmetry of the new vertices, their addition preserves isomorphism. Somewhat remarkably, the same idea applies to quantum isomorphism of higher-arity constraint functions.

▶ **Lemma 29.** *Let $\mathcal{F}$ and $\mathcal{G}$ be compatible sets of constraint functions. If $Z(K) = Z(K_{\mathcal{F}\to\mathcal{G}})$ for every planar #CSP($\mathcal{F}$) instance K, then $\mathcal{F} \cong_{qc} \mathcal{G}$.*

We now present a proof sketch of Lemma 29. See the full version for a full proof. Let 0_F and 0_G be new domain elements. For each $F \in \mathcal{F}$, $G \in \mathcal{G}$, define constraint functions F' and G' on $V(\mathcal{F}) \sqcup \{0_F\}$ and $V(\mathcal{G}) \sqcup \{0_G\}$, by letting

$$F'_{\mathbf{x}} = \begin{cases} F_{\mathbf{x}} & \mathbf{x} \in V(\mathcal{F})^n \\ \gamma & \mathbf{x} = (0_F, 0_F, \ldots, 0_F, c),\ c \neq 0_F \\ 0 & \text{otherwise} \end{cases};\quad G'_{\mathbf{x}} = \begin{cases} G_{\mathbf{x}} & \mathbf{x} \in V(\mathcal{G})^n \\ \gamma & \mathbf{x} = (0_G, 0_G, \ldots, 0_G, c),\ c \neq 0_G \\ 0 & \text{otherwise} \end{cases}$$

for some fixed $\gamma \in \mathbb{R} \setminus \{0\}$. Let $\mathcal{F}' = \{F' \mid F \in \mathcal{F}\}$, $\mathcal{G}' = \{G' \mid G \in \mathcal{G}\}$, with $V(\mathcal{F}') = V(\mathcal{F}) \sqcup \{0_F\}$ and $V(\mathcal{G}') = V(\mathcal{G}) \sqcup \{0_G\}$.

F' and G' are designed to simultaneously satisfy three properties. First, defining $\mathbb{R}$-weighted graphs X' and Y' from F' and G' by summing the first $n-2$ indices as in Lemma 26, we have $X'_{0_F v} = \gamma \neq 0$ for every $v \in V(X') \setminus \{0_F\}$, and similarly for Y'. Thus X' and Y' are connected, so F' and G' are projectively connected.

Second, we wish to obtain $Z^{0_F}(K) = Z^{0_G}(K)$ for every planar 1-labeled #CSP($\mathcal{F}' \oplus \mathcal{G}'$) instance $K = (V, C)$. Then Lemma 25 asserts that 0_F and 0_G are in the same orbit of $\mathrm{Qut}(\mathcal{F}' \oplus \mathcal{G}')$, Hence, since F' and G' are projectively connected, $0_F \in V(\mathcal{F}')$, and $0_G \in V(\mathcal{G}')$, Lemma 28 gives $\mathcal{F}' \cong_{qc} \mathcal{G}'$. Let $K = (V, C)$. To obtain $Z^{0_F}(K) = Z^{0_G}(K)$, consider the following. Let v_0 be the labeled variable in V. When v_0 takes value $0_F \in V(\mathcal{F}')$, all variables must take values in $V(\mathcal{F}')$ (otherwise the assignment contributes 0 to the partition function, as $F' \oplus G'$ takes value 0 unless its inputs are all in $V(\mathcal{F}')$ or all in $V(\mathcal{G}')$). Furthermore, by construction of $\mathcal{F}'$, any constraint function applied to variable v_0 evaluates to 0 unless most of its other arguments also take value 0_F. This fixes more variables to 0_F, and the effect cascades to other constraint functions applied to those variables and so on. Any nonzero constraint with a variable fixed to 0_F always evaluates to γ. Let D be the set of such constraints. Remaining constraints in $C \setminus D$ apply some F' to inputs in only $V(\mathcal{F})$, so F', by construction, acts as the original F. Therefore, the sub-instance of K induced by constraints $C \setminus D$ is effectively a planar #CSP($\mathcal{F}$) instance, so $Z^{0_F}(K)$ is expressible as $\gamma^{|D|}$ times the sum of $Z(K')$ for various planar #CSP($\mathcal{F}$) instances K'. Similarly, by the symmetry of $\mathcal{F}'$ and $\mathcal{G}'$, $Z^{0_G}(K)$ is expressible as $\gamma^{|D|}$ times the sum of $Z(K'')$ for matching planar #CSP($\mathcal{G}$) instances K'', and by assumption each $Z(K') = Z(K'')$. Hence $Z^{0_F}(K) = Z^{0_G}(K)$.

Third, upon obtaining the quantum isomorphism $\mathcal{U}$ between $\mathcal{F}'$ and $\mathcal{G}'$, we must recover a quantum isomorphism between the original $\mathcal{F}$ and $\mathcal{G}$. Define the matrix $\widehat{\mathcal{U}} = (u_{uv})_{u \in V(\mathcal{G}), v \in V(\mathcal{F})}$ (in other words, we eliminate from $\mathcal{U}$ the row and column corresponding to the new vertices 0_G and 0_F, respectively). $\mathcal{F}'$ and $\mathcal{G}'$ were constructed so that, if γ is chosen to be sufficiently large, then $\widehat{\mathcal{U}}$ is a quantum permutation matrix, and furthermore defines a quantum isomorphism between $\mathcal{F}$ and $\mathcal{G}$.

References

1 Albert Atserias, Laura Mančinska, David E. Roberson, Robert Šámal, Simone Severini, and Antonios Varvitsiotis. Quantum and non-signalling graph isomorphisms. *Journal of Combinatorial Theory, Series B*, 136:289–328, 2019.

2 Andrei A. Bulatov. The complexity of the counting constraint satisfaction problem. *J. ACM*, 60(5), October 2013. doi:10.1145/2528400.

3 Jin-Yi Cai and Xi Chen. *Complexity Dichotomies for Counting Problems*, volume 1. Cambridge University Press, 2017. doi:10.1017/9781107477063.002.

4 Jin-Yi Cai and Xi Chen. Complexity of counting csp with complex weights. *J. ACM*, 64(3), June 2017. doi:10.1145/2822891.

5 Jin-Yi Cai, Xi Chen, and Pinyan Lu. Nonnegative weighted #csp: An effective complexity dichotomy. *SIAM J. Comput.*, 45(6):2177–2198, 2016. doi:10.1137/15M1032314.

6 Martin Dyer and David Richerby. An effective dichotomy for the counting constraint satisfaction problem. *SIAM Journal on Computing*, 42(3):1245–1274, 2013. doi:10.1137/100811258.

7 Heng Guo and Tyson Williams. The complexity of planar boolean #csp with complex weights. *Journal of Computer and System Sciences*, 107:1–27, 2020.

8 László Lovász. Operations with structures. *Acta Mathematica Hungarica*, 18(3-4):321–328, 1967.

9 László Lovász. The rank of connection matrices and the dimension of graph algebras. *European Journal of Combinatorics*, 27(6):962–970, 2006.

10 Martino Lupini, Laura Mančinska, and David Roberson. Nonlocal games and quantum permutation groups. *Journal of Functional Analysis*, 279, December 2017. doi:10.1016/j.jfa.2020.108592.

11 Laura Mančinska and David E. Roberson. Quantum isomorphism is equivalent to equality of homomorphism counts from planar graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 661–672, 2020. doi:10.1109/FOCS46700.2020.00067.

12 David Roberson. Private communication, 2023.

13 Leslie G. Valiant. Holographic algorithms. *SIAM Journal on Computing*, 37(5):1565–1594, 2008.

14 Shuzhou Wang. Quantum Symmetry Groups of Finite Spaces. *Communications in Mathematical Physics*, 195(1):195–211, July 1998. doi:10.1007/s002200050385.

15 S. L. Woronowicz. Compact matrix pseudogroups. *Communications in Mathematical Physics*, 111(4):613–665, December 1987. doi:10.1007/BF01219077.

A Appendix: An alternate approach to connectivity

The proof of Lemma 28 makes use of the 2-dimensional orbits, or *orbitals*, of $\mathrm{Qut}(\mathcal{F} \oplus \mathcal{G})$. This construction, as mentioned earlier, does not extend to dimensions higher than 2. This is why we, via projective connectivity, effectively require that $\mathcal{F}$ and $\mathcal{G}$ contain a *binary* connected constraint function in the hypothesis of Lemma 28. To satisfy this hypothesis, we ensure that the modified constraint functions F' and G' in the proof of Lemma 29 are projectively connected. In this appendix, we present a different construction, due to Roberson [12], which, rather than modify the existing constraint functions, adds new binary connected constraint functions to $\mathcal{F}$ and $\mathcal{G}$, while preserving quantum isomorphism. This removes the need for projective connectivity entirely, simplifies the proof of Lemma 28, and could simplify the proof of Lemma 29, since it is no longer necessary that F' and G' be projectively connected (though we still need $Z^{\mathbf{0}_F}(K) = Z^{\mathbf{0}_G}(K)$ for all planar 1-labeled K). Additionally, the alternate construction makes use of two lemmas which should be of independent interest.

First, we extend Definition 3 to gadgets.

▶ **Definition 30** ($\mathbf{K}_{\mathcal{F}\to\mathcal{G}}$). *For compatible constraint function sets $\mathcal{F}$ and $\mathcal{G}$ and $\mathbf{K} \in \mathcal{P}_{\mathcal{F}}$, let $\mathbf{K}_{\mathcal{F}\to\mathcal{G}} \in \mathcal{P}_{\mathcal{G}}$ be the* corresponding *gadget formed by replacing each constraint signature $F_i \in \mathcal{F}$ with the corresponding $G_i \in \mathcal{G}$. Extend this mapping linearly to $\mathfrak{Q}^{\mathcal{P}}_{\mathcal{F}}$.*

The first lemma shows that, viewing intertwiners themselves as constraint functions, we may add "corresponding" pairs of intertwiners (the signature matrices of corresponding quantum $\mathcal{F}$ and $\mathcal{G}$-gadgets – recall Theorem 24) to $\mathcal{F}$ and $\mathcal{G}$, while preserving equivalence on planar #CSP instances.

▶ **Lemma 31** ([12]). *Let $\mathcal{F}$ and $\mathcal{G}$ be compatible constraint function sets. Let $M_F \in C_{\mathrm{Qut}(\mathcal{F})}(m,d)$ and $M_G \in C_{\mathrm{Qut}(\mathcal{G})}(m,d)$ such that $M_F = M(\mathbf{Q})$ and $M_G = M(\mathbf{Q}_{\mathcal{F}\to\mathcal{G}})$ for some quantum $\mathcal{F}$-gadget $\mathbf{Q} \in \mathfrak{Q}^{\mathcal{P}}_{\mathcal{F}}(m,d)$. Let F and G be the constraint functions satisfying $F^{m,d} = M_F$ and $G^{m,d} = M_G$, and let $\mathcal{F}' = \mathcal{F} \cup \{F\}$ and $\mathcal{G}' = \mathcal{G} \cup \{G\}$. Then $Z(K) = Z(K_{\mathcal{F}\to\mathcal{G}})$ for all planar #CSP($\mathcal{F}$) instances K if and only if $Z(K) = Z(K_{\mathcal{F}'\to\mathcal{G}'})$ for all planar #CSP($\mathcal{F}'$) instances K.*

Proof. The backward direction is immediate. Let K be a planar #CSP($\mathcal{F}'$) instance, and let Ω_K be the corresponding Pl-Holant($\mathcal{F}' \mid \mathcal{EQ}$) instance. Create a "quantum signature grid" $\widehat{\Omega}_K \in \mathfrak{Q}^{\mathcal{P}}_{\mathcal{F}}(0,0)$ by replacing every instance of a vertex v assigned F in Ω_K by the equivalent quantum gadget $\mathbf{Q} \in \mathfrak{Q}^{\mathcal{P}}_{\mathcal{F}}$, matching the cyclically-ordered dangling edges of each gadget to the cyclically-ordered incident edges of v (and contracting the edges between the adjacent equality vertices to preserve bipartiteness). Similarly create $\widehat{\Omega}_{K_{\mathcal{F}'\to\mathcal{G}'}} \in \mathfrak{Q}^{\mathcal{P}}_{\mathcal{G}}(0,0)$ by replacing every instance of a vertex assigned G in $\Omega_{K_{\mathcal{F}'\to\mathcal{G}'}}$ with $\mathbf{Q}_{\mathcal{F}\to\mathcal{G}} \in \mathfrak{Q}^{\mathcal{P}}_{\mathcal{G}}$. Then the index-$\alpha$ summand of $\widehat{\Omega}_K$ or $\widehat{\Omega}_{K_{\mathcal{F}'\to\mathcal{G}'}}$ is a planar #CSP($\mathcal{F}$) instance $K^{\alpha}_{\mathcal{F}}$ or planar #CSP($\mathcal{G}$) instance $K^{\alpha}_{\mathcal{G}}$, respectively, and furthermore $K^{\alpha}_{\mathcal{G}} = (K^{\alpha}_{\mathcal{F}})_{\mathcal{F}\to\mathcal{G}}$. Thus

$$\begin{aligned} Z(K) &= \text{Pl-Holant}_{\Omega_K}(\mathcal{F}' \mid \mathcal{EQ}) \\ &= \text{Pl-Holant}_{\widehat{\Omega}_K}(\mathcal{F} \mid \mathcal{EQ}) \\ &= \text{Pl-Holant}_{\widehat{\Omega}_{K_{\mathcal{F}'\to\mathcal{G}'}}}(\mathcal{G} \mid \mathcal{EQ}) \\ &= \text{Pl-Holant}_{\Omega_{K_{\mathcal{F}'\to\mathcal{G}'}}}(\mathcal{G}' \mid \mathcal{EQ}) \\ &= Z(K_{\mathcal{F}'\to\mathcal{G}'}). \end{aligned}$$

◀

An alternate proof would also need the following lemma, which is equivalent to Lemma 31 once Theorem 9 is proved.

▶ **Lemma 32** ([12]). *Suppose $\mathcal{F}$, $\mathcal{G}$, F, G, $\mathcal{F}'$, and $\mathcal{G}'$ satisfy the hypotheses of Lemma 31. Then $\mathcal{F} \cong_{qc} \mathcal{G} \iff \mathcal{F}' \cong_{qc} \mathcal{G}'$.*

Proof. The backward direction is immediate. Let $\mathcal{U}$ be the quantum permutation matrix defining the quantum isomorphism between $\mathcal{F}$ and $\mathcal{G}$. It suffices to show $\mathcal{U}^{\otimes m+d} f = g$, or equivalently, by (3), $\mathcal{U}^{\otimes m} M_F (\mathcal{U}^{\otimes d})^{\dagger} = M_G$. This identity follows from the proof of the quantum Holant theorem. Indeed, while the statement in Theorem 18 only applies to signature grids (gadgets in $\mathfrak{Q}^{\mathcal{P}}_{\mathcal{F}}(0,0)$), the proof may be easily modified to apply to $\mathbf{Q} \in \mathfrak{Q}^{\mathcal{P}}_{\mathcal{F}}(m,d)$ as follows. After inserting $(\mathcal{U}^{\otimes k})^{\dagger}\mathcal{U}^{\otimes k}$ between each pair of gadgets in the Theorem 17 decomposition of (each summand of) $\mathbf{Q}$ and reassociating to convert every $\mathcal{F}$ signature to the corresponding $\mathcal{G}$ signature and fix the internal equality vertices, we must effectively apply an additional $\mathcal{U}$ or $\mathcal{U}^{\dagger}$ to each original dangling edge of $\mathbf{Q}$ to fully transform each equality vertex with a dangling edge back to itself via $\mathcal{U}^{\otimes a} E^{a,b} (\mathcal{U}^{\otimes b})^{\dagger} = E^{a,b}$. Thus $\mathcal{U}^{\otimes m}\, \mathbf{Q} (\mathcal{U}^{\otimes d})^{\dagger} = \mathbf{Q}_{\mathcal{F}\to\mathcal{G}}$, and applying M to both sides gives the result. ◀

Together, Lemmas 31 and 32 show that, to prove Theorem 9, we may assume that $\mathcal{F}$ and $\mathcal{G}$ contain (the constraint functions created from) any intertwiners M_F and M_G which are the signature matrices of corresponding quantum $\mathcal{F}$ and $\mathcal{G}$-gadgets. In particular, we trivially have $(\mathbf{E}^{1,0} \circ \mathbf{E}^{0,1})_{\mathcal{F}\to\mathcal{G}} = (\mathbf{E}^{1,0} \circ \mathbf{E}^{0,1})$, so we may assume $\mathcal{F}$ and $\mathcal{G}$ both contain $M(\mathbf{E}^{1,0} \circ \mathbf{E}^{0,1}) = J$, the all-1s matrix. J is a connected constraint function, so we may immediately apply Lemma 28. Moreover, since J is already binary, we don't have to worry about reducing arity in the proof of Lemma 28.

On the Fine-Grained Complexity of Small-Size Geometric Set Cover and Discrete k-Center for Small k

Timothy M. Chan
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA

Qizheng He
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA

Yuancheng Yu
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA

Abstract

We study the time complexity of the discrete k-center problem and related (exact) geometric set cover problems when k or the size of the cover is small. We obtain a plethora of new results:

- We give the first subquadratic algorithm for *rectilinear discrete 3-center* in 2D, running in $\widetilde{O}(n^{3/2})$ time.
- We prove a lower bound of $\Omega(n^{4/3-\delta})$ for rectilinear discrete 3-center in 4D, for any constant $\delta > 0$, under a standard hypothesis about triangle detection in sparse graphs.
- Given n points and n *weighted* axis-aligned unit squares in 2D, we give the first subquadratic algorithm for finding a minimum-weight cover of the points by 3 unit squares, running in $\widetilde{O}(n^{8/5})$ time. We also prove a lower bound of $\Omega(n^{3/2-\delta})$ for the same problem in 2D, under the well-known APSP Hypothesis. For arbitrary axis-aligned rectangles in 2D, our upper bound is $\widetilde{O}(n^{7/4})$.
- We prove a lower bound of $\Omega(n^{2-\delta})$ for Euclidean discrete 2-center in 13D, under the Hyperclique Hypothesis. This lower bound nearly matches the straightforward upper bound of $\widetilde{O}(n^{\omega})$, if the matrix multiplication exponent ω is equal to 2.
- We similarly prove an $\Omega(n^{k-\delta})$ lower bound for Euclidean discrete k-center in $O(k)$ dimensions for any constant $k \geq 3$, under the Hyperclique Hypothesis. This lower bound again nearly matches known upper bounds if $\omega = 2$.
- We also prove an $\Omega(n^{2-\delta})$ lower bound for the problem of finding 2 boxes to cover the largest number of points, given n points and n boxes in 12D. This matches the straightforward near-quadratic upper bound.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Geometric set cover, discrete k-center, conditional lower bounds

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.34

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2305.01892 [19]

Funding *Timothy M. Chan*: Work supported by NSF Grant CCF-2224271.

Acknowledgements We thank Yinzhan Xu for helpful discussions.

1 Introduction

1.1 The discrete k-center problem for small k

The *Euclidean k-center* problem is well-known in computational geometry and has a long history: given a set P of n points in $\mathbb{R}^d$ and a number k, we want to find k congruent balls covering S, while minimizing the radius. Euclidean 1-center can be solved in linear time for any constant dimension d by standard techniques for low-dimensional linear programming or LP-type problems [21, 24, 27, 41, 50]. In a celebrated paper from SoCG'96, Sharir [44] gave the first $\widetilde{O}(n)$-time[1] algorithm for Euclidean 2-center in $\mathbb{R}^2$, which represented a significant improvement over previous near-quadratic algorithms (the hidden logarithmic factors have since been reduced in a series of subsequent works [29, 16, 49, 23]). The problem is more difficult in higher dimensions: the best time bound for Euclidean 2-center in $\mathbb{R}^d$ is about n^d (see [3, 2] for some results on the $\mathbb{R}^3$ case), and Cabello et al. [14] proved a conditional lower bound, ruling out $n^{o(d)}$-time algorithms, assuming the Exponential Time Hypothesis (ETH). We are not aware of any work specifically addressing the Euclidean 3-center problem.

The k-center problem has also been studied under different metrics. The most popular version after Euclidean is L_∞ or *rectilinear k-center*: here, we want to find k congruent hypercubes covering P, while minimizing the side length of the hypercubes.[2] As expected, the rectilinear version is a little easier than the Euclidean. Sharir and Welzl in SoCG'96 [45] showed that rectilinear 3-center problem in $\mathbb{R}^2$ can be solved in linear time, and that rectilinear 4-center and 5-center in $\mathbb{R}^2$ can be solved in $\widetilde{O}(n)$ time (the logarithmic factors have been subsequently improved by Nussbaum [42]). Katz and Nielsen's work in SoCG'96 [35] implied near-linear-time algorithms for rectilinear 2-center in any constant dimension d, while Cabello et al. in SODA'08 [14] gave an $O(n \log n)$-time algorithm for rectilinear 3-center in any constant dimension d. Cabello et al. also proved a conditional lower bound for rectilinear 4-center, ruling out $n^{o(\sqrt{d})}$-time algorithms under ETH.

In this paper, we focus on a natural variant of the problem called *discrete k-center*, which has also received considerable attention: here, given a set P of n points in $\mathbb{R}^d$ and a number k, we want to find k congruent balls covering P, while minimizing the radius, with the extra constraint that the centers of the chosen balls are from P.[3] The Euclidean discrete 1-center problem can be solved in $O(n \log n)$ time in $\mathbb{R}^2$ by a straightforward application of farthest-point Voronoi diagrams; it can also be solved in $O(n \log n)$ (randomized) time in $\mathbb{R}^3$ with more effort [15], and in subquadratic $\widetilde{O}(n^{2-2/(\lceil d/2 \rceil+1)})$ time for $d \geq 4$ by standard range searching techniques [4, 40]. Agarwal, Sharir, and Welzl in SoCG'97 [6] gave the first subquadratic algorithm for Euclidean discrete 2-center in $\mathbb{R}^2$, running in $\widetilde{O}(n^{4/3})$ time.

One may wonder whether Euclidean discrete 2-center in higher constant dimensions could also be solved in subquadratic time via range searching techniques. No results have been reported, but an $\widetilde{O}(n^\omega)$-time algorithm is not difficult to obtain, where $\omega < 2.373$ denotes the matrix multiplication exponent: by binary search, the problem reduces to finding two balls of a given radius r with centers in S covering S, which is equivalent to finding a pair $p, q \in S$ such that $c_{pq} = \bigvee_{z \in S}(a_{pz} \wedge a_{zq})$ is false, where a_{pz} is true iff p and z has distance more than r – this computation reduces to a Boolean matrix product. This approach works for arbitrary

[1] The $\widetilde{O}$ notation hides polylogarithmic factors.
[2] All squares, rectangles, hypercubes, and boxes are axis-aligned in this paper.
[3] Some authors define the problem slightly more generally, where the constraint is that the centers are from a second input set; in other words, the input consists of two sets of points ("demand points" and "supply points"). The results of this paper will apply to both versions of the problem.

Table 1 Summary of results on k-center for small k in $\mathbb{R}^2$.

k	Euclidean	rectilinear	Euclidean discrete	rectilinear discrete
1	$O(n)$	$O(n)$	$O(n \log n)$	$O(n)$
2	$\widetilde{O}(n)$ [44]	$O(n)$ [45]	$\widetilde{O}(n^{4/3})$ [6]	$\widetilde{O}(n)$
3		$\widetilde{O}(n)$ [45]		$\widetilde{O}(n^{3/2})$ **(new)**

(not necessarily geometric) distance functions. The main question is whether geometry could help in obtaining faster algorithms in the higher-dimensional Euclidean setting, as Agarwal, Sharir, and Welzl were able to exploit successfully in $\mathbb{R}^2$:

▶ **Question 1.** *Is there an algorithm running in faster than n^ω time for the Euclidean discrete 2-center problem in higher constant dimensions?*

We can similarly investigate the rectilinear version of the discrete k-center problem, which is potentially easier. For example, the rectilinear discrete 2-center problem can be solved in $\widetilde{O}(n)$ time in any constant dimension d, by a straightforward application of orthogonal range searching, as reported in several papers [10, 11, 34]. The approach does not seem to work for the rectilinear discrete 3-center problem. Naively, rectilinear discrete 3-center can be reduced to n instances of (some version of) rectilinear discrete 2-center, and solved in $\widetilde{O}(n^2)$ time. However, no better results have been published, leading to the following questions:

▶ **Question 2.** *Is there a subquadratic-time algorithm for the rectilinear discrete 3-center problem?*

▶ **Question 3.** *Are there lower bounds to show that the rectilinear discrete 3-center problem does not have near-linear-time algorithm (and is thus strictly harder than rectilinear discrete 2-center, or rectilinear continuous 3-center)?*

Similar questions may be asked about rectilinear discrete k-center for $k \geq 4$. Here, the complexity of the problem is upper-bounded by $\widetilde{O}(n^{\omega(\lfloor k/2 \rfloor, 1, \lceil k/2 \rceil)})$, where $\omega(a, b, c)$ denotes the exponent for multiplying an $n^a \times n^b$ and an $n^b \times n^c$ matrix: by binary search, the problem reduces to finding k hypercubes of a given edge length r with centers in S covering S, which is equivalent to finding a dominating set of size k in the graph with vertex set S where an edge pz exists iff the distance of p and z is more than r – the dominating set problem reduces to rectangular matrix multiplication with the time bound stated, as observed by Eisenbrand and Grandoni [28]. Note that the difference $\omega(\lfloor k/2 \rfloor, 1, \lceil k/2 \rceil) - k$ converges to 0 as $k \to \infty$ by known matrix multiplication bounds [25] (and is exactly 0 if $\omega = 2$).

As k gets larger compared to d, a better upper bound of $n^{O(dk^{1-1/d})}$ is known for both the continuous and discrete k-center problem under the Euclidean and rectilinear metric [5, 31, 32]. Recently, in SoCG'22, Chitnis and Saurabh [22] (extending earlier work by Marx [38] in the $\mathbb{R}^2$ case) proved a nearly matching conditional lower bound for discrete k-center in $\mathbb{R}^d$, ruling out $n^{o(k^{1-1/d})}$-time algorithms under ETH. However, these bounds do not answer our questions concerning very small k's. In contrast, the conditional lower bounds by Cabello et al. [14] that we have mentioned earlier are about very small k and so are more relevant, but are only for the continuous version of the k-center problem. (The continuous version behaves differently from the discrete version; see Tables 1–2.)

Table 2 Summary of results on k-center for small k in $\mathbb{R}^d$ for an arbitrary constant d. (CLB stands for "conditional lower bound".)

k	Euclidean	rectilinear	Euclidean discrete	rectilinear discrete
1	$O(n)$	$O(n)$	$\widetilde{O}(n^{2-2/(\lceil d/2\rceil+1)})$	$O(n)$
2	$n^{O(d)}$ CLB: $n^{\Omega(d)}$ [14]	$\widetilde{O}(n)$ [35]	$\widetilde{O}(n^\omega)$ CLB: $\Omega(n^{2-\delta})$ **(new)**	$\widetilde{O}(n)$
3		$\widetilde{O}(n)$ [14]	$\widetilde{O}(n^{\omega(1,1,2)})$ CLB: $\Omega(n^{3-\delta})$ **(new)**	$\widetilde{O}(n^2)$ CLB: $\Omega(n^{4/3-\delta})$ **(new)**
4		$n^{O(d)}$ CLB: $n^{\Omega(\sqrt{d})}$ [14]	$\widetilde{O}(n^{\omega(2,1,2)})$ CLB: $\Omega(n^{4-\delta})$ **(new)**	$\widetilde{O}(n^3)$

1.2 The geometric set cover problem with small size k

The decision version of the discrete k-center problem (deciding whether the minimum radius is at most a given value) reduces to a *geometric set cover* problem: given a set P of n points and a set R of n objects, find the smallest subset of objects in R that cover all points of P. Geometric set cover has been extensively studied in the literature, particularly from the perspective of approximation algorithms (since for most types of geometric objects, set cover remains NP-hard); for example, see the references in [18]. Here, we are interested in *exact* algorithms for the case when the optimal size k is a small constant.

For the application to Euclidean/rectilinear k-center, the objects are congruent balls/hypercubes, or by rescaling, unit balls/hypercubes, but other types of objects may be considered, such as arbitrary rectangles or boxes.

We can also consider the *weighted* version of the problem: here, given a set P of n points, a set R of n weighted objects, and a small constant k, we want to find a subset of k objects in R that cover all points of P, while minimizing the total weight of the chosen objects.

A "dual" problem is *geometric hitting set*, which in the weighted case is the following: given a set P of n weighted points, a set R of n objects, and a small constant k, find a subset of k points in P that hit all objects of R, while minimizing the total weight of the chosen points. (The continuous unweighted version, where the chosen points may be anywhere, is often called the *piercing* problem.) In the case of unit balls/hypercubes, hitting set is equivalent to set cover due to self-duality.

For rectangles in $\mathbb{R}^2$ or boxes in $\mathbb{R}^d$, size-2 geometric set cover (unweighted or weighted) can be solved in $\widetilde{O}(n)$ time, like discrete rectilinear 2-center [10, 11, 34], by orthogonal range searching. Analogs to Questions 2–3 may be asked for size-3 geometric set cover for rectangles/boxes.

Surprisingly, the complexity of exact geometric set cover of small size k has not received as much attention, but very recently in SODA'23, Chan [17] initiated the study of similar questions for geometric independent set with small size k, for example, providing subquadratic algorithms and conditional lower bounds for size-4 independent set for boxes.

For larger k, hardness results by Marx and Pilipczuk [39] and Bringmann et al. [13] ruled out $n^{o(k)}$-time algorithms for size-k geometric set cover for rectangles in $\mathbb{R}^2$ and unit hypercubes (or orthants) in $\mathbb{R}^4$, and $n^{o(\sqrt{k})}$-time algorithms for unit cubes (or orthants) in $\mathbb{R}^3$ under ETH. But like the other fixed-parameter intractability results mentioned, these proofs do not appear to imply any nontrivial lower bound for very small k such as $k = 3$.

1.3 New results

New algorithms. In this paper, we answer Question 2 in the affirmative for dimension $d = 2$, by presenting the first subquadratic algorithms for rectilinear discrete 3-center in $\mathbb{R}^2$, and more generally, for (unweighted and weighted) geometric size-3 set cover for unit squares, as well as arbitrary rectangles in $\mathbb{R}^2$. More precisely, the time bounds of our algorithms are:

- $\widetilde{O}(n^{3/2})$ for rectilinear discrete 3-center in $\mathbb{R}^2$ and unweighted size-3 set cover for unit squares in $\mathbb{R}^2$;
- $\widetilde{O}(n^{8/5})$ for weighted size-3 set cover for unit squares in $\mathbb{R}^2$;
- $\widetilde{O}(n^{5/3})$ for unweighted size-3 set cover for rectangles in $\mathbb{R}^2$;
- $\widetilde{O}(n^{7/4})$ for weighted size-3 set cover for rectangles in $\mathbb{R}^2$.

New conditional lower bounds. We also prove the first nontrivial conditional lower bounds on the time complexity of rectilinear discrete 3-center and related size-3 geometric set cover problems. More precisely, our lower bounds are:[4]

- $\Omega(n^{3/2-\delta})$ for weighted size-3 set cover (or hitting set) for unit squares in $\mathbb{R}^2$, assuming the APSP Hypothesis;
- $\Omega(n^{4/3-\delta})$ for rectilinear discrete 3-center in $\mathbb{R}^4$ and unweighted size-3 set cover (or hitting set) for unit hypercubes in $\mathbb{R}^4$, assuming the Sparse Triangle Hypothesis;
- $\Omega(n^{4/3-\delta})$ for unweighted size-3 set cover for boxes in $\mathbb{R}^3$, assuming the Sparse Triangle Hypothesis.

The lower bound in the first bullet is particularly attractive, since it implies that conditionally, our $\widetilde{O}(n^{8/5})$-time algorithm for weighted size-3 set cover for unit squares in $\mathbb{R}^2$ is within a small factor (near $n^{0.1}$) from optimal, and that our $\widetilde{O}(n^{7/4})$-time algorithm for weighted size-3 set cover for rectangles in $\mathbb{R}^2$ is within a factor near $n^{0.25}$ from optimal. The second bullet answers Question 3, implying that rectilinear discrete 3-center is strictly harder than rectilinear discrete 2-center and rectilinear (continuous) 3-center, at least when the dimension is 4 or higher. (In contrast, rectilinear (continuous) 4-center is strictly harder than rectilinear discrete 4-center for sufficiently large constant dimensions [14]; see Table 2.)

In addition, we prove the following conditional lower bounds:

- $\Omega(n^{2-\delta})$ for Euclidean discrete 2-center in $\mathbb{R}^{13}$ and unweighted size-3 set cover (or hitting set) for unit balls in $\mathbb{R}^{13}$, assuming the Hyperclique Hypothesis;
- $\Omega(n^{k-\delta})$ for Euclidean discrete k-center in $\mathbb{R}^{7k}$ and unweighted size-k set cover for unit balls in $\mathbb{R}^{7k}$ for any constant $k \geq 3$, assuming the Hyperclique Hypothesis.

In particular, this answers Question 1 in the negative if $\omega = 2$ (as conjectured by some researchers): geometry doesn't help for Euclidean discrete 2-center when the dimension is a sufficiently large constant. Similarly, the second bullet indicates that the upper bound $\widetilde{O}(n^{\omega(\lfloor k/2 \rfloor, 1, \lceil k/2 \rceil)})$ for Euclidean discrete k-center is basically tight for any fixed $k \geq 3$ in a sufficiently large constant dimension, if $\omega = 2$. (See Tables 1–3.)

Lastly, we prove a lower bound for a standard variant of set cover known as *maximum coverage*: given a set P of n points, a set R of n objects, and a small constant k, find k objects in R that cover the largest number (rather than all) of points of P. Geometric versions of the maximum coverage problem have been studied before from the approximation algorithms perspective (e.g., see [9]). It is also related to "outliers" variants of k-center problems (where we allow a certain number of points to be uncovered), which have also been studied for small

[4] Throughout this paper, δ denotes an arbitrarily small positive constant.

Table 3 Summary of results on size-3 geometric set cover in $\mathbb{R}^2$.

objects	unweighted	weighted
unit squares	$\widetilde{O}(n^{3/2})$ **(new)**	$\widetilde{O}(n^{8/5})$ **(new)**
		CLB: $\Omega(n^{3/2-\delta})$ **(new)**
rectangles	$\widetilde{O}(n^{5/3})$ **(new)**	$\widetilde{O}(n^{7/4})$ **(new)**
		CLB: $\Omega(n^{3/2-\delta})$ **(new)**

k (e.g., see [5]). Recall that the size-2 geometric set cover problem for boxes in $\mathbb{R}^d$ can be solved in $\widetilde{O}(n)$ time (which was why our attention was redirected to the size-3 case). In contrast, we show that maximum coverage for boxes cannot be solved in near-linear time even for size $k = 2$. More precisely, we obtain the following lower bound:

- $\Omega(n^{2-\delta})$ for size-2 maximum coverage for unit hypercubes in $\mathbb{R}^{12}$, assuming the Hyperclique Hypothesis.

What is notable is that this lower bound is tight (up to $n^{o(1)}$ factors), regardless of ω, since there is an obvious $\widetilde{O}(n^2)$-time algorithm for boxes in $\mathbb{R}^d$ by answering n^2 orthogonal range counting queries – our result implies that this obvious algorithm can't be improved!

On hypotheses from fine-grained complexity. Let us briefly state the hypotheses used.

- The *APSP Hypothesis* is among the three most popular hypotheses in fine-grained complexity [46] (the other two being the 3SUM Hypothesis and the Strong Exponential Time Hypothesis): it asserts that there is no $O(n^{3-\delta})$-time algorithm for the all-pairs shortest paths problem for an arbitrary weighted graph with n vertices (and $O(\log n)$-bit integer weights). This hypothesis has been used extensively in the algorithms literature (but less often in computational geometry).
- The *Sparse Triangle Hypothesis* asserts that there is no $O(m^{4/3-\delta})$-time algorithm for detecting a triangle (i.e., a 3-cycle) in a sparse unweighted graph with m edges. The current best upper bound for triangle detection, from a 3-decade-old paper by Alon, Yuster, and Zwick [8], is $\widetilde{O}(m^{2\omega/(\omega+1)})$, which is $\widetilde{O}(m^{4/3})$ if $\omega = 2$. (In fact, a stronger version of the hypothesis asserts that there is no $O(m^{2\omega/(\omega+1)-\delta})$-time algorithm.) As supporting evidence, it is known that certain "listing" or "all-edges" variants of the triangle detection problem have an $O(m^{4/3-\delta})$ lower bound, under the 3SUM Hypothesis or the APSP Hypothesis [43, 48, 20]. See [1, 33] for more discussion on the Sparse Triangle Hypothesis, and [17] for a recent application in computational geometry.
- The *Hyperclique Hypothesis* asserts that there is no $O(n^{k-\delta})$-time algorithm for detecting a size-k hyperclique in an ℓ-uniform hypergraph with n vertices, for any fixed $k > \ell \geq 3$. See [37] for discussion on this hypothesis, and [12, 17, 36] for some recent applications in computational geometry, including Künnemann's breakthrough result on conditional lower bounds for Klee's measure problem [36].

Techniques. Traditionally, in computational geometry, subquadratic algorithms with "intermediate" exponents between 1 and 2 tend to arise from the use of nonorthogonal range searching [4] (Agarwal, Sharir, and Welzl's $\widetilde{O}(n^{4/3})$-time algorithm for Euclidean discrete 2-center in $\mathbb{R}^2$ [6] being one such example). Our subquadratic algorithms for rectilinear discrete 3-center in $\mathbb{R}^2$ and related set-cover problems, which are about "orthogonal" or axis-aligned objects, are different. A natural first step is to use a $g \times g$ grid to divide into

cases, for some carefully chosen parameter g. Indeed, a grid-based approach was used in some recent subquadratic algorithms by Chan [17] for size-4 independent set for boxes in any constant dimension, and size-5 independent set for rectangles in $\mathbb{R}^2$ (with running time $\widetilde{O}(n^{3/2})$ and $\widetilde{O}(n^{4/3})$ respectively). However, discrete 3-center or rectangle set cover is much more challenging than independent set (for one thing, the 3 rectangles in the solution may intersect each other). To make the grid approach work, we need new original ideas (notably, a sophisticated argument to assign grid cells to rectangles, which is tailored to the 2D case). Still, the entire algorithm description fits in under 3 pages.

Our conditional lower bounds for rectilinear discrete 3-center and the corresponding set cover problem for unit hypercubes are proved by reduction from unweighted or weighted triangle finding in graphs. It turns out there is a simple reduction in $\mathbb{R}^2$ by exploiting weights. However, lower bounds in the unweighted case (and thus the original rectilinear discrete 3-center problem) are much trickier. We are able to design a clever, simple reduction in $\mathbb{R}^6$ by hand, but reducing the dimension down to 4 is far from obvious and we end up employing a *computer-assisted search*, interestingly. The final construction is still simple, and so is easy to verify by hand.

Our conditional lower bound proofs for Euclidean discrete 2-center, and more generally discrete k-center, are inspired by a recent conditional hardness proof by Bringmann et al. [12] from SoCG'22 on a different problem. Specifically, they proved that deciding whether the intersection graph of n unit hypercubes in $\mathbb{R}^{12}$ has diameter 2 requires near-quadratic time under the Hyperclique Hypothesis. A priori, this diameter problem doesn't seem related to discrete k-center; moreover, it was a rectilinear problem, not Euclidean (and we know that in contrast, rectilinear discrete 2-center has a near-linear upper bound!). Our contribution is in realizing that Bringmann et al.'s approach is useful for Euclidean discrete 2-center and k-center, surprisingly. To make the proof work though, we need some new technical ideas (in particular, an extra dimension for the $k = 2$ case, and multiple extra dimensions for larger k, with carefully designed coordinate values). Still, the final proof is not complicated to follow.

Our conditional lower bound for size-2 maximum coverage for boxes is also proved using a similar technique, but again the adaptation is nontrivial, and we introduce some interesting counting arguments that proceed a bit differently from Bringmann et al.'s original proof for diameter (a problem that does not involve counting).

2 Subquadratic Algorithms for Size-3 Set Cover for Rectangles in $\mathbb{R}^2$

In this section, we describe the most basic version of our subquadratic algorithm to solve the size-3 geometric set cover problem for weighted rectangles in $\mathbb{R}^2$. The running time is $\widetilde{O}(n^{16/9})$. Refinements of the algorithm will be described in the full version of the paper, where we will improve the time bound further to $\widetilde{O}(n^{7/4})$, or even better for the unweighted case and unit square case. The rectilinear discrete 3-center problem in $\mathbb{R}^2$ reduces to the unweighted unit square case by standard techniques [15, 30].

We begin with a lemma giving a useful geometric data structure:

▶ **Lemma 1.** *For a set P of n points and a set R of n weighted rectangles in $\mathbb{R}^2$, we can build a data structure in $\widetilde{O}(n)$ time and space, to support the following kind of queries: given a pair of rectangles $r_1, r_2 \in R$, we can find a minimum-weight rectangle $r_3 \in R$ (if it exists) such that P is covered by $r_1 \cup r_2 \cup r_3$, in $\widetilde{O}(1)$ time.*

Proof. By orthogonal range searching [4, 26] on P, we can find the minimum/maximum x- and y-values among the points of P in the complement of $r_1 \cup r_2$ in $\widetilde{O}(1)$ time (since the complement can be expressed as a union of $O(1)$ orthogonal ranges). As a result, we obtain

the minimum bounding box b enclosing $P \setminus (r_1 \cup r_2)$. To finish, we find a minimum-weight rectangle in R enclosing b; this is a "rectangle enclosure" query on R and can be solved in $\widetilde{O}(1)$ time, since it also reduces to orthogonal range searching (the rectangle $[x^-, x^+] \times [y^-, y^+]$ encloses the rectangle $[\xi^-, \xi^+] \times [\eta^-, \eta^+]$ in $\mathbb{R}^2$ iff the point (x^-, x^+, y^-, y^+) lies in the box $(-\infty, \xi^-] \times [\xi^+, \infty) \times (-\infty, \eta^-] \times [\eta^+, \infty)$ in $\mathbb{R}^4$). ◀

▶ **Theorem 2.** *Given a set P of n points and a set R of n weighted rectangles in $\mathbb{R}^2$, we can find 3 rectangles $r_1^*, r_2^*, r_3^* \in R$ of minimum total weight (if they exist), such that P is covered by $r_1^* \cup r_2^* \cup r_3^*$, in $\widetilde{O}(n^{16/9})$ time.*

Proof. Let B_0 be the minimum bounding box enclosing P (which touches 4 points). If a rectangle of R has an edge outside of B_0, we can eliminate that edge by extending the rectangle, making it unbounded.

Let g be a parameter to be determined later. Form a $g \times g$ (non-uniform) grid, where each column/row contains $O(n/g)$ rectangle vertices.

Step 1. For each pair of rectangles $r_1, r_2 \in R$ that have vertical edges in a common column or horizontal edges in a common row, we query the data structure in Lemma 1 to find a minimum-weight rectangle $r_3 \in R$ (if exists) such that $P \subset r_1 \cup r_2 \cup r_3$, and add the triple $r_1 r_2 r_3$ to a list L. The number of queried pairs $r_1 r_2$ is $O(g \cdot (n/g)^2) = O(n^2/g)$, and so this step takes $\widetilde{O}(n^2/g)$ total time.

Step 2. For each rectangle $r_1 \in R$ and each of its horizontal (resp. vertical) edges e_1, define $\gamma^-(e_1)$ and $\gamma^+(e_1)$ to be the leftmost and rightmost (resp. bottommost and topmost) grid cell that intersects e_1 and contains a point of P not covered by r_1. We can naively find $\gamma^-(e_1)$ and $\gamma^+(e_1)$ by enumerating the $O(g)$ grid cells intersecting e_1 and performing $O(g)$ orthogonal range queries; this takes $\widetilde{O}(gn)$ total time. For each rectangle $r_2 \in R$ that has an edge intersecting $\gamma^-(e_1)$ or $\gamma^+(e_1)$, we query the data structure in Lemma 1 to find a minimum-weight rectangle $r_3 \in R$ (if exists) such that $P \subset r_1 \cup r_2 \cup r_3$, and add the triple $r_1 r_2 r_3$ to the list L. The total number of queried pairs $r_1 r_2$ is $O(n \cdot n/g) = O(n^2/g)$, and so this step again takes $\widetilde{O}(n^2/g)$ total time. (This entire Step 2, and the definition of $\gamma^-(\cdot)$ and $\gamma^+(\cdot)$, might appear mysterious at first, but their significance will be revealed later in Step 3.)

Step 3. We guess the column containing each of the vertical edges of r_1^*, r_2^*, r_3^* and the row containing each of the horizontal edges of r_1^*, r_2^*, r_3^*; there are at most 12 edges and so $O(g^{12})$ choices. Actually, 4 of the 12 edges are eliminated after extension, and so the number of choices can be lowered to $O(g^8)$.

After guessing, we know which grid cells are completely inside r_1^*, r_2^*, r_3^* and which grid cells intersect which edges of r_1^*, r_2^*, r_3^*. We may assume that the vertical edges from different rectangles in $\{r_1^*, r_2^*, r_3^*\}$ are in different columns, and the horizontal edges from different rectangles in $\{r_1^*, r_2^*, r_3^*\}$ are in different rows: if not, $r_1^* r_2^* r_3^*$ would have already been found in Step 1. In particular, we know combinatorially what the arrangement of r_1^*, r_2^*, r_3^* looks like, even though we do not know the precise coordinates and identities of r_1^*, r_2^*, r_3^*.

We classify each grid cell γ into the following types (see Figure 1):

- Type A: γ is completely contained in some r_j^* ($j \in \{1, 2, 3\}$). Here, we *assign* γ to each such r_j^*.
- Type B: γ is not of type A, and intersects an edge of exactly one rectangle r_j^*. We *assign* γ to this r_j^*. Observe that points in $P \cap \gamma$ can only be covered by r_j^*.

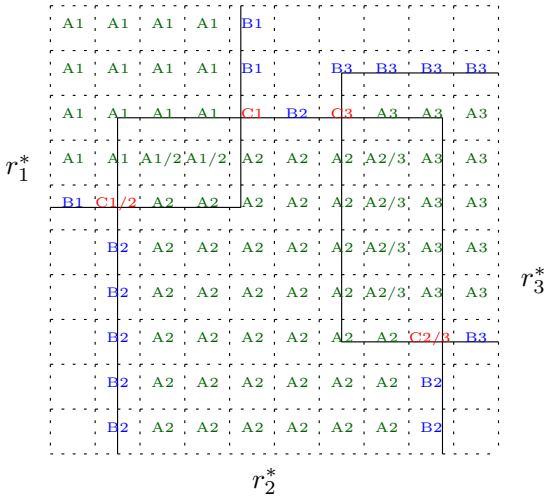


Figure 1 Proof of Theorem 2: grid cells in Step 3. The letter in a cell indicates its type (A, B, or C), and the number (or numbers) in a cell indicates the index (or indices) $j \in \{1, 2, 3\}$ of the rectangle r_j^* that the cell is assigned to.

- Type C: γ is not of type A, and intersects edges from two different rectangles in $\{r_1^*, r_2^*, r_3^*\}$. W.l.o.g., suppose that γ intersects a horizontal edge e_1^* of r_1^* and a vertical edge e_2^* of r_2^*; note that the intersection point $v^* = e_1^* \cap e_2^*$ lies on the boundary of the union $r_1^* \cup r_2^* \cup r_3^*$. By examining the arrangement of $\{r_1^*, r_2^*, r_3^*\}$, we know that at least one of the following is true: (i) we can walk horizontally from v^* to an endpoint of e_1^* (or a point at infinity) while staying on the boundary of $r_1^* \cup r_2^* \cup r_3^*$, or (ii) we can walk vertically from v^* to an endpoint of e_2^* (or a point at infinity) while staying on the boundary of $r_1^* \cup r_2^* \cup r_3^*$.

  If (i) is true, we *assign* γ to r_1^*. Observe that if there is a point in $P \cap \gamma$ not covered by r_1^* (and if the guesses are correct), then γ must be equal to $\gamma^-(e_1^*)$ or $\gamma^+(e_1^*)$ (as defined in Step 2), and so $r_1^* r_2^* r_3^*$ would have already been found in Step 2. This is because except for γ, the grid cells encountered while walking from v^* to that endpoint of e_1^* can intersect only r_1^* and so points in those cells can only be covered by r_1^*.

  If (ii) is true, we *assign* γ to r_2^* for a similar reason.

Note that there are at most $O(1)$ grid cells γ of type C; and the grid cells γ of type B form $O(1)$ contiguous blocks. Let ρ_j be the union of all grid cells assigned to r_j^*. Then ρ_j is a rectilinear polygon of $O(1)$ complexity. We compute the minimum/maximum x- and y-values of the points in $P \cap \rho_j$, by orthogonal range searching in $\widetilde{O}(1)$ time. As a result, we obtain the minimum bounding box b_j enclosing $P \cap \rho_j$. We find a minimum-weight rectangle $r_j \in R$ enclosing b_j, by a rectangle enclosure query (reducible to orthogonal range searching, as before). If $P \setminus (r_1 \cup r_2 \cup r_3) = \emptyset$ (testable by orthogonal range searching), we add the triple $r_1 r_2 r_3$ (which should coincide with $r_1^* r_2^* r_3^*$, if it has not been found earlier and if the guesses are correct) to L. The total time over all guesses is $\widetilde{O}(g^8)$.

At the end, we return a minimum-weight triple in L. The overall running time is $\widetilde{O}(g^8 + n^2/g + gn)$. Setting $g = n^{2/9}$ yields the theorem. ◀

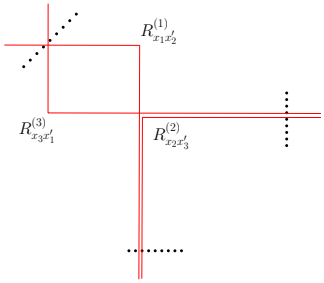


Figure 2 Reduction from the minimum-weight triangle problem to weighted size-3 set cover for orthants in $\mathbb{R}^2$.

3 Conditional Lower Bounds for Size-3 Set Cover for Boxes

In this section, we prove conditional lower bounds for size-3 set cover for boxes in certain dimensions (rectilinear discrete 3-center is related to size-3 set cover for unit hypercubes). We begin with the weighted version, which is more straightforward and has a simple proof, and serves as a good warm-up to the more challenging, unweighted version later.

3.1 Weighted size-3 set cover for unit squares in $\mathbb{R}^2$

An *orthant* (also called a dominance range) refers to a d-sided box in $\mathbb{R}^d$ which is unbounded along each of the d dimensions. (Note that orthants may be oriented in 2^d ways.) To obtain a lower bound for the unit square or unit hypercube case, it suffices to obtain a lower bound for the orthant case, since we can just replace each orthant with a hypercube with a sufficiently large side length M, and then rescale by a $1/M$ factor.

▶ **Theorem 3.** *Given a set P of n points and a set R of n weighted orthants in $\mathbb{R}^2$, finding 3 orthants in R of minimum total weight that cover P requires $\Omega(n^{3/2-\delta})$ time for any constant $\delta > 0$, assuming the APSP Hypothesis.*

Proof. The APSP Hypothesis is known to be equivalent [47] to the hypothesis that finding a minimum-weight triangle in a weighted graph with n vertices requires $\Omega(n^{3-\delta})$ time for any constant $\delta > 0$. We will reduce the minimum-weight triangle problem on a graph with n vertices and m edges ($m \in [n, n^2]$) to the weighted size-3 set cover problem for $O(m)$ points and orthants in $\mathbb{R}^2$. Thus, if there is an $O(m^{3/2-\delta})$-time algorithm for the latter problem, there would be an algorithm for the former problem with running time $O(m^{3/2-\delta}) \leq O(n^{3-2\delta})$, refuting the hypothesis.

Let $G = (V, E)$ be the given weighted graph with n vertices and m edges. Without loss of generality, assume that all edge weights are in $[0, 0.1]$, and that $V \subset [0, 0.1]$, i.e., vertices are labelled by numbers that are rescaled to lie in $[0, 0.1]$. Assume that $0 \in V$ and $0.1 \in V$.

The reduction. For each vertex $t \in V$, we create three points $(t, 1+t)$, $(2, t)$, and $(1+t, -1)$ (call them of type 1, 2, and 3, respectively).

Create the following orthants in $\mathbb{R}^2$:

$$\begin{aligned}
\forall x_1x'_2 \in E:\ R^{(1)}_{x_1x'_2} &= (-\infty, 1+x'_2) \times (-\infty, 1+x_1] && \text{(type 1)}\\
\forall x_2x'_3 \in E:\ R^{(2)}_{x_2x'_3} &= [1+x_2, \infty) \times (-\infty, x'_3) && \text{(type 2)}\\
\forall x_3x'_1 \in E:\ R^{(3)}_{x_3x'_1} &= (x'_1, \infty) \times [x_3, \infty) && \text{(type 3)}
\end{aligned}$$

The weight of each orthant is set to be the number of points it covers plus the weight of the edge it represents. The total number of points and orthants is $O(n)$ and $O(m)$ respectively. The reduction is illustrated in Figure 2.

Correctness. We prove that the minimum-weight triangle in G has weight w (where $w \in [0, 0.3]$) iff the optimal weighted size-3 set cover has weight $3n + w$.

Any feasible solution (if exists) must use an orthant of each type, since the point $(0, 1)$ of type 1 (resp. the point $(2, 0.1)$ of type 2, and the point $(1.1, -1)$ of type 3) can only be covered by an orthant of type 1 (resp. 3 and 2). So, the three orthants in the optimal solution must be of the form $R^{(1)}_{x_1x_2'}$, $R^{(2)}_{x_2x_3'}$ and $R^{(3)}_{x_3x_1'}$ for some $x_1x_2', x_2x_3', x_3x_1' \in E$.

If $x_1 < x_1'$, some point (of type 1) would be uncovered; on the other hand, if $x_1 > x_1'$, some point (of type 1) would be covered twice, and the total weight would then be at least $3n + 1$. Thus, $x_1 = x_1'$. Similarly, $x_2 = x_2'$ and $x_3 = x_3'$. So, $x_1x_2x_3$ forms a triangle in G. We conclude that the minimum-weight solution $R^{(1)}_{x_1x_2}$, $R^{(2)}_{x_2x_3}$ and $R^{(3)}_{x_3x_1}$ correspond to the minimum-weight triangle $x_1x_2x_3$ in G. ◀

3.2 Unweighted size-3 set cover for boxes in $\mathbb{R}^3$

Our preceding reduction uses weights to ensure equalities of two variables representing vertices. For the unweighted case, this does not work. We propose a different way to force equalities, by using an extra dimension and extra sides (i.e., using boxes instead of orthants), with some carefully chosen coordinate values.

▶ **Theorem 4.** *Given a set P of n points and a set R of n unweighted axis-aligned boxes in $\mathbb{R}^3$, deciding whether there exist 3 boxes in R that cover P requires $\Omega(n^{4/3-\delta})$ time for any constant $\delta > 0$, assuming the Sparse Triangle Hypothesis.*

Proof. We will reduce the triangle detection problem on a graph with m edges to the unweighted size-3 set cover problem for $O(m)$ points and boxes in $\mathbb{R}^3$. Thus, if there is an $O(m^{4/3-\delta})$-time algorithm for the latter problem, there would be an algorithm for the former problem with running time $O(m^{4/3-\delta})$, refuting the hypothesis.

Let $G = (V, E)$ be the given unweighted sparse graph with n vertices and m edges ($n \le m$). Without loss of generality, assume that $V \subset [0, 0.1]$, and $0 \in V$ and $0.1 \in V$.

The reduction. For each vertex $t \in V$, create six points

$$\begin{array}{rrrl}
(-1+t, & 0, & 2+t) & \text{(type 1)}\\
(1+t, & 0, & -2+t) & \text{(type 2)}\\
(2+t, & -1+t, & 0) & \text{(type 3)}\\
(-2+t, & 1+t, & 0) & \text{(type 4)}\\
(0, & 2+t, & -1+t) & \text{(type 5)}\\
(0, & -2+t, & 1+t) & \text{(type 6)}
\end{array}$$

Create the following boxes in $\mathbb{R}^3$:

$$\begin{array}{lrcccccc}
\forall x_1x_2' \in E: & R^{(1)}_{x_1x_2'} & = & (-1+x_1, 1+x_1) & \times & [-2+x_2', 2+x_2'] & \times & \mathbb{R}\\
\forall x_2x_3' \in E: & R^{(2)}_{x_2x_3'} & = & [-2+x_3', 2+x_3'] & \times & \mathbb{R} & \times & (-1+x_2, 1+x_2)\\
\forall x_3x_1' \in E: & R^{(3)}_{x_3x_1'} & = & \mathbb{R} & \times & (-1+x_3, 1+x_3) & \times & [-2+x_1', 2+x_1']
\end{array}$$

(call them of type 1, 2, and 3, respectively).

Correctness. We prove that a size-3 set cover exists iff a triangle exists in G.

Any feasible solution (if exists) must use a box of each type, since the point $(-1, 0, 2)$ of type 1 (resp. the point $(2, -1, 0)$ of type 3, and the point $(0, 2, -1)$ of type 5) can only be covered by a box of type 3 (resp. 2 and 1). So, the three boxes in a feasible solution must be of the form $R^{(1)}_{x_1x'_2}$, $R^{(2)}_{x_2x'_3}$ and $R^{(3)}_{x_3x'_1}$ for some $x_1x'_2, x_2x'_3, x_3x'_1 \in E$.

Consider points of type 1 with the form $(-1+t, 0, 2+t)$. The box $R^{(2)}_{x_2x'_3}$ cannot cover any of them due to the third dimension. The box $R^{(1)}_{x_1x'_2}$ covers all such points corresponding to $t > x_1$, and the box $R^{(3)}_{x_3x'_1}$ covers all such points corresponding to $t \le x'_1$. So, all points of type 1 are covered iff $x_1 \le x'_1$. Similarly, all points of type 2 are covered iff $x'_1 \le x_1$. Thus, all points of type 1–2 are covered iff $x_1 = x'_1$. By a symmetric argument, all points of type 3–4 are covered iff $x_3 = x'_3$; and all points of type 5–6 are covered iff $x_2 = x'_2$. We conclude that a feasible solution exists iff a triangle $x_1x_2x_3$ exists in G. ◀

We remark that the boxes above can be made fat, with side lengths between 1 and a constant (by replacing $\mathbb{R}$ with an interval of a sufficiently large constant length).

3.3 Unweighted size-3 set cover for unit hypercubes in $\mathbb{R}^4$

Our preceding lower bound for unweighted size-3 set cover for boxes in $\mathbb{R}^3$ immediately implies a lower bound for orthants (and thus unit hypercubes) in $\mathbb{R}^6$, since the point (x, y, z) is covered by the box $[a^-, a^+] \times [b^-, b^+] \times [c^-, c^+]$ in $\mathbb{R}^3$ iff the point (x, x, y, y, z, z) is covered by the orthant $[a^-, \infty) \times (-\infty, a^+] \times [b^-, \infty) \times (-\infty, b^+] \times [c^-, \infty) \times (-\infty, c^+]$ in $\mathbb{R}^6$.

A question remains: can the dimension 6 be lowered? Intuitively, there seems to be some wastage in the above construction: there are several 0's in the coordinates of the points, and several $\mathbb{R}$'s in the definition of the boxes, and these get doubled after the transformation to 6 dimensions. However, it isn't clear how to rearrange coordinates to eliminate this wastage: we would have to give up this nice symmetry of our construction, and there are too many combinations to try. We ended up writing a computer program to exhaustively try all these different combinations, and eventually find a construction that lowers the dimension to 4! Once it is found, correctness is straightforward to check, as one can see in the proof below.

▶ **Theorem 5.** *Given a set P of n points and a set R of n unweighted orthants in $\mathbb{R}^4$, deciding whether there exists a size-3 set cover requires $\Omega(n^{4/3-\delta})$ time for any constant $\delta > 0$, assuming the Sparse Triangle Hypothesis.*

Proof. We will reduce the triangle detection problem on a graph with m edges to the unweighted size-3 set cover problem for $O(m)$ points and orthants in $\mathbb{R}^4$.

Let $G = (V, E)$ be the given unweighted graph with n vertices and m edges ($n \le m$). Without loss of generality, assume that $V \subset [0, 0.1]$, and $0 \in V$ and $0.1 \in V$.

The reduction. For each vertex $t \in V$, create six points

$(0+t,$	$2+t,$	$-0.5,$	$-0.5)$	(type 1)
$(2-t,$	$0-t,$	$-0.5,$	$-0.5)$	(type 2)
$(1-t,$	$0.5,$	$1+t,$	$1.5)$	(type 3)
$(0.5,$	$1+t,$	$0.5,$	$2-t)$	(type 4)
$(-0.5,$	$-0.5,$	$2-t,$	$0-t)$	(type 5)
$(-0.5,$	$-0.5,$	$0+t,$	$1+t)$	(type 6)

Create the following orthants in $\mathbb{R}^4$:

$$\begin{aligned}
\forall x_1x_2' \in E:\ R^{(1)}_{x_1x_2'} &= [0+x_1,+\infty) \times [0-x_1,+\infty) \times (-\infty,1+x_2') \times (-\infty,2-x_2')\\
\forall x_2x_3' \in E:\ R^{(2)}_{x_2x_3'} &= (-\infty,1-x_2] \times (-\infty,1+x_2] \times (0+x_3',+\infty) \times (0-x_3',+\infty)\\
\forall x_3x_1' \in E:\ R^{(3)}_{x_3x_1'} &= (-\infty,2-x_1') \times (-\infty,2+x_1') \times (-\infty,2-x_3] \times (-\infty,1+x_3]
\end{aligned}$$

(call them of type 1, 2, and 3, respectively).

Correctness. We prove that a size-3 set cover exists iff a triangle exists in G.

Any feasible solution (if exists) must use an orthant of each type, as one can easily check (like before). So, the three orthants in a feasible solution must be of the form $R^{(1)}_{x_1x_2'}$, $R^{(2)}_{x_2x_3'}$ and $R^{(3)}_{x_3x_1'}$ for some $x_1x_2', x_2x_3', x_3x_1' \in E$.

Consider points of type 1 with the form $(0+t, 2+t, -0.5, -0.5)$. The orthant $R^{(2)}_{x_2x_3'}$ cannot cover any of them due to the third dimension. The orthant $R^{(1)}_{x_1x_2'}$ covers all such points corresponding to $t \geq x_1$, and the orthant $R^{(3)}_{x_3x_1'}$ covers all such points corresponding to $t < x_1'$. So, all points of type 1 are covered iff $x_1 \leq x_1'$. By similar arguments, it can be checked that all points of type 2 are covered iff $x_1 \geq x_1'$; all points of type 3 are covered iff $x_2 \leq x_2'$; all points of type 4 are covered iff $x_2 \geq x_2'$; all points of type 5 are covered iff $x_3 \leq x_3'$; all points of type 6 are covered iff $x_3 \geq x_3'$. We conclude that a feasible solution exists iff a triangle $x_1x_2x_3$ exists in G. ◀

In the computer search, we basically tried different choices of points with coordinate values of the form $c \pm t$ or c for some constant c, and orthants defined by intervals of the form $[c \pm x_j, +\infty)$ or $(-\infty, c \pm x_j]$ (closed or open) for some variable x_j (or x_j'). The constraints are not exactly easy to write down, but are self-evident as we simulate the correctness proof above. Naively, the number of cases is in the order of 10^{14}, but can be drastically reduced to about 10^7 with some optimization and careful pruning of the search space. The C++ code is not long (under 150 lines) and, after incorporating pruning, runs in under a second.

It is now straightforward to modify the above lower bound proof for unweighted orthants (or unit hypercubes) in $\mathbb{R}^4$ to the rectilinear discrete 3-center problem in $\mathbb{R}^4$. In the full version of the paper, we also prove a higher conditional lower bound for weighted size-6 set cover for rectangles in $\mathbb{R}^2$.

4 Conditional Lower Bound for Euclidean Discrete 2-Center

In this section, we prove our conditional lower bound for the Euclidean discrete 2-center problem in a sufficiently large constant dimension. The general structure of our proof is inspired by Bringmann et al.'s recent conditional hardness proof [12] for the problem of computing diameter of box intersection graphs in $\mathbb{R}^{12}$, specifically, testing whether the diameter is more than 2. (Despite the apparent dissimilarities of the two problems, what led us to initially suspect that the ideas there might be useful is that both problems are concerned with paths of length 2 in certain geometrically defined graphs, and both problems have a similar "quantifier structure", after unpacking the problem definitions.) Extra ideas are needed, as we are dealing with the Euclidean metric instead of boxes; we end up needing an extra dimension, with carefully tuned coordinate values, to make the proof work.

▶ **Theorem 6.** *For any constant $\delta > 0$, there is no $O(n^{2-\delta})$-time algorithm for Euclidean discrete 2-center in $\mathbb{R}^{13}$, assuming the Hyperclique Hypothesis.*

Proof. We will reduce the problem of detecting a 6-hyperclique in a 3-uniform hypergraph with n vertices, to the Euclidean discrete 2-center problem on $N = O(n^3)$ points in $\mathbb{R}^{13}$. Thus, if there is an $O(N^{2-\delta})$-time algorithm for the latter problem, there would be $O(n^{6-3\delta})$-time algorithm for the former problem, refuting the Hyperclique Hypothesis.

Let $G = (V, E)$ be the given 3-uniform hypergraph. By a standard color-coding technique [7], we may assume that G is 6-partite, i.e., V is partitioned into 6 parts $V_1, \ldots, V_6$, and each edge in E consists of 3 vertices from 3 different parts. The goal is to decide the existence of a 6-hyperclique, i.e., $(x_1, \ldots, x_6) \in V_1 \times \cdots \times V_6$ such that $\{x_i, x_j, x_k\} \in E$ for all distinct $i, j, k \in \{1, \ldots, 6\}$.

Without loss of generality, assume that $V \subset [0, 1]$, i.e., vertices are labelled by numbers that are rescaled to lie in $[0, 1]$. Let $f, g : [0, 1] \to [0, 1]$ be some injective functions satisfying $f(x)^2 + g(x)^2 = 1$. For example, we can take $f(x) = \cos x$ and $g(x) = \sin x$; or alternatively, avoiding trigonometric functions, $f(x) = x$ and $g(x) = \sqrt{1 - x^2}$; or avoiding irrational functions altogether, $f(x) = 2x/(x^2 + 1)$ and $g(x) = (x^2 - 1)/(x^2 + 1)$. (With the last two options, by rounding to $O(\log n)$ bits of precision, it is straightforward to make our reduction work in the standard integer word RAM model.)

The reduction. We construct the following set S of $O(n^3)$ points in $\mathbb{R}^{13}$:

1. For each $(x_1, x_2, x_3) \in V_1 \times V_2 \times V_3$ with $\{x_1, x_2, x_3\} \in E$, create the point
$$p_{x_1x_2x_3} = (f(x_1), g(x_1), f(x_2), g(x_2), f(x_3), g(x_3), 0, 0, 0, 0, 0, 0, 1).$$
2. Similarly, for each $(x_4, x_5, x_6) \in V_4 \times V_5 \times V_6$ such that $\{x_4, x_5, x_6\} \in E$, create the point
$$q_{x_4x_5x_6} = (0, 0, 0, 0, 0, 0, f(x_4), g(x_4), f(x_5), g(x_5), f(x_6), g(x_6), -1).$$
3. For each $(v_i, v_j, v_k) \in V_i \times V_j \times V_k$ with distinct i, j, k such that $\{v_i, v_j, v_k\} \notin E$, $\{i, j, k\} \neq \{1, 2, 3\}$, and $\{i, j, k\} \neq \{4, 5, 6\}$, create a point $z_{v_iv_jv_k}$: the coordinates in dimensions $2i-1, 2i$ are $-f(v_i), -g(v_i)$, and similarly the coordinates in dimensions $2j-1, 2j, 2k-1, 2k$ are $-f(v_j), -g(v_j), -f(v_k), -g(v_k)$, respectively; the 13-th coordinate is
$$\phi_{ijk} = |\{1, 2, 3\} \cap \{i, j, k\}| - 1.5 \in \{-0.5, 0.5\};$$
and all other coordinates are 0. For example, if $i = 1, j = 2, k = 4$,
$$z_{v_1v_2v_4} = (-f(v_1), -g(v_1), -f(v_2), -g(v_2), 0, 0, -f(v_4), -g(v_4), 0, 0, 0, 0, 0.5).$$
4. Finally, add two auxiliary points $s_\pm = (0, \ldots, 0, \pm 3.5)$.

We solve the discrete 2-center problem on the above point set S, and return true iff the minimum radius is strictly less than $\sqrt{10.25}$.

Correctness. Suppose there exists a 6-hyperclique $(x_1, \ldots, x_6) \in V_1 \times \cdots \times V_6$ in G. We claim that every point of S has distance strictly less than $\sqrt{10.25}$ from $p_{x_1x_2x_3}$ or $q_{x_4x_5x_6}$. Thus, S can be covered by 2 balls centered at $p_{x_1x_2x_3}$ and $q_{x_4x_5x_6}$ with radius less than $\sqrt{10.25}$. To verify the claim, consider a point $z_{v_1v_2v_4} \in S$ for a triple $(v_1, v_2, v_4) \in V_1 \times V_2 \times V_4$ with $\{v_1, v_2, v_4\} \notin E$. Observe that the distance between the points $(f(v_\ell), g(v_\ell))$ and $(-f(x_\ell), -g(x_\ell))$ in $\mathbb{R}^2$ is at most 2, with equality iff $v_\ell = x_\ell$. On the other hand, the distance between $(f(v_\ell), g(v_\ell))$ and $(0, 0)$ is 1, and the distance between $(0, 0)$ and $(-f(x_\ell), -g(x_\ell))$ is 1. Thus,
$$\|z_{v_1v_2v_4} - p_{x_1x_2x_3}\|^2 \leq 2^2 + 2^2 + 1 + 1 + 0 + 0 + (0.5 - 1)^2 \leq 10.25,$$

with equality iff $v_1 = x_1$ and $v_2 = x_2$. Furthermore,

$$\|z_{v_1v_2v_4} - q_{x_4x_5x_6}\|^2 \;\le\; 1+1+0+2^2+1+1+(0.5+1)^2 \;\le\; 10.25,$$

with equality iff $v_4 = x_4$. Since $\{x_1, x_2, x_4\} \in E$, we cannot have simultaneously $v_1 = x_1$, $v_2 = x_2$, and $v_4 = x_4$. So, $z_{v_1v_2v_4}$ has distance strictly less than $\sqrt{10.25}$ from $p_{x_1x_2x_3}$ or $q_{x_4x_5x_6}$. Similarly, the same holds for $z_{v_iv_jv_k} \in S$ for all other choices of i, j, k. Points $p_{x'_1x'_2x'_3} \in S$ have distance at most $\sqrt{2+2+2+0+0+0+0} < \sqrt{10.25}$ from $p_{x_1x_2x_3}$, and similarly, points $q_{x'_4x'_5x'_6} \in S$ have distance less than $\sqrt{10.25}$ from $q_{x_4x_5x_6}$. Finally, the auxiliary point s_+ has distance at most $\sqrt{1+1+1+0+0+0+2.5^2} < \sqrt{10.25}$ from $p_{x_1x_2x_3}$, and similarly the point s_- has distance less than $\sqrt{10.25}$ from $q_{x_4x_5x_6}$.

On the reverse direction, suppose that the minimum radius for the discrete 2-center problem on S is strictly less than $\sqrt{10.25}$. Note that the distance between s_+ and $z_{v_iv_jv_k}$ is at least $\sqrt{1+1+1+0+0+0+3^2} > \sqrt{10.25}$, and the distance between s_+ and $q_{x_4x_5x_6}$ is at least $\sqrt{0+0+0+1+1+1+4.5^2} > \sqrt{10.25}$. Thus, in order to cover s_+, one of the two centers must be equal to $p_{x_1x_2x_3}$ for some $\{x_1, x_2, x_3\} \in E$. Similarly, in order to cover s_-, the other center must be equal to $q_{x_4x_5x_6}$ for some $\{x_4, x_5, x_6\} \in E$. Then for every $(v_1, v_2, v_4) \in V_1 \times V_2 \times V_4$ with $\{v_1, v_2, v_4\} \notin E$, the point $z_{v_1v_2v_4}$ has distance strictly less than $\sqrt{10.25}$ from $p_{x_1x_2x_3}$ or $q_{x_4x_5x_6}$. By the above argument, we cannot have $v_1 = x_1$ and $v_2 = x_2$ and $v_4 = x_4$. It follows that $\{x_1, x_2, x_4\} \in E$. Similarly, $\{x_i, x_j, x_k\} \in E$ for all other choices of i, j, k. We conclude that $\{x_1, \ldots, x_6\}$ is a 6-hyperclique. ◀

From the same proof (after rescaling), we immediately get a near-quadratic conditional lower bound for unweighted size-2 geometric set cover for unit balls in $\mathbb{R}^{13}$. In the full version of the paper, we extend the proof to Euclidean discrete k-center for larger constant k, with more technical effort and more delicate handling of the extra dimensions. This is interesting: discrete k-center seems even farther away from graph diameter, but in a way, our proof shows that discrete k-center is a better problem to illustrate the full power of Bringmann et al.'s technique [12].

In the full version of the paper, we also adapt the approach to prove a conditional lower bound for size-2 maximum coverage for boxes. The proof uses a different way to enforce conditions like $\{x_1, x_2, x_4\} \in E$, via an interesting counting argument – we encourage the readers to take a look at the full version.

5 Conclusions

In this paper, we have obtained a plethora of nontrivial new results on a fundamental class of problems in computational geometry related to discrete k-center and size-k geometric set cover for small values of k. (See Tables 1–3.) In particular, we have a few results where the upper bounds and conditional lower bounds are close:

- For weighted size-3 set cover for rectangles in $\mathbb{R}^2$, we have given the first subquadratic $\widetilde{O}(n^{7/4})$-time algorithm, and an $\Omega(n^{3/2-\delta})$ lower bound under the APSP Hypothesis.
- For Euclidean discrete k-center (or unweighted size-k set cover for unit balls) in $\mathbb{R}^{O(k)}$, we have proved an $\Omega(n^{k-\delta})$ lower bound under the Hyperclique Hypothesis, which is near optimal if $\omega = 2$.
- For size-2 maximum coverage for boxes in a sufficiently large constant dimension, we have proved an $\Omega(n^{2-\delta})$ lower bound under the Hyperclique Hypothesis, which is near optimal.

For all of our results, we have managed to find simple proofs (each with 1–3 pages). We view the simplicity and accessibility of our proofs as an asset – they would make good examples illustrating fine-grained complexity techniques in computational geometry. Generally speaking, there has been considerable development on fine-grained complexity in the broader algorithms community over the last decade [46], but to a lesser extent in computational geometry. A broader goal of this paper is to encourage more work at the intersection of these two areas. We should emphasize that while our conditional lower bound proofs may appear simple in hindsight, they are not necessarily easy to come up with; for example, see one of our proofs that require computer-assisted search (Theorem 5).

As many versions of the problems studied here still do not have matching upper and lower bounds, our work raises many interesting open questions. For example:

- Is it possible to make our subquadratic algorithm for rectilinear discrete 3-center in $\mathbb{R}^2$ work in dimension 3 or higher?
- Is it possible to make our conditional lower bound proof for rectilinear discrete 3-center in $\mathbb{R}^4$ work in dimension 2 or 3?
- Is it possible to make our conditional lower bound for Euclidean discrete 2-center in $\mathbb{R}^{13}$ work in dimension 3?
- Is it possible to make our conditional lower bound for size-2 maximum coverage for boxes in $\mathbb{R}^{12}$ work in dimension 2 or 3?
- Although we have ruled out subquadratic algorithms for Euclidean discrete 2-center in $\mathbb{R}^{13}$, could geometry still help in beating n^ω time if $\omega > 2$?

We should remark that some of these questions could be quite difficult. In fine-grained complexity, there are many examples of basic problems that still do not have tight conditional lower bounds (to mention one well-known geometric example, Künnemann's recent FOCS'22 paper [36] has finally obtained a near-optimal conditional lower bound for Klee's measure problem in $\mathbb{R}^3$, but tight lower bounds in dimension 4 and higher are still not known for non-combinatorial algorithms). Still, we hope that our work would inspire more progress in both upper and lower bounds for this rich class of problems.

References

1 Amir Abboud, Karl Bringmann, Seri Khoury, and Or Zamir. Hardness of approximation in P via short cycle removal: Cycle detection, distance oracles, and beyond. In *Proc. 54th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1487–1500, 2022. doi:10.1145/3519935.3520066.

2 Pankaj K. Agarwal, Rinat Ben Avraham, and Micha Sharir. The 2-center problem in three dimensions. *Comput. Geom.*, 46(6):734–746, 2013. Preliminary version in SoCG'10. doi:10.1016/j.comgeo.2012.11.005.

3 Pankaj K. Agarwal, Alon Efrat, and Micha Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIAM J. Comput.*, 29(3):912–953, 1999. doi:10.1137/S0097539795295936.

4 Pankaj K. Agarwal and Jeff Erickson. Geometric range searching and its relatives. In *Advances in Discrete and Computational Geometry*, volume 223 of *Contemporary Mathematics*, pages 1–56. AMS Press, 1999. URL: http://jeffe.cs.illinois.edu/pubs/survey.html.

5 Pankaj K. Agarwal and Cecilia Magdalena Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33(2):201–226, 2002. doi:10.1007/s00453-001-0110-y.

6 Pankaj K Agarwal, Micha Sharir, and Emo Welzl. The discrete 2-center problem. *Discrete & Computational Geometry*, 20(3):287–305, 1998. Preliminary version in SoCG'97.

7 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.

8 Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997. doi:10.1007/BF02523189.

9 Ashwinkumar Badanidiyuru, Robert Kleinberg, and Hooyeon Lee. Approximating low-dimensional coverage problems. In *Proc. 28th ACM Symposium on Computational Geometry (SoCG)*, pages 161–170, 2012. doi:10.1145/2261250.2261274.

10 Sergei Bespamyatnikh and David G. Kirkpatrick. Rectilinear 2-center problems. In *Proc. 11th Canadian Conference on Computational Geometry (CCCG)*, 1999. URL: http://www.cccg.ca/proceedings/1999/fp55.pdf.

11 Sergei Bespamyatnikh and Michael Segal. Rectilinear static and dynamic discrete 2-center problems. In *Proc. 6th Workshop on Algorithms and Data Structures (WADS)*, pages 276–287, 1999. doi:10.1007/3-540-48447-7_28.

12 Karl Bringmann, Sándor Kisfaludi-Bak, Marvin Künnemann, André Nusser, and Zahra Parsaeian. Towards sub-quadratic diameter computation in geometric intersection graphs. In *Proc. 38th International Symposium on Computational Geometry (SoCG)*, pages 21:1–21:16, 2022. doi:10.4230/LIPIcs.SoCG.2022.21.

13 Karl Bringmann, Sándor Kisfaludi-Bak, Michal Pilipczuk, and Erik Jan van Leeuwen. On geometric set cover for orthants. In *Proc. 27th Annual European Symposium on Algorithms (ESA)*, pages 26:1–26:18, 2019. doi:10.4230/LIPIcs.ESA.2019.26.

14 Sergio Cabello, Panos Giannopoulos, Christian Knauer, Dániel Marx, and Günter Rote. Geometric clustering: Fixed-parameter tractability and lower bounds with respect to the dimension. *ACM Trans. Algorithms*, 7(4):43:1–43:27, 2011. Preliminary version in SODA'08. doi:10.1145/2000807.2000811.

15 Timothy M. Chan. Geometric applications of a randomized optimization technique. *Discret. Comput. Geom.*, 22(4):547–567, 1999. doi:10.1007/PL00009478.

16 Timothy M. Chan. More planar two-center algorithms. *Computational Geometry*, 13(3):189–198, 1999.

17 Timothy M. Chan. Finding triangles and other small subgraphs in geometric intersection graphs. In *Proc. 34th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2023. To appear. URL: https://arxiv.org/abs/2211.05345.

18 Timothy M. Chan and Qizheng He. Faster approximation algorithms for geometric set cover. In *Proc. 36th International Symposium on Computational Geometry (SoCG)*, pages 27:1–27:14, 2020. doi:10.4230/LIPIcs.SoCG.2020.27.

19 Timothy M. Chan, Qizheng He, and Yuancheng Yu. On the fine-grained complexity of small-size geometric set cover and discrete k-center for small k. arXiv preprint, 2023. arXiv:2305.01892.

20 Timothy M. Chan, Virginia Vassilevska Williams, and Yinzhan Xu. Hardness for triangle problems under even more believable hypotheses: Reductions from Real APSP, Real 3SUM, and OV. In *Proc. 54th ACM Symposium on Theory of Computing (STOC)*, pages 1501–1514, 2022. doi:10.1145/3519935.3520032.

21 Bernard Chazelle and Jirı Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. *Journal of Algorithms*, 21(3):579–597, 1996.

22 Rajesh Chitnis and Nitin Saurabh. Tight lower bounds for approximate & exact k-center in $\mathbb{R}^d$. In *Proc. 38th International Symposium on Computational Geometry (SoCG)*, pages 28:1–28:15, 2022. doi:10.4230/LIPIcs.SoCG.2022.28.

23 Jongmin Choi and Hee-Kap Ahn. Efficient planar two-center algorithms. *Comput. Geom.*, 97:101768, 2021. doi:10.1016/j.comgeo.2021.101768.

24 Kenneth L. Clarkson. Las Vegas algorithms for linear and integer programming when the dimension is small. *Journal of the ACM*, 42(2):488–499, 1995.

25 Don Coppersmith. Rapid multiplication of rectangular matrices. *SIAM J. Comput.*, 11(3):467–471, 1982. doi:10.1137/0211037.

26 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008. URL: https://www.worldcat.org/oclc/227584184.

27 Martin E. Dyer. On a multidimensional search technique and its application to the Euclidean one-centre problem. *SIAM Journal on Computing*, 15(3):725–738, 1986.

28 Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theor. Comput. Sci.*, 326(1-3):57–67, 2004. doi:10.1016/j.tcs.2004.05.009.

29 David Eppstein. Faster construction of planar two-centers. In *Proc. 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 131–138, 1997. URL: http://dl.acm.org/citation.cfm?id=314161.314198.

30 Greg N. Frederickson and Donald B. Johnson. The complexity of selection and ranking in $X + Y$ and matrices with sorted columns. *J. Comput. Syst. Sci.*, 24(2):197–208, 1982. doi:10.1016/0022-0000(82)90048-4.

31 R. Z. Hwang, R. C. Chang, and Richard C. T. Lee. The searching over separators strategy to solve some NP-hard problems in subexponential time. *Algorithmica*, 9(4):398–423, 1993. doi:10.1007/BF01228511.

32 R. Z. Hwang, Richard C. T. Lee, and R. C. Chang. The slab dividing approach to solve the Euclidean p-center problem. *Algorithmica*, 9(1):1–22, 1993. doi:10.1007/BF01185335.

33 Ce Jin and Yinzhan Xu. Removing additive structure in 3SUM-based reductions. *CoRR*, abs/2211.07048, 2022. doi:10.48550/arXiv.2211.07048.

34 Matthew J. Katz, Klara Kedem, and Michael Segal. Discrete rectilinear 2-center problems. *Comput. Geom.*, 15(4):203–214, 2000. doi:10.1016/S0925-7721(99)00052-8.

35 Matthew J. Katz and Frank Nielsen. On piercing sets of objects. In *Proc. 12th Annual Symposium on Computational Geometry (SoCG)*, pages 113–121, 1996. doi:10.1145/237218.237253.

36 Marvin Künnemann. A tight (non-combinatorial) conditional lower bound for Klee's measure problem in 3D. In *Proc. 63rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 555–566, 2022. doi:10.1109/FOCS54457.2022.00059.

37 Andrea Lincoln, Virginia Vassilevska Williams, and R. Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In *Proc. 29th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1236–1252, 2018. doi:10.1137/1.9781611975031.80.

38 Dániel Marx. Efficient approximation schemes for geometric problems? In *Proc. 13th Annual European Symposium of Algorithms (ESA)*, pages 448–459, 2005. doi:10.1007/11561071_41.

39 Dániel Marx and Michal Pilipczuk. Optimal parameterized algorithms for planar facility location problems using voronoi diagrams. In *Proc. 23rd Annual European Symposium on Algorithms (ESA)*, pages 865–877, 2015. doi:10.1007/978-3-662-48350-3_72.

40 Jirí Matoušek. Reporting points in halfspaces. *Comput. Geom.*, 2:169–186, 1992. doi:10.1016/0925-7721(92)90006-E.

41 Nimrod Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983.

42 Doron Nussbaum. Rectilinear p-piercing problems. In *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 316–323, 1997. doi:10.1145/258726.258828.

43 Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *Proc. 42nd ACM Symposium on Theory of Computing (STOC)*, pages 603–610, 2010. doi:10.1145/1806689.1806772.

44 Micha Sharir. A near-linear algorithm for the planar 2-center problem. *Discrete & Computational Geometry*, 18(2):125–134, 1997. Preliminary version in SoCG'96.

45 Micha Sharir and Emo Welzl. Rectilinear and polygonal p-piercing and p-center problems. In *Proc. 12th Annual Symposium on Computational Geometry (SoCG)*, pages 122–132, 1996. doi:10.1145/237218.237255.

46 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the ICM*, volume 3, pages 3431–3472. World Scientific, 2018. URL: https://people.csail.mit.edu/virgi/eccentri.pdf.

47 Virginia Vassilevska Williams and R. Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM*, 65(5):27:1–27:38, 2018. Preliminary version in FOCS'10. doi:10.1145/3186893.

48 Virginia Vassilevska Williams and Yinzhan Xu. Monochromatic triangles, triangle listing and APSP. In *Proc. 61st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 786–797, 2020. doi:10.1109/FOCS46700.2020.00078.

49 Haitao Wang. On the planar two-center problem and circular hulls. In *Proc. 36th International Symposium on Computational Geometry (SoCG)*, pages 68:1–68:14, 2020. doi:10.4230/LIPIcs.SoCG.2020.68.

50 Emo Welzl. Smallest enclosing disks (balls and ellipsoids). In *New Results and New Trends in Computer Science*, pages 359–370. Springer, 1991.

Ortho-Radial Drawing in Near-Linear Time

Yi-Jun Chang
National University of Singapore, Singapore

Abstract

An *orthogonal drawing* is an embedding of a plane graph into a grid. In a seminal work of Tamassia (SIAM Journal on Computing 1987), a simple combinatorial characterization of angle assignments that can be realized as bend-free orthogonal drawings was established, thereby allowing an orthogonal drawing to be described combinatorially by listing the angles of all corners. The characterization reduces the need to consider certain geometric aspects, such as edge lengths and vertex coordinates, and simplifies the task of graph drawing algorithm design.

Barth, Niedermann, Rutter, and Wolf (SoCG 2017) established an analogous combinatorial characterization for *ortho-radial drawings*, which are a generalization of orthogonal drawings to *cylindrical grids*. The proof of the characterization is existential and does not result in an efficient algorithm. Niedermann, Rutter, and Wolf (SoCG 2019) later addressed this issue by developing quadratic-time algorithms for both testing the realizability of a given angle assignment as an ortho-radial drawing without bends and constructing such a drawing.

In this paper, we improve the time complexity of these tasks to near-linear time. We establish a new characterization for ortho-radial drawings based on the concept of a *good sequence*. Using the new characterization, we design a simple greedy algorithm for constructing ortho-radial drawings.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Graph drawing, ortho-radial drawing, topology-shape-metric framework

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.35

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2305.00425

1 Introduction

A *plane graph* is a *planar graph* $G = (V, E)$ with a *combinatorial embedding* $\mathcal{E}$. The combinatorial embedding $\mathcal{E}$ fixes a circular ordering $\mathcal{E}(v)$ of the edges incident to each vertex $v \in V$, specifying the counter-clockwise ordering of these edges surrounding v in the drawing. An *orthogonal drawing* of a plane graph is a drawing of G such that each edge is drawn as a sequence of horizontal and vertical line segments. For example, see Figure 1 for an orthogonal drawing of K_4 with 4 bends. Alternatively, an orthogonal drawing of G can be seen as an embedding of G into a grid such that the edges of G correspond to internally disjoint paths in the grid. Orthogonal drawing is one of the most classical drawing styles studied in the field of graph drawing, and it has a wide range of applications, including VLSI circuit design [7, 40], architectural floor plan design [33], and network visualization [5, 22, 26, 30].

The topology-shape-metric framework. One of the most fundamental quality measures of orthogonal drawings is the number of *bends*. The *bend minimization* problem, which asks for an orthogonal drawing with the smallest number of bends, has been extensively studied over the past 40 years [14, 16, 17, 38, 39, 25]. In a seminal work, Tamassia [39] introduced the *topology-shape-metric* framework to tackle the bend minimization problem. Tamassia showed that an orthogonal drawing can be described combinatorially by an *orthogonal representation*, which consists of an assignment of an angle of degree in $\{90°, 180°, 270°, 360°\}$ to each corner and a designation of the *outer face*. Specifically, Tamassia [39] showed that an orthogonal representation can be realized as an orthogonal drawing with zero bends if and only if the following two conditions are satisfied:

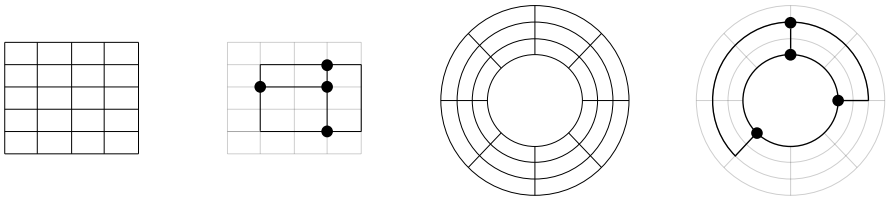


Figure 1 A grid, an orthogonal drawing, a cylindrical grid, and an ortho-radial drawing.

(O1) The sum of angles around each vertex is 360°.

(O2) The sum of angles around each face with k corners is $(k+2)\cdot 180°$ for the outer face and is $(k-2)\cdot 180°$ for the other faces.

An orthogonal representation is *valid* if it satisfies the above conditions (O1) and (O2). Given a valid orthogonal representation, an orthogonal drawing realizing the orthogonal representation can be computed in linear time [29, 39]. This result (shape → metric) allows us to reduce the task of finding a bend-minimized orthogonal drawing (topology → metric) to the conceptually much simpler task of finding a bend-minimized valid orthogonal representation (topology → shape). By focusing on orthogonal representations, we may neglect certain geometric aspects of graph drawing such as edge lengths and vertex coordinates, making the task of algorithm design easier. In particular, given a fixed combinatorial embedding, the task of finding a bend-minimized orthogonal representation can be easily reduced to the computation of a min-cost flow [39].

1.1 Ortho-radial drawing

Ortho-radial drawing is a natural generalization of orthogonal drawing to *cylindrical grids*, whose grid lines consist of concentric circles and straight lines emanating from the center of the circles. Formally, an ortho-radial drawing is defined as a planar embedding where each edge is drawn as a sequence of lines that are either a circular arc of some circle centered on the origin or a line segment of some straight line passing through the origin. We do not allow a vertex to be drawn on the origin, and we do not allow an edge to pass through the origin in the drawing. For example, see Figure 1 for an ortho-radial drawing of K_4 with two bends.

The study of ortho-radial drawing is motivated by its applications [4, 23, 42] in network visualization [41]. For example, ortho-radial drawing is naturally suitable for visualizing metro systems with radial routes and circle routes.

There are three types of faces in an ortho-radial drawing. The face that contains an unbounded region is called the *outer face*. The face that contains the origin is called the *central face*. The remaining faces are called *regular faces*. It is possible that the outer face and the central face are the same face.

Given a plane graph, an *ortho-radial representation* is defined as an assignment of an angle to each corner together with a designation of the central face and the outer face. Barth, Niedermann, Rutter, and Wolf [2] showed that an ortho-radial representation can be realized as an ortho-radial drawing with zero bends if the following three conditions are satisfied:

(R1) The sum of angles around each vertex is 360°.

(R2) The sum s of angles around each face F with k corners satisfies the following.

- $s = (k-2)\cdot 180°$ if F is a regular face.
- $s = k\cdot 180°$ if F is either the central face or the outer face, but not both.
- $s = (k+2)\cdot 180°$ if F is both the central face and the outer face.

(R3) There exists a choice of the *reference edge* $e^\star$ such that the ortho-radial representation does not contain a *strictly monotone cycle*.

Intuitively, this shows that the ortho-radial representations that can be realized as ortho-radial drawings with zero bends can be characterized similarly by examining the angle sum around each vertex and each face, with the additional requirement that the representation does not have a strictly monotone cycle.

The definition of a strictly monotone cycle is technical and depends on the choice of the reference edge $e^\star$, so we defer its formal definition to a subsequent section. The reference edge $e^\star$ is an edge in the contour of the outer face and is required to lie on the outermost circular arc used in an ortho-radial drawing. Informally, a strictly monotone cycle has a structure that is like a loop of ascending stairs or a loop of descending stairs, so a strictly monotone cycle cannot be drawn. The necessity of (R1)–(R3) is intuitive to see. The more challenging and interesting part of the proof in [2] is to show that these three conditions are actually sufficient.

1.2 Previous methods

The proof by Barth, Niedermann, Rutter, and Wolf [2] that conditions (R1)–(R3) are necessary and sufficient is only *existential* in that it does not yield efficient algorithms to check the validity of a given ortho-radial representation and to construct an ortho-radial drawing without bends realizing a given ortho-radial representation.

Checking (R1) and (R2) can be done in linear time in a straightforward manner. The difficult part is to design an efficient algorithm to check (R3). The most naive approach of examining all cycles costs exponential time. The subsequent work by Niedermann, Rutter, and Wolf [35] addressed this gap by showing an $O(n^2)$-time algorithm to decide whether a strictly monotone cycle exists for a given reference edge $e^\star$, where n is the number of vertices in the input graph. They also show an $O(n^2)$-time algorithm to construct an ortho-radial drawing without bends, for any given ortho-radial representation with a reference edge $e^\star$ that does not lead to a strictly monotone cycle.

Rectangulation. The idea behind the proof of Barth, Niedermann, Rutter, and Wolf [2] is a reduction to the easier case where each regular face is *rectangular*. For this case, they provided a proof that conditions (R1)–(R3) are necessary and sufficient, and they also provided an efficient drawing algorithm via a reduction to a flow computation given that (R1)–(R3) are satisfied. For any given ortho-radial representation with n vertices, it is possible to add $O(n)$ additional edges to turn it into an ortho-radial representation where each regular face is rectangular. A major difficulty in the proof of [2] is that they need to ensure that the addition of the edges preserves not only (R1) and (R2) but also (R3). The lack of an efficient algorithm to check whether (R3) is satisfied is precisely the reason that the proof of [2] does not immediately lead to a polynomial-time algorithm.

Quadratic-time algorithms. The above issue was addressed in a subsequent work by Niedermann, Rutter, and Wolf [35]. They provided an $O(n^2)$-time algorithm to find a strictly monotone cycle if one exists, given a fixed choice of the reference edge $e^\star$. This immediately leads to an $O(n^2)$-time algorithm to decide whether a given ortho-radial representation, with a fixed reference edge $e^\star$, admits an ortho-radial drawing. Moreover, combining this $O(n^2)$-time algorithm with the proof of [2] discussed above yields an $O(n^4)$-time drawing algorithm. The time complexity is due to the fact that $O(n)$ edge additions are needed for rectangulation, for each edge addition there are $O(n)$ candidate reference edges to consider, and to test the feasibility of each candidate edge they need to run the $O(n^2)$-time algorithm to test whether the edge addition creates a strictly monotone cycle.

The key idea behind the $O(n^2)$-time algorithm for finding a strictly monotone cycle is a structural theorem that if there is a strictly monotone cycle, then there is a unique outermost one which can be found by a *left-first* DFS starting from any edge in the outermost strictly monotone cycle. The DFS algorithm costs $O(n)$ time. Guessing an edge in the outermost monotone cycle adds an $O(n)$ factor overhead in the time complexity.

Using further structural insights on the augmentation process of [2], the time complexity of the above $O(n^4)$-time drawing algorithm can be lowered to $O(n^2)$ [35]. The reason for the quadratic time complexity is that for each of the $O(n)$ edge additions, a left-first DFS starting from the newly added edge is needed to test whether the addition of this edge creates a strictly monotone cycle.

1.3 Our new method

For both validity testing (checking whether a given angle assignment induces a strictly monotone cycle) and drawing (finding a geometric embedding realizing a given ortho-radial representation), the two algorithms in [35] naturally cost $O(n^2)$ time, as they both require performing left-first DFS $O(n)$ times.

In this paper, we present a new method for ortho-radial drawing that is not based on rectangulation and left-first DFS. We design a simple $O(n \log n)$-time greedy algorithm that simultaneously accomplishes both validity testing and drawing, for the case where the reference edge $e^\star$ is fixed. If a reference edge $e^\star$ is not fixed, our algorithm costs $O(n \log^2 n)$ time, where the extra $O(\log n)$ factor is due to a binary search over the set of candidates for the reference edge. At a high level, our algorithm tries to construct an ortho-radial drawing in a piece-by-piece manner. If at some point no progress can be made in that the current partial drawing cannot be further extended, then the algorithm can identify a strictly monotone cycle to certify the non-existence of a drawing.

Good sequences. The core of our method is the notion of a *good sequence*, which we briefly explain below. An ortho-radial representation satisfying (R1) and (R2), with a fixed reference edge $e^\star$, determines whether an edge e is a vertical edge (i.e., e is drawn as a segment of a straight line passing through the origin) or horizontal (i.e., e is drawn as a circular arc of some circle centered on the origin). Let E_{h} denote the set of horizontal edges, oriented in the clockwise direction, and let $\mathcal{S}_{\mathsf{h}}$ denote the set of connected components induced by E_{h}. Note that each component $S \in \mathcal{S}_{\mathsf{h}}$ is either a path or a cycle. The exact definition of a good sequence is technical, so we defer it to a subsequent section. Intuitively, a good sequence is an ordering of $\mathcal{S}_{\mathsf{h}} = (S_1, S_2, \ldots, S_k)$, where $k = |\mathcal{S}_{\mathsf{h}}|$, that allows us to design a simple linear-time greedy algorithm constructing an ortho-radial drawing in such a way that S_1 is drawn on the circle $r = k$, S_2 is drawn on the circle $r = k - 1$, and so on.

In general, a good sequence might not exist, even if the given ortho-radial representation admits an ortho-radial drawing. In such a case, we show that we may add *virtual edges* to transform the ortho-radial representation into one where a good sequence exists. We will design a greedy algorithm for adding virtual edges and constructing a good sequence. In each step, we add virtual vertical edges to the current graph and append a new element $S \in \mathcal{S}_{\mathsf{h}}$ to the end of our sequence. In case we are unable to find any suitable $S \in \mathcal{S}_{\mathsf{h}}$ to extend the sequence, we can extract a strictly monotone cycle to certify the non-existence of an ortho-radial drawing. A major difference between our method and the approach based on rectangulation in [2, 35] is that the cost for adding a new virtual edge is only $O(\log n)$ in our algorithm. As we will later see, in our algorithm, in order to identify new virtual edges to be added, we only need to do some simple local checks such as calculating the sum of angles, and there is no need to do a full left-first DFS to test whether a newly added edge creates a strictly monotone cycle.

Open questions. While we show a nearly linear-time algorithm for the (shape → metric)-step (i.e., from ortho-radial representations to ortho-radial drawings), essentially nothing is known about the (topology → shape)-step (from planar graphs to ortho-radial representations). While the task of finding a *bend-minimized orthogonal representation* of a given plane graph can be easily reduced to the computation of a minimum cost flow [39], such a reduction does not apply to ortho-radial representations, as network flows do not work well with the notion of strictly monotone cycles. It remains an open question as to whether a bend-minimized ortho-radial representation of a plane graph can be computed in polynomial time.

1.4 Related work

The bend minimization problem for orthogonal drawings for planar graphs of maximum degree 4 without a fixed combinatorial embedding is NP-hard [24, 25]. If the combinatorial embedding is fixed, the topology-shape-metric framework of Tamassia [39] reduces the bend minimization problem to a min-cost flow computation. The algorithm of Tamassia [39] costs $O(n^2 \log n)$ time. The time complexity was later improved to $O\left(n^{7/4}\sqrt{\log n}\right)$ [25] and then to $O\left(n^{3/2} \log n\right)$ [14]. A recent $O(n \operatorname{poly} \log n)$-time planar min-cost flow algorithm [20] implies that the bend minimization problem can be solved in $O(n \operatorname{poly} \log n)$ time if the combinatorial embedding is fixed.

If the combinatorial embedding is not fixed, the NP-hardness result of [24, 25] can be bypassed if the first bend on each edge does not incur any cost [9] or if we restrict ourselves to some special class of planar graphs. In particular, for planar graphs with maximum degree 3, it was shown that the bend-minimization can be solved in polynomial time [16]. After a series of improvements [13, 17, 19], we now know that a bend-minimized orthogonal drawing of a planar graph with maximum degree 3 can be computed in $O(n)$ time [17].

The topology-shape-metric framework [39] is not only useful in bend minimization, but it is also, implicitly or explicitly, behind the graph drawing algorithms for essentially all computational problems in orthogonal drawing and its variants, such as morphing orthogonal drawings [8], allowing vertices with degree greater than 4 [15, 31, 36], restricting the direction of edges [18, 21], drawing cluster graphs [10], and drawing dynamic graphs [11].

The study of ortho-radial drawing by Barth, Niedermann, Rutter, and Wolf [2, 35] extended the topology-shape-metric framework [39] to accommodate cylindrical grids. Before these works [2, 35], a combinatorial characterization of drawable ortho-radial representation is only known for paths, cycles, and theta graphs [28], and for the special case where the graph is 3-regular and each regular face in the ortho-radial representation is a rectangle [27].

1.5 Organization

In Section 2, we discuss the basic graph terminology used in this paper, review some results in previous works [2, 35], and state our main theorems. In Section 3, we give a technical overview of our proof. We conclude in Section 4 with discussions on possible future directions.

2 Preliminaries

Throughout the paper, let $G = (V, E)$ be a planar graph of maximum degree at most 4 with a fixed combinatorial embedding $\mathcal{E}$ in the sense that, for each vertex $v \in V$, a circular ordering $\mathcal{E}(v)$ of its incident edges is given to specify the counter-clockwise ordering of these edges surrounding v in a planar embedding. As we will discuss in the full version of the paper, we may assume that the input graph G is *simple* and *biconnected*. In this section, we introduce some basic graph terminology and review some results from the paper [3], which is a merge of the two papers [2, 35] on ortho-radial drawing.

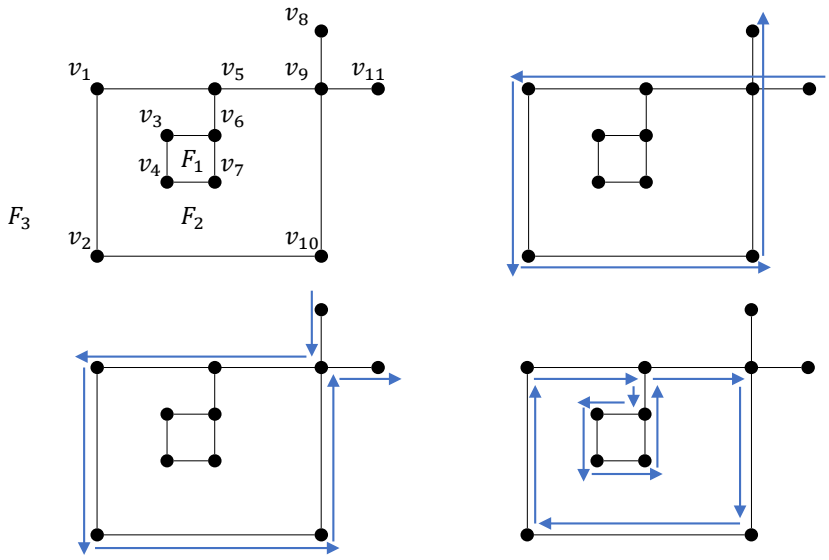


Figure 2 A non-crossing-free path, a crossing-free path, and a facial cycle.

Paths and cycles. Unless otherwise stated, all edges, paths, and cycles are assumed to be directed. We write $\overline{e}$, $\overline{P}$, and $\overline{C}$ to denote the *reversal* of an edge e, a path P, and a cycle C, respectively. We allow paths and cycles to have repeated vertices and edges. We say that a path or a cycle is *simple* if it does not have repeated vertices. Following [3], we say that a path or a cycle is *crossing-free* if it satisfies the following conditions:

- The path or the cycle does not contain repeated undirected edges.
- For each vertex v that appears multiple times in the path or the cycle, the ordering of the edges incident to v appearing in the path or the cycle respects the ordering of $\mathcal{E}(v)$ or its reversal.

Although a crossing-free path or a crossing-free cycle might touch a vertex multiple times, the path or the cycle never crosses itself. For any face F, we define the *facial cycle* C_F to be the clockwise traversal of its contour. In general, a facial cycle might not be a simple cycle as it can contain repeated edges. If we assume that G is biconnected, then each facial cycle of G must be a simple crossing-free cycle. See Figure 2 for an illustration of different types of paths and cycles. The path $(v_{11}, v_9, v_5, v_1, v_2, v_{10}, v_9, v_8)$ is not crossing-free as the path crosses itself at v_9. The path $(v_8, v_9, v_5, v_1, v_2, v_{10}, v_9, v_{11})$ is crossing-free since it respects the ordering $\mathcal{E}(v)$ for $v = v_9$. The cycle $C = (v_1, v_5, v_6, v_3, v_4, v_7, v_6, v_5, v_9, v_{10}, v_2)$ is the facial cycle of F_2. The cycle C is not a crossing-free cycle as it traverses the undirected edge $\{v_5, v_6\}$ twice, from opposite directions.

Ortho-radial representations and drawings. A *corner* is an ordered pair of undirected edges (e_1, e_2) incident to v such that e_2 immediately follows e_1 in the counter-clockwise circular ordering $\mathcal{E}(v)$. Given a planar graph $G = (V, E)$ with a fixed combinatorial embedding $\mathcal{E}$, an ortho-radial representation $\mathcal{R} = (\phi, F_\mathsf{c}, F_\mathsf{o})$ of G is defined by the following components:

- An assignment ϕ of an angle $a \in \{90°, 180°, 270°\}$ to each corner of G.
- A designation of a face of G as the central face F_c.
- A designation of a face of G as the outer face F_o.

For the special case where v has only one incident edge e, we view (e, e) as a 360° corner. This case does not occur if we consider biconnected graphs.

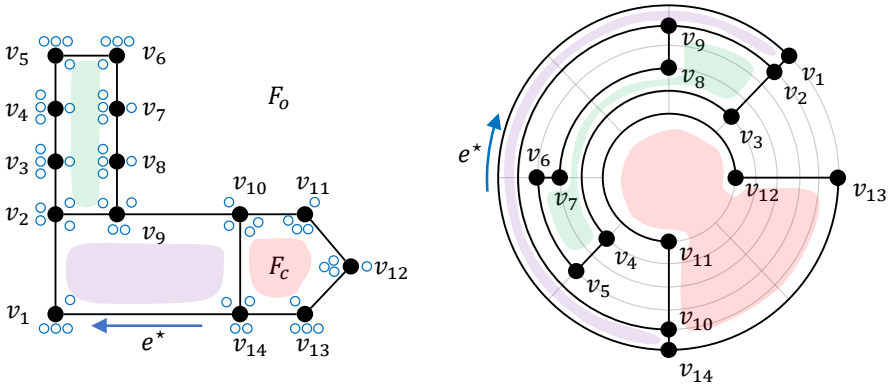


Figure 3 A drawing of an ortho-radial representation with a reference edge, where the small blue circles in the left figure denote the angles in the representation that are realized in the right figure.

An ortho-radial representation $\mathcal{R} = (\phi, F_c, F_o)$ is *drawable* if the representation can be realized as an ortho-radial drawing of G with zero bends such that the angle assignment ϕ is satisfied, the central face F_c contains the origin, the outer face F_o contains an unbounded region.

Recall that, by the definition of ortho-radial drawing, in an ortho-radial drawing with zero bends, each edge is either drawn as a line segment of a straight line passing the origin or drawn as a circular arc of a circle centered at the origin. We also consider the setting where the *reference edge* $e^\star$ is fixed. In this case, there is an additional requirement that the reference edge $e^\star$ has to lie on the outermost circular arc used in the drawing and follows the clockwise direction. If such a drawing exists, we say that $(\mathcal{R}, e^\star)$ is *drawable*. See Figure 3 for an example of a drawing of an ortho-radial representation $\mathcal{R}$ with the reference edge $e^\star = (v_{14}, v_5)$. In the figure, we use $\circ$, $\circ\circ$, and $\circ\circ\circ$ to indicate a 90°, a 180°, and a 270° angle assigned to a corner, respectively.

It was shown in [3] that $(\mathcal{R}, e^\star)$ is drawable if and only if the ortho-radial representation $\mathcal{R}$ satisfies (R1) and (R2) and the reference edge $e^\star$ does not lead to a strictly monotone cycle. Since it is straightforward to test whether (R1) and (R2) are satisfied in linear time, from now on, unless otherwise stated, we assume that (R1) and (R2) are satisfied for the ortho-radial representation $\mathcal{R}$ under consideration.

Combinatorial rotations. Consider a 2-length path $P = (u, v, w)$ that passes through v such that $u \neq w$. Given the angle assignment ϕ, P is either a 90° left turn, a straight line, or a 90° right turn. We define the *combinatorial rotation* of P as follows.

$$\mathsf{rotation}(P) = \begin{cases} -1, & P \text{ is a } 90^\circ \text{ left turn,} \\ 0, & P \text{ is a straight line,} \\ 1, & P \text{ is a } 90^\circ \text{ right turn.} \end{cases}$$

More formally, let $S = (e_1, \ldots, e_k)$ be the contiguous subsequence of edges starting from $e_1 = \{u, v\}$ and ending at $e_k = \{v, w\}$ in the circular ordering $\mathcal{E}(v)$ of the undirected edges incident to v. Then $\sum_{j=1}^{k-1} \phi(e_j, e_{j+1}) - 180^\circ$ equals the degree of the turn of P at the intermediate vertex v, so the combinatorial rotation of P is $\mathsf{rotation}(P) = \left(\sum_{j=1}^{k-1} \phi(e_j, e_{j+1}) - 180^\circ\right) / 90^\circ$.

For the special case where $u = w$, the rotation of $P = (u, v, u)$ can be a 180° left turn, in which case $\mathsf{rotation}(P) = -2$, or a 180° right turn, in which case $\mathsf{rotation}(P) = 2$. For example, consider the directed edge $e = (u, v)$ where P first goes from u to v along the right side of e and then goes from v back to u along the left side of e. Then P is considered a 180° left turn, and similarly, $\overline{P}$ is considered a 180° right turn. In particular, if $P = (u, v, u)$ is a subpath of a facial cycle C, then P is always considered as a 180° left turn, and so $\mathsf{rotation}(P) = -2$.

For a crossing-free path P of length more than 2, we define $\mathsf{rotation}(P)$ as the sum of the combinatorial rotations of all 2-length subpaths of P. Similarly, for a cycle C of length more than 2, we define $\mathsf{rotation}(C)$ as the sum of the combinatorial rotations of all 2-length subpaths of C. Same as [2, 35], based on this notion, we may restate condition (R2).

(R2′) For each face F, the combinatorial rotation of its facial cycle C_F satisfies the following:

$$\mathsf{rotation}(C_F) = \begin{cases} 4, & F \text{ is a regular face,} \\ 0, & F \text{ is either the central face or the outer face, but not both,} \\ -4, & F \text{ is both the central face and the outer face.} \end{cases}$$

For example, consider the ortho-radial representation shown in Figure 3. The path $P = (v_{10}, v_{11}, v_{12}, v_{13}, v_{14})$ has $\mathsf{rotation}(P) = -1$ since it makes two 90° left turns and one 90° right turn. The cycle $C = (v_{10}, v_{11}, v_{12}, v_{13}, v_{14})$ is the facial cycle of the central face, and it has $\mathsf{rotation}(C) = 0$.

We briefly explain the equivalence between the new and the old definitions of (R2). If F is a regular face with k corners, then in the original definition of (R2), it is required that the sum s of angles around F is $s = (k-2) \cdot 180°$. Since the facial cycle C_F traverses the contour of F in the clockwise direction, the number of 90° right turns minus the number of 90° left turns must be exactly 4. Therefore, $s = (k-2) \cdot 180°$ is the same as $\mathsf{rotation}(C_F) = 4$, as each 90° right turn contributes 1 and each 90° left turn contributes -1 in the calculation of $\mathsf{rotation}(C_F)$.

Interior and exterior regions of a cycle. Any cycle C partitions the remaining graph into two parts. If C is a facial cycle, then one part is empty. The direction of C is clockwise with respect to one of the two parts. The part with respect to which C is clockwise, together with C itself, is called the *interior* of C. Similarly, the part with respect to which C is counter-clockwise, together with C itself, is called the *exterior* of C. In particular, if a vertex v lies in the interior of C, then v must be in the exterior of $\overline{C}$.

This above definition is consistent with the notion of facial cycle in that any face F is in the interior of its facial cycle C_F. Depending on the context, the interior or the exterior of a cycle can be viewed as a subgraph, a set of vertices, a set of edges, or a set of faces. For example, consider the cycle $C = (v_1, v_2, v_{10}, v_9, v_5)$ of the plane graph shown in Figure 2. The interior of C is the subgraph induced by v_8, v_{11}, and all vertices in C. The exterior of C is the subgraph induced by v_3, v_4, v_6, v_7, and all vertices in C. The cycle C partitions the faces into two parts: The interior of C contains F_3, and the exterior of C contains F_1 and F_2.

Let C be a simple cycle oriented in such a way that the outer face F_{o} lies in its exterior. Following [3], we say that C is *essential* if the central face F_{c} is in the interior of C. Otherwise we say that C is *non-essential*. The following lemma was proved in [3].

▶ **Lemma 1** ([3]). *Let C be a simple cycle oriented in such a way that the outer face F_o lies in its exterior, then the combinatorial rotation of C satisfies the following:*

$$\mathsf{rotation}(C) = \begin{cases} 4, & C \text{ is an essential cycle,} \\ 0, & C \text{ is a non-essential cycle.} \end{cases}$$

In the above lemma, we implicitly assume that (R1) and (R2) are satisfied. The intuition behind the lemma is that an essential cycle behaves like the facial cycle of the outer face or the central face, and a non-essential cycle behaves like the facial cycle of a regular face.

Subgraphs. When we take a subgraph H of G, the combinatorial embedding, the angle assignment, the central face, and the outer face of H are inherited from G naturally. For example, suppose that $\mathcal{E}(v) = (e_1, e_2, e_3)$ with $\phi(e_1, e_2) = 90°$, $\phi(e_2, e_3) = 90°$, and $\phi(e_3, e_1) = 180°$ in G. Suppose that v is incident only to two edges e_1 and e_2 in H, then the angle assignment ϕ_H for the two corners surrounding v in H will be $\phi_H(e_1, e_2) = 90°$ and $\phi_H(e_2, e_1) = 270°$.

Each face of G is contained in exactly one face of H, A face in H can contain multiples faces of G. A face of H is said to be the central face if it contains the central face of G. Similarly, A face of H is said to be the outer face if it contains the outer face of G.

For example, consider the subgraph H induced by $\{v_2, v_3, \ldots, v_9\}$ in the ortho-radial representation in Figure 3. In H, v_9 is incident to only two edges $e_1 = \{v_8, v_9\}$ and $e_2 = \{v_2, v_9\}$, and the angle assignment ϕ_H for the two corners surrounding v_9 in H are $\phi_H(e_1, e_2) = 90°$ and $\phi_H(e_2, e_1) = 270°$. The outer face and the central face of H are the same.

Defining direction via reference paths. Following [3], for any two edges $e = (u, v)$ and $e' = (x, y)$, we say that a crossing-free path P is a reference path for e and e' if P starts at u or v and ends at x or y such that P does not contain any of the edges in $\{e, \overline{e}, e', \overline{e'}\}$. Given a reference path P for $e = (u, v)$ and $e' = (x, y)$, we define the *combinatorial direction* of e' with respect to e and P as follows.

$$\mathsf{direction}(e, P, e') = \begin{cases} \mathsf{rotation}(e \circ P \circ e'), & P \text{ starts at } v \text{ and ends at } x, \\ \mathsf{rotation}(\overline{e} \circ P \circ e') + 2, & P \text{ starts at } u \text{ and ends at } x, \\ \mathsf{rotation}(e \circ P \circ \overline{e'}) - 2, & P \text{ starts at } v \text{ and ends at } y, \\ \mathsf{rotation}(\overline{e} \circ P \circ \overline{e'}), & P \text{ starts at } u \text{ and ends at } y. \end{cases}$$

Here $P \circ Q$ denotes the concatenation of the paths P and Q. An edge e is interpreted as a 1-length path. In the definition of $\mathsf{direction}(e, P, e')$, we allow the possibility that a reference path P consists of a single vertex. If $v = x$ and $u \neq w$, then we may choose P to be the 0-length path consisting of a single vertex $v = x$, in which case $\mathsf{direction}(e, P, e')$ is simply the combinatorial rotation of the 2-length path (u, v, y). We do not consider the cases where $e = e'$ or $e = \overline{e'}$.

Consider the reference edge $e = (v_{14}, v_1)$ in the ortho-radial representation of Figure 3. We measure the direction of $e' = (v_8, v_9)$ from e with different choices of the reference path P. If $P = (v_1, v_2, v_9)$, then $\mathsf{direction}(e, P, e') = \mathsf{rotation}(e \circ P \circ \overline{e'}) - 2 = -1$. If $P = (v_{14}, v_{10}, v_9)$, then we also have $\mathsf{direction}(e, P, e') = \mathsf{rotation}(\overline{e} \circ P \circ \overline{e'}) = -1$. If we select $P = (v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8)$, then we get a different value of $\mathsf{direction}(e, P, e') = \mathsf{rotation}(e \circ P \circ e') = 3$. As we will later discuss, $\mathsf{direction}(e, P, e') \bmod 4$ is invariant under the choice of P.

In the definition of $\mathsf{direction}(e, P, e')$, the additive $+2$ in $\mathsf{rotation}(\overline{e} \circ P \circ e') + 2$ is due to the fact that the actual path that we intend to consider is $e \circ \overline{e} \circ P \circ e'$, where we make a $180°$ right turn in $e \circ \overline{e}$, which contributes $+2$ in the calculation of the combinatorial rotation. Similarly, the additive -2 in $\mathsf{rotation}(e \circ P \circ \overline{e'}) - 2$ is due to the fact that the actual path that we intend to consider is $e \circ P \circ \overline{e'} \circ e'$, where we make a $180°$ left turn in $\overline{e'} \circ e'$. There is no additive term in $\mathsf{rotation}(\overline{e} \circ P \circ \overline{e'})$ because of the cancellation of the $180°$ right turn $e \circ \overline{e}$ and the $180°$ left turn $\overline{e'} \circ e'$. The reason why $e \circ \overline{e}$ has to be a right turn and $\overline{e'} \circ e'$ has to be a left turn will be explained later.

See Figure 4 for an example of the calculation of an edge direction. The direction of $e = (u_1, u_2)$ with respect to $e^\star = (v_1, v_2)$ and the reference path $P = (v_1, v_5, v_4, u_1)$ can be calculated by $\mathsf{rotation}(\overline{e^\star} \circ P \circ e') + 2 = 1$ according to the formula above, where the additive $+2$ is due to the $180°$ right turn at $e^\star \circ \overline{e^\star}$.

Edge directions. Imagining that the origin is the south pole, in an ortho-radial drawing with zero bends, each edge e is either drawn in one of the following four directions:

- e points towards the *north* direction if e is drawn as a line segment of a straight line passing the origin, where e is directed away from the origin.
- e points towards the *south* direction if e is drawn as a line segment of a straight line passing the origin, where e is directed towards the origin.
- e points towards the *east* direction if e is drawn as a circular arc of a circle centered at the origin in the clockwise direction.
- e points towards the *west* direction if e is drawn as a circular arc of a circle centered at the origin in the counter-clockwise direction.

We say that e is a *vertical* edge if e points towards north or south. Otherwise, we say that e is a *horizontal* edge. We argue that as long as (R1) and (R2) are satisfied, the direction of any edge e is uniquely determined by the ortho-radial representation.

For the reference edge $e^\star$, it is required that $e^\star$ points east, and so $\overline{e^\star}$ points west. Consider any edge e that is neither $e^\star$ nor $\overline{e^\star}$. It is clear that the value of $\mathsf{direction}(e^\star, P, e)$ determines the direction of e in that the direction of e is forced to be east, south, west, or north if $\mathsf{direction}(e^\star, P, e) \bmod 4$ equals 0, 1, 2, or 3, respectively. For example, in the ortho-radial representation of Figure 3, the edge $e' = (v_8, v_9)$ is a vertical edge in the north direction, as we have calculated that $\mathsf{direction}(e^\star, P, e') \bmod 4 = 3$.

▶ **Lemma 2** ([3]). *For any two edges e and e', the value of $\mathsf{direction}(e, P, e') \bmod 4$ is invariant under the choice of the reference path P.*

The above lemma shows that $\mathsf{direction}(e^\star, P, e) \bmod 4$ is invariant under the choice of the reference path P, so the direction of each edge in an ortho-radial representation is well defined, even for the case that $(\mathcal{R}, e^\star)$ might not be drawable. Given the reference edge $e^\star$, we let E_h denote the set of all horizontal edges in the east direction, and let E_v denote the set of all vertical edges in the north direction.

Horizontal segments. We require that in a drawing of $(\mathcal{R}, e^\star)$, the reference edge $e^\star$ lies on the outermost circular arc used in the drawing, so not every edge in $\overline{C_{F_\circ}}$ is eligible to be a reference edge. To determine whether an edge $e \in \overline{C_{F_\circ}}$ is eligible to be a reference edge, we need to introduce some terminology.

Given the reference edge $e^\star$, the set E_v of vertical edges in the north direction and the set E_h of horizontal edges in the east direction are fixed. Let $\mathcal{S}_\mathsf{h}$ denote the set of connected components induced by E_h. Each component $S \in \mathcal{S}_\mathsf{h}$ is either a path or a cycle, and so in any drawing of $\mathcal{R}$, there is a circle C centered at the origin such that S must be drawn as C or a circular arc of C. We call each component $S \in \mathcal{S}_\mathsf{h}$ a *horizontal segment*.

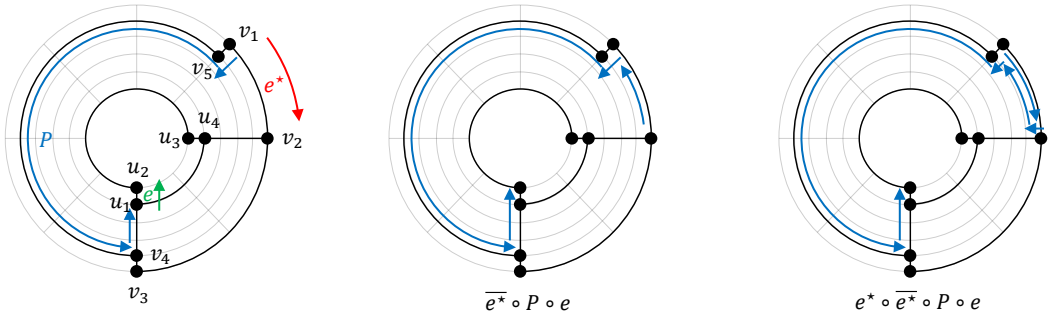


Figure 4 The calculation of $\mathsf{direction}(e^\star, P, e)$.

Each horizontal segment $S \in \mathcal{S}_\mathsf{h}$ is written as a sequence of vertices $S = (v_1, v_2, \ldots, v_s)$, where s is the number of vertices in S, such that $(v_i, v_{i+1}) \in E_\mathsf{h}$ for each $1 \leq i < s$. If S is a cycle, then we additionally have $(v_s, v_1) \in E_\mathsf{h}$, so $S = (v_1, v_2, \ldots, v_s)$ is a circular order. When S is a cycle, we use modular arithmetic on the indices so that $v_{s+1} = v_1$. We write $\mathcal{N}_\mathsf{north}(S)$ to denote the set of vertical edges $e = (x, y) \in E_\mathsf{v}$ such that $x \in S$. Similarly, $\mathcal{N}_\mathsf{south}(S)$ is the set of vertical edges $e = (x, y) \in E_\mathsf{v}$ such that $y \in S$. We assume that the edges in $\mathcal{N}_\mathsf{north}(S)$ and $\mathcal{N}_\mathsf{south}(S)$ are ordered according to the indices of their endpoints in S. The ordering is sequential if S is a path and is circular if S is a cycle. Consider the ortho-radial representation $\mathcal{R}$ given in Figure 3 as an example. The horizontal segment $S = (v_{10}, v_9, v_2)$ has $\mathcal{N}_\mathsf{south}(S) = ((v_{11}, v_{10}), (v_8, v_9), (v_3, v_2))$ and $\mathcal{N}_\mathsf{north}(S) = ((v_{10}, v_{14}), (v_2, v_1))$.

Observe that $\mathcal{N}_\mathsf{north}(S) = \emptyset$ for the horizontal segment $S \in \mathcal{S}_\mathsf{h}$ that contains $e^\star$ is a necessary condition that a drawing of $\mathcal{R}$ where $e^\star$ lies on the outermost circular arc exists. This condition can easily be checked in linear time.

Spirality. Intuitively, $\mathsf{direction}(e, P, e')$ quantifies the degree of *spirality* of e' with respect to e and P. Unfortunately, Lemma 2 does not hold if we replace $\mathsf{direction}(e, P, e') \bmod 4$ with $\mathsf{direction}(e, P, e')$. A crucial observation made in [3] is that such a replacement is possible if we add some restrictions about the positions of e, e', and P. See the following lemma.

▶ **Lemma 3** ([3]). *Let C and C' be essential cycles such that C' lies in the interior of C. Let e be an edge on C. Let e' be an edge on C'. The value of $\mathsf{direction}(e, P, e')$ is invariant under the choice of the reference path P, over all paths P in the interior of C and in the exterior of C'.*

Recall that we require a reference path to be crossing-free. This requirement is crucial in the above lemma. If we allow P to be a general path that is not crossing-free, then we may choose P in such a way that P repeatedly traverses a *non-essential* cycle many times, so that $\mathsf{direction}(e, P, e')$ can be made arbitrarily large and arbitrarily small.

Setting $e = e^\star$ and $C = \overline{C_{F_\mathsf{o}}}$ in the above lemma, we infer that $\mathsf{direction}(e^\star, P, e')$ is determined once we fix an essential cycle C' that contains e' and only consider reference paths P that lie in the exterior of C'. The condition for the lemma is satisfied because $\overline{C_{F_\mathsf{o}}}$ is the outermost essential cycle in that all other essential cycles are in the interior of $\overline{C_{F_\mathsf{o}}}$. The reason why we set $C = \overline{C_{F_\mathsf{o}}}$ and not $C = C_{F_\mathsf{o}}$ is that F_o has to be in the exterior of C. Note that the assumption that G is biconnected ensures that each facial cycle is simple.

Let C be an essential cycle and let e be an edge in C. In view of the above, following [3], we define the *edge label* $\ell_C(e)$ of e with respect to C as the value of $\mathsf{direction}(e^\star, P, e)$, for any choice of reference path P in the exterior of C'. For the special case that $e = e^\star$ and $C = \overline{C_{F_\mathsf{o}}}$,

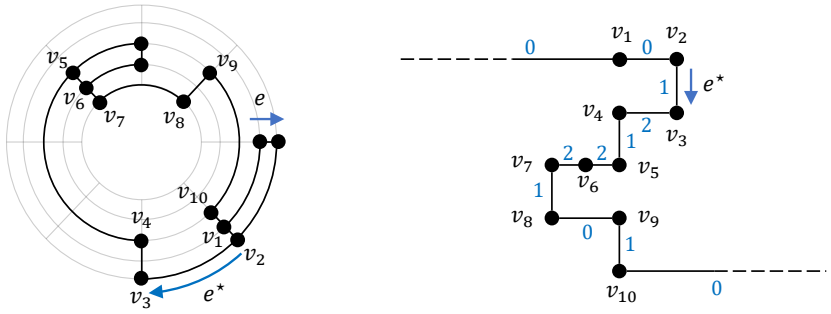


Figure 5 Changing the reference edge to e leads to a strictly monotone cycle.

we let $\ell_C(e) = 0$. Intuitively, the value $\ell_C(e)$ quantifies the degree of spirality of e from $e^\star$ if we restrict ourselves to the exterior of C. Consider the edge $e = (u_1, u_2)$ in the essential cycle $C = (u_1, u_2, u_3, u_4)$ in Figure 4 as an example. We have $\ell_C(e) = \mathsf{direction}(e^\star, P, e) = 1$, since the reference path $P = (v_1, v_5, v_4, u_1)$ lies in exterior of C.

We briefly explain the formula of $\mathsf{direction}(e, P, e')$: As discussed earlier, in the definition of $\mathsf{direction}(e, P, e')$, the additive $+2$ in $\mathsf{rotation}(\overline{e} \circ P \circ e') + 2$ is due to the fact that the actual path that we want to consider is $e \circ \overline{e} \circ P \circ e'$, where we make a 180° right turn in $e \circ \overline{e}$. The reason why $e \circ \overline{e}$ has to be a right turn is because of the scenario considered in Lemma 3, where e is an edge in C. To ensure that we stay in the interior of C in the traversal from e to e' via the path $e \circ \overline{e} \circ P \circ e'$, the 180° turn of $e \circ \overline{e}$ has to be a right turn. The remaining part of the formula of $\mathsf{direction}(e, P, e')$ can be explained similarly.

Monotone cycles. We are now ready to define the notion of strictly monotone cycles used in (R3). We say that an essential cycle C is *monotone* if all its edge labels $\ell_C(e)$ are non-negative or all its edge labels $\ell_C(e)$ are non-positive. Let C be an essential cycle that is monotone. If C contains at least one positive edge label, then we say that C is *increasing*. If C contains at least one negative edge label, then we say that C is *decreasing*. We say that C is *strictly monotone* if C is either decreasing or increasing but not both.

Intuitively, an increasing cycle is like a loop of descending stairs, and a decreasing cycle is like a loop of ascending stairs, so they are not drawable. It was proved in [3] that $(\mathcal{R}, e^\star)$ is drawable if and only if it does not contain a strictly monotone cycle. Recall again that, throughout the paper, unless otherwise stated, we assume that the given ortho-radial representation already satisfies (R1) and (R2).

▶ **Lemma 4** ([3]). *An ortho-radial representation $\mathcal{R}$, with a fixed reference edge $e^\star$ such that $\mathcal{N}_{\mathsf{north}}(S) = \emptyset$ for the horizontal segment $S \in \mathcal{S}_{\mathsf{h}}$ that contains $e^\star$, is drawable if and only if it does not contain a strictly monotone cycle.*

Consider Figure 5 as an example. The ortho-radial representation $\mathcal{R}$ is drawable with the reference edge $e^\star$. If we change the reference edge to e, then $(\mathcal{R}, e)$ become undrawable, as the essential cycle $C = (v_1, v_2, \ldots, v_{10})$ is increasing. With respect to the reference edge e, all the edge labels on the cycle C are non-negative, with some of them being positive. We are ready to state our main results.

▶ **Theorem 5.** *There is an $O(n \log n)$-time algorithm $\mathcal{A}$ that outputs either a drawing of $(\mathcal{R}, e^\star)$ or a strictly monotone cycle of $(\mathcal{R}, e^\star)$, for any given ortho-radial representation $\mathcal{R}$ of an n-vertex biconnected simple graph, with a fixed reference edge $e^\star$ such that $\mathcal{N}_{\mathsf{north}}(S) = \emptyset$ for the horizontal segment $S \in \mathcal{S}_{\mathsf{h}}$ that contains $e^\star$.*

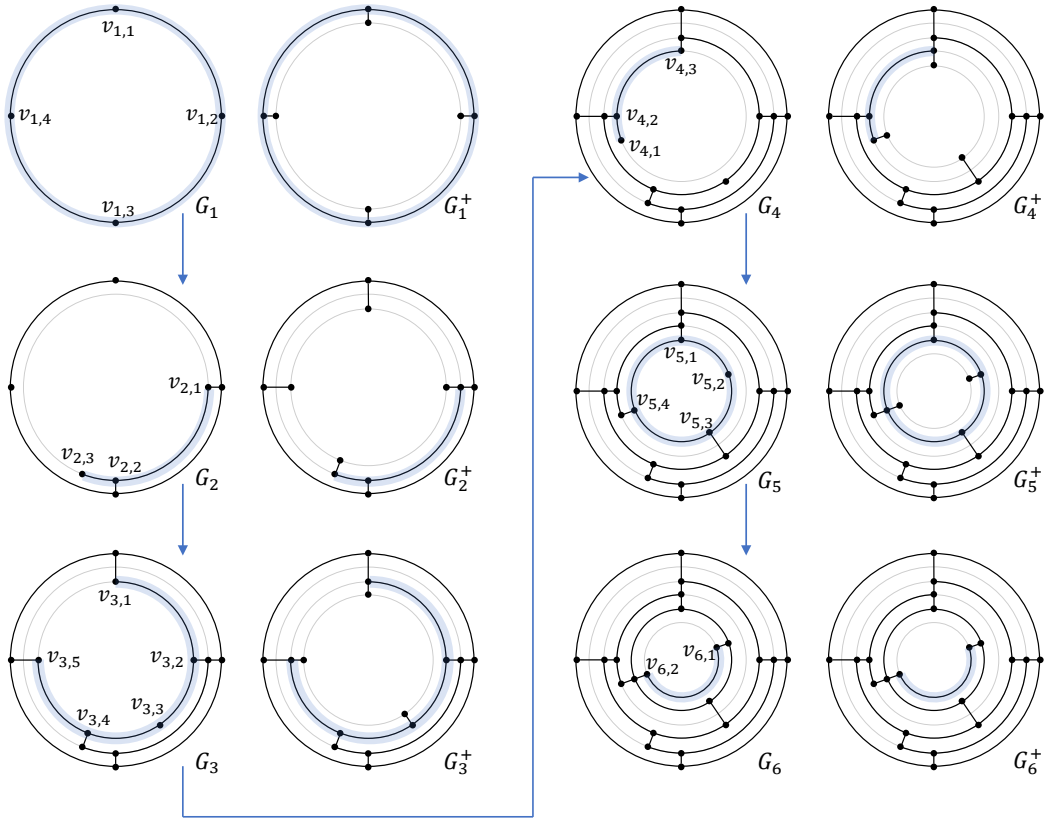


Figure 6 Constructing a good drawing for a good sequence.

The above theorem improves the previous algorithm of [35] which costs $O(n^2)$ time. If the output of $\mathcal{A}$ is a strictly monotone cycle, then the cycle certifies the non-existence of a drawing, by Lemma 4. We also extend the above theorem to the case where the reference edge is not fixed.

▶ **Theorem 6.** *There is an $O(n \log^2 n)$-time algorithm $\mathcal{A}$ that decides whether an ortho-radial representation $\mathcal{R}$ of an n-vertex biconnected simple graph is drawable. If $\mathcal{R}$ is drawable, then $\mathcal{A}$ also computes a drawing of $\mathcal{R}$.*

The proofs of Theorems 5 and 6 are left to the full version of the paper.

3 Technical overview

Let $A = (S_1, S_2, \ldots, S_k)$ be any sequence of k horizontal segments. We consider the following terminology for each $1 \leq i \leq k$, where k is the length of the sequence A.

- Let G_i be the subgraph of G induced by the horizontal edges in $S_1, S_2, \ldots, S_i$ and the set of all vertical edges whose both endpoints are in $S_1, S_2, \ldots, S_i$. Let F_i be the central face of G_i, and let C_i be the facial cycle of F_i.
- We extend the notion $\mathcal{N}_{\mathsf{south}}(S)$ to a sequence of horizontal segments: $\mathcal{N}_{\mathsf{south}}(S_1, S_2, \ldots, S_i)$ is defined as the set of all vertical edges $e = (x, y) \in E_{\mathsf{v}}$ such that $y \in C_i$ and $x \notin C_i$.
- Let G_i^+ be the subgraph of G induced by all the edges in G_i together with the edge set $\mathcal{N}_{\mathsf{south}}(S_1, S_2, \ldots, S_i)$. Let F_i^+ be the central face of G_i^+, and let C_i^+ be the facial cycle of F_i^+.

For each vertical edge $e = (x, y) \in \mathcal{N}_{\mathsf{south}}(S_1, S_2, \ldots, S_i)$, the south endpoint x appears exactly once in C_i^+. We circularly order the edges $e = (x, y) \in \mathcal{N}_{\mathsf{south}}(S_1, S_2, \ldots, S_i)$ according to the position of the south endpoint x in the circular ordering of C_i^+. Take the graph $G = G_6$ in Figure 6 as an example. In this graph, there are 6 horizontal segments, shaded in Figure 6:

$$S_1 = (v_{1,1}, v_{1,2}, v_{1,3}, v_{1,4}), \quad S_2 = (v_{2,1}, v_{2,2}, v_{2,3}), \quad S_3 = (v_{3,1}, v_{3,2}, v_{3,3}, v_{3,4}, v_{3,5}),$$
$$S_4 = (v_{4,1}, v_{4,2}, v_{4,3}), \quad S_5 = (v_{5,1}, v_{5,2}, v_{5,3}, v_{5,4}, v_{5,5}), \quad S_6 = (v_{6,1}, v_{6,2}).$$

With respect to the sequence $A = (S_1, S_2, \ldots, S_6)$, Figure 6 shows the graphs G_i and G_i^+, for all $1 \le i \le 6$. For example, for $i = 2$, we have:

$$\mathcal{N}_{\mathsf{south}}(S_1, S_2) = ((v_{3,1}, v_{1,1})(v_{3,2}, v_{2,1}), (v_{3,4}, v_{2,3}), (v_{3,5}, v_{1,4})),$$
$$\mathcal{N}_{\mathsf{north}}(S_2) = ((v_{2,1}, v_{1,2}), (v_{2,2}, v_{1,3}))$$
$$C_2 = (v_{1,1}, v_{1,2}, v_{2,1}, v_{2,2}, v_{2,3}, v_{2,2}, v_{1,3}, v_{1,4}),$$
$$C_2^+ = (v_{1,1}, v_{3,1}, v_{1,1}, v_{1,2}, v_{2,1}, v_{3,2}, v_{2,1}, v_{2,2}, v_{2,3}, v_{3,4}, v_{2,3}, v_{2,2}, v_{1,3}, v_{1,4}, v_{3,5}, v_{1,4}).$$

Here $\mathcal{N}_{\mathsf{south}}(S_1, S_2)$, C_2, and C_2^+ are circular orderings, and $\mathcal{N}_{\mathsf{north}}(S_2)$ is a sequential ordering, as S_2 is a path.

Good sequences. We say that a sequence of horizontal segments $A = (S_1, S_2, \ldots, S_k)$ is *good* if A satisfies the following conditions.

(S1) S_1 is the reversal of the facial cycle of the outer face F_o, i.e., $S_1 = \overline{C_{F_o}}$.

(S2) For each $1 < i \le k$, $\mathcal{N}_{\mathsf{north}}(S_i)$ satisfies the following requirements.

- $\mathcal{N}_{\mathsf{north}}(S_i) \neq \emptyset$.
- If S_i is a path, then $\mathcal{N}_{\mathsf{north}}(S_i)$ is a contiguous subsequence of $\mathcal{N}_{\mathsf{south}}(S_1, S_2, \ldots, S_{i-1})$.
- If S_i is a cycle, then $\mathcal{N}_{\mathsf{north}}(S_i) = \mathcal{N}_{\mathsf{south}}(S_1, S_2, \ldots, S_{i-1})$.

Clearly, if $A = (S_1, S_2, \ldots, S_k)$ is good, then $(S_1, S_2, \ldots, S_i)$ is also good for each $1 \le i < k$. In general, a good sequence that covers the set of all horizontal segments might not exist for a given $(\mathcal{R}, e^\star)$. In particular, in order to satisfy (S1), it is necessary that the cycle $\overline{C_{F_o}}$ is a horizontal segment. The sequence $A = (S_1, S_2, \ldots, S_6)$ shown in Figure 6 is a good sequence.

If $A = (S_1, S_2, \ldots, S_k)$ is good, then we can find a drawing of G_k in linear time by fixing the drawing of $S_1, S_2, \ldots, S_k$ sequentially, as the definition of a good sequence allows us to safely place S_i below $S_1, S_2, \ldots, S_{i-1}$ and above $S_{i+1}, S_{i+2}, \ldots, S_k$. The following lemma is proved formally in the full version of the paper.

▶ **Lemma 7.** *For a given good sequence $A = (S_1, S_2, \ldots, S_k)$, an ortho-radial drawing of G_k without bends can be constructed in time $O\left(\sum_{i=1}^{k} |S_i|\right)$.*

See Figure 6 for an example of a drawing of G_k produced by the algorithm of Lemma 7.

Constructing a good sequence. In order to use Lemma 7 to compute an ortho-radial drawing of $(\mathcal{R}, e^\star)$, we need to find a good sequence $A = (S_1, S_2, \ldots, S_k)$ with $G_k = G$. However, such a good sequence might not exist even if $(\mathcal{R}, e^\star)$ is drawable. We will show that as long as $(\mathcal{R}, e^\star)$ is drawable, we can always add some *virtual edges* to the graph so that such a good sequence exists and can be computed efficiently. The first step of the algorithm is a simple preprocessing step to ensure the following two properties:

- The facial cycle of the outer face is a horizontal segment.
- Each vertex is incident to a horizontal segment.

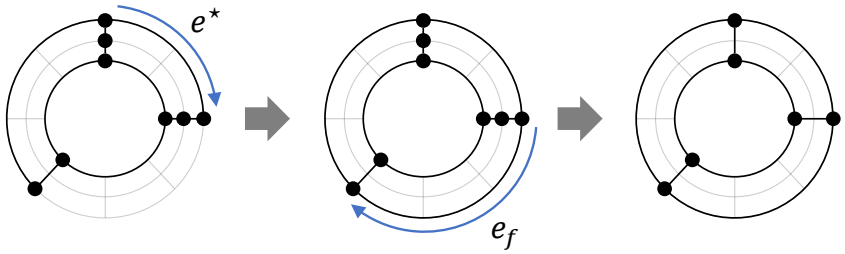


Figure 7 The preprocessing steps.

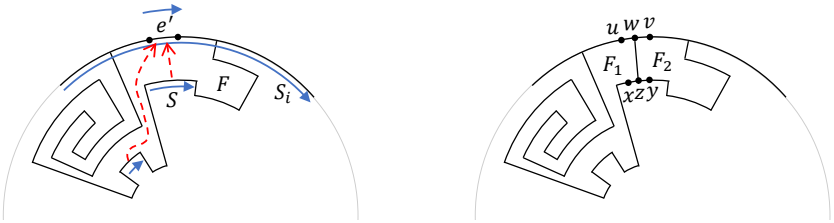


Figure 8 Adding a virtual vertical edge in a regular face.

See Figure 7 for the algorithm of the preprocessing step. The addition of the edge e_f ensures that $\overline{C_{F_o}}$ is a horizontal segment. To ensure that each vertex is on a horizontal segment, some degree-2 vertices are removed by smoothing.

The above two properties alone are not sufficient to guarantee the existence of a good sequence $A = (S_1, S_2, \ldots, S_k)$ with $G_k = G$, as there could be horizontal segment S such that $\mathcal{N}_{\mathsf{north}}(S) = \emptyset$ and $S \neq \overline{C_{F_o}}$. Such a horizontal segment S can never be added to a good sequence, as the definition of a good sequence requires all horizontal segments in the sequence to be non-empty. To deal with this issue, we consider the following eligibility criterion for adding a *virtual* vertical edge incident to such a horizontal segment S:

- Let $A = (S_1, S_2, \ldots, S_k)$ be the current good sequence. Let $S \notin A$ be a horizontal segment such that $\mathcal{N}_{\mathsf{north}}(S) = \emptyset$ and $S \neq \overline{C_{F_o}}$. Let F be the face such that $\overline{S}$ is a subpath of C_F. We say that S is *eligible* for adding a virtual edge if there exists an edge $e' \in C_F$ with $e' \in S_i$ for some $1 \leq i \leq k$ such that either $\mathsf{rotation}(e' \circ \cdots \circ \overline{S}) = 2$ or $\mathsf{rotation}(\overline{S} \circ \cdots \circ e') = 2$ along the cycle C_F.

See Figure 8 for an illustration of adding a virtual edge. In the figure, there are two horizontal segments along the contour of F that are eligible for adding a virtual edge due to $e' \in S_i$. The rotation criterion for eligibility is to ensure that the new faces created due to the virtual edge still satisfy (R2). The condition $\mathcal{N}_{\mathsf{north}}(S) = \emptyset$ ensures that immediately after adding the virtual edge, we may append S to the end of the sequence A.

Our algorithm to construct a good sequence is a simple greedy algorithm: We repeatedly find horizontal segments that can be appended to the current good sequence and repeatedly add virtual edges, until no further such operations can be done. A straightforward implementation of the greedy algorithm, which checks all remaining horizontal segments in each step, takes $O(n^2)$ time. In the full version of the paper, we will present a more efficient implementation that costs only $O(n \log n)$ time.

Extracting a strictly monotone cycle. In the full version of the paper, we prove that if the above greedy algorithm stops with a good sequence $A = (S_1, S_2, \ldots, S_k)$ that does not cover all horizontal segments, then a strictly monotone cycle of the original graph G, without any

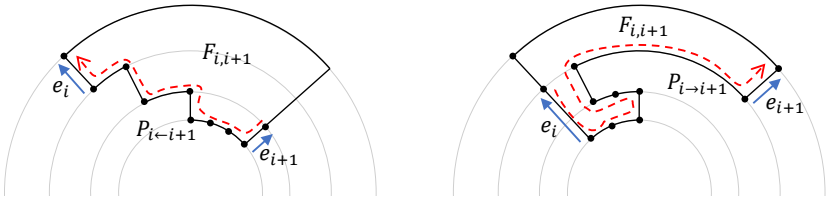


Figure 9 Face types $(*, \sqcup)$ and $(\sqcup, *)$.

virtual edges, can be found to certify the non-existence of a drawing. Let $(e_1, e_2, \ldots, e_s)$ be the circular ordering of $\mathcal{N}_{\mathsf{south}}(A)$. Note that $\{e_1, e_2, \ldots, e_s\}$ is the set of all edges connecting a vertex in G_k and a vertex not in G_k. The proof is achieved by a careful analysis of the structure of the faces involving $\{e_1, e_2, \ldots, e_s\}$. We show that the fact that no more progress can be made in the greedy algorithm forces the parts of the contours of these faces that are not in G_k to form ascending or descending patterns in a consistent manner, so we are able to extract a strictly monotone cycle in G by considering the edges in these facial cycles.

Face types. For each $1 \leq i \leq s$, we write $F_{i,i+1}$ to denote the unique face F such that C_F contains both e_i and $\overline{e_{i+1}}$. Note that $v_{s+1} = v_1$ because $(e_1, e_2, \ldots, e_s)$ is a circular ordering. Consider the face $F_{i,i+1}$, for some $1 \leq i \leq s$. We define $P_{i\leftarrow i+1}$ as the subpath of $C_{F_{i,i+1}}$ starting at $\overline{e_{i+1}}$ and ending at e_i. We write $P_{i\rightarrow i+1} = \overline{P_{i\leftarrow i+1}}$. We write $Z_{i\leftarrow i+1} = (z_1, z_2, \ldots)$ to denote the string of numbers such that z_l is the rotation of the subpath of $P_{i\leftarrow i+1}$ consisting of the first l edges. Similarly, we let $Z_{i\rightarrow i+1} = (z_1, z_2, \ldots)$ be the string of numbers such that z_l is the rotation of the subpath of $P_{i\rightarrow i+1}$ consisting of the first l edges. We define the types $(*, \sqcup)$, $(\sqcup, *)$, $(\sqcup, \sqcup)$, and $(-)$, as follows.

- $F_{i,i+1}$ is of type $(*, \sqcup)$ if $0 \circ 1^c \circ 2$, for some $c \geq 1$, is a prefix of $Z_{i\leftarrow i+1}$.
- $F_{i,i+1}$ is of type $(\sqcup, *)$ if $0 \circ (-1)^c \circ (-2)$, for some $c \geq 1$, is a prefix of $Z_{i\rightarrow i+1}$.
- $F_{i,i+1}$ is of type $(\sqcup, \sqcup)$ if $F_{i,i+1}$ is both of type $(\sqcup, *)$ and of type $(*, \sqcup)$.
- $F_{i,i+1}$ is of type $(-)$ if $Z_{i\leftarrow i+1} = 0 \circ 1^c \circ 2$ for some $c \geq 1$.

In other words, $F_{i,i+1}$ is of type $(-)$ if the subpath of the facial cycle of $F_{i,i+1}$ that connects the south endpoints of e_{i+1} and e_i is a horizontal straight line in the west direction. By considering $P_{i\rightarrow i+1} = \overline{P_{i\leftarrow i+1}}$, equivalently, $F_{i,i+1}$ is of type $(-)$ if $Z_{i\rightarrow i+1} = 0 \circ (-1)^c \circ (-2)$ for some $c \geq 1$.

Consider the good sequence $A = (S_1, S_2)$ of Figure 6 as an example, where we let $\mathcal{N}_{\mathsf{south}}(S_1, S_2) = (e_1, e_2, e_3, e_4)$, where $e_1 = (v_{3,1}, v_{1,1})$, $e_2 = (v_{3,2}, v_{2,1})$, $e_3 = (v_{3,4}, v_{2,3})$, and $e_4 = (v_{3,5}, v_{1,4})$. The facial cycle of the face $F_{1,2}$ is $(v_{3,1}, v_{1,1}, v_{2,1}, v_{3,2})$. We have $P_{1\rightarrow 2} = (v_{1,1}, v_{3,1}, v_{3,2}, v_{2,1})$ and $Z_{1\rightarrow 2} = (0, -1, -2)$, so $F_{1,2}$ is of type $(-)$.

Intuitively, the face $F_{i,i+1}$ is of type $(\sqcup, *)$ if $P_{i\rightarrow i+1}$ makes two 90° left turns before making any right turns, and the first 90° left turn is made at x_i. These two 90° left turns form a $\sqcup$-shape. Similarly, the face $F_{i,i+1}$ is of type $(*, \sqcup)$ if $P_{i\leftarrow i+1}$ makes two 90° right turns before making any left turns, and the first 90° right turn is made at x_{i+1}. These two 90° right turns form a $\sqcup$-shape. See Figure 9 for illustrations of faces of types $(*, \sqcup)$ and $(\sqcup, *)$. In the left part of the figure, we have $Z_{i\leftarrow i+1} = (0, 1, 1, 1, 2, 1, 2, 1, 2)$, so $F_{i,i+1}$ is of type $(*, \sqcup)$. In the right part of the figure, we have $Z_{i\rightarrow i+1} = (0, -1, -1, -2, -3, -3, -2, -1, -2)$, so $F_{i,i+1}$ is of type $(\sqcup, *)$. We show that one of the following holds, which intuitively implies the existence of a strictly monotone cycle.

- All faces $F_{i,i+1}$ are of type $(-)$ and $(\sqcup, *)$, and at least one face $F_{i,i+1}$ is of type $(\sqcup, *)$.
- All faces $F_{i,i+1}$ are of type $(-)$ and $(*, \sqcup)$, and at least one face $F_{i,i+1}$ is of type $(*, \sqcup)$.

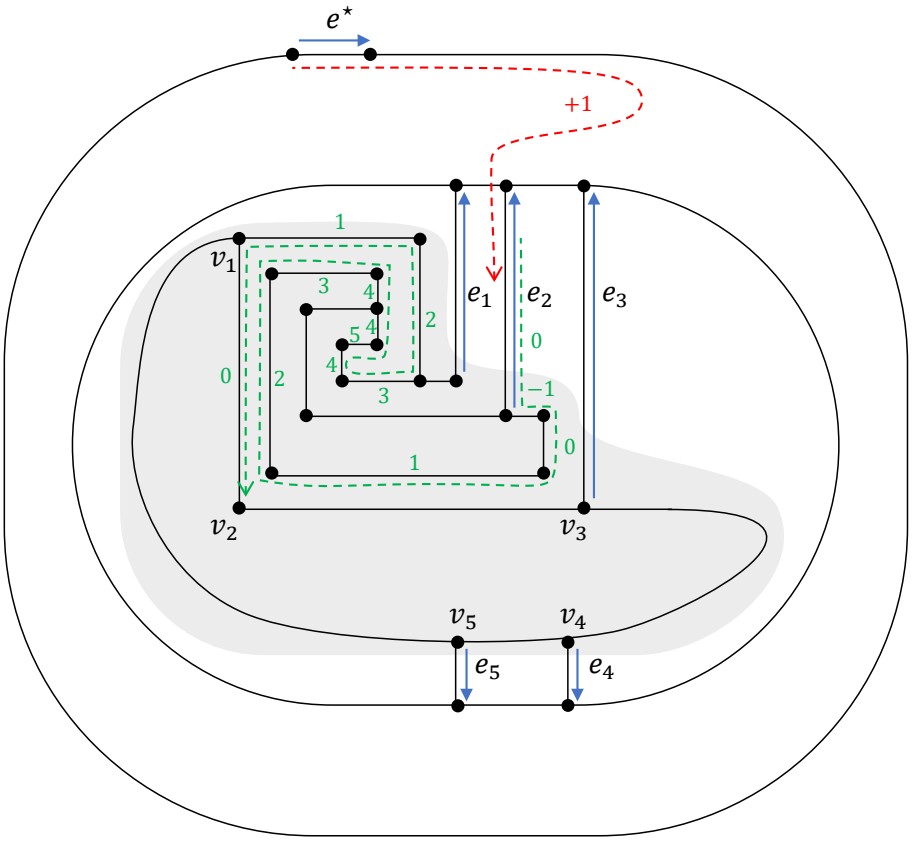


Figure 10 Extracting a strictly monotone cycle $C = (v_1, v_2, \ldots, v_5)$.

Consider Figure 10 for an example of extracting a strictly monotone cycle. In the figure, the shaded part corresponds to the part of the graph that is not in G_k. In this example, $\mathcal{N}_{\mathsf{south}}(A) = (e_1, e_2, \ldots, e_5)$. The faces $F_{5,1}$, $F_{1,2}$, and $F_{2,3}$ are of type $(*, \sqcup)$. The faces $F_{3,4}$ and $F_{4,5}$ are of type $(-)$. The cycle $C = (v_1, v_2, \ldots, v_5)$ is strictly monotone, as it is increasing. We can calculate that $\ell_C((v_1, v_2)) = 1$ by first going from $e^\star$ to $\overline{e_2}$ via a crossing-free path P and then going from $\overline{e_2}$ to (v_1, v_2) along the path $P_{2\to 3}$, as (v_1, v_2) is an intermediate edge of $P_{2\to 3}$. The first part has rotation 1 and the second part has rotation 0, so the overall rotation is 1. Similarly, we can calculate that $\ell_C(e) = 0$ for each remaining edge e in C.

4 Conclusions

In this paper, we presented a near-linear time algorithm to decide whether a given ortho-radial representation is drawable, improving upon the previous quadratic-time algorithm [35]. If the representation is drawable, then our algorithm outputs an ortho-radial drawing realizing the representation. Otherwise, our algorithm outputs a strictly monotone cycle to certify the non-existence of such a drawing. Given the broad applications of the topology-shape-metric framework in orthogonal drawing, we anticipate that our new ortho-radial drawing algorithm will be relevant and useful in future research in this field.

While there has been extensive research in orthogonal drawing, much remains unknown about the computational complexity of basic optimization problems in ortho-radial drawing. In particular, the problem of finding an ortho-radial representation that minimizes the number of bends has only been addressed by a practical algorithm [34] that has no provable guarantees. It remains an intriguing open question to determine to what extent bend minimization is polynomial-time solvable for ortho-radial drawing. To the best of our knowledge, even deciding whether a given plane graph admits an ortho-radial drawing *without bends* is not known to be polynomial-time solvable.

Given an ortho-radial representation, can we find an ortho-radial drawing with the smallest number of layers (i.e., the number of concentric circles) in polynomial time? As discussed in the full version of the paper, if a good sequence is given, then our algorithm can output a layer-minimized drawing. For the general case where a good sequence might not exist, our algorithm does not have the layer-minimization guarantee, as there is some flexibility in the choice of virtual edges to add, and selecting different virtual edges results in different good sequences. There was a series of work in finding *compact* orthogonal drawings according to various complexity measures [1, 6, 12, 32, 37]. To what extent the ideas developed in these works can be applied to ortho-radial drawings?

References

1 Michael J. Bannister, David Eppstein, and Joseph A. Simons. Inapproximability of orthogonal compaction. *Journal of Graph Algorithms and Applications*, 16(3):651–673, 2012. doi:10.7155/jgaa.00263.

2 Lukas Barth, Benjamin Niedermann, Ignaz Rutter, and Matthias Wolf. Towards a Topology-Shape-Metrics Framework for Ortho-Radial Drawings. In Boris Aronov and Matthew J. Katz, editors, *33rd International Symposium on Computational Geometry (SoCG)*, volume 77 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:16, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2017.14.

3 Lukas Barth, Benjamin Niedermann, Ignaz Rutter, and Matthias Wolf. A topology-shape-metrics framework for ortho-radial graph drawing. *arXiv preprint*, 2021. arXiv:2106.05734v1.

4 Hannah Bast, Patrick Brosi, and Sabine Storandt. Metro maps on flexible base grids. In *17th International Symposium on Spatial and Temporal Databases*, pages 12–22, 2021.

5 Carlo Batini, Enrico Nardelli, and Roberto Tamassia. A layout algorithm for data flow diagrams. *IEEE Transactions on Software Engineering*, SE-12(4):538–546, 1986.

6 Michael A. Bekos, Carla Binucci, Giuseppe Di Battista, Walter Didimo, Martin Gronemann, Karsten Klein, Maurizio Patrignani, and Ignaz Rutter. On turn-regular orthogonal representations. *Journal of Graph Algorithms and Applications*, 26(3):285–306, 2022. doi:10.7155/jgaa.00595.

7 Sandeep N Bhatt and Frank Thomson Leighton. A framework for solving VLSI graph layout problems. *Journal of Computer and System Sciences*, 28(2):300–343, 1984.

8 Therese Biedl, Anna Lubiw, Mark Petrick, and Michael Spriggs. Morphing orthogonal planar graph drawings. *ACM Transactions on Algorithms (TALG)*, 9(4):1–24, 2013.

9 Thomas Bläsius, Ignaz Rutter, and Dorothea Wagner. Optimal orthogonal graph drawing with convex bend costs. *ACM Trans. Algorithms*, 12(3):33:1–33:32, 2016.

10 Ulrik Brandes, Sabine Cornelsen, Christian Fieß, and Dorothea Wagner. How to draw the minimum cuts of a planar graph. *Computational Geometry*, 29(2):117–133, 2004.

11 Ulrik Brandes and Dorothea Wagner. Dynamic grid embedding with few bends and changes. In *International Symposium on Algorithms and Computation*, pages 90–99. Springer, 1998.

12 Stina S Bridgeman, Giuseppe Di Battista, Walter Didimo, Giuseppe Liotta, Roberto Tamassia, and Luca Vismara. Turn-regularity and optimal area drawings of orthogonal representations. *Computational Geometry*, 16(1):53–93, 2000.

13 Yi-Jun Chang and Hsu-Chun Yen. On bend-minimized orthogonal drawings of planar 3-graphs. In *33rd International Symposium on Computational Geometry (SoCG 2017)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.

14 Sabine Cornelsen and Andreas Karrenbauer. Accelerated bend minimization. *JGAA*, 16(3):635–650, 2012.

15 Giuseppe Di Battista, Walter Didimo, Maurizio Patrignani, and Maurizio Pizzonia. Orthogonal and quasi-upward drawings with vertices of prescribed size. In *Proceedings of the 7th International Symposium on Graph Drawing (GD)*, pages 297–310. Springer Berlin Heidelberg, 1999.

16 Giuseppe Di Battista, Giuseppe Liotta, and Francesco Vargiu. Spirality and optimal orthogonal drawings. *SIAM Journal on Computing*, 27(6):1764–1811, 1998.

17 Walter Didimo, Giuseppe Liotta, Giacomo Ortali, and Maurizio Patrignani. Optimal orthogonal drawings of planar 3-graphs in linear time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 806–825. SIAM, 2020.

18 Walter Didimo, Giuseppe Liotta, and Maurizio Patrignani. On the complexity of HV-rectilinear planarity testing. In *International Symposium on Graph Drawing (GD)*, pages 343–354. Springer, 2014.

19 Walter Didimo, Giuseppe Liotta, and Maurizio Patrignani. Bend-minimum orthogonal drawings in quadratic time. In *International Symposium on Graph Drawing and Network Visualization (GD)*, pages 481–494. Springer, 2018.

20 Sally Dong, Yu Gao, Gramoz Goranci, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Guanghao Ye. Nested dissection meets ipms: Planar min-cost flow in nearly-linear time. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 124–153. SIAM, 2022.

21 Stephane Durocher, Stefan Felsner, Saeed Mehrabi, and Debajyoti Mondal. Drawing HV-restricted planar graphs. In *Latin American Symposium on Theoretical Informatics (LATIN)*, pages 156–167. Springer, 2014.

22 Markus Eiglsperger, Carsten Gutwenger, Michael Kaufmann, Joachim Kupke, Michael Jünger, Sebastian Leipert, Karsten Klein, Petra Mutzel, and Martin Siebenhaller. Automatic layout of UML class diagrams in orthogonal style. *Information Visualization*, 3(3):189–208, 2004.

23 Martin Fink, Magnus Lechner, and Alexander Wolff. Concentric metro maps. In *Proceedings of the Schematic Mapping Workshop (SMW)*, 2014.

24 Michael Formann, Torben Hagerup, James Haralambides, Michael Kaufmann, Frank Thomson Leighton, Antonios Symvonis, Emo Welzl, and G Woeginger. Drawing graphs in the plane with high resolution. *SIAM Journal on Computing*, 22(5):1035–1052, 1993.

25 Ashim Garg and Roberto Tamassia. A new minimum cost flow algorithm with applications to graph drawing. In *Proceedings of the Symposium on Graph Drawing (GD)*, pages 201–216. Springer Berlin Heidelberg, 1997.

26 Carsten Gutwenger, Michael Jünger, Karsten Klein, Joachim Kupke, Sebastian Leipert, and Petra Mutzel. A new approach for visualizing UML class diagrams. In *Proceedings of the 2003 ACM symposium on Software visualization*, pages 179–188, 2003.

27 Mahdieh Hasheminezhad, S Mehdi Hashemi, Brendan D McKay, and Maryam Tahmasbi. Rectangular-radial drawings of cubic plane graphs. *Computational Geometry*, 43(9):767–780, 2010.

28 Mahdieh Hasheminezhad, S Mehdi Hashemi, and Maryam Tahmasbi. Ortho-radial drawings of graphs. *Australasian Journal of Combinatorics*, 44:171–182, 2009.

29 Min-Yu Hsueh. *Symbolic layout and compaction of integrated circuits*. PhD thesis, University of California, Berkeley, 1980.

30 Steve Kieffer, Tim Dwyer, Kim Marriott, and Michael Wybrow. Hola: Human-like orthogonal network layout. *IEEE transactions on visualization and computer graphics*, 22(1):349–358, 2015.

31 Gunnar W. Klau and Petra Mutzel. Quasi-orthogonal drawing of planar graphs. Technical Report MPI-I-98-1-013, Max-Planck-Institut für Informatik, Saarbrücken, 1998.

32 Gunnar W Klau and Petra Mutzel. Optimal compaction of orthogonal grid drawings. In *Proceedings of the 7th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 304–319. Springer, 1999.

33 Robin S. Liggett and William J. Mitchell. Optimal space planning in practice. *Computer-Aided Design*, 13(5):277–288, 1981. Special Issue Design optimization. doi:10.1016/0010-4485(81)90317-1.

34 Benjamin Niedermann and Ignaz Rutter. An integer-linear program for bend-minimization in ortho-radial drawings. In *International Symposium on Graph Drawing and Network Visualization*, pages 235–249. Springer, 2020.

35 Benjamin Niedermann, Ignaz Rutter, and Matthias Wolf. Efficient Algorithms for Ortho-Radial Graph Drawing. In Gill Barequet and Yusu Wang, editors, *35th International Symposium on Computational Geometry (SoCG)*, volume 129 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 53:1–53:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2019.53.

36 Achilleas Papakostas and Ioannis G Tollis. Efficient orthogonal drawings of high degree graphs. *Algorithmica*, 26(1):100–125, 2000.

37 Maurizio Patrignani. On the complexity of orthogonal compaction. *Computational Geometry*, 19(1):47–67, 2001.

38 James A Storer. The node cost measure for embedding graphs on the planar grid. In *Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 201–210, 1980.

39 Roberto Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444, 1987.

40 Leslie G Valiant. Universality considerations in VLSI circuits. *IEEE Transactions on Computers*, 100(2):135–140, 1981.

41 Hsiang-Yun Wu, Benjamin Niedermann, Shigeo Takahashi, Maxwell J. Roberts, and Martin Nöllenburg. A survey on transit map layout – from design, machine, and human perspectives. *Computer Graphics Forum*, 39(3):619–646, 2020. doi:10.1111/cgf.14030.

42 Yingying Xu, Ho-Yin Chan, and Anthony Chen. Automated generation of concentric circles metro maps using mixed-integer optimization. *International Journal of Geographical Information Science*, pages 1–26, 2022.

Approximation Algorithms for Network Design in Non-Uniform Fault Models

Chandra Chekuri
Department of Computer Science, University of Illinois, Urbana-Champaign, Urbana, IL, USA

Rhea Jain
Department of Computer Science, University of Illinois, Urbana-Champaign, Urbana, IL, USA

Abstract

Classical network design models, such as the Survivable Network Design problem (SNDP), are (partly) motivated by robustness to faults under the assumption that any subset of edges upto a specific number can fail. We consider *non-uniform* fault models where the subset of edges that fail can be specified in different ways. Our primary interest is in the flexible graph connectivity model [1, 3, 4, 8], in which the edge set is partitioned into *safe* and *unsafe* edges. Given parameters $p, q \geq 1$, the goal is to find a cheap subgraph that remains p-connected even after the failure of q unsafe edges. We also discuss the bulk-robust model [6, 2] and the relative survivable network design model [19]. While SNDP admits a 2-approximation [32], the approximability of problems in these more complex models is much less understood even in special cases. We make two contributions.

Our first set of results are in the flexible graph connectivity model. Motivated by a conjecture that a constant factor approximation is feasible when p and q are fixed, we consider two special cases. For the s-t case we obtain an approximation ratio that depends only on p, q whenever $p + q > pq/2$ which includes $(p, 2)$ and $(2, q)$ for all $p, q \geq 1$. For the global connectivity case we obtain an $O(q)$ approximation for $(2, q)$, and an $O(p)$ approximation for $(p, 2)$ and $(p, 3)$ for any $p \geq 1$, and for $(p, 4)$ when p is even. These are based on an augmentation framework and decomposing the families of cuts that need to be covered into a small number of uncrossable families.

Our second result is a poly-logarithmic approximation for a generalization of the bulk-robust model when the "width" of the given instance (the maximum number of edges that can fail in any particular scenario) is fixed. Via this, we derive corresponding approximations for the flexible graph connectivity model and the relative survivable network design model. We utilize a recent framework due to Chen et al. [17] that was designed for handling group connectivity.

2012 ACM Subject Classification Theory of computation → Routing and network design problems

Keywords and phrases non-uniform faults, network design, approximation algorithm

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.36

Category Track A: Algorithms, Complexity and Games

Funding *Chandra Chekuri*: Supported in part by NSF grants CCF-1910149 and CCF-1907937.
Rhea Jain: Supported in part by NSF grant CCF-1907937.

Acknowledgements We thank Qingyun Chen for clarifications on a proof in [17]. We thank Joseph Cheriyan for pointers and helpful comments on flexible graph connectivity. The initial impetus for our work on this topic came from [8].

1 Introduction

The Survivable Network Design Problem (SNDP) is an important problem in combinatorial optimization that generalizes many well-known problems related to connectivity and is also motivated by practical problems related to the design of fault-tolerant networks. The input to this problem is an undirected graph $G = (V, E)$ with non-negative edge costs $c : E \to \mathbb{R}_+$ and a collection of source-sink pairs $(s_1, t_1), \ldots, (s_h, t_h)$, each with an integer connectivity requirement r_i. The goal is to find a minimum-cost subgraph H of G such that

H has r_i connectivity for each pair (s_i, t_i). We focus on edge-connectivity requirements in this paper.[1] SNDP contains as special cases classical problems such as s-t shortest path, minimum spanning tree (MST), minimum k-edge-connected subgraph (k-ECSS), Steiner tree, Steiner forest and several others. It is NP-Hard and APX-Hard to approximate. There is a 2-approximation via the iterated rounding technique [32].

A pair (s, t) that is k-edge-connected in G is robust to the failure of *any* set of $k-1$ edges. In various settings, the set of edges that can fail can be correlated and/or exhibit non-uniform aspects. We are interested in network design in such settings, and discuss a few models of interest that have been studied in the (recent) past. We start with the flexible graph connectivity model (flex-connectivity for short) that was the impetus for our work.

Flexible graph connectivity. In this model, first introduced by Adjiashvili [1] and studied in several recent papers [3, 4, 5, 8, 7], the input is an edge-weighted undirected graph $G = (V, E)$ where the edge set E is partitioned to *safe* edges $\mathcal{S}$ and *unsafe* edges $\mathcal{U}$. The assumption, as the names suggest, is that unsafe edges can fail while safe edges cannot. We say that a vertex-pair (s, t) is (p, q)-flex-connected in a subgraph H of G if s and t are p-edge-connected after deleting from H any subset of at most q unsafe edges. The input, as in SNDP, consists of G and h source-sink pairs; the i'th pair now specifies a (p_i, q_i)-flex-connectivity requirement. The goal is to find a min-cost subgraph H of G such that for each i, s_i and t_i are (p_i, q_i)-flex-connected in H. We refer to this as the Flex-SNDP problem. Note that Flex-SNDP generalizes SNDP in two ways[2].

Bulk-robust network design. This fairly general non-uniform model was introduced by Adjiashvili, Stiller and Zenklusen [6]. Here an explicit *scenario* set $\Omega = \{F_1, F_2, \ldots, F_m\}$ is given as part of the input where each $F_j \subseteq E$. The goal is to find a min-cost subgraph H of G such that each of the given pairs (s_i, t_i) remains connected in $H - F_j$ for each $j \in [m]$. We consider a slight generalization of this problem in which each scenario is now a pair $(F_j, \mathcal{K}_j)$ where $\mathcal{K}_j$ is a set of source-sink pairs. As earlier, the goal is to find a min-cost subgraph H of G such that for each $j \in [m]$, each pair (s_i, t_i) in $\mathcal{K}_j$ is connected in $H - F_j$. The *width* of the failure scenarios is $\max_{1 \le j \le \ell} |F_j|$. We use Bulk-SNDP to refer to this problem.

The advantage of the bulk-robust model is that one can specify arbitrarily correlated failure patterns, allowing it to capture many well studied problems in network design. We observe that SNDP and Flex-SNDP problem can be cast as special cases of Bulk-SNDP model where the width is $\max_i(r_i - 1)$ in the former case, and $\max_i(p_i + q_i - 1)$ in the latter case. The slight generalization on Bulk-SNDP described above also allows us to model a new problem recently proposed by Dinitz, Koranteng, and Kortsarz [19] called **Relative Survivable Network Design** (RSNDP). This problem allows one to ask for higher connectivity even when the underlying graph G has small cuts. The input is an edge-weighted graph $G = (V, E)$ and source-sink pairs (s_i, t_i) each with requirement r_i; the goal is to find a min-cost subgraph H of G such that for each $F \subseteq E$ with $|F| < r_i$, (s_i, t_i) is connected in $H - F$ if s_i and t_i are connected in $G - F$. It is easy to see that RSNDP is a special case of Bulk-SNDP with width at most $\max_i(r_i - 1)$. A disadvantage of Bulk-SNDP is that scenarios have to be explicitly listed, while the other models discussed specify failure scenarios implicitly. However, when connectivity requirements are small/constant, one can reduce to Bulk-SNDP by explicitly listing the failure sets.

---

[1] In the literature the term EC-SNDP and VC-SNDP are used to distinguish edge and vertex connectivity requirements. We use SNDP in place of EC-SNDP.

[2] If all edges are safe ($E = \mathcal{S}$), then $(p, 0)$-flex-connectivity is equivalent to p-edge-connectivity. Similarly, if all edges are unsafe ($E = \mathcal{U}$), then $(1, q-1)$-flex-connectivity is equivalent to q-edge-connectivity.

While SNDP admits a 2-approximation, the approximability of network design in the preceding models is not well-understood. The known results mostly focus on two special cases: (i) the single pair case where there is only one pair (s,t) with a connectivity requirement and (ii) the spanning or global connectivity case when all pairs of vertices have identical connectivity requirement. Even in the single pair case, there are results that show that problems in the non-uniform models are hard to approximate to poly-logarithmic or almost-polynomial factors when the connectivity requirement is not bounded [6, 5]. Further, natural LP relaxations in some cases can also be shown to have large integrality gaps [16]. Motivated by these negative results and practical considerations, we focus our attention on Flex-SNDP when the max connectivity requirement p, q are small, and similarly on Bulk-SNDP when the width is small. Other network design problems with similar hardness results have admitted approximation ratios that depend on the max connectivity requirement (for example, VC-SNDP [12, 18, 38] and (s,t) case of Bulk-Robust [6]).

1.1 Our contribution

We are mainly motivated by Flex-SNDP and insights for it via Bulk-SNDP. We make two broad contributions. Our first set of results is on special cases of Flex-SNDP for which we obtain constant factor approximations. Our second contribution is a poly-logarithmic approximation for Flex-SNDP, Bulk-SNDP, and RSNDP when the requirements are small.

We use the terminology (p,q)-Flex-ST to refer to the single-pair problem with requirement (p,q). We use the term (p,q)-FGC to refer to the spanning/global-connectivity problem where all pairs of vertices have the (p,q)-flex-connectivity requirement (the term FGC is to be consistent with previous usage [3, 8]).

(p,q)-FGC. Adjiashvili et al. [3] considered $(1,1)$-FGC and obtained a constant factor approximation that was subsequently improved to 2 by Boyd et al. [8]. [8] obtained several results for (p,q)-FGC including a 4-approximation for $(p,1)$-FGC, a $(q+1)$-approximation for $(1,q)$-FGC, and a $O(q \log n)$-approximation for (p,q)-FGC. The first non-trivial case of small p, q for which we did not know a constant factor is $(2,2)$-FGC. We prove several results that, as a corollary, yield constant factor approximation for small values of p, q.

▶ **Theorem 1.** *For any $q \geq 0$ there is a $(2q+2)$-approximation for $(2,q)$-FGC. For any $p \geq 1$ there is a $(2p+4)$-approximation for $(p,2)$-FGC, and a $(4p+4)$-approximation for $(p,3)$-FGC. Moreover, for all* even *$p \geq 2$ there is an $(6p+4)$-approximation for $(p,4)$-FGC.*

▶ Remark 2. In independent work Bansal et al. [7] obtained an $O(1)$-approximation for $(p,2)$-FGC for any $p \geq 1$ (6 when p is even and 20 when p is odd). More broadly, they obtain constant factor approximations for a special class of augmentation problems and demonstrate several interesting applications.

(p,q)-Flex-ST. Adjiashivili et al. [4] considered $(1,q)$-Flex-ST and $(p,1)$-Flex-ST and obtained several results. They described a q-approximation for $(1,q)$-Flex-ST and a $(p+1)$-approximation for $(p,1)$-Flex-ST; when p is a fixed constant they obtain a 2-approximation. Also implicit in [6] is an $O(q(p+q)\log n)$-approximation algorithm for (p,q)-Flex-ST that runs in $n^{O(p+q)}$-time. No constant factor approximation was known when $p, q \geq 2$ with $(2,2)$-Flex-ST being the first non-trivial case. We prove a constant factor approximation for this and several more general settings via the following theorem.

▶ **Theorem 3.** *For all p, q where $(p+q) > pq/2$, there is an $O((p+q)^{O(p)})$-approximation algorithm for (p,q)-Flex-ST that runs in $n^{O(p+q)}$ time. In particular, there is an $O(1)$ approximation for $(p,2)$ and $(2,q)$-Flex-ST when p, q are fixed constants.*

Flex-SNDP, Bulk-SNDP, and RSNDP. We show that these problems admit poly-logarithmic approximation algorithms when the width/connectivity requirements are small.

▶ **Theorem 4.** *There is a randomized algorithm that yields an $O(k^4 \log^7 n)$-approximation for Bulk-SNDP on instances with width at most k, and runs in expected polynomial time.*

▶ **Corollary 5.** *There is a randomized algorithm that yields an $O(q(p+q)^3 \log^7 n)$- approximation for Flex-SNDP when $(p_i, q_i) \leq (p, q)$ for all pairs (s_i, t_i), and runs in expected $n^{O(q)}$-time.*

▶ **Corollary 6.** *There is a randomized algorithm that yields an $O(k^4 \log^7 n)$-approximation for RSNDP where k is the maximum connectivity requirement, and runs in expected polynomial time.*

As far as we are aware of, no previous approximation algorithms were known for SNDP versions of flexible graph connectivity (with both $p, q \geq 2$) or bulk-robustness.

1.2 Overview of techniques and related work

Network design has substantial literature. We describe closely related work and results to put ours in context.

SNDP and related connectivity problems. SNDP is a canonical problem in network design for connectivity that captures many problems. We refer the reader to some older surveys [28, 36] on approximation algorithms for connectivity problems, and several recent papers with exciting progress on TSP and weighted Tree and Cactus augmentation. Frank's books is an excellent source for polynomial-time solvable exact algorithms [23]. For SNDP, the augmentation approach was pioneered in [41], and was refined in [25]. These led to $2H_k$ approximation where k is the maximum connectivity requirement. Jain's iterated rounding approach [32] obtained a 2-approximation. The nice structural results that underpin the algorithms for SNDP have been extended to *element* connectivity introduced in [33]; consequently, Elem-SNDP also admits a 2-approximation [21]. VC-SNDP has posed non-trivial technical challenges; the problem is not constant factor approximable when the maximum connectivity requirement is large [12]. In a breakthrough result, [18], Chuzhoy and Khanna gave an $O(k^3 \log n)$ approximation via a reduction to element connectivity, where k is the maximum connectivity. Nutov [38] improves this to an $O(k^2)$ approximation for the single-source VC-SNDP case; however, there has been no further progress in obtaining an $f(k)$-approximation for the general VC-SNDP problem.

Flexible Graph Connectivity. Flexible graph connectivity has been a topic of recent interest, although the model was introduced earlier in the context of a single pair [1]. Adjiashvili, Hommelsheim and Mühlenthaler [3] introduced FGC (which is the same as $(1,1)$-FGC) and pointed out that it generalizes the well-known MST and 2-ECSS problems. Several approximation algorithms for various special cases of FGC and Flex-ST were obtained by Adjiashvili et al.[3] and Boyd et al. [8], as described in Section 1.1.

Adjiashvili et al. [5] also showed hardness results in the single pair setting. They prove that $(1,k)$-Flex-ST in directed graphs is at least as hard as directed Steiner tree which implies poly-logarithmic factor inapproximability [31]. They prove that $(k,1)$-Flex-ST in

directed graphs is at least as hard to approximate as directed Steiner forest (which has almost polynomial factor hardness [20]). The hardness results are when k is part of the input and large, and show that approximability of network design in this model is substantially different from the edge-connectivity model.

Bulk-Robust Network Design. This model was initiated in [6]. They obtained an $O(\log n + \log m)$ approximation for the Bulk-Robust spanning tree problem (as a special case of the more general matroid basis problem). The authors show that the *directed* single pair problem (Bulk-Robust shortest path) is very hard to approximate. The hardness reduction motivated the definition of width. The authors obtain an $O(k^2 \log n)$-approximation for Bulk-Robust shortest path via a nice reduction to the Set Cover problem and the use of the augmentation approach that we build upon here. For the special case of $k = 2$ the authors obtain an $O(1)$-approximation. Adjiashvili [2] showed that if the graph is planar then one can obtain an $O(k^2)$-approximation for both Bulk-Robust shortest path and spanning tree problems – he uses the augmentation approach from [6] and shows that the corresponding covering problem in each augmentation phase corresponds to a Set Cover problem that admits a constant factor approximation. As far as we are aware, there has not been any progress on the general setting beyond the spanning tree and shortest path cases.

Relative Network Design. This model was introduced in very recent work [19]. The authors obtain a 2-approximation for the spanning case via nice use of the iterated rounding technique even though the requirement function is not skew-supermodular. They also obtain a simple 2-approximation when the maximum requirement is 2. They obtain a $\frac{27}{4}$-approximation for the (s,t)-case when the maximum demand is 3.

Survivable Network Design for Group Connectivity. As we remarked, one part of this work builds on the recent framework of Chen et al. [17]. Their main motivation was to address the approximability of the survivable network design problem with group connectivity requirements. We refer the reader to [24, 30, 31, 14, 13, 17] and pointers to the extensive work on the approximability of these problems.

Our Techniques. As we remarked, the non-uniform models have been difficult to handle for existing algorithmic techniques. The structures that underpins the known algorithms for SNDP (primal-dual [41] and iterated rounding [32]) are skew-supermodularity of the requirement function and submodularity of the cut function in graphs. Since non-uniform models do not have such clean structural properties, these known techniques cannot be applied directly. Another technique for network design, based on several previous works, is augmentation. In the augmentation approach we start with an initial set of edges F_0 that partially satisfy the connectivity constraints. We then augment F_0 with a set F in the graph $G - F_0$; the augmentation is typically done to increase the connectivity by one unit for pairs that are not yet satisfied. We repeat this process in several stages until all connectivity requirements are met. The utility of the augmentation approach is that it allows one to reduce a higher-connectivity problem to a series of problems that solve a potentially simpler $\{0,1\}$-connectivity problem. An important tool in this area is a 2-approximation for covering an *uncrossable* function (a formal definition is given in Section 3) [41].

In trying to use the augmentation approach for Flex-SNDP and its special cases, we see that the resulting functions are usually not uncrossable. To prove Theorems 1 and 3, we overcome this difficulty by decomposing the family of cuts to be covered in the augmentation

problem into a sequence of cleverly chosen uncrossable subfamilies. Our structural results hold for certain range of values of p and q and hint at additional structure that may be available to exploit in future work. Boyd et al. also show a connection to *capacitated* network design (also implicitly in [5]) which has been studied in several works [25, 9, 10, 11]. This model generalizes standard edge connectivity by allowing each edge e to have an integer capacity $u_e \geq 1$. One can reduce capacitated network design to standard edge connectivity by replacing each edge e with u_e parallel edges, blowing up the approximation factor by $\max_e u_e$. Boyd et al. show that $(1,k)$-Flex-SNDP and $(k,1)$-Flex-SNDP can be reduced to Cap-SNDP with maximum capacity k. While this reduction does not extend when $p, q \geq 2$, it provides a useful starting point that we exploit for Theorem 3.

We use a completely different algorithmic approach to prove Theorem 4. We rely on a recent novel framework of Chen, Laekhanukit, Liao, and Zhang [17] to tackle survivable network design in group connectivity setting. They used the seminal work of Räcke [39] on probabilistic approximation of capacitated graphs via trees, and the group Steiner tree rounding techniques of Garg, Konjevod and Ravi [24], and subsequent developments [27]. We adapt their ideas to handle the augmentation problem for Flex-SNDP and Bulk-SNDP. We refer the reader to Section 4 since the framework is technical.

Organization. Section 2 sets up the relevant background on the LP relaxations for Flex-SNDP and Bulk-SNDP. Section 3 outlines the proofs of Theorems 1 and 3. Section 4 outlines the proofs of Theorems 4 and the resulting corollaries 5 and 6. This paper combines and extends results from two preliminary versions; [16] for the first set of results discussed in Section 3, and [15] for the second set of results discussed in Section 4. A full version of this paper will be made publicly available in the near future.

2 Preliminaries

Throughout the paper we will assume that we are given an undirected graph $G = (V, E)$ along with a cost function $c : E \to \mathbb{R}_{\geq 0}$. When we say that H is a subgraph of $G = (V, E)$ we implicitly assume that H is an edge-induced subgraph, i.e. $H = (V, F)$ for some $F \subseteq E$. For any subset of edges $F \subseteq E$ and any set $S \subseteq V$ we use the notation $\delta_F(S)$ to denote the set of edges in F that have exactly one endpoint in S. We may drop F if it is clear from the context. For all discussion of flex-connectivity, we will let $\mathcal{S}$ denote the set of safe edges and $\mathcal{U}$ denote the set of unsafe edges.

Flex-SNDP LP Relaxation. We describe an LP relaxation for (p, q)-Flex-SNDP problem. Recall that we are given a set of h terminal pairs $(s_i, t_i) \subseteq V \times V$ and the goal is to choose a min-cost subset of the edges F such that in the subgraph $H = (V, F)$, s_i and t_i are (p, q)-flex-connected for any $i \in [h]$. Let $\mathcal{C} = \{S \subset V \mid \exists i \in [h] \text{ s.t. } |S \cap \{s_i, t_i\}| = 1\}$ be the set of all vertex sets that separate some terminal pair. For a set of edges F to be feasible for the given (p, q)-Flex-SNDP instance, we require that for all $S \in \mathcal{C}$, $|\delta_F(S) \setminus B| \geq p$ for any $B \subseteq \mathcal{U}$ with $|B| \leq q$. We can write cut covering constraints expressing this condition, but these constraints are not adequate by themselves. To improve this LP, we consider the connection to capacitated network design: we give each safe edge a capacity of $p + q$, each unsafe edge a capacity of p, and require $p(p + q)$ connectivity for the terminal pairs; it is not difficult to verify that this is a valid constraint. These two sets of constraints yield the following LP relaxation with variables $x_e \in [0, 1]$, $e \in E$.

$$\begin{aligned}
\min \sum_{e\in E} c(e)x_e & \\
\text{subject to} \sum_{e\in\delta(S)-B} x_e \geq p & \qquad S\in\mathcal{C}, B\subseteq\mathcal{U}, |B|\leq q \\
(p+q)\sum_{e\in\delta(S)\cap\mathcal{S}} x_e + p\sum_{e\in\delta(S)\cap\mathcal{U}} x_e \geq p(p+q) & \qquad S\in\mathcal{C} \\
x_e\in[0,1] & \qquad e\in E
\end{aligned}$$

The following lemma borrows ideas from [8, 10].

▶ **Lemma 7.** *The Flex-SNDP LP relaxation can be solved in $n^{O(q)}$ time. For (p,q)-FGC, it can be solved in polynomial time.*

Proof. We show a polynomial time separation oracle for the given LP. Suppose we are given some vector $x\in[0,1]^{|E|}$. We first check if the capacitated min-cut constraints are satisfied. This can be done in polynomial time by giving every safe edge a weight of $p+q$ and every unsafe edge a weight of p, and checking that the min-cut value is at least $p(p+q)$. If it is not, we can find the minimum cut and output the corresponding violated constraint. Suppose all capacitated constraints are satisfied. Then, for each $B\subseteq\mathcal{U}, |B|\leq q$ we remove B and check that for each $s,t\in T$, the s-t min-cut value in the graph $G-B$ with edge-capacities given by x is at least p. Since there are at most $n^{O(q)}$ such possible sets B, we get our desired separation oracle.

In the FGC case, if there is a remaining unsatisfied constraint, then there must must be some $S\subset V$ and some $B\subseteq\mathcal{U}$, $|B|\leq q$, such that $\sum_{e\in\delta(S)-B} x_e < p$. In particular, $\sum_{e\in\mathcal{S}\cap\delta(S)-B} x_e < p$ and $\sum_{e\in\mathcal{U}\cap\delta(S)-B} x_e < p$. We claim that the total weight (according to weights $(p+q)$ for safe edges and p for unsafe edges) going across $\delta(S)$ is at most $2p(p+q)$: at most $(p+q)p$ from $\mathcal{S}\cap(\delta(S)-B)$, at most p^2 from $\mathcal{U}\cap(\delta(S)-B)$, and at most pq from B. Recall that the min-cut of the graph according the weights has already been verified to be at least $p(p+q)$. Hence, any violated cut from the first set of constraints corresponds to 2-approximate min-cut. It is known via Karger's theorem that there are at most $O(n^4)$ 2-approximate min-cuts in a graph, and moreover they can also be enumerated in polynomial time [34, 35]. We can enumerate all 2-approximate min-cuts and check each of them to see if they are violated. To verify whether a candidate cut S is violated we consider the unsafe edges in $\delta(S)\cap\mathcal{U}$ and sort them in decreasing order of x_e value. Let B' be a prefix of this sorted order of size $\min\{q, |\delta(S)\cap\mathcal{U}|\}$. It is easy to see that that $\delta(S)$ is violated iff it is violated when $B=B'$. Thus, we can verify all candidate cuts efficiently. ◀

Bulk-SNDP LP Relaxation. We can similarly define an LP relaxation for the Bulk-SNDP problem. As above, we have a variable $x_e\in[0,1]$ for each edge $e\in E$.

$$\begin{aligned}
\min \sum_{e\in E} c(e)x_e & \\
\text{subject to} \sum_{e\in\delta(S)\setminus F_j} x_e \geq 1 & \qquad \forall (F_j,\mathcal{K}_j)\in\Omega, S \text{ separates a terminal pair in } \mathcal{K}_j \\
x_e\in[0,1] &
\end{aligned}$$

Note that the LP has a separation oracle that runs in time polynomial in n and m: for each scenario $(F_j,\mathcal{K}_j)$, we can remove F_j from the graph and check that the minimum (u,v)-cut is at least 1 for each $(u,v)\in\mathcal{K}_j$.

Augmentation. The results of this paper rely on the augmentation framework. We first discuss Flex-SNDP. Suppose $G = (V, E), \{s_i, t_i\}_{i \in [h]}$ is an instance of (p, q)-Flex-SNDP. We observe that $(p, 0)$-Flex-SNDP instance can be solved via 2-approximation to EC-SNDP. Hence, we are interested in $q \geq 1$. Let F_1 be a feasible solution for the $(p, q-1)$-Flex-SNDP instance. This implies that for any cut S that separates a terminal pair we have $|\delta_{F_1 \cap \mathcal{S}}(S)| \geq p$ or $|\delta_{F_1}(S)| \geq p + q - 1$. We would like to augment F_1 to obtain a feasible solution to satisfy the (p, q) requirement. Define a function $f : 2^{|V|} \to \{0, 1\}$ where $f(S) = 1$ iff (i) S separates a terminal pair and (ii) $|\delta_{F_1 \cap \mathcal{S}}(S)| < p$ *and* $|\delta_{F_1}(S)| = p + q - 1$. We call S a *violated* cut with respect to F_1. Since F_1 satisfies $(p, q-1)$ requirement, if $|\delta_{F_1 \cap \mathcal{S}}| < p$ it must be the case that $|\delta_{F_1}(S)| \geq p + q - 1$. The following lemma is simple.

▶ **Lemma 8.** *Suppose $F_2 \subseteq E \setminus F_1$ is a feasible cover for f, that is, $\delta_{F_2}(S) \geq f(S)$ for all S. Then $F_1 \cup F_2$ is a feasible solution to (p, q)-Flex-SNDP.*

The augmentation problem is then to find a min-cost subset of edges to cover f in $G - F_1$. The key observation is that the augmentation problem does not distinguish between safe and unsafe edges and hence we can rely on traditional connectivity augmentation ideas. Note that if we instead tried to augment from $(p-1, q)$ to (p, q)-flex-connectivity, we would still need to distinguish between safe and unsafe edges. The following lemma shows that the LP relaxation for the original instance provides a valid cut-covering relaxation for the augmentation problem.

▶ **Lemma 9.** *Let $x \in [0, 1]^{|E|}$ be a feasible LP solution for a given instance of (p, q)-Flex-Steiner. Let F_1 be a feasible solution that satisfies $(p, q-1)$ requirements for the terminal. Then, for any violated cut $S \subseteq V$ in (V, F_1), we have $\sum_{e \in \delta(S) \setminus F_1} x_e \geq 1$.*

Adjiashvili et al [6] also define a corresponding augmentation problem for the bulk robust network design model as follows: given an instance to Bulk-SNDP, let $\Omega_\ell = \bigcup_{j \in [m]} \{(F, \mathcal{K}_j) : |F| \leq \ell \text{ and } F \subseteq F_j\}$. Let $H_\ell \subseteq E$ be a subset of edges that satisfy the constraints defined by the scenarios in Ω_ℓ. Then, a solution to the augmentation problem from $\ell - 1$ to ℓ is a set of edges H' such that $H_{\ell-1} \cup H'$ satisfies the constraints defined by scenarios in Ω_ℓ. It is not difficult to verify that any solution to the original instance Ω is also a solution to any of the augmentation problems, and any solution satisfying all scenarios in Ω_k also satisfies all scenarios in Ω, where k is the width of the Bulk-SNDP instance.

3 Uncrossability-Based Approximation Algorithms

In this section, we prove Theorem 1 and Theorem 3. Recall that we are given a graph $G = (V, E)$ with cost function on the edges $c : E \to \mathbb{R}_{\geq 0}$ and a partition of the edge set E into safe edges $\mathcal{S}$ and unsafe edges $\mathcal{U}$. In the (p, q)-FGC problem, our goal is to find the cheapest set of edges such that every cut has either p safe edges or $p + q$ total edges. The (p, q)-Flex-ST problem is similar, except that we are also given $s, t \in V$ as part of the input and we focus only on cuts separating s from t.

We start by providing some necessary background on uncrossable/ring families and submodularity of the cut function. We then prove a simple $O(q)$-approximation for $(2, q)$-FGC by directly applying existing algorithms for covering uncrossable functions. Next, we devise a framework for augmentation when the requirement function is not uncrossable. Finally, we prove our results for special cases of (p, q)-FGC and (p, q)-Flex-ST using this framework.

Uncrossable functions and families. Uncrossable functions are a general class of requirement functions that are an important ingredient in network design [41, 26, 28, 36].

▶ **Definition 10.** *A function $f : 2^V \to \{0,1\}$ is* uncrossable *if for every $A, B \subseteq V$ such that $f(A) = f(B) = 1$, one of the following is true: (i) $f(A \cup B) = f(A \cap B) = 1$, (ii) $f(A - B) = f(B - A) = 1$. A family of cuts $\mathcal{C} \subset 2^V$ is an uncrossable* family *if the indicator function $f_{\mathcal{C}} : 2^V \to \{0,1\}$ with $f(S) = 1$ iff $S \in \mathcal{C}$, is uncrossable.*

For a graph $G = (V, E)$, a requirement function $f : 2^V \to \{0,1\}$, and a subset of edges $A \subseteq E$, we say a set $S \subseteq V$ is violated with respect to A, f if $f(S) = 1$ and $\delta_A(S) = \emptyset$. The following important result gives a 2-approximation algorithm for the problem of covering an uncrossable requirement function.

▶ **Theorem 11** ([41]). *Let $G = (V, E)$ be an edge-weighted graph and let $f : 2^V \to \{0,1\}$ be an uncrossable function. Suppose there is an efficient oracle that for any $A \subseteq E$ outputs all the minimal violated sets of f with respect to A. Then there is an efficient 2-approximation for the problem of finding a minimum cost subset of edges that covers f.*

A special case of uncrossable family of sets is a *ring family*. We say that an uncrossable family $\mathcal{C} \subseteq 2^V$ is a ring family if the following conditions hold: (i) if $A, B \in \mathcal{C}$ and A, B properly intersect[3] then $A \cap B$ and $A \cup B$ are in $\mathcal{C}$ and (ii) there is a unique minimal set in $\mathcal{C}$. We observe that if $\mathcal{C}$ is an uncrossable family such that there is a vertex s contained in every $A \in \mathcal{C}$ then $\mathcal{C}$ is automatically a ring family. Theorem 11 can be strengthened for this case. There is an optimum algorithm to find a min-cost cover of a ring family – see [37, 38, 22].

In order to use Theorem 11 in the augmentation framework, we need to be able efficiently find all the minimal violated sets of the family. As above, we let F_1 denote a feasible solution for the $(p, q-1)$-Flex-SNDP instance. For any fixed p, q, we can enumerate all minimal violated sets in $n^{O(p+q)}$ time by trying all possible subsets of $p + q - 1$ edges in F_1. In the context of (p, q)-FGC, the total number of violated cuts in the augmentation problem is bounded by $O(n^4)$. See [8] and the proof of Lemma 7 for details.

For the following sections on (p, q)-FGC, we let $\mathcal{C}$ denote the family of violated cuts. Note that such families are symmetric, since $\delta(S) = \delta(V - S)$. For any two sets $A, B \in \mathcal{C}$, if $A \cup B = V$ then by symmetry, $V - A, V - B \in \mathcal{C}$. In this case, $V - A = B - A$ and $V - B = A - B$, so A and B uncross. Therefore, when proving uncrossability of A and B, we assume without loss of generality that $(A \cup B) \neq V$.

Submodularity and posimodularity of the cut function. It is well-known that the cut function of an undirected graph is symmetric and submodular. Submodularity implies that for all $A, B \subseteq V$, $|\delta(A)| + |\delta(B)| \geq |\delta(A \cap B)| + |\delta(A \cup B)|$. Symmetry and submodularity also implies posimodularity: for all $A, B \subseteq V$, $|\delta(A)| + |\delta(B)| \geq |\delta(A - B)| + |\delta(B - A)|$.

3.1 An $O(q)$-approximation for $(2, q)$-FGC

The following lemma shows that the augmentation problem for increasing flex-connectivity from $(2, q-1)$ to $(2, q)$, for any $q \geq 1$ corresponds to covering an uncrossable function.

▶ **Lemma 12.** *The set of all violated cuts when augmenting from $(2, q-1)$-FGC to $(2, q)$-FGC is uncrossable.*

[3] A, B properly intersect if $A \cap B \neq \emptyset$ and $A - B, B - A \neq \emptyset$.

The preceding lemma yields a $2(q+1)$-approximation for $(2,q)$-FGC as follows. We start with a 2-approximation for $(2,0)$-FGC that can be obtained by using an algorithm for 2-ECSS. Then for we augment in q-stages to go from a feasible solution to $(2,0)$-FGC to $(2,q)$-FGC. The cost of augmentation in each stage is at most OPT where OPT is the cost of an optimum solution to $(2,q)$-FGC. We can use the known 2-approximation algorithm in each augmentation stage since the family is uncrossable. Recall from Section 2 that the violated cuts can be enumerated in polynomial time, and hence the primal-dual 2-approximation for covering an uncrossable function can be implemented in polynomial-time. This leads to the claimed approximation and running time.

3.2 Identifying Uncrossable Subfamilies

We have seen that the augmentation problem from $(2,q-1)$-FGC to $(2,q)$-FGC leads to covering an uncrossable function. Boyd et al. [8] showed that augmenting from $(p,0)$-FGC to $(p,1)$-FGC also leads to an uncrossable function for any $p \geq 1$. However this approach fails for most cases of augmenting from $(p,q-1)$ to (p,q) (see [16] for examples). However, in certain cases, we can take a more sophisticated approach where we consider the violated cuts in a small number of stages. In each stage, we choose a subfamily of the violated cuts that is uncrossable. In such cases, we can obtain a $2k$-approximation for the augmentation problem, where k is the upper bound on the number of stages.

Suppose we want to augment from $(p,q-1)$ to (p,q) (for either FGC or the Flex-ST setting). Let $G=(V,E)$ be the original input graph, and let F be the set of edges we have already included. Recall that a cut $\emptyset \neq A \subsetneq V$ is violated iff $|\delta_F(A)| = p+q-1$, $|\delta_{F\cap\mathcal{S}}(A)| < p$, and in the Flex-ST case, A separates s from t. Instead of attempting to cover all violated sets at once, we do so in stages. In each stage we consider the violated cuts based on the number of safe edges. We begin by covering all violated sets with no safe edges, then with one safe edge, and iterate until all violated sets are covered. This is explained in Algorithm 1 below.

Algorithm 1 Augmenting from $(p,q-1)$ to (p,q) in stages.

$F' \leftarrow F$
for $i = 0,\ldots,p-1$ **do**
  $\mathcal{C}_i \leftarrow \{S : S \text{ is violated and } |\delta_{F'\cap\mathcal{S}}(S)| = i\}$
  $F'_i \leftarrow$ approximation algorithm to cover cuts in $\mathcal{C}_i$
  $F' \leftarrow F' \cup F'_i$
end for
return F'

3.3 Approximating (p,q)-FGC for $q \leq 4$

In this section, we show that the above approach works to augment from $(p,q-1)$-FGC to (p,q)-FGC whenever $q \leq 3$ and also for $q=4$ when p is even. The only unspecified part Algorithm 1 is to cover cuts in $\mathcal{C}_i$ in the i'th stage. If we can prove that $\mathcal{C}_i$ forms an uncrossable family then we can obtain a 2-approximation in each stage. First, we prove a generic and useful lemma regarding cuts in $\mathcal{C}_i$.

For the remaining lemmas, we let $F_i \subseteq E$ denote the set of edges F' at the start of iteration i. In other words, F_i is a set of edges such that for all $\emptyset \neq A \subsetneq V$, if $|\delta_{F_i}(A)| = p+q-1$, then $|\delta_{F_i\cap\mathcal{S}}(A)| \geq i$.

▶ **Lemma 13.** *Fix an iteration $i \in \{0, \ldots, p-1\}$. Let $\mathcal{C}_i$ be as defined in Algorithm 1. Then, if $A, B \in \mathcal{C}_i$ and*
1. $|\delta_{F_i}(A \cap B)| = |\delta_{F_i}(A \cup B)| = p + q - 1$, *or*
2. $|\delta_{F_i}(A - B)| = |\delta_{F_i}(B - A)| = p + q - 1$

then A and B uncross, i.e. $A \cap B, A \cup B \in \mathcal{C}_i$ or $A - B, B - A \in \mathcal{C}_i$.

Note that the preceding lemma holds for the high-level approach. Now we focus on cases where we can prove that $\mathcal{C}_i$ is uncrossable.

▶ **Lemma 14.** *Fix an iteration $i \in \{0, \ldots, p-1\}$. Let $\mathcal{C}_i$ be as defined in Algorithm 1. Then, for $q \leq 3$, $\mathcal{C}_i$ is uncrossable.*

Proof. Suppose $A, B \subseteq V$ such that $\delta_{F_i}(A)$ and $\delta_{F_i}(B)$ both have exactly i safe and $p + q - 1 - i$ unsafe edges. Suppose for the sake of contradiction that they do not uncross. By Lemma 13, one of $\delta_{F_i}(A \cap B)$ and $\delta_{F_i}(A \cup B)$ must have at most $p + q - 2$ edges, and the same holds for $\delta_{F_i}(A - B)$ and $\delta_{F_i}(B - A)$. Without loss of generality, suppose $\delta_{F_i}(A \cap B)$ and $\delta_{F_i}(A - B)$ each have at most $p + q - 2$ edges. By the assumptions on F_i, they must both have at least p safe edges, hence they each have at most $q - 2$ unsafe edges. Note that $\delta_{F_i}(A) \subseteq \delta_{F_i}(A - B) \cup \delta_{F_i}(A \cap B)$, hence $\delta_{F_i}(A)$ can have at most $2(q - 2)$ unsafe edges. When $q \leq 3$, $2(q-2) < q$, which implies that $\delta_{F_i}(A)$ has strictly more than $p-1$ safe edges, a contradiction. Notice that $\delta_{F_i}(A) \subseteq \delta_{F_i}(B-A) \cup \delta_{F_i}(A \cup B)$, $\delta_{F_i}(B) \subseteq \delta_{F_i}(A-B) \cup \delta_{F_i}(A \cup B)$, and $\delta_{F_i}(B) \subseteq \delta_{F_i}(B - A) \cup \delta_{F_i}(A \cap B)$; therefore the same argument follows regardless of which pair of sets each have strictly less than $p + q - 2$ edges. ◀

▶ **Corollary 15.** *For any $p \geq 2$ there is a $(2p + 4)$-approximation for $(p, 2)$-FGC and a $(4p + 4)$-approximation for $(p, 3)$-FGC.*

Can we extend the preceding lemma for $q = 4$? It turns out that it does work when p is even but fails for odd $p \geq 3$.

▶ **Lemma 16.** *Fix an iteration $i \in \{0, \ldots, p-2\}$. Let $\mathcal{C}_i$ be as defined in Algorithm 1. Then, for $q = 4$, $\mathcal{C}_i$ is uncrossable. Furthermore, if p is an* even *integer, $\mathcal{C}_{p-1}$ is uncrossable.*

The preceding lemma leads to a $(6p + 4)$-approximation for $(p, 4)$-FGC when p is even by augmenting from a feasible solution to $(p, 3)$, since we pay an additional cost of $2p \cdot OPT$. The preceding lemma also shows that the bottleneck for odd p is in covering $\mathcal{C}_{p-1}$. It may be possible to show that $\mathcal{C}_{p-1}$ separates into a constant number of uncrossable families leading to an $O(p)$-approximation for $(p, 4)$-FGC for all p. The first non-trivial case is when $p = 3$.

3.4 An $O(1)$-Approximation for Flex-ST

In this section, we provide a constant factor approximation for (p, q)-Flex-ST for all fixed p, q that satisfy $2(p + q) > pq$. We follow the general approach outlined in 3.2 with some modifications. In particular, we start with a stronger set of edges F than in the FGC case above. Let $E' \subseteq E$ denote a feasible solution to $(p, q - 1)$-Flex-ST.

Recall from Section 1 the capacitated network design problem, in which each edge has an integer capacity $u_e \geq 1$. Consider an instance of the $(p(p + q))$-Cap-ST problem on G where every safe edge is given a capacity of $p + q$ and every unsafe edge is given a capacity of p. Our goal is to find the cheapest set of edges that support a flow of $(p(p + q))$ from s to t. It is easy to see that any solution to (p, q)-Flex-ST is also a feasible solution for this capacitated problem: every s-t cut either has at least p safe edges or at least $p + q$ total edges, and either case gives a capacity of at least $p(p + q)$. As mentioned in Section 1, there exists a

$2\max_e(u_e) = 2(p+q)$ approximation for this problem. Let $E'' \subseteq E$ be such a solution, and note that $\text{cost}(E'') \leq 2(p+q) \cdot OPT$, where OPT denotes the cost of an optimal solution to (p,q)-Flex-ST.

Let $F = E' \cup E''$. We redefine $\mathcal{C}$ to limit ourselves to the set of violated cuts containing s, i.e. $\mathcal{C} = \{A \subset V : s \in A, t \notin A, |\delta_{F\cap\mathcal{S}}(A)| < p, |\delta_{F\cap\mathcal{U}}(A)| = p+q-1\}$. By symmetry, it suffices to only consider cuts containing s, since covering a set also covers its complement. Following the discussion in Section 3.2, we use Algorithm 1 to cover violated cuts in stages based on the number of safe edges. However, unlike the spanning case, the sets $\mathcal{C}_i$ are not uncrossable in the single pair setting, even for $(2,2)$-Flex-ST. In this case, we aim to further partition $\mathcal{C}_i$ into subfamilies that we can cover efficiently.

For the remaining lemmas, we let $F_i \subseteq E$ denote the set of edges F' at the start of iteration i. In other words, F_i is a set of edges such that for all cuts A separating s from t, if $|\delta_{F_i}(A)| = p+q-1$, then $|\delta_{F_i\cap\mathcal{S}}(A)| \geq i$. We begin with a structural lemma.

▶ **Lemma 17.** *Fix an iteration $i \in \{0,\ldots,p-1\}$. Let $\mathcal{C}_i$ be as defined in Algorithm 1. Let $A, B \in \mathcal{C}_i$. Then, either*
1. *$A \cup B, A \cap B \in \mathcal{C}_i$, i.e. A and B uncross, or*
2. *$\max(\delta_{F_i\cap\mathcal{S}}(A\cap B), \delta_{F_i\cap\mathcal{S}}(A\cup B)) \geq p$.*

Consider a flow network on the graph (V, F_i) with safe edges given a capacity of $(p+q)$ and unsafe edges given a capacity of p. Since $E'' \subseteq F_i$, F_i satisfies the $p(p+q)$-Cap-ST requirement. Therefore, the minimum capacity s-t cut and thus the maximum s-t flow value is at least $p(p+q)$. Since capacities are integral, there is some integral max flow f. By flow decomposition, we can decompose f into a set $\mathcal{P}$ of $|f|$ paths, each carrying a flow of 1, and we can find $\mathcal{P}$ in polynomial time.

For each $\mathcal{Q} \subseteq \mathcal{P}$ where $|\mathcal{Q}| = i$, we define a subfamily of violated cuts $\mathcal{C}_i^{\mathcal{Q}}$ as follows. Let $\mathcal{Q} = P_1, \ldots, P_i$. Then, $A \in \mathcal{C}_i$ is in $\mathcal{C}_i^{\mathcal{Q}}$ iff there exist distinct edges $e_1, \ldots, e_i \in \mathcal{S}$ satisfying:
1. $\forall j \in [i]$, $\delta_{F_i}(A) \cap P_j = \{e_j\}$,
2. $\delta_{F_i\cap\mathcal{S}}(A) = \{e_1, \ldots, e_i\}$.

Informally, $\mathcal{C}_i^{\mathcal{Q}}$ is the set of all violated cuts that intersect the paths of $\mathcal{Q}$ exactly once and on a distinct safe edge each.

▶ **Lemma 18.** *Suppose $pq < 2p + 2q$. If $A \in \mathcal{C}_i$, then there exists some $\mathcal{Q} \subseteq \mathcal{P}$, $|\mathcal{Q}| = i$, such that $A \in \mathcal{C}_i^{\mathcal{Q}}$.*

The above lemma shows that $\cup_{\mathcal{Q}\subseteq\mathcal{P},|\mathcal{Q}|=i}\mathcal{C}_i^{\mathcal{Q}} = \mathcal{C}_i$. Therefore, it suffices to cover each $\mathcal{C}_i^{\mathcal{Q}}$.

▶ **Lemma 19.** *For any $\mathcal{Q} \subseteq \mathcal{P}$, $|\mathcal{Q}| = i$, $\mathcal{C}_i^{\mathcal{Q}}$ is a ring family.*

We combine the above lemmas to obtain a constant factor approximation for covering $\mathcal{C}_i$.

▶ **Lemma 20.** *Suppose $2(p+q) > pq$ and p, q are fixed. Then, there exists an algorithm that runs in $n^{O(p+q)}$ time to cover all cuts in $\mathcal{C}_i$ with cost at most $\binom{p^2+2pq}{i} \cdot OPT$.*

The above lemma gives us Theorem 3 as a corollary. At the beginning of each augmentation step, before running Algorithm 1, we compute a solution to the $(p(p+q))$-Cap-ST problem, which we can do with cost at most $(p+q) \cdot OPT$. Summing over q augmentation iterations gives us the desired $(p+q)^{O(p)}$ approximation ratio.

▶ Remark 21. The approximation factor in Theorem 3 can be optimized slightly. For example, the algorithm we describe gives a 5-approximation for $(2,2)$-Flex-ST. We omit the details of this optimization in this paper and instead focus on showing constant factor for fixed p, q.

4 Approximating the Augmentation Problem for (p, q)-Flex-SNDP

In this section, we prove Theorem 4 and the resulting Corollaries 5 and 6. We begin with some background on Räcke's capacity-based probabilistic tree embeddings and Tree Rounding algorithms for Group Steiner tree. We then present the algorithm and analysis for Bulk-SNDP.

4.1 Räcke Tree Embeddings

The results in this section use Räcke's capacity-based probabilistic tree embeddings. We borrow the notation from [17]. Given $G = (V, E)$ with capacity $x : E \to \mathbb{R}^+$ on the edges, a capacitated tree embedding of G is a tree $\mathcal{T}$, along with two mapping functions $\mathcal{M}_1 : V(\mathcal{T}) \to V(G)$ and $\mathcal{M}_2 : E(\mathcal{T}) \to 2^{E(G)}$ that satisfy some conditions. $\mathcal{M}_1$ maps each vertex in $\mathcal{T}$ to a vertex in G, and has the additional property that it gives a one-to-one mapping between the leaves of $\mathcal{T}$ and the vertices of G. $\mathcal{M}_2$ maps each edge $(a, b) \in E(\mathcal{T})$ to a path in G between $\mathcal{M}_1(a)$ and $\mathcal{M}_1(b)$. For notational convenience we view the two mappings as a combined mapping $\mathcal{M}$. For a vertex $u \in V(G)$ we use $\mathcal{M}^{-1}(u)$ to denote the leaf in $\mathcal{T}$ that is mapped to u by $\mathcal{M}_1$. For an edge $e \in E(G)$ we use $\mathcal{M}^{-1}(e) = \{f \in E(\mathcal{T}) \mid e \in \mathcal{M}_2(f)\}$. It is sometimes convenient to view a subset $S \subseteq V(G)$ both as vertices in G and also corresponding leaves of $\mathcal{T}$.

The mapping $\mathcal{M}$ induces a capacity function $y : E(\mathcal{T}) \to \mathbb{R}_+$ as follows. Consider $f = (a, b) \in E(\mathcal{T})$. $\mathcal{T} - f$ induces a partition (A, B) of $V(T)$ which in turn induces a partition/cut (A', B') of $V(G)$ via the mapping $\mathcal{M}$: A' is the set of vertices in G that correspond to the leaves in A and similarly B'. We then set $y(f) = \sum_{e \in \delta(A')} x(e)$, in other words $y(f)$ is the capacity of cut (A', B') in G. The mapping also induces loads on the edges of G. For each edge $e \in G$, we let $\mathrm{load}(e) = \sum_{f \in E(\mathcal{T}) : e \in M(f)} y(f)$. The relative load or *congestion* of e is $\mathrm{rload}(e) = \mathrm{load}(e)/x(e)$. The congestion of G with respect to a tree embedding $(\mathcal{T}, \mathcal{M})$ is defined as $\max_{e \in E(G)} \mathrm{rload}(e)$. Given a probabilistic distribution $\mathcal{D}$ on trees embeddings of (G, x) we let $\beta_{\mathcal{D}} = \max_{e \in E(G)} \mathbf{E}_{(\mathcal{T}, \mathcal{M}) \sim \mathcal{D}} \mathrm{rload}(e)$ denote the maximum expected congestion. Räcke showed the following fundamental result on probabilistic embeddings of a capacitated graph into trees.

▶ **Theorem 22** ([39]). *Given a graph G and $x : E(G) \to \mathbb{R}^+$, there exists a probability distribution $\mathcal{D}$ on tree embeddings such that $\beta_{\mathcal{D}} = O(\log |V(G)|)$. All trees in the support of $\mathcal{D}$ have height at most $O(\log(nC))$, where C is the ratio of the largest to smallest capacity in x. Moreover, there is randomized polynomial-time algorithm that can sample a tree from the distribution $\mathcal{D}$.*

In the rest of the paper we use β to denote the guarantee provided by the preceding theorem where $\beta = O(\log n)$ for a graph on n nodes. In order to use these probabilistic embeddings to route flow, we need the following corollary, where we use $\mathrm{maxflow}^z_H(A, B)$ to denote the maxflow between two disjoint vertex subsets A, B in a capacitated graph H with capacities given by $z : E(H) \to \mathbb{R}_+$.

▶ **Corollary 23.** *Let $\mathcal{D}$ be the distribution guaranteed in Theorem 22. Let $A, B \in V(G)$ be two disjoint sets. Then*

- **(i)** *for any tree $(\mathcal{T}, \mathcal{M})$ in $\mathcal{D}$, $\mathrm{maxflow}^x_G(A, B) \leq \mathrm{maxflow}^y_{\mathcal{T}}(\mathcal{M}^{-1}(A), \mathcal{M}^{-1}(B))$ and*
- **(ii)** $\frac{1}{\beta} \mathbf{E}_{(\mathcal{T}, \mathcal{M}) \sim \mathcal{D}}[\mathrm{maxflow}^y_{\mathcal{T}}(\mathcal{M}^{-1}(A), \mathcal{M}^{-}(B))] \leq \mathrm{maxflow}^x_G(A, B)$.

4.2 Group Steiner Tree, Set Connectivity and Tree Rounding

The group Steiner tree problem was introduced in [40] and studied in approximation by Garg, Konjevod and Ravi [24]. The input is an edge-weighted graph $G = (V, E)$, a root vertex $r \in V$, and k groups $S_1, S_2, \ldots, S_k$ where each $S_i \subseteq V$. The goal is to find a min-weight subgraph H of G such there is a path in H from r to each group S_i (that is, to some vertex in S_i). The approximability of this problem has attracted substantial attention. Garg et al. [24] described a randomized algorithm to round a fractional solution to a cut-based LP relaxation when G is a tree – it achieves a $O(\log n \log k)$-approximation.

Set Connectivity is a generalization of group Steiner tree problem. Here we are given pairs of sets $(S_1, T_1), (S_2, T_2), \ldots, (S_k, T_k)$ and the goal is to find a min-cost subgraph H such that there is an (S_i, T_i) path in H for each i. Chalermsook, Grandoni and Laekhanukit [13] studied Survivable Set Connectivity problem, motivated by earlier work in [29]. Here each pair (S_i, T_i) has a connectivity requirement r_i which implies that one seeks r_i edge-disjoint paths between S_i and T_i in the chosen subgraph H; [13] obtained a bicriteria-approximation via Räcke tree and group Steiner tree rounding. The recent work of Chen et al [17] uses related but more sophisticated ideas to obtain the first true approximation for this problem. They refer to the problem as Group Connectivity problem and obtain an $O(r^3 \log r \log^7 n)$-approximation where $r = \max_i r_i$ connectivity requirement (see [17] for more precise bounds).

Oblivious tree rounding. In [13] a randomized oblivious algorithm based on the group Steiner tree rounding from [24] is described. This is useful since the sets to be connected during the course of their algorithm are implicitly generated. We encapsulate their result in the following lemma. The tree rounding algorithm in [13, 17] is phrased slightly differently since they combine aspects of group Steiner rounding and the congestion mapping that comes from Räcke trees. We separate these two explicitly to make the idea more transparent. We refer to the algorithm from the lemma below as TreeRounding.

▶ **Lemma 24** ([13, 17]). *Consider an instance of Set Connectivity on an n-node tree $T = (V, E)$ with height h and let $x : E \to [0, 1]$. Suppose $A, B \subseteq V$ are disjoint sets and suppose $K \subseteq E$ such that x restricted to K supports a flow of $f \leq 1$ between A and B. There is a randomized algorithm that is oblivious to A, B, K (hence depends only on x and value f) that outputs a subset $E' \subseteq E$ such that (i) The probability that $E' \cap K$ connects A to B is at least a fixed constant ϕ and (ii) For any edge $e \in E$, the probability that $e \in E'$ is $\min\{1, O(\frac{1}{f} h \log^2 n) x(e)\}$.*

4.3 Rounding Algorithm for the Augmentation Problem

We adapt the algorithm and analysis in [17] to Bulk-SNDP. Let β be the expected congestion given by Theorem 22. Consider an instance of Bulk-SNDP specified by a graph $G : (V, E)$ with cost function $c : E \to \mathbb{R}_{\geq 0}$ and a set of scenarios $\Omega = \{(F_j, \mathcal{K}_j) : j \in [m]\}$. Assume we have a partial solution H satisfying all scenarios in $\Omega_{\ell-1}$. We augment H to satisfy scenarios in Ω_ℓ.

We start by obtaining a solution $\{x_e\}_{e \in E \setminus H}$ for the LP relaxation. Let $E' = E \setminus H$. We define $\text{LARGE} = \{e \in E' : x_e \geq \frac{1}{4\ell\beta}\}$, and $\text{SMALL} = \{e \in E' : x_e < \frac{1}{4\ell\beta}\}$. The LP has paid for each $e \in \text{LARGE}$ a cost of at least $c(e)/(4\ell\beta)$, hence adding all of them to H will cost $O(\ell\beta \cdot \text{OPT}_{\text{LP}})$. If $\text{LARGE} \cup H$ is a feasible solution to the augmentation problem, then we are done since we obtain a solution of cost $O(\ell \log n \cdot \text{OPT}_{\text{LP}})$. Thus, the interesting case is when $\text{LARGE} \cup H$ is *not* a feasible solution.

Following [17] we employ a Räcke tree based rounding. A crucial step is to set up a capacitated graph appropriately. We can assume, with a negligible increase in the fractional cost, that for each edge $e \in E'$, $x(e) = 0$ or $x(e) \geq \frac{1}{n^3}$; this can be ensured by rounding down to 0 the fractional value of any edge with very small value, and compensating for this loss by scaling up the fractional value of the other edges by a factor of $(1 + 1/n)$. It is easy to check that the new solution satisfies the cut covering constraints, and we have only increased the cost of the fractional solution by a $(1 + 1/n)$-factor. In the subsequent steps we can ignore edges with $x_e = 0$ and assume that there are no such edges.

Consider the original graph $G = (V, E)$ where we set a capacity for each $e \in E$ as follows. If $e \in \text{LARGE} \cup H$ we set $\tilde{x}_e = \frac{1}{4\ell\beta}$. Otherwise we set $\tilde{x}_e = x_e$. Since the ratio of the largest to smallest capacity is $O(n^3)$, the height of any Räcke tree for G with capacities $\tilde{x}$ is at most $O(\log n)$. Then, we repeatedly sample Räcke trees. For each tree, we sample edges by the rounding algorithm given by Chalermsook et al in [13] (see Section 4.2 for details). A formal description of the algorithm is provided below where t' and t are two parameters that control the number of trees sampled and the number of times we run the tree rounding algorithm in each sampled tree. We will analyze the algorithm by setting both t and t' to $\Theta(\ell \log n)$.

Algorithm 2 Approximating the Bulk-SNDP Augmentation Problem from $\ell - 1$ to ℓ.

$H \leftarrow$ partial solution satisfying scenarios in Ω_ℓ
$\{x\}_{e \in E} \leftarrow$ fractional solution to the LP
$\text{LARGE} \leftarrow \{e \in E' : x_e \geq \frac{1}{4\ell\beta}\}$
$\text{SMALL} \leftarrow \{e \in E' : x_e < \frac{1}{4\ell\beta}\}$
$H \leftarrow H \cup \text{LARGE}$
if H is a feasible solution satisfying scenarios in $\Omega_{\ell+1}$ **then return** H
else

$$\tilde{x}_e \leftarrow \begin{cases} \frac{1}{4\ell\beta} & e \in H \\ x_e & \text{otherwise} \end{cases}$$

end if
$\mathcal{D} \leftarrow$ Räcke tree distribution for $(G, \tilde{x})$
for $i = 1, \ldots t'$ **do**
  Sample a tree $(\mathcal{T}, \mathcal{M}, y) \sim \mathcal{D}$
  for $j = 1, \ldots, t$ **do**
    $H' \leftarrow$ output of oblivious TreeRounding algorithm on $(G, \mathcal{T})$
    $H \leftarrow H \cup \mathcal{M}(H')$
  end for
end for
return H

4.4 Analysis

For the remainder of of this analysis, we denote as H the partial solution after buying edges in LARGE. We will assume, following earlier discussion, that H does not satisfy all requirements specified by scenarios in Ω_ℓ. This implies that there must be some F such that $|F| \leq \ell$, $F \subseteq F_i$ for some $i \in [m]$, and $\exists (u, v) \in \mathcal{K}_i$ such that u and v are disconnected in $(V, H \setminus F)$. Since H satisfies all scenarios in $\Omega_{\ell-1}$, it must be the case that F has exactly ℓ edges. We call such an F a violating set. There are at most $\binom{|H|}{\ell}$ violating edge sets, and since $|H| \leq n^2$, this is upper bounded by $O(n^{2\ell})$. We say that a set of edges $H' \subseteq E \setminus H$ is a feasible augmentation for violating edge set F if $\forall i \in [m]$ such that $F \subseteq F_i$, $\forall (u, v) \in \mathcal{K}_i$, there is a path from u to v in $(H \cup H') \setminus F$. The following is a simple observation.

▷ Claim 25. $H' \subseteq E \setminus H$ is a feasible solution to the augmentation problem iff for each violating edge set F, H' is a feasible augmentation for F.

The preceding observation allows us to focus on a fixed violating edge set F, and ensure that the algorithm outputs a set H' that is a feasible augmentation for F with high probability. We observe that the algorithm is oblivious to F. Thus, if we obtain a high probability bound for a fixed F, since there are $O(n^{2\ell})$ violating edge sets, we can use the union bound to argue that H' is feasible solution for *all* violating edge sets. For the remainder of this section, until we do the final cost analysis, we work with a fixed violating edge set F. For ease of notation, we let $\mathcal{K}_F = \bigcup_{i \in [m]: F \subseteq F_i} \mathcal{K}_i$ be the set of terminal pairs that need to be connected in $(H \cup H') \setminus F$.

Consider a tree $(\mathcal{T}, \mathcal{M}, y)$ in the Räcke distribution for the graph G with capacities $\tilde{x}$. We let $\mathcal{M}^{-1}(F)$ denote the set of all tree edges corresponding to edges in F, i.e. $\mathcal{M}^{-1}(F) = \cup_{e \in F} \mathcal{M}^{-1}(e)$. We call $(\mathcal{T}, \mathcal{M}, y)$ *good* with respect to F if $y(\mathcal{M}^{-1}(F)) \leq \frac{1}{2}$; equivalently, F blocks a flow of at most $\frac{1}{2}$ in $\mathcal{T}$.

▶ **Lemma 26.** *For a violating edge set F, a randomly sampled Räcke tree $(\mathcal{T}, \mathcal{M}, y)$ is good with respect to F with probability at least $\frac{1}{2}$.*

Given the preceding lemma, a natural approach is to sample a good tree $\mathcal{T}$ and hope that $\mathcal{T} \setminus M^{-1}(F)$ still has good flow between each terminal pair. However, since we rounded down all edges in $\text{LARGE} \cup H$, it is possible that $\mathcal{M}^{-1}(F)$ contains an edge whose removal would disconnect a terminal pair in $\mathcal{T}$, even if $\mathcal{T}$ is good. See [17] for a more detailed discussion and example.

We note that our goal is to find a set of edges $H' \subseteq E$ such that each terminal pair in $\mathcal{K}_F$ has a path in $(H' \cup H) \setminus F$; these paths must exist in the original graph, even if they do not exist in the tree. Therefore, instead of looking directly at paths in $\mathcal{T}$, we focus on obtaining paths through components that are already connected in $(V(G), H \setminus F)$. The rest of the argument is to show that sufficiently many iterations of TreeRounding on any good tree $\mathcal{T}$ for F will yield a feasible set H' for F.

4.5 Shattered Components, Set Connectivity and Rounding

Let $\mathbb{Q}_F$ be the set of connected components in the subgraph induced by $H \setminus F$. We use vertex subsets to denote components. Let $\mathcal{T}$ be a good tree for F. We say that a connected component $Q \in \mathbb{Q}_F$ is *shattered* if it is disconnected in $\mathcal{T} \setminus \mathcal{M}^{-1}(F)$, else we call it *intact*. For each $(u, v) \in \mathcal{K}_F$, let $Q_u \in \mathbb{Q}_F$ be the component containing u, and $Q_v \in \mathbb{Q}_F$ be the component containing v. Note that Q_u may be the same as Q_v for some $(u, v) \in \mathcal{K}_F$, but if F is a violating edge set then there is at least one pair $(u, v) \in \mathcal{K}_F$ such that $Q_u \neq Q_v$. Now, we define a Set Connectivity instance that is induced by F and $\mathcal{T}$. Consider two disjoint vertex subsets $A, B \subset V$. We say that (A, B) partitions the set of shattered components if each shattered component Q is fully contained in A or fully contained in B. Formally let

$$Z_F = \{(A \cup Q_u, B \cup Q_v) : (A, B) \text{ partitions the shattered components}, (u, v) \in \mathcal{K}_F\}.$$

In other words, Z_F is set of all partitions of shattered components that separate some pair $(u, v) \in \mathcal{K}_F$. Since the leaves of $\mathcal{T}$ are in one to one correspondence with $V(G)$ we can view Z_F as inducing a Set Connectivity instance in $\mathcal{T}$; technically we need to consider the pairs $\{(\mathcal{M}^{-1}(A), \mathcal{M}^{-1}(B)) \mid (A, B) \in Z_F\}$; however, for simplicity we conflate the leaves of $\mathcal{T}$ with $V(G)$. We claim that it suffices to find a feasible solution that connects the pairs defined by Z_F in the tree $\mathcal{T}$.

▶ **Lemma 27.** *Let $E' \subseteq \mathcal{T} \setminus \mathcal{M}^{-1}(F)$. Suppose there exists a path in $E' \subseteq \mathcal{T} \setminus \mathcal{M}^{-1}(F)$ connecting A to B for all $(A,B) \in Z_F$. Then, there is an u-v path for each $(u,v) \in \mathcal{K}_F$ in $(\mathcal{M}(E') \cup H) \setminus F$.*

Routing flow. We now argue that $(\mathcal{T}, \mathcal{M}, y)$ routes sufficient flow for each pair in Z_F without using the edges in $\mathcal{M}^{-1}(F)$; in other words y is fractional solution (modulo a scaling factor) to the Set Connectivity instance Z_F in the graph/forest $\mathcal{T} \setminus \mathcal{M}^{-1}(F)$. We can then appeal to TreeRounding lemma to argue that it will connect the pairs in Z_F without using any edges in F.

▶ **Lemma 28.** *Let $(A,B) \in Z_F$. Let $S \subset V_{\mathcal{T}}$ such that $A \subseteq S$ and $B \subseteq V_{\mathcal{T}} \setminus S$. Then $y(\delta_{\mathcal{T} \setminus \mathcal{M}^{-1}(F)}(S)) \geq \frac{1}{4\ell\beta}$.*

Bounding Z_F. A second crucial property is a bound on $|Z_F|$, the number of pairs in the Set Connectivity instance induced by F and a good tree $\mathcal{T}$ for F.

▶ **Lemma 29.** *For a good tree $\mathcal{T}$, $|Z_F| \leq 2^{2\ell\beta}|\mathcal{K}_F|$.*

4.6 Correctness and Cost

The following two lemmas show that by taking a union bound over all violating edge sets F and applying the Tree Rounding lemma 24, one can show that the algorithm outputs a feasible augmentation solution with probability at least $\frac{1}{2}$.

▶ **Lemma 30.** *Suppose $\mathcal{T}$ is good for a violating edge set F. Then after $t = O(\ell log n)$ rounds of TreeRounding with flow parameter $\frac{1}{4\ell\beta}$, the probability that H' is* not *a feasible augmentation for F is at most $(1-\phi)^t|Z_F| \leq 1/4$.*

▶ **Lemma 31.** *The algorithm outputs a solution H' such that $H \cup H'$ is a feasible augmentation to the given instance with probability at least $\frac{1}{2}$.*

Now we analyze the expected cost of the edges output by the algorithm for augmentation with respect to $\mathrm{OPT}_{\mathrm{LP}}$, the cost of the fractional solution.

▶ **Lemma 32.** *The total expected cost of the algorithm is $O(\ell^3 \log^7 n) \cdot OPT_{LP}$.*

Combining the correctness and cost analysis we obtain the following.

▶ **Lemma 33.** *There is a randomized $O(\ell^3 \log^7 n)$-approximation algorithm for the Bulk-SNDP Augmentation problem from $\ell - 1$ to ℓ. The algorithm runs in time polynomial in n and α, where α is the amount of time it takes to solve the LP.*

To prove Theorem 4, we start with a solution from $\ell = 0$ and iteratively solve k augmentation problems. Since the LP for Bulk-SNDP can be solved in polynomial time (see Section 2), we obtain a polynomial time $O(k^4 \log^7 n)$-approximation algorithm.

For Flex-SNDP, recall that $(p,0)$-Flex-SNDP is equivalent to EC-SNDP where every terminal pair has connectivity requirement $r_i = p$. Therefore, we can start with a 2-approximate solution to $(p,0)$-Flex-SNDP and apply Lemma 33 q times. In this case, the maximum width is $p+q$, so we get an overall approximation ratio of $O(q(p+q)^3 \log^7 n)$. Recall from Section 2 that the LP can be solved in $n^{O(q)}$ time, giving us Corollary 5.

Finally, for Relative SNDP, there is an LP relaxation described in [19] that can be solved in polynomial time, even when k is not fixed. We can modify Algorithm 2 for RSNDP by solving this LP relaxation instead and following the same rounding algorithm. This, along with the reduction to Bulk-SNDP discussed in Section 1, completes the proof of Corollary 6.

References

1 David Adjiashvili. Fault-tolerant shortest paths – Beyond the uniform failure model, 2013. doi:10.48550/arXiv.1301.6299.

2 David Adjiashvili. Non-uniform robust network design in planar graphs. *arXiv preprint*, 2015. arXiv:1504.05009.

3 David Adjiashvili, Felix Hommelsheim, and Moritz Mühlenthaler. Flexible graph connectivity. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 13–26. Springer, 2020.

4 David Adjiashvili, Felix Hommelsheim, and Moritz Mühlenthaler. Flexible graph connectivity. *Mathematical Programming*, 192(1):409–441, 2022.

5 David Adjiashvili, Felix Hommelsheim, Moritz Mühlenthaler, and Oliver Schaudt. Fault-Tolerant Edge-Disjoint s-t Paths – Beyond Uniform Faults. In Artur Czumaj and Qin Xin, editors, *18th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2022)*, volume 227 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:19, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SWAT.2022.5.

6 David Adjiashvili, Sebastian Stiller, and Rico Zenklusen. Bulk-robust combinatorial optimization. *Mathematical Programming*, 149(1):361–390, 2015.

7 Ishan Bansal, Joseph Cheriyan, Logan Grout, and Sharat Ibrahimpur. Improved approximation algorithms by generalizing the primal-dual method beyond uncrossable functions, 2022. To appear in Proc. of ICALP 2023. doi:10.48550/arXiv.2209.11209.

8 Sylvia Boyd, Joseph Cheriyan, Arash Haddadan, and Sharat Ibrahimpur. Approximation algorithms for flexible graph connectivity. *Mathematical Programming*, pages 1–24, 2023. Preliminary version appeared in Proc. of FSTTCS 2021. doi:10.1007/s10107-023-01961-5.

9 R. D. Carr, L. K. Fleischer, V. J. Leung, and C. A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 106–115. Society for Industrial and Applied Mathematics, 2000.

10 Deeparnab Chakrabarty, Chandra Chekuri, Sanjeev Khanna, and Nitish Korula. Approximability of capacitated network design. *Algorithmica*, 72(2):493–514, 2015.

11 Deeparnab Chakrabarty, Ravishankar Krishnaswamy, Shi Li, and Srivatsan Narayanan. Capacitated network design on undirected graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 71–80. Springer, 2013.

12 Tanmoy Chakraborty, Julia Chuzhoy, and Sanjeev Khanna. Network design for vertex connectivity. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 167–176, 2008.

13 Parinya Chalermsook, Fabrizio Grandoni, and Bundit Laekhanukit. On survivable set connectivity. In *Proceedings of the 2015 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 25–36. Society for Industrial and Applied Mathematics, 2015. doi:10.1137/1.9781611973730.3.

14 Chandra Chekuri, Guy Even, Anupam Gupta, and Danny Segev. Set connectivity problems in undirected graphs and the directed steiner network problem. *ACM Transactions on Algorithms (TALG)*, 7(2):1–17, 2011.

15 Chandra Chekuri and Rhea Jain. Approximating flexible graph connectivity via räcke tree based rounding, 2022. doi:10.48550/arXiv.2211.08324.

16 Chandra Chekuri and Rhea Jain. Augmentation based approximation algorithms for flexible network design, 2022. doi:10.48550/arXiv.2209.12273.

17 Qingyun Chen, Bundit Laekhanukit, Chao Liao, and Yuhao Zhang. Survivable network design revisited: Group-connectivity, 2022. Full version of paper in Proceedings of IEEE FOCS 2022. doi:10.48550/arXiv.2204.13648.

18 J. Chuzhoy and S. Khanna. An $O(k^3 \log n)$-approximation algorithm for vertex-connectivity survivable network design. *Theory of Computing*, 8:401–413, 2012.

19 Michael Dinitz, Ama Koranteng, and Guy Kortsarz. Relative Survivable Network Design. In Amit Chakrabarti and Chaitanya Swamy, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022)*, volume 245 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 41:1–41:19, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX/RANDOM.2022.41.

20 Yevgeniy Dodis and Sanjeev Khanna. Design networks with bounded pairwise distance. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 750–759, 1999.

21 L. Fleischer, K. Jain, and D. P. Williamson. Iterative rounding 2-approximation algorithms for minimum-cost vertex connectivity problems. *Journal of Computer and System Sciences*, 72(5):838–867, 2006.

22 András Frank. Kernel systems of directed graphs. *Acta Sci. Math.(Szeged)*, 41(1-2):63–76, 1979.

23 András Frank. *Connections in combinatorial optimization*, volume 38. Oxford University Press Oxford, 2011.

24 Naveen Garg, Goran Konjevod, and Ramamoorthi Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. *Journal of Algorithms*, 37(1):66–84, 2000. Preliminary version in Proc. of ACM-SIAM SODA 1998.

25 M. X. Goemans, A. V. Goldberg, S. Plotkin, D. B. Shmoys, E. Tardos, and D. P. Williamson. Improved approximation algorithms for network design problems. In *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 223–232, 1994.

26 M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In *Approximation algorithms for NP-hard problems*, pages 144–191. PWS Publishing Company, Boston, MA, 1997.

27 Fabrizio Grandoni, Bundit Laekhanukit, and Shi Li. $O(\log^2 k \log\log k)$-approximation algorithm for directed steiner tree: a tight quasi-polynomial-time algorithm. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 253–264, 2019.

28 A. Gupta and J. Könemann. Approximation algorithms for network design: A survey. *Surveys in Operations Research and Management Science*, 16(1):3–20, 2011.

29 Anupam Gupta, Ravishankar Krishnaswamy, and Ramamoorthi Ravi. Tree embeddings for two-edge-connected network design. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1521–1538. SIAM, 2010.

30 Eran Halperin, Guy Kortsarz, Robert Krauthgamer, Aravind Srinivasan, and Nan Wang. Integrality ratio for group steiner trees and directed steiner trees. *SIAM Journal on Computing*, 36(5):1494–1511, 2007.

31 Eran Halperin and Robert Krauthgamer. Polylogarithmic inapproximability. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 585–594, 2003.

32 K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

33 Kamal Jain, Ion Măndoiu, Vijay V Vazirani, and David P Williamson. A primal-dual schema based approximation algorithm for the element connectivity problem. *Journal of Algorithms*, 45(1):1–15, 2002.

34 David R Karger. Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm. In *Soda*, volume 93, pages 21–30, 1993.

35 David R Karger. Minimum cuts in near-linear time. *Journal of the ACM (JACM)*, 47(1):46–76, 2000.

36 G. Kortsarz and Z. Nutov. Approximating minimum cost connectivity problems. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2010.

37 Z. Nutov. Approximating Steiner networks with node-weights. *SIAM Journal on Computing*, 39(7):3001–3022, 2010.

38 Z. Nutov. Approximating minimum-cost connectivity problems via uncrossable bifamilies. *ACM Transactions on Algorithms (TALG)*, 9(1):1, 2012.

39 Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 255–264, New York, NY, USA, 2008. Association for Computing Machinery. doi:10.1145/1374376.1374415.

40 Gabriele Reich and Peter Widmayer. Beyond steiner's problem: A vlsi oriented generalization. In *International Workshop on Graph-theoretic Concepts in Computer Science*, pages 196–210. Springer, 1989.

41 D. P. Williamson, M. X. Goemans, M. Mihail, and V. V. Vazirani. A primal-dual approximation algorithm for generalized Steiner network problems. *Combinatorica*, 15(3):435–454, 1995.

Sublinear Algorithms and Lower Bounds for Estimating MST and TSP Cost in General Metrics

Yu Chen
EPFL, Lausanne, Switzerland

Sanjeev Khanna
University of Pennsylvania, Philadelphia, PA, USA

Zihan Tan
DIMACS, Rutgers University, NJ, USA

Abstract

We consider the design of sublinear space and query complexity algorithms for estimating the cost of a minimum spanning tree (MST) and the cost of a minimum traveling salesman (TSP) tour in a metric on n points. We start by exploring this estimation task in the regime of $o(n)$ space, when the input is presented as a stream of all $\binom{n}{2}$ entries of the metric in an arbitrary order (a metric stream). For any $\alpha \geq 2$, we show that both MST and TSP cost can be α-approximated using $\tilde{O}(n/\alpha)$ space, and moreover, $\Omega(n/\alpha^2)$ space is necessary for this task. We further show that even if the streaming algorithm is allowed p passes over a metric stream, it still requires $\tilde{\Omega}(\sqrt{n/\alpha p^2})$ space.

We next consider the well-studied semi-streaming regime. In this regime, it is straightforward to compute MST cost exactly even in the case where the input stream only contains the edges of a weighted graph that induce the underlying metric (a graph stream), and the main challenging problem is to estimate TSP cost to within a factor that is strictly better than 2. We show that in graph streams, for any $\varepsilon > 0$, any one-pass $(2-\varepsilon)$-approximation of TSP cost requires $\Omega(\varepsilon^2 n^2)$ space. On the other hand, we show that there is an $\tilde{O}(n)$ space two-pass algorithm that approximates the TSP cost to within a factor of 1.96.

Finally, we consider the query complexity of estimating metric TSP cost to within a factor that is strictly better than 2 when the algorithm is given access to an $n \times n$ matrix that specifies pairwise distances between n points. The problem of MST cost estimation in this model is well-understood and a $(1+\varepsilon)$-approximation is achievable by $\tilde{O}(n/\varepsilon^{O(1)})$ queries. However, for estimating TSP cost, it is known that an analogous result requires $\Omega(n^2)$ queries even for $(1,2)$-TSP, and for general metrics, no algorithm that achieves a better than 2-approximation with $o(n^2)$ queries is known. We make progress on this task by designing an algorithm that performs $\tilde{O}(n^{1.5})$ distance queries and achieves a strictly better than 2-approximation when either the metric is known to contain a spanning tree supported on weight-1 edges or the algorithm is given access to a minimum spanning tree of the graph. Prior to our work, such results were only known for the special cases of graphic TSP and $(1,2)$-TSP.

In terms of techniques, our algorithms for metric TSP cost estimation in both streaming and query settings rely on estimating the *cover advantage* which intuitively measures the cost needed to turn an MST into an Eulerian graph. One of our main algorithmic contributions is to show that this quantity can be meaningfully estimated by a sublinear number of queries in the query model. On one hand, the fact that a metric stream reveals pairwise distances for all pairs of vertices provably helps algorithmically. On the other hand, it also seems to render useless techniques for proving space lower bounds via reductions from well-known hard communication problems. Our main technical contribution in lower bounds is to identify and characterize the communication complexity of new problems that can serve as canonical starting point for proving metric stream lower bounds.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Minimum spanning tree, travelling salesman problem, streaming algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.37

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2203.14798

Funding *Yu Chen*: Supported by ERC Starting Grant 759471.
Sanjeev Khanna: Supported in part by NSF awards CCF-1934876 and CCF-2008305.
Zihan Tan: Supported by a grant to DIMACS from the Simons Foundation (820931).

1 Introduction

The minimum spanning tree (MST) problem and the metric traveling salesman (TSP) problem are among the most well-studied combinatorial optimization problems with a long and rich history. The two problems are intimately connected to one another, as many approximation algorithms for metric TSP use a minimum spanning tree as a starting point for efficiently constructing an approximate solution. In particular, any algorithm for estimating the MST cost to within a factor of α immediately implies an algorithm for estimating the metric TSP cost to within a factor of 2α. In this work, we consider the design of sublinear space and query complexity algorithms for estimating the cost of a minimum spanning tree (MST) and the cost of a minimum metric traveling salesman (TSP) tour in an n-vertex weighted undirected graph G. An equivalent view of both problems is that we are given an $n \times n$ matrix w specifying pairwise distances between them, where the entry $w[u, v]$ corresponds to the weight of the shortest path from u to v in G. It is clear that any algorithm that works with a weighted graph as input also works when the input is presented as the complete metric. However, the converse is not true. For instance, no single-pass streaming algorithm can obtain a finite approximation to the diameter (or even determine the connectivity) of a graph in $o(n)$ space when the graph is presented as a sequence of edges (a *graph stream*). But if instead we are presented a stream of n^2 entries of the metric matrix w (a *metric stream*), there is a trivial $\tilde{O}(1)$ space algorithm for this problem – simply track the largest entry seen.

1.1 Our Results

In the first part of this work, we explore the power and limitations of graph and metric streams for MST and TSP cost estimation. We start by exploring this estimation task in the regime of $o(n)$ space in the streaming model. It is easy to show that no finite approximation to MST/TSP cost is achievable in this regime when the input stream simply contains the edges of a weighted graph that induce the underlying metric (a graph stream). However, we show that this state of affairs changes completely if the input is instead presented as all entries of the shortest-path-distance metric induced by the input graph (a metric stream).

▶ **Theorem 1.** *For any $\alpha > 1$, there is a randomized one-pass α-approximation streaming algorithm for MST cost estimation in metric streams using $\tilde{O}(n/\alpha)$ space.*

Note that this also immediately gives a one-pass $\tilde{O}(n/\alpha)$-space algorithm for TSP cost estimation for any $\alpha \geq 2$ by simply doubling the MST cost estimate. The result above is in sharp contrast to what is achievable in graph streams. Using a simple reduction from the Index problem, we can show the following lower bound for graph streams.

▶ **Theorem 2.** *For any $\alpha > 1$, any randomized p-pass α-approximation streaming algorithm for MST cost estimation in graph streams requires $\tilde{\Omega}(n/p)$ space.*

We next show that there are limits to the power of metric streams, and in particular, any non-trivial approximation of MST cost still requires polynomial space even if we allow multiple passes over the stream.

▶ **Theorem 3.** *For any $\alpha > 1$, any randomized one-pass α-approximation streaming algorithm for MST cost estimation in metric streams requires $\Omega(n/\alpha^2)$ space.*

▶ **Theorem 4.** *For any $\alpha > 1$, any randomized p-pass α-approximation streaming algorithm for MST cost estimation requires $\tilde{\Omega}(\sqrt{n/\alpha p^2})$ space.*

Table 1 summarizes our results for MST (and TSP) cost estimation in the regime of $o(n)$ space.

Table 1 Summary of results for MST-cost estimation streaming algorithms.

Stream Type	MST estimation		
	# of passes	Approximation ratio	Upper or Lower bounds
Metric Stream	1	1	$\tilde{O}(n)$ (trivial)
	1	α	$\tilde{O}(n/\alpha)$ (Theorem 1), $\tilde{\Omega}(n/\alpha^2)$ (Theorem 3)
	p	α	$\tilde{\Omega}(\sqrt{n/\alpha p^2})$ (Theorem 4)
Graph Stream	1	1	$\tilde{\Theta}(n)$ (trivial)
	p	any	$\Omega(n/p)$ (Theorem 2)

We next consider the well-studied semi-streaming regime when the streaming algorithm is allowed to use $\tilde{O}(n)$ space. In this regime, it is straightforward to design a deterministic one-pass streaming algorithm to compute MST cost exactly even in graph streams, and this in turn, immediately gives an $\tilde{O}(n)$ space algorithm to estimate TSP cost to within a factor of 2. Thus in the semi-streaming regime, the key challenging problem is to estimate TSP cost to within a factor that is strictly better than 2. A special case of this problem, *graphic TSP cost* estimation, where the input metric corresponds to the shortest-path distances induced by an unweighted undirected graph, was studied in [4], and the authors gave an $\tilde{O}(n)$ space randomized one-pass streaming algorithm that achieves an (11/6)-approximation even in the setting of graph streams. This ratio was recently improved[1] by Behnezhad, Roghani, Rubinstein, and Saberi to 1.83 [2]. However, no analogous result is known for general TSP. We show that there is in fact a good reason for this state of affairs:

▶ **Theorem 5.** *For any $0 < \epsilon < 1$, any randomized one-pass $(2-\varepsilon)$-approximation streaming algorithm for TSP cost estimation in graph streams requires $\Omega(\epsilon^2 n^2)$ space.*

However, we show that the situation changes considerably once we allow two passes and indeed there is now a deterministic $\tilde{O}(n)$ space algorithm that achieves better than a 2-approximation to TSP cost.

▶ **Theorem 6.** *There is a deterministic two-pass 1.96-approximation algorithm for TSP cost estimation in graph streams using $\tilde{O}(n)$ space.*

We note that an interesting remaining question here is if a similar result is achievable using one pass when the input is a metric stream. As a step towards understanding the power of metric streams in semi-streaming regime, we show that any one-pass algorithm that computes TSP cost exactly requires $\Omega(n^2)$ space. Table 2 summarizes our results for TSP cost estimation in the regime of semi-streaming space.

[1] In their paper, they give an $\tilde{O}(n)$-time 1.83-approximation algorithm, which can be easily turned into a one-pass streaming algorithm with space $\tilde{O}(n)$ with the same approximation ratio.

Table 2 Summary of results for TSP-cost estimation streaming algorithms. The statements and proofs for entries marked by (∗) is deferred to the full version.

Stream Type	TSP estimation		
	# of passes	Approximation ratio	Upper or Lower bounds
Metric Stream	1	1	$\Omega(n^2)$*
	1	$2-\varepsilon$	Open
Graph Stream	1	2	$\tilde{\Theta}(n)$ (trivial)
	1	$2-\varepsilon$	$\tilde{\Omega}(\varepsilon^2 n^2)$ (Theorem 5)
	2	1.96	$\tilde{O}(n)$ (Theorem 6)

The second part of our paper focuses on the design of sublinear query complexity algorithms for TSP cost estimation. The related problem of estimating the MST cost using sublinear queries was first studied in the graph adjacency-list model by Chazelle, Rubinfeld, and Trevisan [3]. The authors gave an $\tilde{O}(dW/\varepsilon^2)$-time algorithm to estimate the MST cost to within a factor of $(1+\varepsilon)$ in a graph where the average degree is d, and all edge costs are integers in $\{1,\ldots,W\}$. For certain parameter regimes this gives a sublinear time algorithm for estimating the MST cost, but in general, this run-time need not be sublinear. In fact, it is not difficult to show that in general, even checking if a graph is connected requires $\Omega(n^2)$ queries in the graph adjacency-list model, and hence no finite approximation to MST cost can be achieved in $o(n^2)$ queries. However, the situation changes if one restricts attention to the *metric* MST problem where the edge weights satisfy the triangle inequality, and the algorithm is given access to an $n \times n$ matrix w specifying pairwise distances between vertices. Czumaj and Sohler [6] showed that for any $\varepsilon > 0$, there exists an $\tilde{O}(n/\varepsilon^{O(1)})$ query algorithm that returns a $(1+\varepsilon)$-approximate estimate of the metric MST cost. This result immediately implies an $\tilde{O}(n/\varepsilon^{O(1)})$ time algorithm to estimate the TSP cost to within a factor of $(2+\varepsilon)$ for any $\varepsilon > 0$. In sharp contrast to this result, so far no $o(n^2)$ query algorithms are known to approximate metric TSP cost to a factor that is strictly better than 2. In this work, we consider sublinear query algorithms for TSP cost when the algorithm is given query access to the $n \times n$ distance matrix w. *We will assume throughout the paper that all entries of w are positive integers.*

For the special case of graphic TSP, where the metric corresponds to shortest path distances of some underlying connected unweighted graph, the algorithm of Chen, Kannan, and Khanna [4] combined with the recent result of Behnezhad [1] (which builds on the work of Yoshida et al. [8] and Onak et al. [7]), gives an $\tilde{O}(n)$-query (27/14)-approximation algorithm for estimating graphic TSP cost. The authors in [4] also show that there exists an $\varepsilon_0 > 0$, such that any algorithm that estimates the cost of graphic TSP (or even $(1,2)$-TSP) to within a $(1+\varepsilon_0)$-factor, necessarily requires $\Omega(n^2)$ queries. Later on, Behnezhad, Roghani, Rubinstein, and Saberi [2] improved the graphic TSP result by giving an $\tilde{O}(n)$-query 1.83-approximation, and they also gave an $\tilde{O}(n)$-query $(1.5+\varepsilon)$-approximation algorithm for $(1,2)$-TSP. This leaves open the following question: Is there an $o(n^2)$ query algorithm to estimate TSP cost to a factor strictly better than 2 when the metric is *arbitrary*?

We make progress on this question by designing an $\tilde{O}(n^{1.5})$-query algorithm that achieves a strictly better than 2-approximation when either the metric is known to contain a spanning tree supported on weight-1 edges or the algorithm is given access to a minimum spanning tree of the graph. Prior to our work, such results were only known for the special cases of graphic TSP and $(1,2)$-TSP.

▶ **Theorem 7.** *There is a randomized algorithm, that, given access to an n-point metric w with the promise that w contains a minimum spanning tree supported only on weight-1 edges, estimates with high probability the metric TSP cost to within a factor of $(2-\varepsilon_0)$ for some universal constant $\varepsilon_0 > 0$, by performing $\tilde{O}(n^{1.5})$ queries to w.*

We note that the setting of Theorem 7 captures as a special case graphic TSP but is considerably more general, and hence difficult.

▶ **Theorem 8.** *There is a randomized algorithm, that, given access to an n-point metric w and an arbitrary minimum spanning tree of the complete graph with edge weights given by w, estimates with high probability the metric TSP cost to within a factor of $(2-\varepsilon_0)$ for some universal constant $\varepsilon_0 > 0$, by performing $\tilde{O}(n^{1.5})$ queries to w.*

In what follows, we give an overview of the techniques underlying our results.

1.2 Technical Overview

1.2.1 Overview of Algorithmic Techniques

Our streaming algorithm for MST estimation (Theorem 1) utilizes a rather natural idea. We sample $O(n/\alpha)$ vertices and maintain a MST T' over them. For the remaining vertices, we maintain an estimate of the cost of connecting them to the nearest vertex in T'. We show that these estimates can be suitably combined to obtain an α-approximation of MST cost. In this subsection we focus on providing a high-level overview of the algorithms for TSP estimation.

It is well-known that $\mathsf{MST} \leq \mathsf{TSP} \leq 2 \cdot \mathsf{MST}$ holds for any graph/metric, since we can construct a TSP-tour by doubling all edges of a MST (and then shortcut the obtained walk into a tour). Since the MST cost of a graph/metric can be exactly computed by a one-pass $\tilde{O}(n)$ space algorithm (the greedy algorithm) in the streaming model, and can be approximated to within a factor of $(1+\varepsilon)$ by performing $\tilde{O}(n)$ queries in the query model [6], to obtain a factor $(2-\varepsilon)$ approximation for TSP, it suffices to establish either $\mathsf{TSP} \geq (1+\varepsilon) \cdot \mathsf{MST}$ or $\mathsf{TSP} \leq (2-\varepsilon) \cdot \mathsf{MST}$ holds. From the approach due to [5], the minimum weight of a perfect matching on the set of all odd-degree vertices in an MST can immediately gives us the answer. However, obtaining a good approximation to the minimum weight of such a perfect matching appears hard to do, both for semi-streaming algorithms and for a query algorithm that performs $o(n^2)$ queries, even if we are given an MST at the start. To get around this issue, we consider an alternative measure, called the *cover advantage*, that turns out to be more tractable in both models.

Cover Advantage. Let T be a MST of the input graph/metric. For an edge $f \in E(T)$ and an edge $e \notin E(T)$, we say that f is *covered* by e, iff f belongs to the unique tree-path in T connecting the endpoints of e. For a set E' of edges, we denote by $\mathsf{cov}(E', T)$ the set of all edges in $E(T)$ that are covered by at least one edge in E'. The *cover advantage* of E', denoted by $\mathsf{adv}(E')$, is defined to be the total weight of all edges in $\mathsf{cov}(E', T)$ minus the total weight of all edges in E'. Intuitively, if a single-edge set $\{e\}$ where $e = (u, v)$ has cover advantage c, then we can construct a tour by starting from some Euler-tour of T and replacing the segment corresponding to the tree path of T connecting u to v by the single edge e, and thereby "saving a cost of c" from $2 \cdot \mathsf{MST}$, the cost of the Euler-tour obtained by doubling MST edges. Generalizing this idea, we show that if there exists a set E' with cover advantage bounded away from 0 (at least $\varepsilon \cdot \mathsf{MST}$), then $\mathsf{TSP} \leq (2-\varepsilon/2) \cdot \mathsf{MST}$. Conversely, if there does not exist any set E' with cover advantage close to $\mathsf{MST}/2$ (say

at least $(1/2 - \varepsilon/2) \cdot \mathsf{MST}$), then $\mathsf{TSP} \geq (1 + \varepsilon) \cdot \mathsf{MST}$. In fact, we show that the same hold for a more restricted notion called *special cover advantage*, which is defined to be the maximum cover advantage of any subset E' of edges that have at least one endpoint being a special vertex in T (a vertex v is called a *special vertex* of T iff $\deg_T(v) \neq 2$). Therefore, to obtain a better-than-factor-2 approximation for TSP, it suffices to obtain a constant-factor approximation for the maximum cover advantage or the maximum special cover advantage.

Estimating maximum cover advantage in the streaming setting. We construct a one-pass streaming algorithm $O(1)$-estimating the maximum cover advantage, which leads to a two-pass algorithm in Theorem 6 where in the first pass we only compute an MST of the input graph. We store edges with substantial cover advantage with respect to the MST in a greedy manner. Since all edges appear in the stream, it can be shown that, if we end up not discovering a large cover advantage, then the real maximum cover advantage is indeed small (bounded away from $\mathsf{MST}/2$).

Estimating maximum cover advantage in the query model. The task of obtaining a constant-factor approximation to maximum cover advantage turns out to be distinctly more challenging in the query model, even if we are given explicit access to an MST of the metric. The design of sublinear query algorithms for estimating cover advantage is indeed our central algorithmic contribution. We design an $\tilde{O}(n^{1.5})$-query algorithm for estimating the maximum cover advantage when either an MST is explicitly given or we can assume that the MST is supported on weight-1 edges. Note that the latter case generalizes graphic TSP studied in [4].

The algorithms for these two cases share several similarities. To illustrate the ideas behind them, it might be instructive to consider the following two examples. In the first example, we are given an MST T on V that has at most $O(\sqrt{n})$ leaves. We can simply query the distances between all pairs $u, v \in V$ where u is a special vertex of T, and then use the obtained information to compute the maximum special cover advantage, which takes $\tilde{O}(n^{1.5})$ queries since there can be at most $O(\sqrt{n})$ special vertices (or in fact we can even query the distances between all pairs of special vertices in T and compute the minimum weight perfect matching on them). In the second example, we are given an MST T on V which is a star graph centered at a vertex $r \in V$, and all edges have weight 1. Note that, since all edge weights are integers, in this case the distances between every pair of vertices in $V \setminus \{r\}$ is either 1 or 2, and it is not hard to see that the maximum cover advantage is exactly the size of a maximum weight-1 matching on $V \setminus \{r\}$. Therefore, we can adapt the algorithm from [1] to obtain an $O(1)$-approximation of the maximum weight-1 matching size, using $\tilde{O}(n)$ queries. Note that, in this case we obtain an estimate of the maximum cover advantage without computing a set of edges that achieves it.

Taking a step back, we observe that, in the first example where the number of special vertices is small, the cover advantage can be computed in a local and exhaustive manner, while in the second example where the number of special vertices is large, the cover advantage has to be estimated in a global and "superficial" manner. Intuitively, our query algorithms interpolate between these two approaches in an organic manner.

We now provide more details of our query algorithms in the two special cases. We first consider the special case where we are given the structure of an MST.

When MST is given. We root the given MST T at an arbitrary vertex. For each vertex $v \in V$, we say that it is *light* iff the subtree of T rooted at v, denoted by T_v, contains at most $\sqrt{n}$ vertices, and we call T_v a *light subtree* of T. On the one hand, the cover advantages

that are local at some light subtree (achieved by edges with both endpoints in the same light subtree) can be efficiently estimated in an exhaustive manner. On the other hand, if we peel off all light subtrees from T, then the remaining subtree, that we denote by T', contains at most $\sqrt{n}$ leaves, and therefore the special cover advantage achieved by any set of edges with at least one endpoint being a special vertex of T' can also be computed in an exhaustive manner. The only type of cover advantages that is not yet computed are the one achieved by edges with endpoints in different light subtrees. We then observe that the light subtrees hanged at T are similar to the edges of a star graph hanged at its root, and eventually manage to adapt the algorithm from [1] in a delicate way to estimate the cover advantage by edges of this type in a global manner.

When MST consists of only weight-1 edges. This special case appears trickier since we do not know the structure of an MST at the start, and there may not even be a unique MST. To circumvent this, we need to utilize the following technical result of [4]: Let G_1 be the graph on V induced by all weight-1 edges in the given metric, then if G_1 contains a size-s matching consisting of only edges in 2-edge-connected components of G_1, then $\mathsf{TSP} \leq 2n - \Omega(s)$. This result allows us to construct a local procedure that explores some neighborhood of the unknown graph G_1 up to a certain size, such that in the end we either reconstruct a size-$\sqrt{n}$ subgraph of the (locally) unique MST, or certify that a set of $\Omega(\sqrt{n})$ vertices belong to some 2-edge-connected components of G_1, which will be later collected to estimate the maximum weight-1 matching size.

We then use this local procedure on a set of vertices randomly sampled from V. Let T be an MST. Intuitively, if the total size of light subtrees of T is non-negligible, then with high probability some of the sampled vertices will lie in light subtrees of T, and we can obtain an estimate of the local cover advantage within subtrees. If the total size of light subtrees is negligible, then T', the subtree obtained from T by peeling off all light subtrees, has roughly the same size as T, which means that T is close to the first instructive example mentioned before – a tree with only $O(\sqrt{n})$ special vertices. Then we can apply the local procedure to $\Omega(\sqrt{n})$ sampled vertices, to almost reconstruct the whole tree T, and the rest of the algorithm is similar to the algorithm in the first special case.

1.2.2 Overview of Lower Bound Techniques

As our algorithmic results illustrate that metric streams are more powerful than graph streams, it is perhaps not surprising that proving space lower bounds for metrics streams turns out to be a more challenging task that requires new tools. To illustrate this point, it might be instructive to consider the following simplified versions of metric and graph streams. Let G be a graph.

- Unweighted Graph Stream: a sequence that contains all edges of $E(G)$, and the same edge may appear more than once in the stream;
- Unweighted Metric Stream: a sequence that contains, for each pair u, v of $V(G)$, a symbol $f(u, v)$ indicating whether or not the edge (u, v) belongs to $E(G)$.

Note that in unweighted metric streams, the non-edge information between pairs of vertices is also explicitly given (as the edge information), as opposed to being given implicitly in the unweighted graph stream. This seemingly unimportant distinction, unexpectedly, makes proving lower bounds for several problems much harder in unweighted metric streams than unweighted graph streams.

For example, consider the problem of deciding whether the input graph is a clique. On the one hand, to prove a space lower bound for streaming algorithms in unweighted graph streams, we consider the following two-player one-way communication game: Alice is given a

graph G_A and Bob is given a graph G_B on a common vertex set V, and Alice and Bob want to decide if $G_A \cup G_B$ is the complete graph on V. It is easy to show that this communication game has back-and-forth communication complexity $\Omega(n^2)$. In fact, Alice's input graph G_A can be viewed as a vector $x^A \in \{0,1\}^{\binom{V}{2}}$ and Bob's input graph G_B can be viewed as a vector $x^B \in \{0,1\}^{\binom{V}{2}}$, where the coordinate $x^A_{(u,u')}$ indexed by the pair u, u' of vertices in V indicates whether or not the edge (u, u') appears in graph G_A, and similarly $x^B_{(u,u')}$ indicates whether or not the edge (u, u') appears graph G_B. It is then easy to see that the two players need to detect whether or not the bitwise-OR of vectors x^A and x^B is the all-one vector, which requires $\Omega(n^2)$-bits information exchange even in the back-and-forth communication model. On the other hand, in the corresponding two-player one-way communication game for unweighted metric streams, Alice and Bob are each given a set of edge/non-edge information, with the promise that the edge/non-edge information between each pair of vertices appears in at least one of the player's input. There is a one-bit protocol: Alice simply sends to Bob a signal indicating whether or not in her input there is non-edge information between any pair of vertices, and Bob outputs "Not a Clique" iff either he sees Alice's "non-edge" signal or he sees a non-edge information in his input.

The distinction that all non-edge information is explicitly given in the unweighted stream seems to fail all reductions from standard problems (like Disjointness and Index) to prove lower bounds. Therefore, in the lower bound proofs of Theorem 3 and Theorem 4, we identify new "primitive" graph-theoretic problems, prove communication lower bounds for them, and then reduce them to MST-estimation problems. Here we briefly provide some ideas for the proof of Theorem 4.

We consider the special type of metrics, in which the distance between every pair of vertices is either 1 or a large enough real number. Intuitively, the problem of estimating the MST cost is equivalent to the problem of estimating the number of connected components of the graph induced by all weight-1 edges, which is essentially a graph-theoretic problem in unweighted metric streams.

As a first step, we consider the following problem: given an unweighted metric stream, decide whether the underlying graph is a perfect matching or a perfect matching minus one edge. Unlike the previous clique-identification problem, we show that the corresponding two-player communication game for this problem has communication complexity $\Omega(n)$ in the back and forth communication model, even if the complete edge/non-edge information is split between Alice and Bob. The proof is by analyzing the information complexity of any protocol for the problem, We construct several similar input combinations for Alice and Bob, among which some cross-combination lead to different answers, and then lower bound the mutual information between the protocol transcript and the players' inputs.

However, this perfect matching vs perfect matching minus one edge problem is not sufficient for our purpose, since a perfect matching graph on n vertices has $n/2$ connected components, while a perfect matching minus one edge graph on n vertices has $n/2 - 1$ connected components, and the ratio between $n/2$ and $n/2 - 1$ are too small to provide a space lower bound for α-approximation of the number of connected components. To fix this issue, we next consider a generalization of this problem, called the *Clique or Independent Set* problem ($\mathsf{COI}_{a,b}$) parametrized by two integers a, b. In this problem, we are required to decide whether the input graph is the disjoint union of b cliques of size a each (Yes case) or it is a disjoint union of $(b-1)$ cliques of size a each and an independent set of size a (No case). Note that if $a = 2$ and $b = n/2$ then this problem is exactly the perfect matching vs perfect matching minus one edge problem. Now if we let $a \gg b$, then the ratio between numbers of connected components in Yes case and in No case is $(a + b - 1)/b = \Omega(a/b)$, which is

enough for giving a space lower bound for $o(a/b)$-approximation streaming algorithms for MST estimation. For the proof of the communication lower bound of problem $\mathsf{COI}_{a,b}$, we first consider the special case $\mathsf{COI}_{a,2}$ and show that the communication complexity is $\tilde{\Omega}(1)$ via a Hellinger distance analysis on transcript distributions on certain input combinations, and then use a direct sum type argument to show that the communication complexity of $\mathsf{COI}_{a,b}$ is $\tilde{\Omega}(b)$. Both steps use techniques similar to the ones used in the proof of communication lower bound for the perfect matching vs perfect matching minus one edge problem. Now for a given approximation ratio $\alpha > 1$, setting $a = \Theta(\sqrt{\alpha n})$ and $b = \Theta(\sqrt{n/\alpha})$ yields the desired communication lower bound, which then implies the space lower bounds for streaming algorithms.

1.3 Organization

Due to the limit of space, in the remainder of the paper we only present the proof sketches of one of our algorithmic results, which best illustrates the utilization of cover advantage. We first introduce the notion of cover advantage in Section 2. We then we sketch the proof of Theorem 6 in Section 3 and sketch the proof of Theorem 8 in Section 4. The proofs of all other theorems are deferred to the full version (in the appendix).

2 Cover Advantage

In this section, we introduce the notion of cover advantage, which is a key notion that captures the gap between the MST cost and the TSP cost. Our TSP cost estimation algorithms in both streaming and query settings will crucially utilize this notion. Due to the limit of space, the proofs of some lemmas presented in this section are deferred to the full version.

At a high-level, the TSP estimation algorithms in this paper are based on converting an MST T of the input graph/metric into a spanning Eulerian subgraph. A trivial approach is to simply double all edges in T obtaining a 2-approximation. A more clever approach due to Christofides [5] instead makes T Eulerian by adding a minimum weight perfect matching on odd-degree vertices in T, obtaining a 3/2-appproximation. However, computing a good approximation to the minimum weight perfect matching on a set of vertices appears hard to do either in the semi-streaming setting or with sublinear number of queries. We instead identify the more tractable notion, called the cover advantage, that can be efficiently implemented in the semi-streaming and query model.

We say that an edge f of tree T is *covered* by an edge e that may or may not belong to T, iff $f \in E(P_e^T)$; and we say that f is covered by a set E' of edges, iff it is covered by some edge of E'. We denote by $\mathsf{cov}(e)$ the set of all edges of T that are covered by e, and define $\mathsf{cov}(E') = \bigcup_{e \in E'} \mathsf{cov}(e)$.

Let T' be a subtree of T. For each edge $e \notin E(T)$, we define $\mathsf{cov}(e, T') = \mathsf{cov}(e) \cap E(T')$. Similarly, for a set E' of edges, we define $\mathsf{cov}(E', T') = \bigcup_{e \in E'} \mathsf{cov}(e, T')$. Clearly, $\mathsf{cov}(E', T') = \mathsf{cov}(E') \cap E(T')$.

We define the *cover advantage* of a set E' of edges on a subtree T' of T, denoted by $\mathsf{adv}(E', T')$, to be $\mathsf{adv}(E', T') = w(\mathsf{cov}(E', T')) - w(E')$. The *optimal cover advantage* of a subtree T', denoted by $\mathsf{adv}(T')$, is defined to be the maximum cover advantage of any set E' of edges that have at least one endpoint lying in $V(T')$ on T'. The *optimal special cover advantage* of a subtree T', denoted by $\mathsf{adv}^*(T')$, is defined to be the maximum cover advantage of any set E' of edges that have at least one endpoint being a special vertex of T' (a vertex v is a special vertex of T' iff $\deg_{T'}(v) \neq 2$). Clearly, by definition, $\mathsf{adv}(T') \geq \mathsf{adv}^*(T') \geq 0$.

The next two lemmas show that the optimal cover advantage and the optimal special cover advantage of any subtree can be computed using a small number of queries.

▶ **Lemma 9.** *There is an algorithm, that given a subtree T' of T, computes the optimal cover advantage of T' as well as a set E' of edges achieving the optimal cover advantage of T', by performing at most $O(n \cdot |V(T')|)$ queries.*

▶ **Lemma 10.** *There is an algorithm, that given a subtree T' of T, computes the optimal special cover advantage of T' as well as a set E' of edges achieving the optimal special cover advantage of T', by performing $O(n \cdot k_{T'})$ queries, where $k_{T'}$ is the number of special vertices in T'.*

The following lemma is crucial to our algorithms. It shows that the high cover advantage of edge-disjoint subtrees of an MST translates into a TSP tour whose cost is bounded away from 2 times the MST cost.

▶ **Lemma 11.** *Let T be an MST on a set V of vertices, and let $\mathcal{T}$ be a set of edge-disjoint subtrees of T. Then $\mathsf{TSP} \leq 2 \cdot \mathsf{MST} - \frac{1}{2} \cdot \sum_{T' \in \mathcal{T}} \mathsf{adv}(T')$.*

Proof. We introduce some definitions before providing the proof.

Let E' be a set of edges that do not belong to $E(T)$. We define the multi-graph $H_{T,E'}$ as follows. Its vertex set is $V(H_{T,E'}) = V$. Its edge set is the union of (i) the set E'; and (ii) the set $E_{[T,E']}$ that contains, for each edge $f \in E(T)$, 2 copies of f iff f is covered by an even number of edges in E', 1 copy of f iff f is covered by an odd number of edges in E'. Equivalently, graph $H_{T,E'}$ can be obtained from the following iterative algorithm. Throughout, we maintain a graph $\hat{H}$ on the vertex set V, that initially contains two copies of each edge of $E(T)$. We will maintain the invariant that, over the course of the algorithm, for each edge f of $E(T)$, graph $\hat{H}$ contains either one copy or two copies of f. We then process edges of E' one-by-one (in arbitrary order) as follows. Consider now an edge $e \in E'$ and the tree-path P_e^T. We add one copy of edge e to $\hat{H}$. Then for each edge $f \in E(P_e^T)$, if currently the graph $\hat{H}$ contains 2 copies of f, then we remove one copy of it from $\hat{H}$; if currently the graph $\hat{H}$ contains 1 copy of f, then we add one copy of it into $\hat{H}$. Clearly after each iteration of processing some edge of E', the invariant still holds. It is also easy to see that the resulting graph we obtain after processing all edges of E' is exactly the graph $H_{T,E'}$ defined above.

We prove the following observation.

▶ **Observation 12.** *For any set E', graph $H_{T,E'}$ is Eulerian.*

Proof. Consider the algorithm that produces the graph $H_{T,E'}$. Initially, graph $\hat{H}$ contains 2 copies of each edge of T, and is therefore Eulerian. It is easy to see that, in the iteration of processing the edge $e \in E'$, we only modify the degrees of vertices in the cycle $e \cup P_e^T$. Specifically, for each vertex in the cycle $e \cup P_e^T$, either its degree is increased by 2 (if a copy is added to both of its incident edges in the cycle), or its degree is decreased by 2 (if a copy is removed from both of its incident edges in the cycle), or its degree remains unchanged (if a copy is removed from one of its incident edges, and a copy is added to the other incident edge). Therefore, the graph $\hat{H}$ remains Eulerian after this iteration, and it follows that the resulting graph $H_{T,E'}$ is Eulerian. ◀

We now provide the proof of Lemma 11. Denote $\mathcal{T} = \{T_1, \ldots, T_k\}$. For each index $1 \leq i \leq k$, let E_i^* be the set of edges that achieves the maximum cover advantage on T_i. Denote $E^* = \bigcup_{1 \leq i \leq k} E_i^*$, and then we let E' be the random subset of E^* that includes

each edge of E^* independently with probability $1/2$. We will show that the expected total weight of all edges in $E(H_{T,E'})$ is at most $2 \cdot \mathsf{MST}(G) - \frac{1}{2} \cdot \sum_{1 \le i \le k} \mathsf{adv}(T_i)$. Note that this implies that there exists a subset E^{**} of E^*, such that the weight of graph $H_{T,E^{**}}$ is at most $2 \cdot \mathsf{MST}(G) - \frac{1}{2} \cdot \sum_{1 \le i \le k} \mathsf{adv}(T_i)$. Combined with Observation 12 and the fact that TSP is upper bounded by the total cost of any connected Eulerian graph, this implies $\mathsf{TSP} \le 2 \cdot \mathsf{MST}(G) - \frac{1}{2} \cdot \sum_{1 \le i \le k} \mathsf{adv}(T_i)$, completing the proof of Lemma 11.

We now show that $\mathbb{E}[w(H_{T,E'})] \le 2 \cdot \mathsf{MST}(G) - \frac{1}{2} \cdot \sum_{1 \le i \le k} \mathsf{adv}(T_i)$. From the definition of graph $H_{T,E'}$, $E(H_{T,E'}) = E' \cup E_{[T,E']}$. On one hand, from the construction of set E', $\mathbb{E}[w(E')] = w(E^*)/2$. On the other hand, for each edge $f \in \mathsf{cov}(E^*)$, with probability $1/2$ graph $H_{T,E'}$ contains 1 copy of it, and with probability $1/2$ graph $H_{T,E'}$ contains 2 copies of it. Therefore, $\mathbb{E}[w(E_{[T,E']})] = 2 \cdot w(E(T)) - w(\mathsf{cov}(E^*))/2$. Note that subtrees $\{T_i\}_{1 \le i \le k}$ are edge-disjoint, so the edge sets $\{\mathsf{cov}(E^*, T_i)\}_{1 \le i \le k}$ are mutually disjoint. Altogether,

$$\begin{aligned}
\mathbb{E}[w(H_{T,E'})] &= 2 \cdot \mathsf{MST} - \frac{w(\mathsf{cov}(E^*)) - w(E^*)}{2} \\
&\le 2 \cdot \mathsf{MST} - \frac{1}{2} \cdot \sum_{1 \le i \le k} \Big(w(\mathsf{cov}(E^*, T_i)) - w(E^*_i) \Big) \\
&\le 2 \cdot \mathsf{MST} - \frac{1}{2} \cdot \sum_{1 \le i \le k} \Big(w(\mathsf{cov}(E^*_i, T_i)) - w(E^*_i) \Big) \\
&= 2 \cdot \mathsf{MST} - \frac{1}{2} \cdot \sum_{1 \le i \le k} \mathsf{adv}(T_i).
\end{aligned}$$

◀

Complementing Lemma 11, the next lemma shows that, if the special cover advantage is low, then the TSP cost is close to 2 times the MST cost.

▶ **Lemma 13.** *Let T' be any subtree of an MST T. Then $\mathsf{TSP} \ge 2 \cdot w(T') - 2 \cdot \mathsf{adv}^*(T')$.*

Proof. Let π^* be the optimal TSP-tour that visits all vertices of V, so $\mathsf{TSP} = w(\pi^*)$. Let π be the tour obtained from π^* by deleting all vertices of $T \setminus T'$, so π is a tour that visits all vertices of $V(T')$, and, from triangle inequality, $w(\pi^*) \ge w(\pi)$. We now show that $E(\pi)$ can be partitioned into two subsets $E(\pi) = E_0 \cup E_1$, such that $E(T') \subseteq \mathsf{cov}(E_0)$ and $E(T') \subseteq \mathsf{cov}(E_1)$.

Let V' be the set of all odd-degree vertices in T', so $|V'|$ is even. Denote $V' = \{v_1, v_2, \ldots, v_{2k}\}$, where the vertices are index according to the order in which they appear in π. For each $1 \le i \le 2k$, we define edge $e_i = (v_i, v_{i+1})$ and define E^i_π to be the set of all edges traversed by π between vertices v_i and v_{i+1}. Clearly, $\mathsf{cov}(e_i) \subseteq \mathsf{cov}(E^i_\pi)$. We define $E_0 = \bigcup_{0 \le i \le k-1} E^{2i}_\pi$ and $E_1 = \bigcup_{0 \le i \le k-1} E^{2i+1}_\pi$.

Consider now the tour π' induced by edges of $e_1, \ldots, e_{2k}$. Clearly, π' is a tour that visits all vertices of V'. We define sets $F_0 = \{e_{2i} \mid 0 \le i \le k-1\}$ and $F_1 = \{e_{2i+1} \mid 0 \le i \le k-1\}$, so $E(\pi') = F_0 \cup F_1$. We now show that $E(T') \subseteq \mathsf{cov}(F_0)$ and $E(T') \subseteq \mathsf{cov}(F_1)$. Note that this implies that $E(T') \subseteq \mathsf{cov}(E_0)$ and $E(T') \subseteq \mathsf{cov}(E_1)$, since

$$\mathsf{cov}(F_0) = \Big(\bigcup_{0 \le i \le k-1} \mathsf{cov}(e_{2i}) \Big) \subseteq \Big(\bigcup_{0 \le i \le k-1} \mathsf{cov}(E^{2i}_\pi) \Big) \subseteq \mathsf{cov}\Big(\bigcup_{0 \le i \le k-1} E^{2i}_\pi \Big) = \mathsf{cov}(E_0),$$

and similarly $\mathsf{cov}(F_1) \subseteq \mathsf{cov}(E_1)$.

In fact, note that F_0 is a perfect matching on V'. Since V' is the set of odd-degree vertices of T', the graph on $V(T')$ induced by edges of $E(T') \cup F_0$ is Eulerian. Therefore, every edge of T' appears in at least two sets of $\{\mathsf{cov}(e') \mid e' \in E(T') \cup M_o\}$. Note that for each $e' \in E(T')$, $\mathsf{cov}(e') = \{e'\}$. Therefore, every edge $e \in E(T')$ appears in at least one set of $\{\mathsf{cov}(e') \mid e' \in F_0\}$, i.e., $E(T') \subseteq \mathsf{cov}(F_0)$. Similarly, we get that $E(T') \subseteq \mathsf{cov}(F_1)$.

From triangle inequality, $w(F_0) + w(F_1) \leq w(E_0) + w(E_1) = w(\pi)$. Note that edges of F_0 and F_1 have both endpoint in V', and moreover, from the definition of V', all vertices of V' are special vertices of T'. Since $E(T') \subseteq \mathsf{cov}(F_0)$, from the definition of $\mathsf{adv}^*(T')$, we get that $\mathsf{adv}^*(T') \geq w(\mathsf{cov}(F_0, T')) - w(F_0) = w(T') - w(F_0)$, and similarly $\mathsf{adv}^*(T') \geq w(T') - w(F_1)$. Therefore, $\mathsf{TSP} = w(\pi^*) \geq w(\pi) = w(E_0) + w(E_1) \geq w(F_0) + w(F_1) \geq 2 \cdot (w(T') - \mathsf{adv}^*(T'))$. ◀

The last two lemmas show that the total cover advantage and the total special cover advantage of a set of edge-disjoint subtrees of an MST can be efficiently and accurately estimated.

▶ **Lemma 14.** *There is an algorithm, that, given a constant $0 < \varepsilon < 1$ and a set $\mathcal{T}$ of edge-disjoint subtrees of T, with high probability, either correctly reports that $\sum_{T' \in \mathcal{T}} \mathsf{adv}(T') \geq \varepsilon \cdot \mathsf{MST}$, or correctly reports that $\sum_{T' \in \mathcal{T}} \mathsf{adv}(T') \leq 2\varepsilon \cdot \mathsf{MST}$, by performing $\tilde{O}((n/\varepsilon^2) \cdot \max\{|V(T')|\}_{T' \in \mathcal{T}})$ queries.*

▶ **Lemma 15.** *There is an algorithm, that, given a constant $0 < \varepsilon < 1$ and a set $\mathcal{T}$ of edge-disjoint subtrees of T, with high probability, either correctly reports that $\sum_{T' \in \mathcal{T}} \mathsf{adv}^*(T') \geq \varepsilon \cdot \mathsf{MST}$, or correctly reports that $\sum_{T' \in \mathcal{T}} \mathsf{adv}^*(T') \leq 2\varepsilon \cdot \mathsf{MST}$, by performing $\tilde{O}((n/\varepsilon^2) \cdot \max\{k_{T'} \mid T' \in \mathcal{T}\})$ queries, where $k_{T'}$ is the number of special vertices in T'.*

3 A Two-Pass Algorithm for TSP Estimation in Graph Streams

In this section, we present a deterministic 2-pass 1.96-approximation algorithm for TSP estimation in graph streams, which uses $\tilde{O}(n)$ space, thus proving Theorem 6. Our algorithm will utilize the notion of cover advantage, introduced in Section 2. This result is in a sharp contrast to Theorem 5 which showed that any single-pass algorithm requires $\Omega(n^2)$ space to obtain a better than 2-approximation.

Algorithm. Let $\alpha, \beta \in (0, 1)$ be two constants whose values will be set later. In the first pass, we simply compute a minimum spanning tree T and its cost $\mathsf{MST} = \sum_{e \in E(T)} w(e)$. Throughout the second pass, we maintain a subset E_{temp} of edges, that is initialized to be $\emptyset$, and will only grow over the course of the algorithm. Upon the arrival of each edge e, we compare $w(e)$ with $w(\mathsf{cov}(e) \setminus \mathsf{cov}(E_{\mathsf{temp}})) = \sum_{f \in \mathsf{cov}(e) \setminus \mathsf{cov}(E_{\mathsf{temp}})} w(f)$. We add the edge e to set E_{temp} iff $w(e) \leq \alpha \cdot w(\mathsf{cov}(e) \setminus \mathsf{cov}(E_{\mathsf{temp}}))$. Let E^* be the set E_{temp} at the end of the algorithm. We then compute $w(\mathsf{cov}(E^*)) = \sum_{e \in \mathsf{cov}(E^*)} w(e)$. If $w(\mathsf{cov}(E^*)) \geq \beta \cdot \mathsf{MST}$, then we output $(2 - \frac{(1-\alpha)\cdot\beta}{2}) \cdot \mathsf{MST}$ as an estimate of TSP; otherwise we output $2 \cdot \mathsf{MST}$. We use the parameters $\alpha = 0.715$ and $\beta = 0.285$, so $(2 - \frac{(1-\alpha)\cdot\beta}{2}) \approx \frac{2}{2\alpha(1-\beta)} \approx 1.96$.

Proof of Correctness. The correctness of the algorithm is guaranteed by the following two claims.

▷ Claim 16. If $w(\mathsf{cov}(E^*)) \geq \beta \cdot \mathsf{MST}$, then $\mathsf{TSP} \leq (2 - \frac{(1-\alpha)\cdot\beta}{2}) \cdot \mathsf{MST}$.

Proof. Let E' be the random subset of E^* that includes each edge of E^* independently with probability $1/2$. We will show that the expected total weight of all edges in graph $E(H_{T,E'})$ is at most $(2 - \frac{(1-\alpha)\cdot\beta}{2}) \cdot \mathsf{MST}$, namely $\mathbb{E}[w(E(H_{T,E'}))] \leq (2 - \frac{(1-\alpha)\cdot\beta}{2}) \cdot \mathsf{MST}$. Note that this implies that there exists a subset E^{**} of E^*, such that $w(E(H_{T,E^{**}})) \leq (2 - \frac{(1-\alpha)\cdot\beta}{2}) \cdot \mathsf{MST}$. Combined with Observation 12, this implies that there is an Eulerian tour of the same cost (using only edges of graph $H_{T,E^{**}}$). Therefore, there is a TSP-tour of at most the same cost, completing the proof of Claim 16.

We now show that $\mathbb{E}[w(E(H_{T,E'}))] \leq (2 - \frac{(1-\alpha)\cdot\beta}{2}) \cdot \mathsf{MST}$. From the definition of graph $H_{T,E'}$, $E(H_{T,E'}) = E' \cup E_{[T,E']}$. On one hand, from the definition of the random subset E', $\mathbb{E}[w(E')] = w(E^*)/2$. On the other hand, for each edge $f \in \mathsf{cov}(E^*)$, with probability $1/2$ graph $H_{T,E'}$ contains 1 copy of it, and with probability $1/2$ graph $H_{T,E'}$ contains 2 copies of it. Therefore, $\mathbb{E}[w(E_{[T,E']})] = 2 \cdot w(E(T)) - w(\mathsf{cov}(E^*))/2 = 2 \cdot \mathsf{MST} - w(\mathsf{cov}(E^*))/2$. Altogether, $\mathbb{E}[w(E(H_{T,E'}))] = 2 \cdot \mathsf{MST} - (w(\mathsf{cov}(E^*)) - w(E^*))/2$. The following observation follows immediately from the algorithm.

▶ **Observation 17.** $w(E^*) \leq \alpha \cdot w(\mathsf{cov}(E^*))$.

From Observation 17,

$$\mathbb{E}[w(E(H_{T,E'}))] \leq 2 \cdot \mathsf{MST} - \frac{(1-\alpha) \cdot w(\mathsf{cov}(E^*))}{2} \leq \left(2 - \frac{(1-\alpha)\cdot\beta}{2}\right) \cdot \mathsf{MST}.$$

This concludes the proof of Claim 16. ◁

We next show that, if we do not find a sufficiently large cover, i.e., the value of $w(\mathsf{cov}(E^*))$ is not sufficiently large compared with MST, then TSP must be bounded away from MST.

▷ Claim 18. If $w(\mathsf{cov}(E^*)) < \beta \cdot \mathsf{MST}$, then $\mathsf{TSP} \geq 2\alpha(1-\beta) \cdot \mathsf{MST}$.

Proof. Recall that set $V_1(T)$ contains all vertices with odd degree in T. Let M be a minimum-cost perfect matching on $V_1(T)$, so $\mathsf{TSP} \geq 2 \cdot w(M)$. We use the following observations. The proof of Observation 19 is straightforward and is deferred to the full version.

▶ **Observation 19.** $\mathsf{cov}(M) = E(T)$.

▶ **Observation 20.** *For each* $e \in M$, $w(\mathsf{cov}(e) \setminus \mathsf{cov}(E^*)) < w(e)/\alpha$.

Proof. We denote by Q_e the shortest-path in G connecting the endpoints of e (where G is the graph underlying the stream). Since Q_e is a subgraph of G, all edges of Q_e will appear in the graph stream of G. Note that $w(Q_e) = w(e)$ and $\mathsf{cov}(e) \subseteq \mathsf{cov}(E(Q_e))$.

We will show that, for every edge $e' \in E(Q_e)$, $w(\mathsf{cov}(e') \setminus \mathsf{cov}(E^*)) < w(e')/\alpha$. Note that the observation follows from this assertion, as

$$w(\mathsf{cov}(e) \setminus \mathsf{cov}(E^*)) \leq w(\mathsf{cov}(E(Q_e)) \setminus \mathsf{cov}(E^*)) \leq \sum_{e' \in E(Q_e)} w(\mathsf{cov}(e') \setminus \mathsf{cov}(E^*)) < \frac{w(Q_e)}{\alpha} = \frac{w(e)}{\alpha}.$$

Consider now any edge $e' \in E(Q_e)$, and assume for contradiction that $w(\mathsf{cov}(e') \setminus \mathsf{cov}(E^*)) \geq w(e')/\alpha$. Note that set E_{temp} only grows over the course of the algorithm that computes set E^*, and so does the set $\mathsf{cov}(E_{\mathsf{temp}})$. Therefore, when e' arrives in the stream, $w(\mathsf{cov}(e') \setminus \mathsf{cov}(E_{\mathsf{temp}})) \geq w(e')/\alpha$ must hold. Then according to the algorithm, the edge e' should be added to E_{temp} right away, which means that edge e' will eventually belong to E^*, leading to $\mathsf{cov}(e') \subseteq \mathsf{cov}(E^*)$ and $w(\mathsf{cov}(e') \setminus \mathsf{cov}(E^*)) = 0$, a contradiction to the assumption that $w(\mathsf{cov}(e') \setminus \mathsf{cov}(E^*)) \geq w(e')/\alpha$. ◀

From Observation 19 and Observation 20, we get that

$$(1-\beta) \cdot \mathsf{MST} \leq \mathsf{MST} - w(\mathsf{cov}(E^*)) = w(\mathsf{cov}(M) \setminus \mathsf{cov}(E^*)) \leq \sum_{e \in M} w(\mathsf{cov}(e) \setminus \mathsf{cov}(E^*)) < w(M)/\alpha.$$

Therefore, $w(M) \geq \alpha(1-\beta) \cdot \mathsf{MST}$. Since $\mathsf{TSP} \geq 2 \cdot w(M)$, we conclude that $\mathsf{TSP} \geq 2\alpha(1-\beta) \cdot \mathsf{MST}$. ◁

4 An $(2-\varepsilon_0)$-Approximation Query Algorithm with a given MST

In this section, we provide a proof sketch of Theorem 8, by showing an algorithm that, given access of a minimum spanning tree of a metric, obtains an $(2-\varepsilon_0)$-approximation of TSP (for $\varepsilon_0 = 2^{-100}$) by performing $\tilde{O}(n^{1.5})$ queries. Note that it suffices for the algorithm to correctly claim either $\mathsf{TSP} \geq (1+\varepsilon_0)\cdot\mathsf{MST}$ or $\mathsf{TSP} \leq (2-\varepsilon_0)\cdot\mathsf{MST}$.

Let T be the input MST and let it be rooted at an arbitrary vertex. We first divide T into a top part and a bottom part as follows. We say that a vertex v is *maximally light*, iff T_v (the subtree of T rooted at v) contains at most $\sqrt{n}$ vertices, but its parent node does not. Let T' be the tree obtained from T by deleting from it, for each maximally light vertex v, all edges and vertices of T_v, and we call T' the top part of T, and call $T \setminus T'$ the bottom part of T. It is easy to show that T' has at most $O(\sqrt{n})$ leaves and therefore at most $O(\sqrt{n})$ special vertices. Moreover, we can efficiently partition T' into a set $\mathcal{P}$ of $O(\sqrt{n})$ vertex-disjoint paths, such that, for each path $P \in \mathcal{P}$, either P contains a single vertex of T, or the total number of vertices in T that has an ancestor in P is at most $\sqrt{n}$. For each path $P \in \mathcal{P}$, we call the subtree of T induced by all vertices of P and all their descendants in T a *segment*. Let $\mathcal{S}$ be the set of all segments.

The algorithm consists of four steps, that are summarized as follows.

1. We compute the special cover advantage of T' using Lemma 10. If $\mathsf{adv}^*(T') \geq 10\varepsilon_0\cdot\mathsf{MST}$, then we claim $\mathsf{TSP} \leq (2-\varepsilon_0)\cdot\mathsf{MST}$.
2. We estimate the total cover advantage of all segments using Lemma 14. If the algorithm reports that $\sum_{S\in\mathcal{S}} \mathsf{adv}(S) \geq 10\varepsilon_0\cdot\mathsf{MST}$, then we claim $\mathsf{TSP} \leq (2-\varepsilon_0)\cdot\mathsf{MST}$.
3. (informal) We estimate the cover advantage involving $T\setminus T'$, with the help of the algorithm in [1]. If the estimation is at least $10\varepsilon_0\cdot\mathsf{MST}$, then we claim $\mathsf{TSP} \leq (2-\varepsilon_0)\cdot\mathsf{MST}$.
4. If we did not terminate and claim $\mathsf{TSP} \leq (2-\varepsilon_0)\cdot\mathsf{MST}$ in any of the previous step, then we claim $\mathsf{TSP} \geq (1+\varepsilon_0)\cdot\mathsf{MST}$.

We first count the total number of queries performed by the algorithm. First, the algorithm from Lemma 10 performs $\tilde{O}(n^{1.5})$ queries, since the number of special vertices in T' is at most $O(\sqrt{n})$. Second, in Step 2, the algorithm from Lemma 14 performs in total $\tilde{O}(n^{1.5})$ queries, since each segment contains $O(\sqrt{n})$ vertices. Lastly, in Step 3 the algorithm performs $\tilde{O}(n^{1.5})$ queries. Altogether, the algorithm performs in total $\tilde{O}(n^{1.5})$ queries.

We now show that the algorithm indeed returns a $(2-\varepsilon_0)$-approximation of TSP. That is, the claim made by the algorithm is correct.

- If in Step 1, the algorithm from Lemma 10 reported $\mathsf{adv}^*(T') \geq 10\varepsilon_0\cdot\mathsf{MST}$. Then from Lemma 11, $\mathsf{TSP} \leq 2\cdot\mathsf{MST} - (10\varepsilon_0)\cdot\mathsf{MST}/2 = (2-5\varepsilon_0)\cdot\mathsf{MST}$ and the claim is correct.
- If in Step 2, the algorithm from Lemma 14 reported $\sum_{S\in\mathcal{S}} \mathsf{adv}(S) \geq 10\varepsilon_0\cdot\mathsf{MST}$. From Lemma 11, $\mathsf{TSP} \leq 2\cdot\mathsf{MST} - (10\varepsilon_0)\cdot\mathsf{MST}/2 = (2-5\varepsilon_0)\cdot\mathsf{MST}$ and the claim is correct.
- If in Step 2, the estimate of the cover advavtage in $T\setminus T'$ is at least $10\varepsilon_0\cdot\mathsf{MST}$, from Lemma 11, we can derive that $\mathsf{TSP} \leq (2-5\varepsilon_0)\cdot\mathsf{MST}$ and the claim is correct.

We assume from now on the algorithm did not terminate and claim $\mathsf{TSP} \leq (2-\varepsilon_0)\cdot\mathsf{MST}$ in the first three steps. We will show that in this case, $\mathsf{TSP} \geq (1+\varepsilon_0)\cdot\mathsf{MST}$.

Let π be an optimal TSP-tour, so $\mathsf{TSP} = w(\pi)$. Intuitively, if the tour π "continuously travels within the top part of T", then the report in Step 1 guarantees that its total cost must be bounded away from MST; if the tour π "continuously travels within the same segments", then the report in Step 2 guarantees that its total cost must be bounded away from MST. Therefore, we only need to consider the case where the tour "constantly jumping between the top and the bottom parts of T and across different segments".

Note that, since the cover advantage that we discovered in Steps 1-2 are low, for every edge $e = (u, v)$, the weight $w(e)$ should be rougly equal to the total weight of the unique u-v path in T. Therefore, if we replace each edge of π with the corresponding path in T, then the total weight of the resulting edge set should be close to TSP. But since the tour jumps between the top and the bottom parts of T and across different segments, the "segment-connecting" edges in T must be covered many times by π, and this can be used to show that $w(\pi)$ is bounded away from MST.

Specifically, we let E'_π the obtained edge set after replacing each edge of $E(\pi)$ with edges in the corresponding path in T, so E'_π may contain many copies of the same edge. For each edge e that has more than 2 copies contained in E'_π, if E'_π contains an odd number of copies of e, then we delete all but one copies from E'_π; if E'_π contains an even number of copies of e, then we delete all but two copies from E'_π. Denote by E' the resulting set of edges, so each edge has at most 2 copies contained in E'. It is easy to verify that the graph induced by edges of E' (with multiplicity) is connected and Eulerian.

Let E'' be the subset of E' that contains all bridges in the graph induced by edges of E' (ignoring multiplicities), and it is easy to verify that each edge has two copies contained in E'. Then from the report of Steps 1-2 of the algorithm, we can show that $w(E') - w(E(\pi)) \geq O(\varepsilon_0) \cdot \mathsf{MST}$ and $w(E') \geq \mathsf{MST} + w(E'')$. Therefore, if $w(E'') \geq \Omega(\varepsilon_0) \cdot \mathsf{MST}$ then we are done. If $w(E'') \leq O(\varepsilon_0) \cdot \mathsf{MST}$, then we can show that the total cost of all edges with at least one endpoints in $T \setminus T'$ must be large, and from the report of Step 3 of the algorithm, we can conclude that $w(E(\pi))$ is bounded away from MST.

5 Future Directions

In this work, we studied the problems of MST and TSP cost estimation in the streaming and query settings. For TSP cost estimation, we introduced and utilized a novel notion called *cover advantage* that may prove useful for solving this problem in other computational models also. In the streaming setting, an interesting open problem is to obtain a one-pass $o(n^2)$-space $(2-\varepsilon)$-approximate estimation of TSP cost in the metric stream. In the query model, we believe a major open problem is to obtain an $o(n^2)$-query $(2-\varepsilon)$-approximate estimation of TSP-cost in general metrics.

References

1. Soheil Behnezhad. Time-optimal sublinear algorithms for matching and vertex cover. *arXiv preprint*, 2021. `arXiv:2106.02942`.
2. Soheil Behnezhad, Mohammad Roghani, Aviad Rubinstein, and Amin Saberi. Sublinear algorithms for tsp via path covers. *arXiv preprint*, 2023. `arXiv:2301.05350`.
3. Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM J. Comput.*, 34(6):1370–1379, 2005.
4. Yu Chen, Sampath Kannan, and Sanjeev Khanna. Sublinear algorithms and lower bounds for metric tsp cost estimation. *arXiv preprint*, 2020. `arXiv:2006.05490`.
5. Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, 1976.
6. Artur Czumaj and Christian Sohler. Estimating the weight of metric minimum spanning trees in sublinear time. *SIAM Journal on Computing*, 39(3):904–922, 2009.

7 Krzysztof Onak, Dana Ron, Michal Rosen, and Ronitt Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1123–1131. Society for Industrial and Applied Mathematics, 2012.

8 Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. Improved constant-time approximation algorithms for maximum matchings and other optimization problems. *SIAM Journal on Computing*, 41(4):1074–1093, 2012.

Quantum Algorithms and Lower Bounds for Linear Regression with Norm Constraints

Yanlin Chen
QuSoft and CWI, Amsterdam, The Netherlands

Ronald de Wolf
QuSoft and CWI, Amsterdam, The Netherlands
University of Amsterdam, The Netherlands

Abstract

Lasso and Ridge are important minimization problems in machine learning and statistics. They are versions of linear regression with squared loss where the vector $\theta \in \mathbb{R}^d$ of coefficients is constrained in either ℓ_1-norm (for Lasso) or in ℓ_2-norm (for Ridge). We study the complexity of quantum algorithms for finding ε-minimizers for these minimization problems. We show that for Lasso we can get a quadratic quantum speedup in terms of d by speeding up the cost-per-iteration of the Frank-Wolfe algorithm, while for Ridge the best quantum algorithms are linear in d, as are the best classical algorithms. As a byproduct of our quantum lower bound for Lasso, we also prove the first classical lower bound for Lasso that is tight up to polylog-factors.

2012 ACM Subject Classification Mathematics of computing → Mathematical optimization; Theory of computation → Quantum computation theory

Keywords and phrases Quantum algorithms, Regularized linear regression, Lasso, Ridge, Lower bounds

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.38

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2110.13086

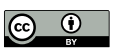
Funding *Ronald de Wolf*: Partially supported by the Dutch Research Council (NWO) through Gravitation-grant Quantum Software Consortium, 024.003.037, and through QuantERA ERA-NET Cofund project QuantAlgo 680-91-034.

Acknowledgements We thank Yi-Shan Wu and Christian Majenz for useful discussions, and Armando Bellante for pointing us to [11].

1 Introduction

1.1 Linear regression with norm constraints

One of the simplest, most useful and best-studied problems in machine learning and statistics is *linear regression*. We are given N data points $\{(x_i, y_i)\}_{i=0}^{N-1}$ where $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$, and want to fit a line through these points that has small error. In other words, we want to find a vector $\theta \in \mathbb{R}^d$ of coefficients such that the inner product $\langle \theta, x \rangle = \sum_{j=1}^{d} \theta_j x_j$ is a good predictor for the y-variable. There are different ways to quantify the error ("loss") of such a θ-vector, the most common being the squared error $(\langle \theta, x \rangle - y)^2$, averaged over the N data points (or over an underlying distribution $\mathcal{D}$ that generated the data). If we let X be the $N \times d$ matrix whose N rows are the x-vectors of the data, then we want to find a $\theta \in \mathbb{R}^d$ that minimizes $\|X\theta - y\|_2^2$. This minimization problem has a well-known closed-form solution: $\theta = (X^T X)^+ X^T y$, where the superscript "+" indicates the Moore-Penrose pseudoinverse.



In practice, unconstrained least-squares regression sometimes has problems with overfitting and often yields solutions θ where all entries are non-zero, even when only a few of the d coordinates in the x-vector really matter and one would really hope for a sparse vector θ [42, see Chapters 2 and 13]. This may be improved by "regularizing" θ via additional constraints. The most common constraints are to require that the ℓ_1-norm or ℓ_2-norm of θ is at most some bound B.[1] Linear regression with an ℓ_1-constraint is called *Lasso* (due to Tibshirani [43]), while with an ℓ_2-constraint it is called *Ridge* (due to Hoerl and Kennard [29]).

Both Lasso and Ridge are widely used for robust regression and sparse estimation in ML problems and elsewhere [44, 15]. Consequently, there has been great interest in finding the fastest-possible algorithms for them. For reasons of efficiency, algorithms typically aim at finding not the exactly optimal solution but an ε-minimizer, i.e., a vector θ whose loss is only an additive ε worse than the minimal-achievable loss. The best known results on the time complexity of classical algorithms for Lasso are an upper bound of $\tilde{\mathcal{O}}(d/\varepsilon^2)$ [28] and a lower bound of $\Omega(d/\varepsilon)$ [16] (which we actually improve to a tight lower bound in this paper, see below); for Ridge the best bound is $\tilde{\Theta}(d/\varepsilon^2)$ [28], which is tight up to logarithmic factors.[2]

1.2 Our results

We focus on the *quantum* complexity of Lasso and Ridge, investigating to what extent quantum algorithms can solve these problems faster. Table 1 summarizes the results. The upper bounds are on time complexity (total number of elementary operations and queries to entries of the input vectors) while the lower bounds are on query complexity (which itself lower bounds time complexity).

Table 1 Classical and quantum upper and lower bounds for Lasso and Ridge.

	Upper bound	**Lower bound**
Lasso	Classical [28]: $\tilde{\mathcal{O}}(d/\varepsilon^2)$ Quantum [*this work*]: $\tilde{\mathcal{O}}(\sqrt{d}/\varepsilon^2)$	Classical [*this work*]: $\tilde{\Omega}(d/\varepsilon^2)$ Quantum [*this work*]: $\Omega(\sqrt{d}/\varepsilon^{1.5})$
Ridge	Classical [28]: $\tilde{\mathcal{O}}(d/\varepsilon^2)$	Classical [28]: $\Omega(d/\varepsilon^2)$ Quantum [*this work*]: $\Omega(d/\varepsilon)$

1.2.1 Lasso

We design a quantum algorithm that finds an ε-minimizer for Lasso in time $\tilde{\mathcal{O}}(\sqrt{d}/\varepsilon^2)$. This gives a quadratic quantum speedup over the best-possible classical algorithm in terms of d, while the ε-dependence remains the same as in the best known classical algorithm.

[1] For ease of presentation we will set $B = 1$. However, one can also set B differently or even do a binary search over its values, finding a good θ for each of those values and selecting the best one at the end. Instead of putting a hard upper bound B on the norm, one may also include it as a penalty term in the objective function itself, by just minimizing the function $\|X\theta - y\|_2^2 + \lambda \|\theta\|$, where λ is a Lagrange multiplier and the norm of θ could be ℓ_1 or ℓ_2 (and could also be squared). This amounts to basically the same thing as our setup.

[2] For such bounds involving additive error ε to be meaningful, one has to put certain normalization assumptions on X and y, which are given in the body of the paper. The $\tilde{\mathcal{O}}$ and $\tilde{\Theta}$-notation hides polylogarithmic factors. It is known that $N = \mathcal{O}((\log d)/\varepsilon^2)$ data points suffice for finding an ε-minimizer, which explains the absence of N as a separate variable in these bounds.

Our quantum algorithm is based on the Frank-Wolfe algorithm, a well-known iterative convex optimization method [22]. Frank-Wolfe, when applied to a Lasso instance, starts at the all-zero vector θ and updates this in $\mathcal{O}(1/\varepsilon)$ iterations to find an ε-minimizer. Each iteration looks at the gradient of the loss function at the current point θ and selects the best among $2d$ directions for changing θ (each of the d coordinates can change positively or negatively, whence $2d$ directions). The new θ will be a convex combination of the previous θ and this optimal direction of change. Note that Frank-Wolfe automatically generates *sparse* solutions: only one coordinate of θ can change from zero to nonzero in one iteration, so the number of nonzero entries in the final θ is at most the number of iterations, which is $\mathcal{O}(1/\varepsilon)$.

Our quantum version of Frank-Wolfe does not reduce the number of iterations, which remains $\mathcal{O}(1/\varepsilon)$, but it does reduce the cost per iteration. In each iteration it selects the best among the $2d$ possible directions for changing θ by using a version of quantum minimum-finding on top of a quantum approximation algorithm for entries of the gradient (which in turn uses amplitude estimation). Both this minimum-finding and our approximation of entries of the gradient will result in approximation errors throughout. Fortunately Frank-Wolfe is a very robust method which still converges if we carefully ensure those quantum-induced approximation errors are sufficiently small.

Our quantum algorithm assumes coherent quantum query access to the entries of the data points (x_i, y_i), as well as a relatively small QRAM (quantum-readable classical-writable classical memory). We use a variant of a QRAM data structure developed by Prakash and Kerenidis [37, 33], to store the nonzero entries of our current solution θ in such a way that we can (1) quickly generate θ as a quantum state, and (2) quickly incorporate the change of θ incurred by a Frank-Wolfe iteration.[3] Because our θ is $\mathcal{O}(1/\varepsilon)$-sparse throughout the algorithm, we only need $\tilde{\mathcal{O}}(1/\varepsilon)$ bits of QRAM.

We also prove a lower bound of $\Omega(\sqrt{d}/\varepsilon^{1.5})$ quantum queries for Lasso, showing that the d-dependence of our quantum algorithm is essentially optimal, while our ε-dependence might still be slightly improvable. Our lower bound strategy "hides" a subset of the columns of the data matrix X by letting those columns have slightly more $+1$s than -1, and observes that an approximate minimizer for Lasso allows us to recover this hidden set. We then use the composition property of the adversary lower bound [12] together with a worst-case to average-case reduction to obtain a quantum query lower bound for this hidden-set-finding problem, and hence for Lasso.

Somewhat surprisingly, no tight *classical* lower bound was known for Lasso prior to this work. To the best of our knowledge, the previous-best classical lower bound was $\Omega(d/\varepsilon)$, due to Cesa-Bianchi, Shalev-Shwartz, and Shamir [16]. As a byproduct of our quantum lower bound, we use the same set-hiding approach to prove for the first time the optimal (up to logarithmic factors) lower bound of $\tilde{\Omega}(d/\varepsilon^2)$ queries for classical algorithms for Lasso.

1.2.2 Ridge

What about Ridge? Because ℓ_2 is a more natural norm for quantum states than ℓ_1, one might hope that Ridge is more amenable to quantum speedup than Lasso. Unfortunately this turns out to be wrong: we prove a quantum lower bound of $\Omega(d/\varepsilon)$ queries for Ridge, using a similar strategy as for Lasso. This shows that the classical linear dependence of the runtime on d cannot be improved on a quantum computer. Whether the ε-dependence can be improved remains an open question.

[3] Each iteration will actually change all nonzero entries of θ because the new θ is a convex combination of the old θ and a vector with one nonzero entry. Our data structure keeps track of a global scalar, which saves us the cost of separately adjusting all nonzero entries of θ in the data structure in each iteration.

1.3 Related work

As already cited in Table 1, Hazan and Koren [28] obtained an optimal classical algorithm for Ridge, and the best known classical algorithm for Lasso. Cesa-Bianchi, Shalev-Shwartz, and Shamir [16] provided a non-optimal classical lower bound for Lasso, and their idea inspired us to hide a subset among the column of the data matrix and to use a Lasso solver to find that subset (our lower bound also benefited from the way composition of the adversary bound was used in [8]).

Du, Hsieh, Liu, You, and Tao [20] also showed a quantum upper bound for Lasso based on quantizing parts of Frank-Wolfe, though their running time $\tilde{\mathcal{O}}(N^{3/2}\sqrt{d})$ is substantially worse than ours. The main goal of their paper was to establish differential privacy, not so much to obtain the best-possible quantum speedup for Lasso. They also claim an $\Omega(\sqrt{d})$ lower bound for quantum algorithms for Lasso [20, Corollary 1], without explicit dependence on ε, but we do not fully understand their proof, which goes via a claimed equivalence with quantum SVMs. Bellante and Zanero [11] recently and independently used similar techniques as we use here for our Lasso upper bound (KP-trees and amplitude estimation) to give a polynomial quantum speedup for the classical matching-pursuit algorithm, which is a heuristic algorithm for the NP-hard problem of linear regression with a sparsity constraint, i.e., with an ℓ_0-regularizer.

Another quantum approach for solving (unregularized) least-squares linear regression is based on the linear-systems algorithm of Harrow, Hassidim, and Lloyd [27]. In this type of approach, the quantum algorithm very efficiently generates a solution vector θ *as a quantum state* $\frac{1}{\|\theta\|_2}\sum_i \theta_i |i\rangle$ (which is incomparable to our goal of returning θ as a classical vector). Chakraborty, Gilyén, and Jeffery [18] used the framework of block-encodings to achieve this. Subsequently Gilyén, Lloyd, and Tang [25] obtained a "dequantized" classical algorithm for (unregularized) least-squares linear regression assuming length square sampling access to the input data, which again is incomparable to our setup. The quantum algorithm was very recently improved with an ℓ_2-regularizer by Chakraborty, Morolia, and Peduri [19], thought still producing the final output as a quantum state rather than as a classical solution.

Norm-constrained linear regression is a special case of convex optimization. Quantum algorithms for various convex optimization problems have received much attention recently. For example, there has been a sequence of quantum algorithms for solving linear and semidefinite programs starting with Brandão and Svore [14, 5, 13, 6, 3]. There have also been some polynomial speedups for matrix scaling [7, 26] and for boosting in machine learning [9, 30], as well as some general speedups for converting membership oracles for a convex feasible set to separation oracles and optimization oracles [17, 4, 2]. On the other hand Garg, Kothari, Netrapalli, and Sherif [24] showed that the number of iterations for first-order algorithms for minimizing non-smooth convex functions cannot be significantly improved on a quantum computer; recently they generalized this result to higher-order algorithms [23]. Finally, there has also been work on quantum speedups for *non-convex* problems, for instance on escaping from saddle points [45].

2 Preliminaries

Throughout the paper, d will always be the dimension of the ambient space $\mathbb{R}^d$, and log without a base will be the binary logarithm. It will be convenient for us to index entries of vectors starting from 0, so the entries x_i of a d-dimensional vector x are indexed by $i \in \{0,\dots,d-1\} = \mathbb{Z}_d$. $\mathcal{U}_N = \mathcal{U}\{0,\dots,N-1\}$ is the discrete uniform distribution over integers $0,1,2,\dots,N-1$.

2.1 Computational model and quantum algorithms

Our computational model is a classical computer (a classical random-access machine) that can invoke a quantum computer as a subroutine. The input is stored in quantum-readable read-only memory (a QROM), whose bits can be queried. The classical computer can also write bits to a quantum-readable classical-writable classical memory (a QRAM). The classical computer can send a description of a quantum circuit to the quantum computer; the quantum computer runs the circuit (which may include queries to the input bits stored in QROM and to the bits stored by the computer itself in the QRAM), measures the full final state in the computational basis, and returns the measurement outcome to the classical computer. In this model, an algorithm has time complexity T if it uses at most T elementary classical operations and quantum gates, quantum queries to the input bits stored in QROM, and quantum queries to the QRAM. The query complexity of an algorithm only measures the number of queries to the input stored in QROM. We call a (quantum) algorithm *bounded-error* if (for every possible input) it returns a correct output with probability at least $9/10$.

We will represent real numbers in computer memory using a number of bits of precision that is polylogarithmic in d, N, and $1/\varepsilon$ (i.e., $\tilde{\mathcal{O}}(1)$ bits). This ensures all numbers are represented throughout our algorithms with negligible approximation error and we will ignore those errors later on for ease of presentation.

The following is a modified version of quantum minimum-finding, which in its basic form is due to Høyer and Dürr [21]. Our proof of the more general version below is given in our full version on arXiv, and is based on a result from [5]. We also use some other Grover-based quantum algorithms as subroutines, described in our full version.

▶ **Theorem 1** (min-finding with an approximate unitary). *Let $\delta_1, \delta_2, \varepsilon \in (0,1)$, $v_0, \ldots, v_{d-1} \in \mathbb{R}$. Suppose we have a unitary $\tilde{A}$ that maps $|j\rangle|0\rangle \to |j\rangle|\Lambda_j\rangle$ such that for every $j \in \mathbb{Z}_d$, after measuring the state $|\Lambda_j\rangle$, with probability $\geq 1-\delta_2$ the first register λ of the measurement outcome satisfies $|\lambda - v_j| \leq \varepsilon$. There exists a quantum algorithm that finds an index j such that $v_j \leq \min_{k\in\mathbb{Z}_d} v_k + 2\varepsilon$ with probability $\geq 1-\delta_1 - 1000\log(1/\delta_1)\cdot\sqrt{2d\delta_2}$, using $1000\sqrt{d}\cdot\log(1/\delta_1)$ applications of $\tilde{A}$ and $\tilde{A}^\dagger$, and $\tilde{\mathcal{O}}(\sqrt{d})$ elementary gates. In particular, if $\delta_2 \leq \delta_1^2/(2000000d\log(1/\delta_1))$, that finds such a j with probability $\geq 1-2\delta_1$.*

2.2 Expected and empirical loss

Let sample set $S = \{(x_i, y_i)\}_{i=0}^{N-1}$ be a set of i.i.d. samples from $\mathbb{R}^d \times \mathbb{R}$, drawn according to an unknown distribution $\mathcal{D}$. A *hypothesis* is a function $h : \mathbb{R}^d \to \mathbb{R}$, and $\mathcal{H}$ denotes a set of hypotheses. To measure the performance of the prediction, we use a convex loss function $\ell : \mathbb{R}^2 \to \mathbb{R}$. The *expected loss* of h with respect to $\mathcal{D}$ is denoted by $L_{\mathcal{D}}(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(h(x), y)]$, and the *empirical loss* of h with respect to S is denoted by $L_S(h) = \frac{1}{N}\sum_{i\in\mathbb{Z}_N} \ell(h(x_i), y_i)$.

▶ **Definition 2.** *Let $\varepsilon > 0$. An $h \in \mathcal{H}$ is an ε-minimizer over $\mathcal{H}$ w.r.t. $\mathcal{D}$ if*

$$L_{\mathcal{D}}(h) - \min_{h'\in\mathcal{H}} L_{\mathcal{D}}(h') \leq \varepsilon.$$

▶ **Definition 3.** *Let $\varepsilon > 0$. An $h \in \mathcal{H}$ is an ε-minimizer over $\mathcal{H}$ w.r.t. sample set S if*

$$L_S(h) - \min_{h'\in\mathcal{H}} L_S(h') \leq \varepsilon.$$

2.3 Linear regression problems and their classical and quantum setup

In linear regression problems, the hypothesis class is the set of linear functions on $\mathbb{R}^d$. The goal is to find a vector θ for which the corresponding hypothesis $\langle \theta, x \rangle$ provides a good prediction of the target y. One of the most natural choices for regression problems is the squared loss

$$\ell(\hat{y}, y) = (\hat{y} - y)^2.$$

We can instantiate the expected and empirical losses as a function of θ using squared loss:

$$L_{\mathcal{D}}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(\langle \theta, x \rangle, y)] = \mathbb{E}_{(x,y)\sim\mathcal{D}}[(\langle \theta, x \rangle - y)^2],$$

$$L_S(\theta) = \frac{1}{N} \sum_{i \in \mathbb{Z}_N} \ell(\langle \theta, x \rangle, y_i) = \frac{1}{N} \sum_{i \in \mathbb{Z}_N} (\langle \theta, x \rangle - y_i)^2.$$

We also write the empirical loss as $L_S(\theta) = \frac{1}{N}\|X\theta - y\|_2^2$, where matrix entry X_{ij} is the jth entry of the vector x_i, and y is the N-dimensional vector with entries y_i. As we will see below, if the instances in the sample set are chosen i.i.d. according to $\mathcal{D}$, and N is sufficiently large, then $L_S(\theta)$ and $L_{\mathcal{D}}(\theta)$ are typically close by the law of large numbers.

In the quantum case, we assume the sample set S is stored in a QROM, which we can access by means of queries to the oracles $O_X : |i\rangle |j\rangle |0\rangle \to |i\rangle |j\rangle |X_{ij}\rangle$ and $O_y : |i\rangle |0\rangle \to |i\rangle |y_i\rangle$.

2.3.1 Lasso

The *least absolute shrinkage and selection operator*, or *Lasso*, is a special case of linear regression with a norm constraint on the vector θ: it restricts solutions to the unit ℓ_1-ball, which we denote by B_1^d. For the purpose of normalization, we require that every sample (x, y) satisfies $\|x\|_\infty \leq 1$ and $|y| \leq 1$.[4] The goal is to find a $\theta \in B_1^d$ that (approximately) minimizes the expected loss. Since the expected loss is not directly accessible, we instead find an approximate minimizer of the empirical loss. Mohri, Rostamizadeh, and Talwalkar [34] showed that with high probability, an approximate minimizer for *empirical* loss is also a good approximate minimizer for *expected* loss.

▶ **Theorem 4** ([34], Theorem 11.16). *Let $\mathcal{D}$ be an unknown distribution over $[-1,1]^d \times [-1,1]$ and $S = \{(x_i, y_i)\}_{i=0}^{N-1}$ be a sample set containing N i.i.d. samples from $\mathcal{D}$. Then, for each $\delta > 0$, with probability $\geq 1 - \delta$ over the choice of S, the following holds for all $\theta \in B_1^d$:*

$$L_{\mathcal{D}}(\theta) - L_S(\theta) \leq 4\sqrt{\frac{2\log(2d)}{N}} + 4\sqrt{\frac{\log(1/\delta)}{2N}}.$$

This theorem implies that if $N = c\log(d/\delta)/\varepsilon^2$ for sufficiently large constant c, then finding (with error probability $\leq \delta$) an ε-minimizer for the *empirical* loss L_S, implies finding (with error probability $\leq 2\delta$ taken both over the randomness of the algorithm and the choice of the sample S) a 2ε-minimizer for the *expected* loss $L_{\mathcal{D}}$.

[4] Note that if $\theta \in B_1^d$ and $\|x\|_\infty \leq 1$, then $|\langle \theta, x \rangle| \leq 1$ by Hölder's inequality.

2.3.2 Ridge

Another special case of linear regression with a norm constraint is *Ridge*, which restricts solutions to the unit ℓ_2-ball B_2^d. For the purpose of normalization, we now require that every sample (x, y) satisfies $\|x\|_2 \leq 1$ and $|y| \leq 1$. Similarly to the Lasso case, Mohri, Rostamizadeh, and Talwalkar [34] showed that with high probability, an approximate minimizer for the empirical loss is also a good approximate minimizer for the expected loss.

▶ **Theorem 5** ([34], Theorem 11.11). *Let $\mathcal{D}$ be an unknown distribution over $B_2^d \times [-1, 1]$ and $S = \{(x_i, y_i)\}_{i=0}^{N-1}$ be a sample set containing N i.i.d. samples from $\mathcal{D}$. Then, for each $\delta > 0$, with probability $\geq 1 - \delta$ over the choice of S, the following holds for all $\theta \in B_2^d$:*

$$L_{\mathcal{D}}(\theta) - L_S(\theta) \leq 8\sqrt{\frac{1}{N}} + 4\sqrt{\frac{\log(1/\delta)}{2N}}.$$

2.4 The KP-tree data structure and efficient state preparation

Kerenidis and Prakash [37, 33] gave a quantum-accessible classical data structure to store a vector θ with support t (i.e., t nonzero entries) to enable efficient preparation of the state

$$|\theta\rangle = \sum_{j \in \mathbb{Z}_d} \sqrt{\frac{|\theta_j|}{\|\theta\|_1}} |j\rangle |sign(\theta_j)\rangle .$$

We modify their data structure such that for arbitrary $a, b \in \mathbb{R}$ and $j \in \mathbb{Z}_d$, we can efficiently update a data structure for the vector θ to a data structure for the vector $a\theta + be_j$, without having to individually update all nonzero entries of the vector. We only give the definition here; for more details and analysis, see our full version on arXiv.

▶ **Definition 6** (KP-tree). *Let $\theta \in \mathbb{R}^d$ have support t. Define a KP-tree KP_θ of θ as:*

- *KP_θ is a rooted binary tree with depth $\lceil \log d \rceil$ and with $\mathcal{O}(t \log d)$ vertices.*
- *The root stores a scalar $A \in \mathbb{R} \setminus \{0\}$ and the support t of θ.*
- *Each edge of the tree is labelled by a bit.*
- *For each $j \in supp(\theta)$, there is one corresponding leaf storing $\frac{\theta_j}{A}$. The number of leaves is t.*
- *The bits on the edges of the path from the root to the leaf corresponding to the j^{th} entry of θ, form the binary description of j.*
- *Each intermediate node stores the sum of its children's absolute values.*

For $\ell \in \mathbb{Z}_{\lceil \log d \rceil}$ and $j \in \mathbb{Z}_{2^\ell}$, we define $KP_\theta(\ell, j)$ as the value of the j^{th} node in the ℓ^{th} layer, i.e., the value stored in the node that we can reach by the path according to the binary representation of j from the root. Also, we let $KP_\theta(0, 0)$ be the sum of all absolute values stored in the leaves. If there is no corresponding j^{th} node in the ℓ^{th} layer (that is, we cannot reach a node by the path according to the binary representation of j from the root), then $KP_\theta(\ell, j)$ is defined as 0. Note that both the numbering of the layer and the numbering of nodes start from 0. In the special case where θ is the all-0 vector, the corresponding tree will just have a root node with $t = 0$.

3 Quantum Algorithm for Lasso

3.1 The classical Frank-Wolfe algorithm

Below is a description of the Frank-Wolfe algorithm with approximate linear solvers. For now this is for an arbitrary convex objective function L and arbitrary compact convex domain $\mathcal{X}$ of feasible solutions; for Lasso we will later instantiate these to the quadratic loss function and

ℓ_1-ball, respectively. Frank-Wolfe finds an ε-approximate solution to a convex optimization problem, using $O(1/\varepsilon)$ iterations. It is a first-order method: each iteration assumes access to the gradient of the objective function at the current point. The algorithm considers the linearization of the objective function, and moves towards a minimizer of this linear function without ever leaving the domain $\mathcal{X}$ (in contrast to for instance projected gradient descent).

Algorithm 1 The Frank-Wolfe algorithm with approximate linear subproblems.

input : number of iterations $T > 0$; convex differentiable function L; compact convex domain $\mathcal{X}$;
Let C_L be the curvature constant of L;
Let θ^0 be an arbitrary point in $\mathcal{X}$;
for $t \leftarrow 0$ **to** T **do**
  $\tau_t = \frac{2}{t+2}$;
  find $s \in \mathcal{X}$ such that $\langle s, \nabla L(\theta^t)\rangle \leq \min_{s' \in \mathcal{X}} \langle s', \nabla L(\theta^t)\rangle + \frac{\tau_t C_L}{4}$;
  $\theta^{t+1} = (1-\tau_t)\theta^t + \tau_t s$;
end
output : θ^T;

The convergence rate of the Frank-Wolfe algorithm is affected by the "non-linearity" of the objective function L, as measured by the curvature constant C_L:

▶ **Definition 7.** *The curvature constant C_L of a convex and differentiable function $L : \mathbb{R}^d \to \mathbb{R}$ with respect to a convex domain $\mathcal{X}$ is defined as*

$$C_L \equiv \sup_{\substack{x,s \in \mathcal{X}, \gamma \in [0,1], \\ y = x + \gamma(s-x)}} \frac{2}{\gamma^2}(L(y) - L(x) - \langle \nabla L(x), (y-x)\rangle).$$

Next we give an upper bound for the curvature constant of the empirical loss function for Lasso.

▶ **Theorem 8.** *Let $S = \{(x_i, y_i)\}_{i=0}^{N-1}$ with all entries of x_i and y_i in $[-1, 1]$. Then the curvature constant C_{L_S} of L_S w.r.t. B_1^d is ≤ 8.*

Proof. We know

$$L_S(\theta) = \frac{1}{N}\|X\theta - y\|_2^2 = \frac{(X\theta - y)^T(X\theta - y)}{N} = \frac{\theta^T X^T X\theta - y^T X\theta - \theta^T X^T y + y^T y}{N},$$

which implies the Hessian of L_S is $\nabla^2 L_S(z) = \frac{2X^T X}{N}$, independent of z. By replacing sup by max because the domain is compact, we have

$$\begin{aligned} C_{L_S} &= \max_{\substack{x,s \in \mathcal{X}, \gamma \in [0,1], \\ y = x + \gamma(s-x)}} \frac{2}{\gamma^2}(L_S(y) - L_S(x) - \langle \nabla L_S(x), (y-x)\rangle) \\ &= \max_{x,s \in \mathcal{X}, \gamma \in [0,1]} \langle (s-x), \nabla^2 L_S \cdot (s-x)\rangle = \max_{x,s \in \mathcal{X}} \frac{2}{N}\|X(s-x)\|_2^2. \end{aligned}$$

Each coefficient of X is at most 1 in absolute value, and $s - x \in 2B_1^d$, hence each entry of the vector $X(s-x)$ has magnitude at most 2. Therefore $\max_{x,y \in B_1^d} \frac{2}{N}\|X(s-x)\|_2^2$ is at most 8. ◀

The original Frank-Wolfe algorithm [22] assumed that the minimization to determine the direction-of-change s was done exactly, without the additive error term $\tau_t C_{L_S}/4$ that we wrote in Algorithm 1. However, the following theorem, due to Jaggi [31], shows that solving approximate linear subproblems is sufficient for the Frank-Wolfe algorithm to converge at an $O(C_{L_S}/T)$ rate, which means one can find an ε-approximate solution with $T = O(C_{L_S}/\varepsilon)$ iterations.

▶ **Theorem 9** ([31], Theorem 1). *For each iteration $t \geq 1$, the corresponding θ^t of Algorithm 1 satisfies*

$$L_S(\theta^t) - \min_{\theta' \in B_1^d} L_S(\theta') \leq \frac{3C_{L_S}}{t+2}.$$

3.2 Approximating the quadratic loss function and entries of its gradient

In this subsection, we give a quantum algorithm to estimate the quadratic loss function $L_S(\theta)$ and entries of its gradient, given query access to entries of the vectors in $S = \{(x_i, y_i)\}_{i=0}^{N-1}$ and given a KP-tree for $\theta \in B_1^d$. One can estimate these numbers with additive error β in time roughly $1/\beta$.

We start with estimating entries of the gradient of the loss function at a given θ:

▶ **Theorem 10.** *Let $\theta \in B_1^d$, and $\beta, \delta > 0$. Suppose we have a KP-tree KP_θ of vector θ and can make quantum queries to $O_{KP_\theta} : |\ell, k\rangle |0\rangle \to |\ell, k\rangle |KP_\theta(\ell, k)\rangle$. One can implement $\tilde{U}_{\nabla L_S} : |j\rangle |0\rangle \to |j\rangle |\Lambda\rangle$ such that for all $j \in \mathbb{Z}_d$, after measuring the state $|\Lambda\rangle$, with probability $\geq 1-\delta$ the first register λ of the outcome will satisfy $|\lambda - \nabla_j L_S(\theta)| \leq \beta$, by using $\tilde{\mathcal{O}}(\frac{\log(1/\delta)}{\beta})$ applications of O_X, $O_X^\dagger$, O_y, $O_y^\dagger$, O_{KP_θ}, $O_{KP_\theta}^\dagger$, and elementary gates.*

Next we show how to estimate the value of the loss function itself at a given θ:

▶ **Theorem 11.** *Let $\theta \in B_1^d$, and $\beta, \delta > 0$. Suppose we have a KP-tree KP_θ of vector θ and can make quantum queries to $O_{KP_\theta} : |\ell, k\rangle |0\rangle \to |\ell, k\rangle |KP_\theta(\ell, k)\rangle$. Then we can implement $\tilde{U}_{L_S} : |0\rangle \to |\Lambda\rangle$ such that after measuring the state $|\Lambda\rangle$, with probability $\geq 1-\delta$ the first register λ of the outcome will satisfy $|\lambda - L_S(\theta)| \leq \beta$, by using $\tilde{\mathcal{O}}(\frac{\log(1/\delta)}{\beta})$ applications of O_X, $O_X^\dagger$, O_y, $O_y^\dagger$, O_{KP_θ}, $O_{KP_\theta}^\dagger$, and elementary gates.*

If we have multiple vectors $\theta^0, \ldots, \theta^{m-1}$, then we can apply the previous theorem conditioned on the index of the vector we care about:

▶ **Corollary 12.** *Let $\theta^0, \theta^1, \ldots, \theta^{m-1} \in B_1^d$, and $\beta, \delta > 0$. Suppose for all $h \in \mathbb{Z}_m$, we have a KP-tree KP_{θ^h} of vector θ^h and can make quantum queries to $O_{KP_\theta} : |h, \ell, k\rangle |0\rangle \to |h, \ell, k\rangle |KP_{\theta^h}(\ell, k)\rangle$. Then we can implement $\tilde{U}_{L_S} : |h\rangle |0\rangle \to |h\rangle |\Lambda\rangle$ such that for all $h \in \mathbb{Z}_m$, after measuring the state $|\Lambda\rangle$, with probability $\geq 1-\delta$ the first register λ of the outcome will satisfy $|\lambda - L_S(\theta^h)| \leq \beta$, by using $\tilde{\mathcal{O}}(\frac{\log(1/\delta)}{\beta})$ applications of O_X, $O_X^\dagger$, O_y, $O_y^\dagger$, O_{KP_θ}, $O_{KP_\theta}^\dagger$, and elementary gates.*

3.3 Quantum algorithms for Lasso with respect to S

In this subsection, we will show how to find an approximate minimizer for Lasso with respect to a given sample set S. The following algorithm simply applies the Frank-Wolfe algorithm to find an ε-minimizer for Lasso with respect to the sample set S given C, a guess for the curvature constant C_{L_S} (which our algorithm does not know in advance). Note that to find an $s \in B_1^d$ such that $\langle s, \nabla L_S(\theta^t)\rangle \leq \min_{s' \in \mathcal{X}} \langle s', \nabla L_S(\theta^t)\rangle + \tau_t C_{L_S}/4$, it suffices to only check

$s \in \{\pm e_0, \ldots, \pm e_{d-1}\}$ because the domain is B_1^d and ∇L_S is a linear function in θ. Also, by Theorem 8, the curvature constant C_{L_S} of loss function L_S is at most 8 because (x_i, y_i) is in $[-1,1]^d \times [-1,1]$ for all $i \in \mathbb{Z}_N$.

Algorithm 2 The algorithm for Lasso with a guess C for the value of the curvature constant.

input : a positive value C; additive error ε;
Let θ^0 be the d-dimensional all-zero vector;
Let $T = 6 \cdot \lceil \frac{C}{\varepsilon} \rceil$;
for $t \leftarrow 0$ **to** T **do**
  $\tau_t = \frac{2}{t+2}$;
  Let $s \in \{\pm e_0, \ldots, \pm e_{d-1}\}$ be such that $\langle \nabla L_S(\theta^t), s \rangle \leq \min_{j' \in \mathbb{Z}_d} -|\nabla_{j'} L_S(\theta^t)| + \frac{C}{8t+16}$;
  $\theta^{t+1} = (1-\tau_t)\theta^t + \tau_t s$;
end
output : θ^T;

It is worth mentioning that Algorithm 2 also outputs an ε-minimizer if its input C equals the curvature constant C_{L_S} approximately instead of exactly. For example, suppose we only know that the curvature constant C_{L_S} is between C and $2C$, where C is the input in Algorithm 2. Then the output of Algorithm 2 is still an ε-minimizer. We can see this by first observing that the error we are allowed to make for the linear subproblem in iteration t is $\frac{C_{L_S}}{4t+8} \geq \frac{C}{8t+16}$, and hence by Theorem 9, after $T = 6 \cdot \lceil \frac{C}{\varepsilon} \rceil$ iterations, the output θ^T is a $\frac{3C}{(T+2)} = \frac{3C}{6 \cdot \lceil C/\varepsilon \rceil + 2}$-minimizer for L_S. Because $\frac{3C}{6 \cdot \lceil C/\varepsilon \rceil + 2} \leq \varepsilon$, the output θ^T is therefore an ε-minimizer.

In the Lasso case, we do not know how to find a positive number C such that $C_{L_S} \in [C, 2C]$, but we know $C_{L_S} \leq 8$ by Theorem 8. Hence we can try different intervals of possible values for C_{L_S}: we apply Algorithm 2 with different input $C = 8, 4, 2, 1, 1/2, \ldots, 2^{-\lceil \log(1/\varepsilon) \rceil}$, and then we collect all outputs of Algorithm 2 with those different inputs, as candidates. After that, we compute the objective values of all those candidates, and output the one with minimum objective value. If $C_{L_S} \in (\varepsilon, 8]$, then at least one of the values we tried for C will be within a factor of 2 of the actual curvature constant C_{L_S}. Hence one of our candidates is an ε-minimizer.

However, we also need to deal with the case that $C_{L_S} \leq \varepsilon$. In this case, we consider the "one-step" version of the Frank-Wolfe algorithm, where the number of iterations is 1. But now we do not estimate $\langle \nabla L_S(\theta^t), s \rangle$ anymore (i.e., we do not solve linear subproblems anymore). We find that the only possible directions are the vertices of the ℓ_1-ball, and θ^0 is the all-zero vector, implying that θ^1, the output of one-step Frank-Wolfe, must be in $I = \{\pm e_0/3, \ldots, \pm e_{d-1}/3\}$ by the update rule of Frank-Wolfe. Besides, $C_{L_S} \leq \varepsilon$ implies that θ^1 is a $\frac{3C_{L_S}}{1+2} \leq \varepsilon$-minimizer for Lasso. Hence we simply output a $v = \arg\min_{v' \in I} L_S(v')$ if $C_{L_S} \leq \varepsilon$.

Combining the above arguments gives the following algorithm:

▶ **Theorem 13.** *Let $S = \{(x_i, y_i)\}_{i=0}^{N-1}$ be the given sample set stored in QROM. For each $\varepsilon \in (0, 0.5)$, there exists a bounded-error quantum algorithm that finds an ε-minimizer for Lasso w.r.t. sample set S using $\tilde{\mathcal{O}}(\frac{\sqrt{d}}{\varepsilon^2})$ time and $\tilde{\mathcal{O}}(\frac{1}{\varepsilon})$ QRAM and classical space.*

Proof. We will implement Algorithm 3 in $\tilde{\mathcal{O}}(\frac{\sqrt{d}}{\varepsilon^2})$ time and $\tilde{\mathcal{O}}(\frac{1}{\varepsilon})$ QRAM space. Below we analyze its different components.

Algorithm 3 The algorithm for Lasso.

input : ε;
Let $v \in \{\pm e_0/3, \ldots, \pm e_{d-1}/3\}$ be such that $L_S(v) - \min_{j\in\mathbb{Z}_d} L_S(\pm e_j/3) \le \varepsilon/10$;
Let candidate set $A = \{v\}$;
for $C \leftarrow 8, 4, 2, 1, \frac{1}{2}, \ldots, 2^{-\lceil\log(1/\varepsilon)\rceil-1}$ **do**
  RUN Algorithm 2 with inputs C and $\varepsilon/10$;
  ADD the output of Algorithm 2 to A;
end
output : $\arg\min_{w\in A} L_S(w)$;

3.3.1 Analysis of Algorithm 2

We first show that we can implement Algorithm 2 in $\tilde{\mathcal{O}}(\frac{\sqrt{d}}{\varepsilon^2})$ time. Because $C_{L_S} \le 8$ (Theorem 8), the number of iterations for Algorithm 2 with input $C = C_{L_S}$ is at most $6 \cdot \lceil\frac{8}{\varepsilon}\rceil$. However, as we mentioned above, we don't know how large C_{L_S} is exactly, so we try all possible inputs (of Algorithm 2) in Algorithm 3. Note that for every input $C \in \{8, 4, 2, 1, \frac{1}{2}, \ldots, 2^{-\lceil\log(1/\varepsilon)\rceil-1}\}$ and for every number of iterations $t \in \{1, \ldots, 6 \cdot \lceil\frac{C}{\varepsilon}\rceil\}$, $\frac{C}{4t+8}$ is at least $\frac{\varepsilon}{10}$, so it suffices to ensure that in each iteration in each of our runs of Algorithm 2, the additive error for the approximate linear subproblem is $\le \frac{\varepsilon}{10}$.

Suppose we have KP_{θ^t} for each iteration t of Algorithm 2, and suppose we can make queries to $O_{KP_{\theta^t}}$, then by Theorem 10, one can implement $\tilde{U}_{\nabla L_S} : |j\rangle|0\rangle \to |j\rangle|\Lambda\rangle$ such that for all $j \in \mathbb{Z}_d$, after measuring the state $|\Lambda\rangle$, with probability $\ge 1 - \frac{\varepsilon^2}{2d\cdot 10^{20}\cdot\log^6(1/\varepsilon)}$ the first register λ of the measurement outcome will satisfy $|\lambda - \nabla_j L_S(\theta)| \le \frac{\varepsilon}{20}$, by using $\tilde{\mathcal{O}}(\frac{\log(d/\varepsilon)}{\varepsilon})$ time and queries to $O_{KP_{\theta^t}}$, $O^\dagger_{KP_{\theta^t}}$. Then by Theorem 1, with failure probability at most $\frac{\varepsilon}{10000\log(1/\varepsilon)}$, one can find $s \in \{\pm e_0, \ldots, \pm e_{d-1}\}$ such that $\langle\nabla L_S(\theta^t), s\rangle \le \min_{j'\in\mathbb{Z}_d} -|\nabla_{j'} L_S(\theta^t)| + 2\cdot\frac{\varepsilon}{20}$, by using $\tilde{\mathcal{O}}(\sqrt{d}\cdot\log(1/\varepsilon))$ applications of $\tilde{U}_{\nabla L_S}$ and $\tilde{U}^\dagger_{\nabla L_S}$, and $\tilde{\mathcal{O}}(\sqrt{d})$ elementary gates.

For each iteration t in Algorithm 2, we also maintain KP_{θ^t} and hence we can make quantum queries to $O_{KP_{\theta^t}}$. The cost for constructing KP_{θ^0} and the cost for updating KP_{θ^t} to $KP_{\theta^{t+1}}$ is $\tilde{\mathcal{O}}(1)$ for both time and space by (shown in our full version). Moreover, the total number of iterations T is at most $6\cdot\lceil\frac{8}{\varepsilon}\rceil$ in Algorithm 2 because $C_{L_S} \le 8$, and hence the space cost for maintaining KP_{θ^t} and implementing $O_{KP_{\theta^t}}$ is $\tilde{\mathcal{O}}(\frac{1}{\varepsilon})$ bits. Hence we can implement Algorithm 2 with failure probability at most $\lceil\frac{8}{\varepsilon}\rceil\cdot\frac{6\varepsilon}{10000\log(1/\varepsilon)}$ using $\tilde{\mathcal{O}}(\frac{\sqrt{d}}{\varepsilon^2})$ time and $\tilde{\mathcal{O}}(\frac{1}{\varepsilon})$ bits of QRAM and classical space.

3.3.2 Analysis of Algorithm 3

Now we show how to implement Algorithm 3 with failure probability at most 1/10 using $\tilde{\mathcal{O}}(\frac{\sqrt{d}}{\varepsilon^2})$ time. By Corollary 12, one can implement $\tilde{U}_{L_S} : |j\rangle|0\rangle \to |j\rangle|\Lambda\rangle$ such that for all $j \in \mathbb{Z}_d$, after measuring the state $|\Lambda\rangle$, with failure probability at most $\frac{1}{2d\cdot 10^{16}}$ the first register λ of the outcome will satisfy $|\lambda - L_S(e_j/3)| \le \varepsilon/20$ using $\tilde{\mathcal{O}}(\frac{1}{\varepsilon})$ time. Then by Theorem 1, with failure probability at most $0.0001 + 1000\cdot\log(1000)\sqrt{\frac{2d}{2d\cdot 10^{16}}} \le \frac{2}{1000}$ we can find $v \in \{\pm e_0/3, \ldots, \pm e_{d-1}/3\}$ such that $L_S(v) - \min_{j\in\mathbb{Z}_d} L_S(\pm e_j/3) \le 2\cdot\varepsilon/20 = \varepsilon/10$ by using $\tilde{\mathcal{O}}(\sqrt{d})$ applications of $\tilde{U}_{L_S}$ and $\tilde{U}^\dagger_{L_S}$ and $\tilde{\mathcal{O}}(\sqrt{d})$ elementary gates, hence $\tilde{\mathcal{O}}(\frac{\sqrt{d}}{\varepsilon})$ time.

Because Algorithm 3 runs Algorithm 2 $\lceil\log(1/\varepsilon)\rceil$ times and each run fails with probability at most $\lceil\frac{8}{\varepsilon}\rceil\cdot\frac{6\varepsilon}{10000\log(1/\varepsilon)}$, the candidate set A, with failure probability $\lceil\frac{8}{\varepsilon}\rceil\cdot\frac{6\varepsilon}{10000\log(1/\varepsilon)}\cdot\lceil\log(1/\varepsilon)\rceil + \frac{2}{1000} \le \frac{1}{20}$, contains an $\frac{\varepsilon}{10}$-minimizer. To output $\arg\min_{w\in A} L_S(w)$, we use

Theorem 11 to evaluate $L_S(w)$ for all $w \in A$ with additive error $\frac{\varepsilon}{10}$ with failure probability at most $\frac{1}{40\log(1/\varepsilon)}$, and hence we find an $\varepsilon/10$-minimizer among A with probability at least $1 - 1/20 - \lceil \log(1/\varepsilon) \rceil \cdot \frac{1}{40\log(1/\varepsilon)} \geq 0.9$. Because the candidate set A contains an $\frac{\varepsilon}{10}$-minimizer for Lasso, the $\frac{\varepsilon}{10}$-minimizer among A is therefore an ε-minimizer for Lasso. The QRAM and classical space cost for each run is at most $\tilde{\mathcal{O}}(\frac{1}{\varepsilon})$ because the space cost for Algorithm 2 is $\tilde{\mathcal{O}}(\frac{1}{\varepsilon})$. Hence the total cost for implementing Algorithm 3 is $\tilde{\mathcal{O}}(\frac{\sqrt{d}}{\varepsilon^2})$ time and $\tilde{\mathcal{O}}(\frac{1}{\varepsilon})$ bits of QRAM and classical space. ◀

3.4 Quantum algorithms for Lasso with respect to $\mathcal{D}$

In the previous subsection, we showed that we can find an ε-minimizer for Lasso with respect to sample set S. Here we show how we can find an ε-minimizer for Lasso with respect to distribution $\mathcal{D}$. First sample a set S of $N = \tilde{\mathcal{O}}((\log d)/\varepsilon^2)$ i.i.d. samples from $\mathcal{D}$, which is the input that will be stored in QROM, and then find an $\varepsilon/2$-minimizer for Lasso with respect to S by Theorem 13. By Theorem 4, with high probability, an $\varepsilon/2$-minimizer for Lasso with respect to S will be an ε-minimizer for Lasso with respect to distribution $\mathcal{D}$. Hence we obtain the following corollary:

▶ **Corollary 14.** *Let $S = \{(x_i, y_i)\}_{i=0}^{N-1}$ be the given sample set, sampled i.i.d. from $\mathcal{D}$. For arbitrary $\varepsilon > 0$, if $N = \tilde{\mathcal{O}}(\frac{\log d}{\varepsilon^2})$, then there exists a bounded-error quantum algorithm that finds an ε-minimizer for Lasso w.r.t. distribution $\mathcal{D}$ using $\tilde{\mathcal{O}}(\frac{\sqrt{d}}{\varepsilon^2})$ queries to O_X, O_y and elementary gates, and using $\tilde{\mathcal{O}}(\frac{1}{\varepsilon})$ space (QRAM and classical bits).*

In our full version on arXiv we show that we can also avoid the usage of QRAM in the above corollary with $\tilde{\mathcal{O}}(1/\varepsilon)$ extra overhead.

▶ **Corollary 15.** *Let $S = \{(x_i, y_i)\}_{i=0}^{N-1}$ be the given sample set, sampled i.i.d. from $\mathcal{D}$. For arbitrary $\varepsilon > 0$, if $N = \tilde{\mathcal{O}}(\frac{\log d}{\varepsilon^2})$, then there exists a bounded-error quantum algorithm that finds an ε-minimizer for Lasso w.r.t. $\mathcal{D}$ using $\tilde{\mathcal{O}}(\frac{\sqrt{d}}{\varepsilon^3})$ queries to O_X, O_y and elementary gates, and using $\tilde{\mathcal{O}}(\frac{1}{\varepsilon})$ classical bits.*

4 Quantum query lower bounds for Lasso

In this section we prove a quantum lower bound of $\Omega(\sqrt{d}/\varepsilon^{1.5})$ queries for Lasso. To show such a lower bound, we define a certain set-finding problem, and show how it can be solved by an algorithm for Lasso. After that, we show that the worst-case set-finding problem can be seen as the composition of two problems, which have query complexities $\Omega(\sqrt{d/\varepsilon})$ and $\Omega(1/\varepsilon)$, respectively. Then the composition property of the quantum adversary bound implies a $\Omega(\sqrt{d/\varepsilon} \cdot 1/\varepsilon) = \Omega(\sqrt{d}/\varepsilon^{1.5})$ query lower bound for Lasso.

4.1 Finding a hidden set W using a Lasso solver

Let $p \in (0, 1/2)$, $W \subset \mathbb{Z}_d$, and $\overline{W} = \mathbb{Z}_d \setminus W$. Define the distribution $\mathcal{D}_{p,W}$ over $(x, y) \in \{-1, 1\}^d \times \{-1, 1\}$ as follows. For each $j' \in \overline{W}$, $x_{j'}$ is generated according to $\Pr[x_{j'} = 1] = \Pr[x_{j'} = -1] = 1/2$, and for each $j \in W$, x_j is generated according to $\Pr[x_j = 1] = 1/2 + p$. And y is generated according to $\Pr[y = 1] = 1$. The goal of the distributional set-finding problem $\mathrm{DSF}_{\mathcal{D}_{p,W}}$ with respect to $\mathcal{D}_{p,W}$ is to output a set $\tilde{W}$ such that $|\tilde{W} \Delta W| \leq w/200$, given M samples from $\mathcal{D}_{p,W}$. One can think of the $M \times d$ matrix of samples as "hiding" the set W: the columns corresponding to $j \in W$ are likely to have more 1s than -1s, while the columns corresponding to $j \in \overline{W}$ have roughly as many 1s as -1s. A Lasso-solver can help us to find the hidden set W approximately. Precisely, algorithms that find an $\varepsilon/8000$-minimizer for Lasso with respect to $\mathcal{D}_{p,W}$ can also find a set $\tilde{W} \subset \mathbb{Z}_d$ such that $|W \Delta \tilde{W}| \leq w/200$.

▶ **Theorem 16.** *Let $\varepsilon \in (2/d, 1/100)$, w be eitehr $\lfloor 1/\varepsilon \rfloor$ or $\lfloor 1/\varepsilon \rfloor - 1$, $p = 1/(2\lfloor 1/\varepsilon \rfloor)$, and $W \subset \mathbb{Z}_d$ be a set of size w. Let θ be an $\varepsilon/8000$-minimizer for Lasso w.r.t. $\mathcal{D}_{p,W}$. Then the set $\tilde{W}$ that contains the indices of the entries of θ whose absolute value is $\geq \varepsilon/3$ satisfies $|W \Delta \tilde{W}| \leq w/200$.*

4.2 Worst-case quantum query lower bound for the set-finding problem

Here we will define the worst-case set-finding problem and then provide a quantum query lower bound for it. Before we step into the query lower bound for the worst-case set-finding problem, we have to introduce the lower bounds for the following problems first.

consider the *exact set-finding problem*: given input $x = x_0 \dots x_{d-1} \in \{0,1\}^d$ with at most w 1s, find the set W of all indices j with $x_j = 1$ (equivalently, learn x). To see the query lower bound for this problem, we consider the identity function where both domain and codomain are $\mathcal{Z} = \{z \in \{0,1\}^d : |z| = w\}$, and give a lower bound for computing this. If we can compute the identity function, then we can simply check the output string $x_0, x_1, \dots, x_{d-1}$ and collect all indices j with $x_j = 1$.

▶ **Theorem 17.** *Let w be an integer satisfying $0 < w \leq d/2$, $W \subset \mathbb{Z}_d$ with size w, and $x \in \{0,1\}^d$ such that $x_j = 1$ if $j \in W$ and $x_{j'} = 0$ if $j' \in \overline{W}$. Suppose we have query access to x. Then every quantum bounded-error algorithm to find W makes at least $\frac{1}{8}\sqrt{dw}$ queries.*

Using the same method, we give a lower bound for the *approximate* set-finding problem $\mathrm{ASF}_{d,w}$, which is to find a set $\tilde{W} \subset \mathbb{Z}_d$ such that $|W \Delta \tilde{W}| \leq w/200$. The intuition is that if we could find such a $\tilde{W}$ then we can "correct" it to W itself using a small number of Grover searches, so finding a good approximation $\tilde{W}$ is not much easier than finding W itself.

▶ **Theorem 18.** *Let w be an integer satisfying $0 < w \leq d/2$, $W \subset \mathbb{Z}_d$ with size w, and $x \in \{0,1\}^d$ such that $x_j = 1$ if $j \in W$ and $x_{j'} = 0$ if $j' \in \overline{W}$. Suppose we have query access to x. Then every bounded-error quantum algorithm that outputs $\tilde{W} \subset \mathbb{Z}_d$ satisfying $|W \Delta \tilde{W}| \leq w/200$ makes $\Omega(\sqrt{dw})$ queries.*

Next we consider the *Hamming-weight distinguisher problem* $\mathrm{HD}_{\ell,\ell'}$: given a $z \in \{0,1\}^N$ of Hamming weight ℓ or ℓ', distinguish these two cases. The adversary bound gives the following bound (a special case of Nayak and Wu [35] based on the polynomial method [10]).

▶ **Theorem 19.** *Let $N \in 2\mathbb{Z}_+$, $z \in \{0,1\}^N$, and $p \in (0, 0.5)$ be multiple of $1/N$. Suppose we have query access to z. Then every bounded-error quantum algorithm that computes $HD_{\frac{N}{2},N(\frac{1}{2}+p)}$ makes $\Omega(1/p)$ queries.*

The above theorem implies a lower bound of $\Omega(1/p)$ queries for $\mathrm{HD}_{\frac{N}{2},N(\frac{1}{2}+p)}$. One can also think of the input bits as ± 1 and in this case, the goal is to distinguish whether the entries add up to 0 or to $2pN$. For convenience, we abuse the notation $\mathrm{HD}_{\frac{N}{2},N(\frac{1}{2}+p)}$ also for the problem with ± 1 inputs. Now we are ready to prove a lower bound for the *worst-case set-finding problem* $\mathrm{WSF}_{d,w,p,N}$: given a matrix $X \in \{-1,1\}^{N \times d}$ where each column-sum is either $2pN$ or 0, the goal is to find a set $\tilde{W} \subset \mathbb{Z}_d$ such that $|\tilde{W} \Delta W| \leq w/200$, where W is the set of indices for those columns whose entries add up to $2pN$ and $w = |W|$. One can see that this problem is actually a composition of the approximate set-finding problem and the Hamming-weight distinguisher problem. Composing the relational problem $\mathrm{ASF}_{d,w}$ with d valid inputs of $\mathrm{HD}_{\frac{N}{2},N(\frac{1}{2}+p)}$, exactly w of which evaluate to 1, we can see that the d-bit string given by the values of $\mathrm{HD}_{\frac{N}{2},N(\frac{1}{2}+p)}$ on these d inputs, is a valid input for $\mathrm{ASF}_{d,w}$. In other words, the set of valid inputs for $\mathrm{WSF}_{d,w,p,N}$, or equivalently, the set of valid inputs for the composed problem $\mathrm{ASF}_{d,w} \circ (\mathrm{HD}_{\frac{N}{2},N(\frac{1}{2}+p)})^d$ is

$$\{(x^{(1)},\dots,x^{(d)}) \in \mathcal{P}^d : |\mathrm{HD}_{\frac{N}{2},N(\frac{1}{2}+p)}(x^{(1)}) \dots \mathrm{HD}_{\frac{N}{2},N(\frac{1}{2}+p)}(x^{(d)})| = w\},$$

where $\mathcal{P} = \{x \in \{0,1\}^N : |x| \in \{N/2, N/2 + pN\}\}$. The next theorem by Belovs and Lee shows that the quantum query complexity of the composed problem $\mathrm{ASF}_{d,w} \circ (\mathrm{HD}_{\frac{N}{2},\frac{N+2pN}{2}})^d$ is at least the product of the complexities of the two composing problems:

▶ **Theorem 20** ([12], Corollary 27). *Let $f \subseteq S \times T$, with $S \subseteq \{0,1\}^d$, be a relational problem with bounded-error quantum query complexity L. Assume that f is efficiently verifiable, that is given some $t \in T$ and oracle access to $x \in S$, there exists a bounded-error quantum algorithm that verifies whether $(x,t) \in f$ using $o(L)$ queries to x. Let $D \subseteq \{0,1\}^N$ and $g : D \to \{0,1\}$ be a Boolean function whose bounded-error quantum query complexity is Q. Then the bounded-error quantum query complexity of the relational problem $f \circ g^d$, restricted to inputs $x \in \{0,1\}^{dN}$ such that $g^d(x) \in S$, is $\Omega(LQ)$.*

Applying Theorem 20 with the lower bounds of Theorem 19 and Theorem 18, we obtain:

▶ **Corollary 21.** *Let $N \in 2\mathbb{Z}_+$ and $p \in (0, 0.5)$ be an integer multiple of $1/N$. Given a matrix $X \in \{-1,+1\}^{N\times d}$ such that there exists a set $W \subseteq \mathbb{Z}_d$ with size w and*

- *For every $j \in W$, $\sum_{i\in\mathbb{Z}_N} X_{ij} = 2pN$.*
- *For every $j' \in \overline{W}$, $\sum_{i\in\mathbb{Z}_N} X_{ij'} = 0$.*

Suppose we have query access to X. Then every bounded-error quantum algorithm that computes $\tilde{W}$ such that $|W \Delta \tilde{W}| \le w/200$, uses $\Omega(\sqrt{dw}/p)$ queries to O_X.

4.3 Worst-case to average-case reduction for the set-finding problem

Our goal is to prove a lower bound for Lasso algorithms that have high success probability w.r.t. the distribution $\mathcal{D}_{p,W}$, yet the lower bound of the previous subsection is for *worst-case* instances. In this subsection, we will connect these by providing a worst-case to average-case reduction for the set-finding problem. After that, by simply combining with the query lower bound for the worst-case set-finding problem and the reduction from the distributional set-finding problem to Lasso, we obtain an $\Omega(\sqrt{d}/\varepsilon^{1.5})$ query lower bound for Lasso.

▶ **Theorem 22.** *Let $N \in 2\mathbb{Z}_+$, $p \in (0, 0.5)$ be an integer multiple of $1/N$, w be a natural number between 2 to $d/2$, and M be a natural number. Suppose $X \in \{-1,+1\}^{N\times d}$ is a valid input for $\mathrm{WSF}_{d,w,p,N}$, and let $W \subset \mathbb{Z}_d$ be the set of the w indices of the columns of X whose entries add up to $2pN$. Let $R \in \mathbb{Z}_N^{M\times d}$ be a matrix whose entries are i.i.d. samples from $\mathcal{U}_N$, and define $X' \in \{-1,1\}^{M\times d}$ as $X'_{ij} = X_{R_{ij}j}$. Then the M vectors $(X'_i, 1)$, where X'_i is the ith row of X' and $i \in \mathbb{Z}_M$, are i.i.d. samples from $\mathcal{D}_{p,W}$.*

Proof. Every entry of R is a sample from $\mathcal{U}_N$, so $X_{R_{ij}j}$ is uniformly chosen from the entries of the jth column of X. Moreover, because every valid input W for $\mathrm{WSF}_{d,w,p,N}$ satisfies that for every $j \in W$, $\Pr_{i\sim\mathcal{U}_N}[X_{ij} = 1] = 1/2 + p$ and for every $j' \in \overline{W}$, $\Pr_{i\sim\mathcal{U}_N}[X_{ij'} = 1] = 1/2$, we know $(X'_i, 1)$ is distributed as $D_{p,W}$. ◀

The above theorem tells us that we can convert an instance of $\mathrm{WSF}_{d,w,p,N}$ to an instance of $\mathrm{DSF}_{\mathcal{D}_{p,W}}$. Note that we can produce matrix R offline and therefore we can construct the oracle $O_{X'} : |i\rangle|j\rangle|0\rangle \to |i\rangle|j\rangle|X_{R_{ij}j}\rangle$ using 1 query to $O_X : |i\rangle|j\rangle|0\rangle \to |i\rangle|j\rangle|X_{ij}\rangle$ (and some other elementary gates, which is irrelevant to the number of queries). Also observe that if $M = 10^{12} \cdot \lceil \log d \rceil \cdot \lfloor 1/\varepsilon \rfloor^2 = \mathcal{O}((\log d)/\varepsilon^2)$ and hence $S' = \{(X'_i, 1)\}_{i=0}^{M-1}$ is a sample

set with M i.i.d. samples from $\mathcal{D}_{p,W}$, then by Theorem 4, with probability $\geq 9/10$, an $\varepsilon/16000$-minimizer for Lasso with respect to S' is also an $\varepsilon/8000$-minimizer for Lasso with respect to distribution $\mathcal{D}_{p,W}$. By Theorem 16, an $\varepsilon/8000$-minimizer for Lasso with respect to distribution $\mathcal{D}_{p,W}$ can be used to output a set $\tilde{W} \subset \mathbb{Z}_d$ such that $|\tilde{W} \Delta W| \leq w/200$, where W is the set of indices for those columns of X whose entries add up to $2pN$. Hence we have a reduction from the worst-case set-finding problem to Lasso. By the reduction above and by plugging $w = \lfloor 1/\varepsilon \rfloor$ and $p = 1/(2\lfloor 1/\varepsilon \rfloor)$ in Corollary 21 (and N an arbitrary natural number such that $pN \in \mathbb{N}$), we obtain a lower bound of $\Omega(\sqrt{d}/\varepsilon^{1.5})$ queries for $\mathrm{WSF}_{d,w,p,N}$, and hence the main result of this section: a lower bound of $\Omega(\sqrt{d}/\varepsilon^{1.5})$ for Lasso.

▶ **Corollary 23.** *Let $\varepsilon \in (2/d, 1/100)$, $w = \lfloor 1/\varepsilon \rfloor$, $p = 1/(2\lfloor 1/\varepsilon \rfloor)$, and $W \subset \mathbb{Z}_d$ with size w. Every bounded-error quantum algorithm that computes an ε-minimizer for Lasso w.r.t. $\mathcal{D}_{p,W}$ uses $\Omega(\sqrt{d}/\varepsilon^{1.5})$ queries.*

4.4 Classical lower bound for Lasso

In the full version of this paper on arXiv we show how this quantum lower bound approach can be modified to prove, for the first time, a lower bound of $\tilde{\Omega}(d/\varepsilon^2)$ on the classical query complexity of Lasso. This lower bound is optimal up to logarithmic factors.

5 Quantum query lower bound for Ridge

Recall that Ridge's setup assumes the vectors in the sample set are normalized in ℓ_2 rather than ℓ_∞ as in Lasso. We modify the distribution to $\mathcal{D}'_{p,W}$ over $(x, y) \in \{-1/\sqrt{d}, 1/\sqrt{d}\}^d \times \{-1, 1\}$ as follows. Let $p \in (0, 1/4)$, $W \subset \mathbb{Z}_d$, and $\overline{W} = \mathbb{Z}_d \setminus W$. For each $j' \in \overline{W}$, $x_{j'}$ is generated according to $\Pr[x_{j'} = -1/\sqrt{d}] = 1/2 + p$; for each $j \in W$, x_j is generated according to $\Pr[x_j = 1/\sqrt{d}] = 1/2 + p$; y is generated according to $\Pr[y = 1] = 1$. Now again we want to solve a distributional set-finding problem with respect to $\mathcal{D}'_{p,W}$, given M samples from $\mathcal{D}'_{p,W}$. Similar to the Lasso case, one can think of the $M \times d$ matrix of samples as "hiding" the set W: the columns corresponding to $j \in W$ are likely to have more $1/\sqrt{d}$'s than $-1/\sqrt{d}$'s, while the columns corresponding to $j \in \overline{W}$ are likely to have more $-1/\sqrt{d}$'s than $1/\sqrt{d}$'s.

In this section let $\theta^* = \sum_{j \in \mathbb{Z}_d} \frac{e_j}{\sqrt{d}} (-1)^{[j \in \overline{W}]}$ and note that for every $\theta \in \mathbb{R}^d$,

$$\begin{aligned}
L_{\mathcal{D}'_{p,W}}(\theta) =& \mathbb{E}_{(x,y)\sim\mathcal{D}'_{p,W}}[\langle \theta, x\rangle^2] - 2\mathbb{E}_{(x,y)\sim\mathcal{D}'_{p,W}}[\langle \theta, x\rangle] + 1 \\
=& (\mathbb{E}_{(x,y)\sim\mathcal{D}'_{p,W}}[\langle \theta, x\rangle^2] - \mathbb{E}_{(x,y)\sim\mathcal{D}'_{p,W}}[\langle \theta, x\rangle]^2) \\
&+ \mathbb{E}_{(x,y)\sim\mathcal{D}'_{p,W}}[\langle \theta, x\rangle]^2 - 2\mathbb{E}_{(x,y)\sim\mathcal{D}'_{p,W}}[\langle \theta, x\rangle] + 1 \\
=& \|\theta\|_2^2 \cdot (1 - 4p^2)/d + (\mathbb{E}_{(x,y)\sim\mathcal{D}'_{p,W}}[\langle \theta, x\rangle] - 1)^2 \\
=& \|\theta\|_2^2 \cdot (1 - 4p^2)/d + (2p\langle \theta, \theta^*\rangle - 1)^2,
\end{aligned}$$

where the third equality holds because $\langle \theta, x\rangle$ is a sum of independent random variables and hence its variance is the sum of the variances of the terms $\theta_i x_i$ (which are $\theta_i^2(1 - 4p^2)/d$).

Next we show that θ^* is the minimizer for Ridge with respect to $\mathcal{D}'_{p,W}$.

▶ **Theorem 24.** *Let $w = \lfloor d/2 \rfloor$ and $W \subset \mathbb{Z}_d$ be a set of size w, and let $\varepsilon \in (1000/d, 1/10000)$ and $p = 1/\lfloor 1/\varepsilon \rfloor$. Then $\theta^* = \sum_{j \in \mathbb{Z}_d} \frac{e_j}{\sqrt{d}} (-1)^{[j \in \overline{W}]}$ is the minimizer for Ridge w.r.t. $\mathcal{D}'_{p,W}$.*

Proof. Let $\theta = \sum_{j\in\mathbb{Z}_d} \theta_j e_j \in B_2^d$ be a minimizer. We want to show $\theta_j = \theta_j^*$ for every $j \in \mathbb{Z}_d$. Note that if $\theta_j \cdot (-1)^{[j\in\overline{W}]} < 0$, then we can flip the sign of θ_j to get a smaller objective value, that is,

$$\begin{aligned} L_{\mathcal{D}'_{p,W}}(\theta') - L_{\mathcal{D}'_{p,W}}(\theta) &= (\|\theta'\|_2^2 - \|\theta\|_2^2)\cdot(1-4p^2)/d + (2p\langle\theta',\theta^*\rangle - 1)^2 - (2p\langle\theta,\theta^*\rangle - 1)^2 \\ &= (2p\langle\theta'-\theta,\theta^*\rangle)(2p\langle\theta'+\theta,\theta^*\rangle - 2) \\ &= (-4p\theta_j\cdot(-1)^{[j\in\overline{W}]})(2p\langle\theta'+\theta,\theta^*\rangle - 2) < 0, \end{aligned}$$

where $\theta' = \sum_{k\in\mathbb{Z}_d\setminus\{j\}} \theta_k e_k - \theta_j e_j$, and the last inequality is because $-4p\theta_j\cdot(-1)^{[j\in\overline{W}]} > 0$ and $2p\langle\theta'+\theta,\theta^*\rangle \le 2p\|\theta'+\theta\|_2\cdot\|\theta^*\|_2 \le 4p \le 1$. Since θ was assumed a minimizer, for all $j\in\mathbb{Z}_d$ the sign of θ_j must be $(-1)^{[j\in\overline{W}]}$.

Second, we show that we must have $|\theta_0| = |\theta_1| = \cdots = |\theta_{d-1}|$. Suppose, towards a contradiction, that this is not the case. Consider $\theta' = \sum_{j\in\mathbb{Z}_d} ue_j\cdot(-1)^{[j\in\overline{W}]}$, where $u = \sqrt{\sum_{j\in\mathbb{Z}_d}|\theta_j|^2/d}$. We have

$$\begin{aligned} L_{\mathcal{D}'_{p,W}}(\theta') - L_{\mathcal{D}'_{p,W}}(\theta) &= (2p\langle\theta'-\theta,\theta^*\rangle)(2p\langle\theta'+\theta,\theta^*\rangle - 2) \\ &= (2p/\sqrt{d})\cdot(du - \sum_{j\in\mathbb{Z}_d}|\theta_j|)\cdot(2p\langle\theta'+\theta,\theta^*\rangle - 2) < 0. \end{aligned}$$

The last inequality holds because again $2p\langle\theta'+\theta,\theta^*\rangle \le 4p \le 1$ and in addition,

$$d\cdot\sum_{j\in\mathbb{Z}_d}|\theta_j|^2 > (\sum_{j\in\mathbb{Z}_d}|\theta_j|)^2$$

by the Cauchy–Schwarz inequality (which is strict if the $|\theta_j|$ are not all equal). Hence if θ is indeed a minimizer, then its entries must all have the same magnitude.

Now we know a minimizer θ must be in the same direction as θ^*, we just don't know yet that the magnitudes of its entries are $1/\sqrt{d}$. Suppose $\|\theta\|_2 = u \le 1$ and $\theta = u\cdot\theta^*$, then

$$L_{\mathcal{D}'_{p,W}}(\theta) = \|\theta\|_2^2\cdot(1-4p^2)/d + (2p\langle\theta,\theta^*\rangle - 1)^2 = (u^2(1-4p^2)/d + (2pu-1)^2).$$

The discriminant of $f(u) = u^2(1-4p^2)/d + (2pu-1)^2$ is less than 0, and $u = \frac{2p}{4p^2+(1-4p^2)/d}$ is the global minimizer of $f(u)$. Note that $u = \frac{2p}{4p^2+(1-4p^2)/d} > 1$, and hence $f(1) \le f(u)$ for every $u \le 1$. Therefore we know θ^* is the minimizer for Ridge with respect to $\mathcal{D}'_{p,W}$. ◀

Next we show that the inner product between the minimizer and an approximate minimizer for Ridge will be close to 1.

▶ **Theorem 25.** *Let $w = \lfloor d/2\rfloor$, $W \subset \mathbb{Z}_d$ be a set of size w, $\varepsilon \in (1000/d, 1/10000)$, and $p = 1/\lfloor 1/\varepsilon\rfloor$. Suppose $\theta \in B_2^d$ is an $\varepsilon/1000$-minimizer for Ridge w.r.t. $\mathcal{D}'_{p,W}$. Then $\langle\theta,\theta^*\rangle \ge 0.999$.*

Proof. Because θ is an $\varepsilon/1000$-minimizer, we have

$$\begin{aligned} &0.001\varepsilon \ge L_{\mathcal{D}'_{p,W}}(\theta) - L_{\mathcal{D}'_{p,W}}(\theta^*) = (1-4p^2)\cdot(\|\theta\|_2^2 - 1)/d + (2p\langle\theta,\theta^*\rangle - 1)^2 - (2p-1)^2 \\ &\implies 2p\langle\theta,\theta^*\rangle \ge 1 - \sqrt{1-4p+4p^2+0.001\varepsilon - (1-4p^2)\cdot(\|\theta\|_2^2-1)/d}. \end{aligned}$$

Letting $z = 4p - 4p^2 - 0.001\varepsilon + (1-4p^2)\cdot(\|\theta\|_2^2 - 1)/d$, we have

$$\begin{aligned} 2p\langle\theta,\theta^*\rangle \geq& 1-\sqrt{1-z} \geq 1-(1-z/2) = z/2 \\ =& 2p - 2p^2 + (1-4p^2)\cdot(\|\theta\|_2^2-1)/d - 0.001\varepsilon, \end{aligned}$$

where the second inequality holds because $z \in (0,1)$. Dividing both sides by $2p$, we have

$$\langle\theta,\theta^*\rangle \geq 1 - p + (1-4p^2)\cdot(\|\theta\|_2^2-1)/(2pd) - 0.0005\varepsilon/p.$$

Because $\theta \in B_2^d$, $p = 1/\lfloor 1/\varepsilon \rfloor$, and $\varepsilon \in (1000/d, 1/10000)$, we get $\langle\theta,\theta^*\rangle \geq 0.999$. ◀

Combining the above theorem with the following theorem, we can see how to relate the entries of an approximate minimizer for Ridge with respect to $\mathcal{D}'_{p,W}$ to the elements of the hidden set W.

▶ **Theorem 26.** *Suppose $\theta \in B_2^d$ satisfies $\langle\theta,\theta^*\rangle \geq 1 - 0.001$. Then $\#\{j \in \mathbb{Z}_d \mid \theta_j \cdot \theta^*_j \leq 0\} \leq d/500$.*

Proof. If $\theta_j \cdot \theta^*_j \leq 0$ then $|\theta_j - \theta^*_j| \geq |\theta^*_j| = \frac{1}{\sqrt{d}}$, hence using Theorem 25 we have

$$\begin{aligned} \frac{1}{d}\#\{j \in \mathbb{Z}_d \mid \theta_j\cdot\theta^*_j \leq 0\} \leq \|\theta-\theta^*\|_2^2 &= \|\theta\|_2^2 + \|\theta^*\|_2^2 - 2\langle\theta,\theta^*\rangle \\ &\leq 2 - 2(1-0.001) = 1/500. \end{aligned}$$

◀

We know $\theta^* = \sum_{j\in\mathbb{Z}_d} \frac{e_j}{\sqrt{d}}(-1)^{[j\in\overline{W}]}$, so by looking at the signs of entries of θ, we can find an index set $\tilde{W} = \{j \in \mathbb{Z}_d : \theta_j > 0\}$ satisfying that $|W\Delta\tilde{W}| \leq d/500 \leq w/200$ because $w = \lfloor d/2 \rfloor$. Therefore, once we have an $\varepsilon/1000$-minimizer for Ridge with respect to $\mathcal{D}'_{p,W}$, we can solve $\mathrm{DSF}_{\mathcal{D}'_{p,W}}$.

With the reduction from $\mathrm{DSF}_{\mathcal{D}'_{p,W}}$ to Ridge, we here show (similar to Lasso) a lower bound for the *worst-case symmetric set-finding problem* $\mathrm{WSSF}_{d,w,p,N}$: given a matrix $X \in \{-1/\sqrt{d}, 1/\sqrt{d}\}^{N\times d}$ where each column-sum is either $2pN/\sqrt{d}$ or $-2pN/\sqrt{d}$, the goal is to find a set $\tilde{W} \subset \mathbb{Z}_d$ such that $|\tilde{W}\Delta W| \leq w/200$, where W is the set of indices for those columns whose entries add up to $2pN/\sqrt{d}$ and $w = |W|$. This problem is again a composition of the approximate set finding problem in Section 4.2 and the Hamming-weight distinguisher problem $\mathrm{HD}_{\ell,\ell'}$ with $\ell = \frac{N}{2} - pN$ and $\ell' = \frac{N}{2} + pN$ up to a scalar $1/\sqrt{d}$. Following the proof of Theorem 19, we prove a lower bound of $\Omega(1/p)$ queries for this problem.

▶ **Theorem 27.** *Let $N \in 2\mathbb{Z}_+$, $z \in \{0,1\}^N$, and $p \in (0, 0.5)$ be an integer multiple of $1/N$. Suppose we have query access to z. Then every bounded-error quantum algorithm that computes $HD_{\frac{N}{2}-pN,\frac{N}{2}+pN}$ makes $\Omega(1/p)$ queries.*

Again we think of the input bits as ± 1 and abuse the notation $\mathrm{HD}_{\frac{N}{2}-pN,\frac{N}{2}+pN}$ for the problem with ± 1 input. Also, by the composition property of the adversary bound from Belovs and Lee [12] (Theorem 20), we have a lower bound of $\Omega(\sqrt{dw}/p)$ for $\mathrm{WSSF}_{d,w,p,N}$ from the $\Omega(\sqrt{dw})$ lower bound for $\mathrm{ASF}_{d,w}$ and the $\Omega(1/p)$ lower bound for $\mathrm{HD}_{\frac{N}{2}-pN,\frac{N}{2}+pN}$.

▶ **Corollary 28.** *Let $N \in 2\mathbb{Z}_+$ and $p \in (0, 0.5)$ be an integer multiple of $1/N$. Given a matrix $X \in \{-1/\sqrt{d}, +1/\sqrt{d}\}^{N\times d}$ such that there exists a set $W \subseteq \mathbb{Z}_d$ with size w and*

- *For every $j \in W$, $\sum_{i\in\mathbb{Z}_N} X_{ij} = 2pN/\sqrt{d}$.*
- *For every $j' \in \overline{W}$, $\sum_{i\in\mathbb{Z}_N} X_{ij'} = -2pN/\sqrt{d}$.*

Then every bounded-error quantum algorithm that computes $\tilde{W}$ such that $|W\Delta\tilde{W}| \leq w/200$, takes $\Omega(\sqrt{dw}/p)$ queries.

The final step for proving a lower bound for Ridge, using the same arguments as in Section 4.3, is to provide a worst-case to average-case reduction for the symmetric set-finding problem. We follow the same proof in Theorem 22 and immediately get the following theorem:

▶ **Theorem 29.** *Let $N \in 2\mathbb{Z}_+$, $p \in (0, 0.5)$ be an integer multiple of $1/N$, w be a natural number between 2 to $d/2$, and M be a natural number. Suppose $X \in \{-1/\sqrt{d}, +1/\sqrt{d}\}^{N\times d}$ is a valid input for $WSSF_{d,w,p,N}$, and let $W \subset \mathbb{Z}_d$ be the set of the w indices of the columns of X whose entries add up to $2pN/\sqrt{d}$. Let $R \in \mathbb{Z}_N^{M\times d}$ be a matrix whose entries are i.i.d. samples from $\mathcal{U}_N$, and define $X' \in \{-1/\sqrt{d}, 1/\sqrt{d}\}^{M\times d}$ as $X'_{ij} = X_{R_{ij}j}$. Then the vectors $(X'_i, 1)$, where X'_i is the ith row of X' and $i \in \mathbb{Z}_M$, are i.i.d. samples from $\mathcal{D}'_{p,W}$.*

By setting $M = 10^{10} \cdot \lceil \log d \rceil \cdot \lfloor 1/\varepsilon \rfloor^2 = \mathcal{O}((\log d)/\varepsilon^2)$ and letting $S' = \{(X'_i, 1)\}_{i=0}^{M-1}$ be a sample set with M i.i.d. samples from $\mathcal{D}'_{p,W}$, with probability $\geq 9/10$, an $\varepsilon/2000$-minimizer for Ridge with respect to S' is also an $\varepsilon/1000$-minimizer for Ridge with respect to distribution $\mathcal{D}'_{p,W}$ from Theorem 5. By Theorem 26 and Theorem 25, an $\varepsilon/1000$-minimizer for Ridge with respect to distribution $\mathcal{D}'_{p,W}$ gives us a set $\tilde{W} \subset \mathbb{Z}_d$ such that $|\tilde{W} \Delta W| \leq w/200$, where W is the set of indices for those columns of X whose entries add up to $2pN/\sqrt{d}$. Hence we have a reduction from the worst-case symmetric set-finding problem to Ridge. By this reduction and by plugging $w = \lfloor d/2 \rfloor$ and $p = 1/\lfloor 1/\varepsilon \rfloor$ in Corollary 28 (and N an arbitrary natural number such that $pN \in \mathbb{N}$), we obtain a lower bound of $\Omega(d/\varepsilon)$ queries for $\mathrm{WSSF}_{d,w,p,N}$, and hence for Ridge as well, which is the main result of this section.

▶ **Corollary 30.** *Let $\varepsilon \in (2/d, 1/1000)$, $w = \lfloor d/2 \rfloor$, $p = 1/\lfloor 1/\varepsilon \rfloor$, and $W \subset \mathbb{Z}_d$ with size w. Every bounded-error quantum algorithm that computes an ε-minimizer for Ridge w.r.t. $\mathcal{D}'_{p,W}$ uses $\Omega(d/\varepsilon)$ queries.*

6 Future work

We mention a few directions for future work:

- While the d-dependence of our quantum bounds for Lasso is essentially optimal, the ε-dependence is not: upper bound $\sqrt{d}/\varepsilon^2$ vs lower bound $\sqrt{d}/\varepsilon^{1.5}$. Can we shave off a $1/\sqrt{\varepsilon}$ factor from our upper bound, maybe using a version of accelerated gradient descent [36] with $O(1/\sqrt{\varepsilon})$ iterations instead of Frank-Wolfe's $O(1/\varepsilon)$ iterations? Or can we somehow improve our lower bound by embedding harder query problems into Lasso?
- Similar question for Ridge: the linear d-dependence of our quantum bounds is tight, but we should improve the ε-dependence of our upper and/or lower bounds. The most interesting outcome would be a quantum algorithm for Ridge with better ε-dependence than the optimal classical complexity of $\tilde{\Theta}(d/\varepsilon^2)$; currently we do not know of any quantum speedup for Ridge.
- Can we speed up some other methods for (smooth) convex optimization? In particular, can we find a classical iterative method where quantum algorithms can significantly reduce the number of iterations, rather than just the cost per iteration as we did here?
- There are many connections between Lasso and Support Vector Machines [32], and there are recent quantum algorithms for optimizing SVMs [38, 41, 39, 1, 40]. We would like to understand this connection better.

References

1 Jonathan Allcock and Chang-Yu Hsieh. A quantum extension of SVM-perf for training nonlinear SVMs in almost linear time. *Quantum*, 4:342, 2020. arXiv:2006.10299.

2 Joran van Apeldoorn. *A quantum view on convex optimization.* PhD thesis, Universiteit van Amsterdam, 2020.

3 Joran van Apeldoorn and András Gilyén. Quantum algorithms for zero-sum games, 2019. arXiv:1904.03180.

4 Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Convex optimization using quantum oracles. *Quantum*, 4:220, 2020. arXiv:1809.00643.

5 Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Quantum SDP-solvers: better upper and lower bounds. *Quantum*, 4:230, 2020. Earlier version in FOCS'17. arXiv:1705.01843.

6 Joran van Apeldoorn and András Gilyén. Improvements in quantum SDP-solving with applications. In *Proceedings of 46th International Colloquium on Automata, Languages, and Programming*, volume 132 of *Leibniz International Proceedings in Informatics*, pages 99:1–99:15, 2019. arXiv:1804.05058.

7 Joran van Apeldoorn, Sander Gribling, Yinan Li, Harold Nieuwboer, Michael Walter, and Ronald de Wolf. Quantum algorithms for matrix scaling and matrix balancing. In *Proceedings of 48th International Colloquium on Automata, Languages, and Programming*, volume 198 of *Leibniz International Proceedings in Informatics*, pages 110:1–17, 2021. arXiv:2011.12823.

8 Simon Apers and Ronald de Wolf. Quantum speedup for graph sparsification, cut approximation and Laplacian solving. In *Proceedings of 61st IEEE Annual Symposium on Foundations of Computer Science*, pages 637–648, 2020. arXiv:1911.07306.

9 Srinivasan Arunachalam and Reevu Maity. Quantum boosting. In *Proceedings of 37th International Conference on Machine Learning (ICML'20)*, 2020. arXiv:2002.05056.

10 Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001. Earlier version in FOCS'98. quant-ph/9802049.

11 Armando Bellante and Stefano Zanero. Quantum matching pursuit: A quantum algorithm for sparse representations. *Physical Review A*, 105:022414, 2022.

12 Aleksandrs Belovs and Troy Lee. The quantum query complexity of composition with a relation, 2020. arXiv:2004.06439.

13 Fernando Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta Svore, and Xiaodi Wu. Quantum SDP solvers: Large speed-ups, optimality, and applications to quantum learning. In *Proceedings of 46th International Colloquium on Automata, Languages, and Programming*, volume 132 of *Leibniz International Proceedings in Informatics*, pages 27:1–27:14, 2019. arXiv:1710.02581.

14 Fernando Brandão and Krysta Svore. Quantum speed-ups for solving semidefinite programs. In *Proceedings of 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 415–426, 2017. arXiv:1609.05537.

15 Peter Bühlmann and Sara van de Geer. *Statistics for High-Dimensional Data: Methods, Theory and Applications*. Springer, 2011.

16 Nicolò Cesa-Bianchi, Shai Shalev-Shwartz, and Ohad Shamir. Efficient learning with partially observed attributes. *Journal of Machine Learning Research*, 12:2857–2878, 2011. arXiv:1004.4421.

17 Shouvanik Chakrabarti, Andrew Childs, Tongyang Li, and Xiaodi Wu. Quantum algorithms and lower bounds for convex optimization. *Quantum*, 4:221, 2020. arXiv:1809.01731.

18 Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. The power of block-encoded matrix powers: improved regression techniques via faster Hamiltonian simulation. In *Proceedings of 46th International Colloquium on Automata, Languages, and Programming*, volume 132 of *Leibniz International Proceedings in Informatics*, pages 33:1–33:14, 2019. arXiv:1804.01973.

19 Shantanav Chakraborty, Aditya Morolia, and Anurudh Peduri. Quantum regularized least squares, 2022. arXiv:2206.13143.

20 Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, Shan You, and Dacheng Tao. Quantum differentially private sparse regression learning, 2020. arXiv:2007.11921.

21 Christoph Dürr and Peter Høyer. A quantum algorithm for finding the minimum, 1996. arXiv:quant-ph/9607014.

22 Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

23 Ankit Garg, Robin Kothari, Praneeth Netrapalli, and Suhail Sherif. Near-optimal lower bounds for convex optimization for all orders of smoothness. In *Proceedings of 35th Conference on Neural Information Processing Systems*, 2021.

24 Ankit Garg, Robin Kothari, Praneeth Netrapalli, and Suhail Sherif. No quantum speedup over gradient descent for non-smooth convex optimization. In *Proceedings of 12th Innovations in Theoretical Computer Science Conference*, volume 185 of *Leibniz International Proceedings in Informatics*, pages 53:1–53:20, 2021. arXiv:2010.01801.

25 András Gilyén, Seth Lloyd, and Ewin Tang. Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension, 2018. arXiv:1811.04909.

26 Sander Gribling and Harold Nieuwboer. Improved quantum lower and upper bounds for matrix scaling. In *Proceedings of 39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022)*, volume 219 of *Leibniz International Proceedings in Informatics*, pages 35:1–35:23, 2022. arXiv:2109.15282.

27 Aram Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for solving linear systems of equations. *Physical Review Letters*, 103(15):150502, 2009. arXiv:0811.3171.

28 Elad Hazan and Tomer Koren. Linear regression with limited observation. In *Proceedings of the 29th International Conference on Machine Learning*, 2012. arXiv:1206.4678 . More extensive version at arXiv:1108.4559.

29 Arthur Hoerl and Robert Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

30 Adam Izdebski and Ronald de Wolf. Improved quantum boosting, 2020. arXiv:2009.08360.

31 Martin Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 427–435, 2013.

32 Martin Jaggi. An equivalence between the Lasso and Support Vector Machines. In Johan Suykens, Marco Signoretto, and Andreas Argyriou, editors, *Regularization, Optimization, Kernels, and Support Vector Machines*. CRC Press, 2014. arXiv:1303.1152.

33 Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. In *Proceedings of 8th Innovations in Theoretical Computer Science Conference*, volume 67 of *Leibniz International Proceedings in Informatics*, pages 49:1–49:21, 2017. arXiv:1603.08675.

34 Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. Adaptive Computation and Machine Learning series. MIT Press, second edition, 2018.

35 Ashwin Nayak and Felix Wu. The quantum query complexity of approximating the median and related statistics. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 384–393. ACM, 1999. arXiv:quant-ph/9804066.

36 Yurii Nesterov. A method for solving the convex programming problem with convergence rate $\mathcal{O}(1/k^2)$. *Proceedings of the USSR Academy of Sciences*, 269:543–547, 1983.

37 Anupam Prakash. *Quantum Algorithms for Linear Algebra and Machine Learning*. PhD thesis, University of California, Berkeley, 2014.

38 Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical Review Letters*, 113(13):130503, 2014. arXiv:1307.0471.

39 Seyran Saeedi and Tom Arodz. Quantum sparse support vector machines, 2019. arXiv:1902.01879.

40 Seyran Saeedi, Aliakbar Panahi, and Tom Arodz. Quantum semi-supervised kernel learning. *Quantum Machine Intelligence*, 3:24, 2021.

41 Maria Schuld and Nathan Killoran. Quantum machine learning in feature Hilbert spaces. *Physical Review Letters*, 122(13):040504, 2019. arXiv:1803.07128.

42 Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014.

43 Robert Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society*, 58:267–288, 1996.

44 Hrishikesh Vinod. A survey of Ridge regression and related techniques for improvements over ordinary least squares. *The Review of Economics and Statistics*, 60(1):121–131, 1978.

45 Chenyi Zhang, Jiaqi Leng, and Tongyang Li. Quantum algorithms for escaping from saddle points. *Quantum*, 5:229, 2021. `arXiv:2007.10253`.

New PRGs for Unbounded-Width/Adaptive-Order Read-Once Branching Programs

Lijie Chen
Miller Institute for Basic Research in Science at University of California at Berkeley, CA, USA

Xin Lyu
University of California at Berkeley, CA, USA

Avishay Tal
University of California at Berkeley, CA, USA

Hongxun Wu
University of California at Berkeley, CA, USA

Abstract

We give the first pseudorandom generators with sub-linear seed length for the following variants of read-once branching programs (roBPs):

1. First, we show there is an explicit PRG of seed length $O(\log^2(n/\varepsilon)\log(n))$ fooling *unbounded-width unordered* permutation branching programs with a single accept state, where n is the length of the program. Previously, [Lee-Pyne-Vadhan RANDOM 2022] gave a PRG with seed length $\Omega(n)$ for this class. For the ordered case, [Hoza-Pyne-Vadhan ITCS 2021] gave a PRG with seed length $\widetilde{O}(\log n \cdot \log 1/\varepsilon)$.
2. Second, we show there is an explicit PRG fooling *unbounded-width unordered* regular branching programs with a single accept state with seed length $\widetilde{O}(\sqrt{n \cdot \log(1/\varepsilon)} + \log(1/\varepsilon))$. Previously, no non-trivial PRG (with seed length less than n) was known for this class (even in the ordered setting). For the ordered case, [Bogdanov-Hoza-Prakriya-Pyne CCC 2022] gave an HSG with seed length $\widetilde{O}(\log n \cdot \log 1/\varepsilon)$.
3. Third, we show there is an explicit PRG fooling width w *adaptive* branching programs with seed length $O(\log n \cdot \log^2(nw/\varepsilon))$. Here, the branching program can choose an input bit to read depending on its current state, while it is guaranteed that on any input $x \in \{0,1\}^n$, the branching program reads each input bit exactly once. Previously, no PRG with a non-trivial seed length is known for this class.

   We remark that there are some functions computable by constant-width adaptive branching programs but not by sub-exponential-width unordered branching programs.

In terms of techniques, we indeed show that the Forbes-Kelly PRG (with the right parameters) from [Forbes-Kelly FOCS 2018] already fools all variants of roBPs above. Our proof adds several new ideas to the original analysis of Forbes-Kelly, and we believe it further demonstrates the versatility of the Forbes-Kelly PRG.

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization

Keywords and phrases pseudorandom generators, derandomization, read-once branching programs

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.39

Category Track A: Algorithms, Complexity and Games

Funding *Lijie Chen*: Supported by a Miller Research Fellowship.
Avishay Tal: Supported by a Sloan Research Fellowship and NSF CAREER Award CCF-2145474.

1 Introduction

One central question in complexity theory is whether randomness is necessary for efficient computation. In the time setting, the question is essentially asking whether $\mathsf{P} = \mathsf{BPP}$. While it is commonly believed that $\mathsf{P} = \mathsf{BPP}$ [23, 16], it is known that establishing this would imply breakthrough lower bounds in complexity theory [13, 17], which seems to be out of reach for current techniques. Therefore, most previous works are devoted to derandomizing sub-classes of BPP. In particular, the class of randomized log-space algorithms (BPL) has attracted a lot of attention, since not only it contains many interesting problems, but also it is indeed possible to give unconditional derandomizations of BPL [22, 27].

A leading approach to derandomize BPL is to construct explicit PRGs for ordered read-once branching programs (see below for a formal definition) with short seed length.

▶ **Definition 1.** *An ordered read-once branching program (roBP) B of length n and width w computes a function $B\colon \{0,1\}^n \to \{0,1\}$. The program has $(n+1)$ layers of states $V_0 \cup V_1 \cup \cdots \cup V_n$ where V_i contains all states in the i-th layer. Being width-w means that $|V_i| \le w$ for every $i \in [n]$. On an input $x \in \{0,1\}^n$, the branching program computes as follows. It starts at a fixed start state $s \in V_0$. Then for every $i = 1, 2, \ldots, n$, it reads the next input bit x_i and updates its state according to a transition function $B_i\colon V_{i-1} \times \{0,1\} \to V_i$ by taking $v_i = B_t(v_{i-1}, x_i)$. Note that the transition function B_i can differ at each time step.*

When we use the program to compute a decision problem, we specify a set $V_{\mathsf{acc}} \subseteq [w]$ of accepting states in the final layer. Let v_n be the final state reached by the branching program on input x. If $v_n \in V_{\mathsf{acc}}$, the branching program accepts, denoted by $B(x) = 1$. Otherwise, the program rejects, denoted by $B(x) = 0$.

Next, we recall the definition of a pseudorandom generator (PRG).

▶ **Definition 2.** *Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$. An ε-PRG for $\mathcal{F}$ is a function $G\colon \{0,1\}^s \to \{0,1\}^n$ such that for every $f \in \mathcal{F}$,*

$$\left| \Pr_{x \in \{0,1\}^n}[f(x) = 1] - \Pr_{x \in \{0,1\}^s}[f(G(x)) = 1] \right| \le \varepsilon.$$

We say that G ε-fools $\mathcal{F}$ if it is an ε-PRG for $\mathcal{F}$. The input length s is the seed length of the PRG G. We say a generator is explicit*, if given as input a seed $x \in \{0,1\}^s$, the output is computable in space $O(s)$.*

In a seminal work, Nisan constructed an explicit PRG that ε-fools length-n width-w ordered roBPs with seed length $O(\log n \cdot \log(nw/\varepsilon))$. Since then, many PRGs with improved seed lengths were constructed for sub-classes of ordered roBPs (see [5, 10, 20] and the references therein), but Nisan's PRG remains the state-of-the-art even for width-4 general roBPs.

Nisan's PRG (and [15, 9]) crucially relies on the following "communication" argument: The first half of the roBP can only communicate $\log w$ bits (describing the state reached at the end of the first half) to the second half. Due to this, it is possible to reuse all but $\log w$ bits from the seed that is used to generate the first half of the pseudorandom input, when generating the second half of the pseudorandom input. Recursively applying the idea gives the $\log n \log(nw/\varepsilon)$ seed length of Nisan's PRG.

However, some researchers have the feeling that this type of argument is inherently limited to having seed length at least $\log^2 n$ [6, 26, 28][1], and different approaches are required to

[1] For example, in [26], "This paradigm seems unlikely to yield pseudorandom generators for general logspace computations that have a seed length of $O(\log^{1.99} n)$."

overcome this $\log^2 n$ barrier. The search for a different paradigm for designing PRGs has motivated the study of models stronger than normal roBPs, with the hope that studying them would inspire us to find new techniques. In particular, two interesting models, unordered roBPs and unbounded-width roBPs, were introduced recently. It turns out that designing PRGs for both models requires inherently new techniques or analysis compared to Nisan's original PRG (or the INW PRG [15]).

Unordered roBPs. Let $\mathcal{B}$ be a class of ordered roBPs. We say a function $g\colon \{0,1\}^n \to \{0,1\}$ is computable by an unordered $\mathcal{B}$ roBP, if there is a function $f\colon \{0,1\}^n \to \{0,1\}$ and a permutation π on $[n]$ such that f is computable by a roBP in $\mathcal{B}$ and $g(x_1,\ldots,x_n) = f(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)})$.

It is known that Nisan's PRG fails to fool unordered roBPs [29]. After a long line of previous works [4, 14, 26, 28, 11, 19, 7], Forbes and Kelly [8] constructed $O(\log^2 n \log(nw/\varepsilon))$-seed-length PRGs fooling length-n width-w unordered roBPs with error ε.

Unbounded-width roBPs. Another recent line of works studied roBPs with unbounded width [12, 25, 24, 3, 18]. Of course, a general roBP with unbounded width can compute any function (even with a single accept state), so we must restrict our attention to sub-classes of such roBPs. The following two sub-classes of roBPs are the most studied ones in the literature.

▶ **Definition 3.** *Let B be an ordered roBP with length n and width w. We say that B is a regular roBP, if for every $t \in [n]$ and every $v \in [w]$, there are exactly 2 pairs $(u,b) \in [w]\times\{0,1\}$ such that $B_t(u,b) = v$. We say that B is a permutation roBP, if for every $t \in [n]$ and every $b \in \{0,1\}$, $B_t(\cdot,b)$ is a permutation on $[w]$.*

In [12], an $\widetilde{O}(\log n \cdot \log 1/\varepsilon)$-seed length PRG with error ε is constructed for ordered unbounded-width permutation roBP with length-n and a single accept state. A later work [25] (building on a prior work [1]) constructed an $\widetilde{O}(\log n \cdot \sqrt{\log(n/\varepsilon)} + \log(1/\varepsilon))$-seed length weighted PRG for the same class.[2]

1.1 Our Results

In this work, we consider two even stronger models of roBPs: (1) roBPs that are *both* unordered and have unbounded width and (2) roBPs that can read input in an adaptive order (that is, the next bit to read can depend on the current state).

1.1.1 Unordered and Unbounded-width roBPs

Given the recent developments on unordered roBPs and on unbounded-width roBPs, a natural question is whether one can construct non-trivial PRGs for unordered *and* unbounded-width (permutation or regular) roBPs. A prior, it is even unclear whether such a class admits *non-explicit* PRGs with short seed length, since the usual probabilistic argument for the existence of PRGs with short seed length does not apply here [12].

Our first result is a $\mathrm{polylog}(n/\varepsilon)$-seed-length PRG for unordered unbounded-width permutation roBPs with a single accept state, significantly improving the previous $\Omega(n)$-seed length PRGs from [18].

[2] A weighted PRG for a class of functions $\mathcal{F}$ is a pair of functions $G\colon \{0,1\}^s \to \{0,1\}^n$ and $\rho\colon \{0,1\}^s \to \mathbb{R}$ such that $\mathbf{E}_{x\in\{0,1\}^s}[\rho(x)f(G(x))]$ is ε-close to $\mathbf{E}_{x\in\{0,1\}^n}[f(x)]$.

▶ **Theorem 4** (Unbounded width permutation BP). *For all integers n and $\varepsilon > 0$, there is an explicit ε-PRG $G\colon \{0,1\}^s \to \{0,1\}^n$ with seed length*

$$s = O(\log n \cdot \log^2(n/\varepsilon))$$

that fools unordered unbounded-width permutation branching programs with a single accept state.

Our second result is a $\widetilde{O}(\sqrt{n}\log(1/\varepsilon))$-seed-length PRG for unordered unbounded-width regular roBPs with a single accept state. No (even non-explicit) non-trivial PRG is known for this class even in the ordered setting; Bogdanov, Hoza, Prakriya, and Pyne [3] has constructed $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$-seed-length HSG for the ordered case.[3]

▶ **Theorem 5** (Unbounded width regular BP). *For all integers n and $\varepsilon > 0$, there is an explicit ε-PRG $G\colon \{0,1\}^s \to \{0,1\}^n$ with seed length*

$$s = O\left(\sqrt{n \log\left(\frac{n}{\varepsilon}\right)} \cdot \log n\right)$$

that fools unordered unbounded-width regular branching programs with a single accept state.

In terms of techniques, we indeed prove that the Forbes-Kelly PRG suffices for the two theorems above. Our analysis carefully modifies the original analysis from [8]. In fact, we prove that a single round of Forbes-Kelly pseudorandom restriction fools unordered unbounded-width regular roBPs (see Theorem 10). Iterating this restriction $O(\log(n/\varepsilon))$ times proves Theorem 4. Unfortunately, it is unclear whether the same iterative construction fools unordered unbounded-width regular roBPs since they are not closed under restrictions. Still, doing the pseudorandom restriction exactly once with the right parameters proves Theorem 5.

1.1.2 Adaptive roBPs

While an unordered roBP can read its input in any order, it cannot change the ordering based on the input it has read so far (*i.e.*, the order is input oblivious). We also consider an even stronger variant of roBPs, called adaptive roBPs, which are programs that can decide the next bit to read given its current state. We formally define them as follows.

▶ **Definition 6.** *An adaptive read-once branching program B of length n and width w computes a function $B\colon \{0,1\}^n \to \{0,1\}$. The program has states $V_0 \cup V_1 \cup \cdots \cup V_n$ where V_i consists of the w states in the i-th layer. On an input $x \in \{0,1\}^n$, the branching program B computes as follows. It starts at a fixed start state $v_0 \in [w]$. Then for every $t = 1, 2, \ldots, n$, it reads the bit $x_{\mathsf{pos}(t-1,v_{t-1})}$ and updates its state according to a transition function $B_t\colon V_{t-1} \times \{0,1\} \to V_t$ by taking $v_t = B_t(v_{t-1}, x_{\mathsf{pos}(t-1,v_{t-1})})$. Here, $\mathsf{pos}\colon V_0 \cup \cdots \cup V_{n-1} \to [n]$ is a function specifying the index of the next bit to read given the current state v_{t-1}. We require that on every input $x \in \{0,1\}^n$, B reads each bit in x exactly once.*

We remark that adaptive roBPs are strictly stronger than unordered roBPs as shown by an example function $f : \{0,1\} \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ as

$$f(b,x,y) = \mathbf{1}[b=0] \cdot \mathbf{1}[x=y] + \mathbf{1}[b=1] \cdot \mathbf{1}[x=y^R].$$

[3] A hitting set generator (HSG) $H\colon \{0,1\}^s \to \{0,1\}$ for a class of functions $\mathcal{F}$ satisfies the following: for every $f \in \mathcal{F}$ such that $\Pr_{x\in\{0,1\}^n}[f(x)=1] > \varepsilon$, there exists $z \in \{0,1\}^s$ such that $f(H(z)) = 1$. Note that a PRG is automatically an HSG, while the converse may not hold.

Here, y^R denotes the reversed string of y. Observe that there is a constant-width adaptive roBP for f. The program first reads b. If $b = 0$, the program reads and compares x and y bit by bit. Otherwise, the program compares x and y^R bit by bit. Moreover, it is easy to see (via a communication complexity argument) that every unordered roBP for f requires exponential width.

Our third result gives $O(\text{polylog}(nw/\varepsilon))$-seed-length PRG for adaptive roBPs. To the best of our knowledge, no explicit PRGs with seed length less than n was known prior to our work.

▶ **Theorem 7.** *For every $n, w \geq 1$ and $\varepsilon > 0$, there is an explicit ε-PRG $G : \{0,1\}^s \to \{0,1\}^n$ fooling width-w adaptive roBPs with seed length $s = O(\log n \cdot \log^2(nw/\varepsilon))$.*

We prove Theorem 7 by adapting the argument in [8]. The key observation allowing us to do so is the following. Suppose B satisfies the read-once promise. Then, for every vertex $v \in V_i$, if we denote by Pre_v (resp. Post_v) the set of possible variables read in any path from the starting state to v (resp. v to the accepting state). It must be the case that Pre_v and Post_v are disjoint for every v. By a delicate argument (Claim 15), we show that this disjointness property is sufficient for applying the key technique of Forbes-Kelley proof: decomposing the branching program by high/low-degree Fourier terms.

Moreover, when the width w of the adaptive roBP is small, we can show that the branching program has bounded Fourier growth (following [7]). In particular, we show that the L-th level Fourier mass of a width-w adaptive roBP is bounded by $O(\log(nw))^{2Lw}$. As shown in [8], for programs with bounded Fourier growth, we can further improve the seed length by a $\log(n)$ factor. Formally, we show

▶ **Theorem 8.** *For every $n, w \geq 1$ and $\varepsilon > 0$, there is an explicit ε-PRG $G : \{0,1\}^s \to \{0,1\}^n$ fooling width-w adaptive roBPs with seed length $s = \widetilde{O}(w \log^2(n/\varepsilon))$.*

Theorem 8 is a direct corollary of the new Fourier growth bound. We briefly comment on how we get the Fourier growth of $O(\log n)^{2Lw}$ for adaptive roBP. Roughly speaking, given a width-w, length-n adaptive roBP B, we construct a related witdh-$2w$, length-n^2 *oblivious* roBP B', such that the Fourier spectrum of B is "dominated" by that of B'. The idea is simple: we duplicate each input of B for n times and get n^2 bits. Now, it is easy to construct a width-$2w$ oblivious roBP running on the n^2 bits to simulate B. (Essentially, the n^2 bits allow us to make n passes over the input, we can use each pass to implement one step of transition of B.)

Although B' has n^2 input bits, we can exploit the promise that B is read-once, and prove the following nice property: For any input $z \in \{0,1\}^{n^2}$, $B'(z)$ depends only on n bits of z (that is to say, there is a subset of n bits from z, such that flipping all other bits of z cannot change the output). This allows us to connect the Fourier weights of B' and B. The details can be found in Appendix A.

2 Preliminaries

For a Boolean predicate P, we use $\mathbf{1}_{\{P\}}$ to denote the indicator of P, which takes value 1 if P holds, value 0 otherwise. We often use U_n to denote the uniform distribution over $\{0,1\}^n$ (when n is clear from the context, we will just write U for simplicity), and $U(X)$ to denote the uniform distribution over a set X. For two strings $\alpha, \beta \in \{0,1\}^n$, we use $\alpha \wedge \beta$ and $\alpha + \beta$ to denote their bit-wise AND and bit-wise XOR, respectively. Similarly, for two distributions D_1, D_2, we use $D_1 \wedge D_2$ (resp. $D_1 + D_2$) to denote the distributions obtained by drawing $\alpha \sim D_1$ and $\beta \sim D_2$ and outputting $\alpha \wedge \beta$ (resp. $\alpha + \beta$).

We always work with the $\{-1,1\}^n$ basis for Boolean function analysis. For a function $f\colon \{-1,1\}^n \to \mathbb{R}$, recall that its Fourier characters indexed by $\alpha \subseteq [n]$, is defined by

$$\widehat{f}(\alpha) = \mathop{\mathbb{E}}_{x\in\{-1,1\}^n}\left[f(x)\cdot\prod_{i\in\alpha} x_i\right].$$

We often use greek letters (such as α, β, γ) to index Fourier characters.

We will need k-wise independent and γ-almost k-wise independent distributions throughout the paper, which look locally uniform and thus fool functions that only depend on a few bits.

▶ **Definition 9.** *Let D be a distribution over $\{0,1\}^n$. We say D is k-wise independent if, for every $f\colon \{0,1\}^n \to [-1,1]$ that depends on at most k bits, we have*

$$\mathop{\mathbb{E}}_{D} f(D) = \mathop{\mathbb{E}}_{U} f(U).$$

If D merely satisfies

$$\left|\mathop{\mathbb{E}}_{D} f(D) - \mathop{\mathbb{E}}_{U} f(U)\right| \le \gamma$$

for every such f, we say that D is γ-almost k-wise independent.

It is possible to sample from a k-wise independent distribution using $O(k\cdot\log n)$ random bits ([30]) and from a γ-almost k-wise independent distribution using $O(k+\log\log n+\log 1/\gamma)$ random bits ([21, 2]).

3 PRGs for Unbounded-width Branching Programs

In this section, we will prove the following theorem, which shows that one round of pseudorandom restriction fools regular branching programs with unbounded width and a single accept state.

▶ **Theorem 10.** *Let B be an unbounded-width regular branching program of length n with starting state $s \in V_0$ and a single accept state $t \in V_n$. Let D, U denote a $2k$-wise independent distribution and a uniform distribution over $\{0,1\}^n$, respectively. Let $T^{(a)}$ denote a $2k$-wise independent distribution over $[a]^n$, and let distribution T be defined as $T_i = \mathbf{1}_{\{T_i^{(a)}=1\}}$ for all $i \in [n]$. Then*

$$|\mathbf{E}[B(U)] - \mathbf{E}[B(D + T \wedge U)]| \le n\cdot(1-1/a)^{k/2}.$$

Since the class of permutation BPs is closed under restrictions, we can iteratively apply Theorem 10 to it with $k = \log(n)$ and $a = 2$. The immediate consequence is that we get a PRG for unordered unbounded-width permutation branching programs.

▶ **Corollary 11** (Restating Theorem 4)**.** *For all integers n and $\varepsilon > 0$, there is an explicit ε-PRG $G\colon \{0,1\}^s \to \{0,1\}^n$ with seed length*

$$s = O(\log n\cdot\log^2(n/\varepsilon))$$

that fools unordered unbounded-width permutation branching programs with a single accept state.

However, when it comes to regular branching programs, this class is no longer closed under restrictions. Hence we can only apply Theorem 10 once and set $k = \widetilde{O}(\sqrt{n})$ and $a = \widetilde{O}(\sqrt{n})$. It remains an interesting open problem to apply iterative restriction for unbounded-width regular branching programs.

▶ **Corollary 12** (Restating Theorem 5). *For all integers n and $\varepsilon > 0$, there is an explicit ε-PRG $G\colon \{0,1\}^s \to \{0,1\}^n$ with seed length*

$$s = O\left(\left(\sqrt{n \log\left(\frac{n}{\varepsilon}\right)} + \log\left(\frac{1}{\varepsilon}\right)\right) \cdot \log n\right)$$

that fools unordered unbounded-width regular branching programs with a single accept state.

We will prove Theorem 10 in Subsection 3.1 and Subsection 3.2. In Subsection 3.3, we prove Corollary 11 and Corollary 12.

3.1 Fourier Decomposition of Regular BPs

Recall that V_i is the set of nodes in the i-th level of our branching program. $s \in V_0$ is the starting point and $t \in V_n$ is the unique accepting state. $x \in \{0,1\}^n$ is the input to our branching program B. In order to work with $\{-1,1\}$ basis, we let $y_i = (-1)^{x_i}$ for all $i \in [n]$.

For any two nodes $a \in V_i$ and $b \in V_j$. We define the indicator $P_{a,b} : \{-1,+1\}^n \to \{0,1\}$,

$$P_{a,b}(y) = \begin{cases} 1 & \text{Starting from } a, \text{ we reach at node } b \text{ on inputs } x_{i+1}, \ldots, x_j; \\ 0 & \text{Otherwise.} \end{cases}$$

Its Fourier expansion is as follows:

$$P_{a,b}(y) = \sum_{\alpha \subseteq \{i+1, i+2, \ldots, j\}} \widehat{P}_{a,b}(\alpha) \cdot \chi_\alpha(y)$$

where the Fourier characters are defined as

$$\chi_\alpha(y) = \prod_{i \in \alpha} y_i.$$

This naturally extends $P_{a,b}$ to $\mathbb{R}^n \to \mathbb{R}$.

Furthermore, we define

$$\bar{P}^{[k]}_{a,b}(y) = \sum_{\substack{\alpha \subseteq \{i+1, i+2, \ldots, j\} \\ |\alpha| = k,\ j \in \alpha}} \widehat{P}_{a,b}(\alpha) \cdot \chi_\alpha(y)$$

which is the sum all the degree k terms that contain y_j.

We also define

$$P^{(k)}_{a,b}(y) = \sum_{\substack{\alpha \subseteq \{i+1, i+2, \ldots, j\} \\ |\alpha| = k,\ i+1 \in \alpha}} \widehat{P}_{a,b}(\alpha) \cdot \chi_\alpha(y)$$

which is the sum all the degree k terms that contain y_{i+1}.

▶ **Fact 13.** *Let D, T, U be the distributions defined in Theorem 10, and let G be a distribution defined as*

$$G_i = \begin{cases} (-1)^{D_i} & T_i = 0, \\ 0 & T_i = 1. \end{cases} \quad \forall i \in [n].$$

Then, we have $\mathbf{E}[B(U)] = \widehat{P}_{s,t}(\varnothing)$ and $\mathbf{E}[B(D + T \wedge U)] = \mathbf{E}_{y \sim G}[P_{s,t}(y)]$.

Proof. Notice that when $y_i = (-1)^{x_i}$ for all i, $B(x) = P_{s,t}(y)$. The first fact holds because for all $\alpha \neq \varnothing$, we have $\mathbf{E}_{y\sim U(\{\pm 1\}^n)}[\chi_\alpha(y)] = 0$. Hence $\mathbf{E}[B(U)] = \mathbf{E}_{y\sim U(\{\pm 1\}^n)}[P_{s,t}(y)] = \widehat{P}_{s,t}(\varnothing)$.

For the second fact, conditioned on an instantiation of T, we define an intermediate distribution G' as

$$G'_i = \begin{cases} (-1)^{D_i} & T_i = 0, \\ (-1)^{U_i} & T_i = 1. \end{cases}.$$

we know that $\mathbf{E}[B(D + T \wedge U)] = \mathbf{E}_{y\sim G'}[P_{s,t}(y)]$.

- When $T_i = 1$, we have $\mathbf{E}_{y\sim G'}[y_i \mid T_i = 1] = \mathbf{E}_{y\sim G}[y_i \mid T_i = 1] = 0$ since y_i is sampled uniformly and independently from $\{\pm 1\}$.
- When $T_i = 0$, we always have $G_i = G'_i = (-1)^{D_i}$.

Hence for all α, we know that

$$\mathbf{E}_{y\sim G'}[\chi_\alpha(y)] = \mathbf{E}_{y\sim G'}\left[\prod_{i\in\alpha} y_i\right] = \mathbf{E}_T\left[\prod_{\substack{i\in\alpha \\ T_i=1}} \mathbf{E}_{y\sim G'}\left[y_i \mid T_i = 1\right] \cdot \mathbf{E}_{y\sim G'}\left[\prod_{\substack{i\in\alpha \\ T_i=0}} y_i \,\middle|\, T\right]\right]$$

$$= \mathbf{E}_T\left[\prod_{\substack{i\in\alpha \\ T_i=1}} \mathbf{E}_{y\sim G}\left[y_i \mid T_i = 1\right] \cdot \mathbf{E}_{y\sim G}\left[\prod_{\substack{i\in\alpha \\ T_i=0}} y_i \,\middle|\, T\right]\right] = \mathbf{E}_{y\sim G}[\chi_\alpha(y)].$$

As a result, $\mathbf{E}_{y\sim G'}[P_{s,t}(y)] = \mathbf{E}_{y\sim G}[P_{s,t}(y)]$. This finishes the proof. ◀

3.2 Bounding the Error

In the error analysis, we follow the approach of Forbes and Kelley [8]. By Fact 13, the result we wish to prove is equivalent to

$$\left|\mathbf{E}_{y\sim G}[P_{s,t}(y)] - \widehat{P}_{s,t}(\varnothing)\right| \leq n \cdot (1 - 1/a)^{k/2}.$$

In the analysis of [8], they considered the decomposition,

$$L_k(y) = \sum_{\substack{\alpha\subseteq\{1,2,\ldots,n\} \\ 0<|\alpha|<k}} \widehat{P}_{s,t}(\alpha) \cdot \chi_\alpha(y),$$

$$P_{s,t}(y) - \widehat{P}_{s,t}(\varnothing) = L_k(y) + \sum_{i=1}^{n} \sum_{m\in V_i} \bar{P}^{[k]}_{s,m}(y) \cdot P_{m,t}(y).$$

Here $L_k(y)$ are the low-degree terms, and $\bar{P}^{[k]}_{s,m}(y) \cdot P_{m,t}(y)$ are the terms that reaches degree k exactly at node $m \in V_i$. The intuition is that the $2k$-wise independent distribution D fools $L_k(y)$ while the high-degree terms are fooled by $T \wedge U$.

However, in order to work for unbounded-width regular branching programs, we have to consider a different decomposition. Let $L_k(y)$ be the same as before. We have

$$P_{s,t}(y) - \widehat{P}_{s,t}(\varnothing) = L_k(y) + \sum_{i=1}^{n} \sum_{m\in V_i} P_{s,m}(y) \cdot P^{(k)}_{m,t}(y).$$

Observe that we are using $P^{(k)}_{m,t}(y)$ instead of $\bar{P}^{[k]}_{m,t}(y)$. The benefit of this decomposition is that now for all y, we have

$$\sum_{m\in V_i} P_{s,m}(y)^2 \leq 1,$$

since from s only one state m can be reached under input y. In contrast, in the original decomposition, $\sum_{m\in V_i} P_{m,t}(y)^2$ could be very large. This difference will be essential in our analysis.

Now we are ready to prove Theorem 10.

Proof of Theorem 10. By our decomposition, we know

$$\left|\mathbf{E}_{y\sim G}[P_{s,t}(y)] - \widehat{P}_{s,t}(\varnothing)\right| \le |\mathbf{E}_{y\sim G}[L_k(y)]| + \sum_{i=1}^{n}\sum_{m\in V_i}\left|\mathbf{E}_{y\sim G}\left[P_{s,m}(y)\cdot P^{(k)}_{m,t}(y)\right]\right|. \tag{1}$$

Since G is k-wise independent, we know $\mathbf{E}_{y\sim G}\left[L_k(y)\right] = 0$. Now we bound the second term. We will need the following fact: For any two (not necessarily independent) sequences of random variables $\{f_m\}_{m\in V_i}, \{g_m\}_{m\in V_i}$, we have

$$\mathbf{E}\left[\sum_{m\in V_i} f_m g_m\right] \le \mathbf{E}\left[\sum_{m\in V_i} f_m^2\right]^{1/2} \mathbf{E}\left[\sum_{m\in V_i} g_m^2\right]^{1/2}.$$

This is the Cauchy-Schwarz Inequality for random variables.

Let $f_m = |P_{s,m}(y)|$ and $g_m = |P^{(k)}_{m,t}(y)|$. We have

$$\sum_{m\in V_i}\left|\mathbf{E}_{y\sim G}\left[P_{s,m}(y)P^{(k)}_{m,t}(y)\right]\right| \le \mathbf{E}_{y\sim G}\left[\sum_{m\in V_i}\left(P_{s,m}(y)\right)^2\right]^{1/2} \mathbf{E}_{y\sim G}\left[\sum_{m\in V_i} P^{(k)}_{m,t}(y)^2\right]^{1/2}$$

We bound these two separately.

We first bound $\mathbf{E}_{y\sim G}\left[\sum_{m\in V_i} P^{(k)}_{m,t}(y)^2\right]$. Suppose

$$P^{(k)}_{m,t}(y) = \sum_{\alpha} c_\alpha \chi_\alpha(y).$$

By $2k$-wise independence of G, we know for all $\alpha \neq \beta$ and $|\alpha| + |\beta| \le 2k$, the cross term

$$\mathbf{E}_{y\sim G}[\chi_\alpha(y)\chi_\beta(y)] = 0.$$

For the square terms, notice that for all $T_i = 1$, we have $y_i = 0$. When $T_i = 0$, $y_i = (-1)^{D_i}$. $T_i = 1$ happens with probability $1/a$. D, T are $2k$-wise independent. Hence when $|\alpha| = k$, we have[4]

$$\begin{aligned}\mathbf{E}_{y\sim G}[\chi_\alpha(y)^2] &= \mathbf{E}_{y\sim D}[\chi_\alpha(y)^2\cdot \mathbf{1}_{\{\forall i\in\alpha, T_i=0\}}]\\ &= \left(1-\frac{1}{a}\right)^k \cdot \mathbf{E}_{y\sim U}[\chi_\alpha(y)^2]\\ &= \left(1-\frac{1}{a}\right)^k.\end{aligned}$$

Hence,

$$\begin{aligned}\mathbf{E}_{y\sim G}\left[\left(P^{(k)}_{m,t}(y)\right)^2\right] &= \mathbf{E}_{y\sim G}\left[\sum_{|\alpha|=k} c_\alpha^2\chi_\alpha(y)^2\right]\\ &= \left(1-\frac{1}{a}\right)^k \sum_{|\alpha|=k} c_\alpha^2 \le \left(1-\frac{1}{a}\right)^k \mathbf{E}_{y\sim U}\left[\left(P_{m,t}(y)\right)^2\right].\end{aligned}$$

[4] For brevity, we use $y\sim U$ to mean that $y\sim U(\{-1,1\}^n)$.

The last step follows from Parseval identity. Summing over all $m \in V_i$ for a fixed i, we get

$$\sum_{m\in V_i} \mathbf{E}_{y\sim G}\left[\left(P^{(k)}_{m,t}(y)\right)^2\right] \le \left(1-\frac{1}{a}\right)^k \sum_{m\in V_i} \mathbf{E}_{y\sim U}\left[\left(P_{m,t}(y)\right)^2\right]$$
$$= \left(1-\frac{1}{a}\right)^k \sum_{m\in V_i} \mathbf{E}_{y\sim U}\left[P_{m,t}(y)\right] = \left(1-\frac{1}{a}\right)^k.$$

The last step is because now $y \sim U$, and the branching program is regular so that $\sum_{m\in V_i} \mathbf{E}_{y\sim U}[P_{m,t}(y)] = 1$.[5]

- On the other hand, as we mentioned, for any y, s can reach a single vertex in V_i, hence

$$\sum_{m\in V_i} \mathbf{E}_{y\sim G}\left[P_{s,m}(y)^2\right] \le 1.$$

Putting these two together, we get

$$|\mathbf{E}_{y\sim G}[P_{s,t}(y)] - \widehat{P}_{s,t}(\varnothing)| \le \sum_{i=1}^{n} \mathbf{E}_{y\sim G}\left[\sum_{m\in V_i}\left(P_{s,m}(y)\right)^2\right]^{1/2} \cdot \mathbf{E}_{y\sim G}\left[\sum_{m\in V_i} P^{(k)}_{m,t}(y)^2\right]^{1/2}$$
$$\le \left(1-\frac{1}{a}\right)^{k/2} n. \qquad \blacktriangleleft$$

3.3 Applications

Finally, we prove Corollary 11 and Corollary 12 in the rest of this section.

Proof of Corollary 11. Let $\{D^{(i)}\}_{i\in[\ell]}, \{T^{(i)}\}_{i\in[\ell]}$ be ℓ independent copies of $2k$-wise independent dsitributions defined in Theorem 10 with $k = \log\left(\frac{n}{\varepsilon}\right) + \log\log(\frac{n}{\varepsilon}) + 1$ and $a = 2$.

We construct pseudorandom distributions $G^{(0)}, G^{(1)}, \ldots, G^{(\ell)}$ with $\ell = \Theta(\log(n/\varepsilon))$. We let G_0 be the set of all one strings in $\{0,1\}^n$ and set

$$G^{(i+1)} = D^{(i)} + T^{(i)} \wedge G^{(i)}.$$

Let branching program $B^{(i)}$ be defined as $B^{(\ell)} = B$ and

$$B^{(i)}(x) = B^{(i+1)}(D^{(i)} + T^{(i)} \wedge x).$$

Since any restriction of a permutation branching program is still a permutation branching program. For any realization of $D^{(i)}$ and $T^{(i)}$, Theorem 10 says that,

$$\left|\mathbf{E}[B^{(i+1)}(U)] - \mathbf{E}[B^{(i)}(U)]\right| = \left|\mathbf{E}_{x\sim D^{(i)}+T^{(i)}\wedge U}[B^{(i)}(x)] - \mathbf{E}[B^{(i)}(U)]\right|$$
$$\le \left(1-\frac{1}{2}\right)^{\log\left(\frac{n}{\varepsilon}\right)+\log\log\left(\frac{n}{\varepsilon}\right)+1} n \le \frac{\varepsilon/2}{\log(\frac{n}{\varepsilon})}.$$

From a standard Chernoff bound, with probability at least $1-\varepsilon/2$, $T^{(1)} \wedge T^{(2)} \wedge \cdots \wedge T^{(\ell)} = 0000\ldots0$. This implies that $\left|\mathbf{E}[B^{(0)}(U)] - \mathbf{E}_{x\sim G^{(0)}}[B^{(0)}(x)]\right| \le \varepsilon/2$ since $B^{(0)}$ does not depend on its input when $T^{(1)} \wedge T^{(2)} \wedge \cdots \wedge T^{(\ell)} = 0000\ldots0$.

[5] This is the only place we use the regularity of the program.

On the other hand, by definition, we know $\mathbf{E}_{B^{(0)},x\sim G^{(0)}}[B^{(0)}(x)] = \mathbf{E}_{x\sim G^{(\ell)}}[B(x)]$. Hence a hybrid argument proves that

$$\begin{aligned}|\mathbf{E}_{x\sim G^{(\ell)}}[B(x)] - \mathbf{E}[B(U)]| &= \left|\mathbf{E}_{x\sim G^{(0)}}[B^{(0)}(x)] - \mathbf{E}[B^{(0)}(U)]\right| + \left|\mathbf{E}[B^{(0)}(U)] - \mathbf{E}[B^{(\ell)}(U)]\right| \\ &\le \varepsilon/2 + \sum_{i=1}^{\ell}\left|\mathbf{E}[B^{(i-1)}(U)] - \mathbf{E}[B^{(i)}(U)]\right| \\ &\le \varepsilon.\end{aligned}$$

◀

Proof of Corollary 12. For regular branching programs, let D and T be $2k$-wise independent distributions defined in Theorem 10 with $k = 2\sqrt{n\log\left(\frac{n}{\varepsilon}\right)} + \log\left(\frac{1}{\varepsilon}\right) + 2$ and $a = \sqrt{\frac{n}{\log\left(\frac{n}{\varepsilon}\right)}}$. We let D' be another independent copy of D.

We construct pseudorandom distribution $G = D + T \wedge D'$. From Theorem 10, we know that

$$|\mathbf{E}_{x\sim D+T\wedge U}[B(x)] - \mathbf{E}[B(U)]| \le n\cdot\left(1-\frac{1}{a}\right)^k \le \varepsilon/2.$$

Let $N = |\{i \mid T_i = 1\}|$. Since T is $2k$-wise independent,

$$\begin{aligned}\mathbf{E}[N^k] &\le \sum_{i_1,i_2,\ldots,i_k\in[n]} \Pr[T_{i_1} = T_{i_2} = \cdots = T_{i_k} = 1] \\ &= n^k \cdot \Pr_{i_1,i_2,\ldots,i_k\in[n]}[T_{i_1} = T_{i_2} = \cdots = T_{i_k} = 1] \\ &= n^k \cdot \prod_{j=1}^{k} \Pr[T_{i_j} = 1 \mid T_{i_1} = T_{i_2} = \cdots = T_{i_{j-1}} = 1] \\ &\le n^k \cdot \prod_{j=1}^{k} (\Pr[i_j \in \{i_1, i_2, \ldots, i_{j-1}\}] + \Pr[T_{i_j} = 1 \mid i_j \notin \{i_1, i_2, \ldots, i_{j-1}\}]) \\ &\le n^k \cdot \left(\frac{k}{n} + \frac{1}{a}\right)^k\end{aligned}$$

From Markov inequality, we get that $\Pr[N \ge 2k] \le 2\cdot(1/2)^k + 2\cdot(n/(2ak))^k$. By the $2k$ wise independence of D',

$$|\mathbf{E}_{x\sim G}[B(x)] - \mathbf{E}_{x\sim D+T\wedge U}[B(x)]| \le 2\cdot(1/2)^k + 2\cdot(n/(2ak))^k \le \varepsilon/2.$$

The seed length is $3k(\log n + \log a) = O\left(\left(\sqrt{n\log\left(\frac{n}{\varepsilon}\right)} + \log\left(\frac{1}{\varepsilon}\right)\right)\cdot\log n\right)$. ◀

▶ Remark 14. We believe the seed length in Corollary 11 can be improved to $O(\log^2 n\cdot\log(n/\varepsilon))$ following the sharper analysis in Section 7.1 of [8]. However, for the simplicity of presentation, we choose to only present it for the seed length of $O(\log n\cdot\log^2(n/\varepsilon))$.

4 PRGs for Adaptive Branching Programs

In this section, we prove our results for adaptive roBPs.

4.1 Decomposition of roBPs

As before, We use $B : \{0,1\}^n \to \{0,1\}$ to denote the adaptive branching program we are analyzing and use $P : \{\pm1\}^n \to \{0,1\}$ to denote the function computed by BP over $\{\pm1\}$ basis. For every input $x \in \{0,1\}^n$, define $y \in \{\pm1\}^n$ as $y_i = (-1)^{x_i}$ for every $i \in [n]$, and

define $P(y) = B(x)$. For any state v in the program, we denote by pos_v the index of the variable queried on state v. We have two outgoing edges from v, one marked with $x_{\mathsf{pos}_v} = 0$ and another with $x_{\mathsf{pos}_v} = 1$.

For any state v in the program, we denote by Pre_v the set of variables read in any path from the starting state to v, and by Post_v the set of variables read in any path from v to the accepting state. We observe that Pre_v and Post_v are disjoint as otherwise there exists a path from the starting state to the accepting state (and passes through v) and reads the same variable twice.

Formally, suppose there exists a vertex v and an index $i \in \mathsf{Pre}_v \cap \mathsf{Post}_v$. We choose a computation path π from starting vertex v_0 to v that queries the set $S \subseteq [n]$ of variables, and a path π' from v to the final layer that queries the set $T \subseteq [n]$, where $i \in S$.

We can construct an input $x \in \{0,1\}^n$ that guides the program to follow the computational path of $\pi \circ \pi'$. Each time the program reads a variable x_j, if x_j has been queried before, this clearly violates the read-once requirement. Otherwise, we can set x_j to make the program follow the path of $\pi \circ \pi'$. However, since know x_i is queried at least twice along the path, there must be some point, where the "read-once" requirement is violated.

Define $P : \{-1,1\}^n \to \{0,1\}$ as the function computed by the program. For a state v in the branching program, we denote by $P_{\to v}$ the event that the path from the starting state passes through v. Note that $P_{\to v}$ can be described as a branching program on the variables Pre_v. We denote by $P_{v\to}$ the sub-program of P starting at v. Note that $P_{v\to}$ can be described as a branching program on the variables Post_v.

4.1.1 Fourier Decomposition for Adaptive BP

Recall that the Fourier representation of any function $f : \{\pm 1\}^n \to \mathbb{R}$ is $\sum_{\alpha\subseteq[n]} \widehat{f}(\alpha)\chi_\alpha(y)$ where $\chi_\alpha(y) = \prod_{i\in\alpha} y_i$ and $\widehat{f}(\alpha) = \mathbf{E}_{y\sim\{\pm1\}^n}[f(y)\cdot\chi_\alpha(y)]$. Furthermore, we have that $\mathbf{E}_{y\sim\{\pm1\}^n}[f(y)^2] = \sum_\alpha \widehat{f}(\alpha)^2$ and $\mathbf{E}_{y\sim\{\pm1\}^n}[f(y)] = \widehat{f}(\emptyset)$.

Let $k \in \mathbb{N}$. Let α be a set of size $\ell > k$. We express $\widehat{P}(\alpha)$ as a sum of products of Fourier coefficients, where the Fourier coefficients come from sub-programs of B. In particular, we have the following claim.

▷ Claim 15. We have

$$\widehat{P}(\alpha) = \sum_{v:|\mathsf{Pre}_v\cap\alpha|=k,\mathsf{pos}_v\in\alpha} \widehat{P_{\to v}}(\alpha\cap\mathsf{Pre}_v)\cdot\widehat{P_{v\to}}(\alpha\cap\mathsf{Post}_v).$$

Proof. By definition

$$\widehat{P}(\alpha) = \mathbf{E}_{y\sim\{\pm1\}^n}[P(y)\cdot\chi_\alpha(y)] = \mathbf{E}_{y\sim\{\pm1\}^n}[P(y)\cdot\mathbf{1}_{\{B \text{ on } y \text{ reads all the variables in } \alpha\}}\cdot\chi_\alpha(y)]$$

where the second equality holds due to the following reason. For any $\beta \subseteq \alpha$ let X_β be the set of strings on which the program reads β and doesn't read $\alpha \setminus \beta$. We note that if $x \in X_\beta$ for some set β which is a strict subset of α, then also $x' := x \oplus e_i$ for $i \in \alpha \setminus \beta$ is in X_β, since the path for both x and x' will be the same (as the path doesn't query x_i). We see that the inputs in X_β can be partitioned to pairs, and each pair contributed 0 to $\mathbf{E}_{y\sim\{\pm1\}^n}[\chi_\alpha(y)]$.

For any x for which $B(x)$ reads all the variables in α, there is a unique state v along the path such that B reads exactly k variables in α before v, and B reads the $k+1$ variable from α immediately on the edge that goes out from v.

Thus, we can partition these paths according to the state v. We observe that v is the state immediately before reading the $k+1$ variable in α if $|\mathsf{Pre}_v \cap \alpha| = k$ and if $\mathsf{pos}_v \in \alpha$. This gives

$$\begin{aligned}\widehat{P}(\alpha) &= \mathbf{E}_{y\sim\{\pm1\}^n}\left[\sum_{v:|\mathsf{Pre}_v\cap\alpha|=k,\mathsf{pos}_v\in\alpha} P_{\to v}(y)P_{v\to}(y)\cdot\chi_\alpha(y)\cdot\mathbf{1}_{\{B \text{ reads all the variables in } \alpha\}}\right]\\ &= \sum_{v:|\mathsf{Pre}_v\cap\alpha|=k,\mathsf{pos}_v\in\alpha} \mathbf{E}_{y\sim\{\pm1\}^n}\left[P_{\to v}(y)P_{v\to}(y)\cdot\chi_\alpha(y)\right]\cdot\mathbf{1}_{\{B \text{ reads all the variables in } \alpha\}}\big]\\ &= \sum_{v:|\mathsf{Pre}_v\cap\alpha|=k,\mathsf{pos}_v\in\alpha} \mathbf{E}_{y\sim\{\pm1\}^n}[P_{\to v}(y)P_{v\to}(y)\cdot\chi_\alpha(y)]\end{aligned}$$

where the last equality follows from the same argument as before by observing that $P_{\to v}(y)P_{v\to}(y)$ is equivalent to the indicator of a program B' that checks that we passed through v and reached the accept state of B.

Now, for every state $v : |\mathsf{Pre}_v \cap \alpha| = k, \mathsf{pos}_v \in \alpha$, we get

$$\begin{aligned}&\mathbf{E}_x[P_{\to v}(y)P_{v\to}(y)\cdot\chi_\alpha(y)]\\ &= \mathbf{E}_{y\in\{0,1\}^{\mathsf{Pre}_v}}[P_{\to v}(y)\chi_{\alpha\cap\mathsf{Pre}_v}(y)]\cdot\mathbf{E}_{y\in\{0,1\}^{\mathsf{Post}_v}}[P_{v\to}(y)\chi_{\alpha\cap\mathsf{Post}_v}(y)]\\ &= \widehat{P_{\to v}}(\alpha\cap\mathsf{Pre}_v)\cdot\widehat{P_{v\to}}(\alpha\cap\mathsf{Post}_v),\end{aligned}$$

which completes the proof. ◁

By summing over all sets of size larger than k we get

$$\begin{aligned}&\sum_{\alpha:|\alpha|>k}\widehat{P}(\alpha)\chi_\alpha(y)\\ &= \sum_{\alpha,v:|\mathsf{Pre}_v\cap\alpha|=k,\mathsf{pos}_v\in\alpha}\widehat{P_{\to v}}(\alpha\cap\mathsf{Pre}_v)\chi_{\alpha\cap\mathsf{Pre}_v}(y)\cdot\widehat{P_{v\to}}(\alpha\cap\mathsf{Post}_v)\chi_{\alpha\cap\mathsf{Post}_v}(y)\\ &= \sum_v\left(\sum_{\alpha':\alpha'\subseteq\mathsf{Pre}_v,|\alpha'|=k}\widehat{P_{\to v}}(\alpha')\chi_{\alpha'}(y)\right)\cdot\left(\sum_{\alpha'':\alpha''\subseteq\mathsf{Post}_v,\mathsf{pos}_v\in\alpha''}\widehat{P_{\to v}}(\alpha'')\chi_{\alpha''}(y)\right).\end{aligned}$$

For each v we denote

$$H_v(y) := \sum_{\alpha':\alpha'\subseteq\mathsf{Pre}_v,|\alpha'|=k}\widehat{P_{\to v}}(\alpha')\chi_{\alpha'}(y)$$

and

$$G_v(y) := \sum_{\alpha'':\alpha''\subseteq\mathsf{Post}_v,\mathsf{pos}_v\in\alpha''}\widehat{P_{\to v}}(\alpha'')\chi_{\alpha''}(y).$$

We observe that $G_v(y)$ is the pos_v-Laplacian[6] of $P_{v\to}$. As such, $G_v(y)$ is a bounded function, i.e., $|G_v(y)| \le 1$ for all $y \in \{\pm1\}^n$.

▶ **Lemma 16.** *For any read-once adaptive branching program B, let P be the function computed by B. We have*

$$P(y) = \mathbf{E}[P(U)] + L(y) + \sum_{v\in V} H_v(y)\cdot G_v(y)$$

where $L(y) = \sum_{1\le|\alpha|\le k}\widehat{P}(\alpha)\chi_\alpha(y)$.

[6] Given a function $f : \{\pm1\}^n \to \mathbb{R}$ and index $i \in [n]$. The i-Laplacian of f is defined as a new function $\mathsf{L}_i f(y) := \frac{f(y)-f(y+e_i)}{2}$, where e_i denotes the i-th unit vector. Observe that $\mathsf{L}_i f(y) = \sum_{S:i\in S}\widehat{f}(S)\chi_S(y)$.

Proof. Any function can be written in the Fourier representation, i.e.

$$P(y) = \sum_{\alpha \subseteq [n]} \widehat{P}(\alpha) \cdot \chi_\alpha(y).$$

Now, we can partition this sum to the sum of sets of size at least k and the sum of sets of size smaller than k,

$$P(y) = \mathbf{E}[P] + L(y) + H(y)$$

where

$$\mathbf{E}[P] = \widehat{P}(\emptyset), \qquad L(y) = \sum_{1 \le |\alpha| \le k} \widehat{P}(\alpha)\chi_\alpha(y) \qquad \text{and} \qquad H(y) = \sum_{|\alpha| > k} \widehat{P}(\alpha)\chi_\alpha(y).$$

We decompose $H(y)$ by the above decomposition. ◀

4.2 Forbes-Kelley PRG fools Adaptive roBP

In this section, we prove that the Forbes-Kelley PRG fools adaptive roBP. First, the following lemma is the analog of [8, Lemma 6.3] for adaptive roBP.

▶ **Lemma 17.** *Let B be a read-once adaptive branching program of size s. Suppose D, T, and U are independently drawn from a $2(k+1)$-wise independent distribution, a $(k+1)$-wise independent distribution, and the uniform distribution over $\{\pm 1\}^n$, respectively. Then,*

$$|\mathbf{E}[P(U)] - \mathbf{E}[P(D + T \wedge U)]| \le s \cdot 2^{-k/2}.$$

Since we are working over ± 1 basis, $T \wedge U$ is a coordinate-wise operation defined as $(T \wedge U)_i = -1$ if and only if $T_i = U_i = -1$, and $D + (T \wedge U)$ is defined as $(D + (T \wedge U))_i = D_i \times (T \wedge U)_i$.

Proof. We use the Decomposition Lemma (Lemma 16):

$$|\mathbf{E}[P] - \mathbf{E}[P(D + T \wedge U)]| \le |\mathbf{E}[L(D + T \wedge U)]| + \sum_{v \in V} |\mathbf{E}[(H_v \cdot G_v)(D + T \wedge U)]|. \tag{2}$$

The first summand in the RHS of Eq. (2) equals zero since $D + T \wedge U$ fools any χ_α for $|\alpha| \le k$. Namely,

$$\mathbf{E}[L(D + T \wedge U)] = \sum_{0 < |\alpha| \le k} \widehat{P}(\alpha) \cdot \mathbf{E}[\chi_\alpha(D + T \wedge U)] = 0.$$

We bound the second summand in the RHS of Eq. (2) term by term. For each $v \in V$:

$$\begin{aligned}
&|\mathbf{E}_{D,T,U}[(H_v \cdot B_{v\to})(D + T \wedge U)]| \\
&\le \mathbf{E}_{D,T}\left[|\mathbf{E}_U[(H_v \cdot B_{v\to})(D + T \wedge U)]|\right] \\
&= \mathbf{E}_{D,T}\left[|\mathbf{E}_U[H_v(D + T \wedge U)]| \cdot |\mathbf{E}_U[G_v(D + T \wedge U)]|\right] && (3) \\
&\le \mathbf{E}_{D,T}\left[|\mathbf{E}_U[H_v(D + T \wedge U)]|\right] && (4) \\
&\le 2^{-k/2}. && \text{(Claim 18)}
\end{aligned}$$

Here, (3) follows by observing that, for every fixed T and D, $H_v(D + T \wedge U)$ and $G_v(D + T \wedge U)$ are independent. (4) is due to that $B_{v\to}$ is bounded. Finally, the last line utilizes a claim that is to be introduced and proved next.

Overall, we get

$$|\mathbf{E}[B(U)] - \mathbf{E}[B(D + T \wedge U)]| \le \sum_{i, v \in V_i} 2^{-k/2} = s \cdot 2^{-k/2},$$

as desired. ◀

The following claim has been used in the proof of Lemma 17. We show its proof now.

▷ Claim 18. Let $H_v : \{\pm 1\}^m \to \mathbb{R}$ be a function whose Fourier spectrum is k-homogeneous, i.e.,

$$H_v(y) = \sum_{\alpha \subseteq [m]:|\alpha|=k} \widehat{H_v}(\alpha) \cdot \chi_\alpha(y).$$

Let D, T, and U denote a $2k$-wise independent distribution, a k-wise independent distribution, and uniform distribution over $\{0,1\}^n$. Then,

$$\mathbf{E}_{D,T}\left[|\mathbf{E}_U[H_v(D + T \wedge U)]|\right] \leq 2^{-k/2} \cdot \sqrt{\sum_\alpha \widehat{H_v}(\alpha)^2}$$

Proof. We verify the claim by direction calculation.

$$\begin{aligned}
&\left(\mathbf{E}_{D,T}\left[|\mathbf{E}_U[H_v(D + T \wedge U)]|\right]\right)^2 \\
&\leq \mathbf{E}_{D,T}\left[\mathbf{E}_U[H_v(D + T \wedge U)]^2\right] \\
&= \mathbf{E}_{D,T,U,U'}\left[H_v(D + T \wedge U) \cdot H_v(D + T \wedge U')\right] \\
&= \sum_{\alpha,\alpha'} \widehat{H_v}(\alpha) \cdot \widehat{H_v}(\alpha') \cdot \mathbf{E}_{D,T,U,U'}[\chi_\alpha(D + T \wedge U) \cdot \chi_{\alpha'}(D + T \wedge U')] \\
&= \sum_{\alpha,\alpha'} \widehat{H_v}(\alpha) \cdot \widehat{H_v}(\alpha') \cdot \mathbf{E}_D[\chi_\alpha(D)\chi_{\alpha'}(D)] \cdot \mathbf{E}_{T,U,U'}[\chi_\alpha(T \wedge U) \cdot \chi_{\alpha'}(T \wedge U')] \\
&= \sum_\alpha \widehat{H_v}(\alpha)^2 \cdot \mathbf{E}_{T,U,U'}[\chi_\alpha(T \wedge U) \cdot \chi_\alpha(T \wedge U')] \quad + \sum_{\alpha \neq \alpha'} |\widehat{H_v}(\alpha)| \cdot |\widehat{H_v}(\alpha')| \cdot 0 \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (D \text{ is } 2k\text{-wise}) \\
&= \sum_\alpha \widehat{H_v}(\alpha)^2 \cdot \mathbf{E}_T[\mathbf{1}_{\{\alpha \cap T = \emptyset\}}] \\
&= 2^{-k} \cdot \sum_\alpha \widehat{H_v}(\alpha)^2. \qquad\qquad\qquad\qquad\qquad\qquad (T \text{ is } k\text{-wise independent}, |\alpha| = k)
\end{aligned}$$

◁

Finally, let us remark that we can also use δ-almost k-wise independent distributions T, D to construct $D + T \wedge U$. Doing an analysis similar as we have done for Claim 18, one can show that

$$|\mathbf{E}[P(U)] - \mathbf{E}[P(D + T \wedge U)]| \leq s \cdot \left(\sqrt{\gamma} + 2^{-k/2} + \sqrt{\gamma}\left(\sum_\alpha |\widehat{H}(\alpha)|\right)\right).$$

The argument is also similar to the one done in [8, Lemma 7.2]. We omit the detail here. Note that, if we can prove a good upper bound of $\sum_\alpha |\widehat{H}(\alpha)|$, we can hope to construct D, T using γ-almost k-wise independent distributions with larger γ. Recall the seed length to sample a γ-almost distribution is $O(\log(1/\gamma) + k + \log\log(n))$, which is smaller than the seed length to sample a perfect k-wise independent distribution by a $\log(n)$ factor for large γ (e.g., when $\gamma \approx 2^{-k}$).

4.2.1 PRG for Adaptive roBP

Given Lemma 17, we prove Theorem 7 now.

Proof of Theorem 7. The proof is nearly identical to that of Corollary 11.

Let $\{D^{(i)}\}_{i\in[\ell]}, \{T^{(i)}\}_{i\in[\ell]}$ be ℓ independent copies of $2k$-wise independent dsitributions defined in Lemma 17 with $k = \log\left(\frac{n}{\varepsilon}\right) + \log\log(\frac{n}{\varepsilon}) + 1$.

We construct pseudorandom distributions $G^{(0)}, G^{(1)}, \ldots, G^{(\ell)}$ with $\ell = \Theta(\log(n/\varepsilon))$. We let G_0 be the trivial PRG that outputs $1^n \in \{0,1\}^n$. Then we set

$$G^{(i+1)} = D^{(i)} + T^{(i)} \wedge G^{(i)}.$$

Define a branching program $B^{(i)}$ as $B^{(\ell)} = B$ and

$$B^{(i)}(x) = B^{(i+1)}(D^{(i)} + T^{(i)} \wedge x).$$

Note that $B^{(i)}$ is a random variable depending on $D^{(i)}$ and $T^{(i)}$. Since the restriction of an adaptive roBP is still a roBP. For any realization of $D^{(i)}$ and $T^{(i)}$, Lemma 17 says that,

$$\left|\mathbf{E}[B^{(i+1)}(U)] - \mathbf{E}[B^{(i)}(U)]\right| \leq \frac{\varepsilon/2}{\log(\frac{n}{\varepsilon})}.$$

From a standard Chernoff bound, with probability at least $1 - \varepsilon/2$, $T^{(1)} \wedge T^{(2)} \wedge \cdots \wedge T^{(\ell)} = 0000\ldots0$. Conditioning on this event, $B^{(0)}$ does not depend on its input, implying that $\left|\mathbf{E}[B^{(0)}(U)] - \mathbf{E}_{x\sim G^{(0)}}[B^{(0)}]\right| \leq \varepsilon/2$ since $B^{(0)}$.

On the other hand, by definition, we know $\mathbf{E}_{B^{(0)},x\sim G^{(0)}}[B^{(0)}(x)] = \mathbf{E}_{x\sim G^{(\ell)}}[B(x)]$. Hence a hybrid argument proves that

$$\begin{aligned}
|\mathbf{E}_{x\sim G^{(\ell)}}[B(x)] - \mathbf{E}[B(U)]| &= \left|\mathbf{E}_{x\sim G^{(0)}}[B^{(0)}(x)] - \mathbf{E}[B^{(0)}(U)]\right| + \left|\mathbf{E}[B^{(0)}(U)] - \mathbf{E}[B^{(\ell)}(U)]\right| \\
&\leq \varepsilon/2 + \sum_{i=1}^{\ell} \left|\mathbf{E}[B^{(i-1)}(U)] - \mathbf{E}[B^{(i)}(U)]\right| \\
&\leq \varepsilon,
\end{aligned}$$

completing the proof. ◀

References

1. AmirMahdi Ahmadinejad, Jonathan A. Kelner, Jack Murtagh, John Peebles, Aaron Sidford, and Salil P. Vadhan. High-precision estimation of random walks in small space. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1295–1306. IEEE, 2020.
2. Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992.
3. Andrej Bogdanov, William M. Hoza, Gautam Prakriya, and Edward Pyne. Hitting sets for regular branching programs. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPIcs*, pages 3:1–3:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.
4. Andrej Bogdanov, Periklis A. Papakonstantinou, and Andrew Wan. Pseudorandomness for read-once formulas. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 240–246. IEEE Computer Society, 2011.
5. Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom generators for regular branching programs. *SIAM J. Comput.*, 43(3):973–986, 2014.
6. Joshua Brody and Elad Verbin. The coin problem and pseudorandomness for branching programs. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 30–39. IEEE Computer Society, 2010.

7 Eshan Chattopadhyay, Pooya Hatami, Omer Reingold, and Avishay Tal. Improved pseudorandomness for unordered branching programs through local monotonicity. In *STOC*, pages 363–375. ACM, 2018.

8 Michael A Forbes and Zander Kelley. Pseudorandom generators for read-once branching programs, in any order. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 946–955. IEEE, 2018.

9 Anat Ganor and Ran Raz. Space pseudorandom generators by communication complexity lower bounds. In Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, volume 28 of *LIPIcs*, pages 692–703. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014.

10 Parikshit Gopalan, Raghu Meka, Omer Reingold, Luca Trevisan, and Salil Vadhan. Better pseudorandom generators from milder pseudorandom restrictions. In *Proc. 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 120–129. IEEE, 2012.

11 Elad Haramaty, Chin Ho Lee, and Emanuele Viola. Bounded independence plus noise fools products. *SIAM J. Comput.*, 47(2):493–523, 2018.

12 William M. Hoza, Edward Pyne, and Salil P. Vadhan. Pseudorandom generators for unbounded-width permutation branching programs. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 7:1–7:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

13 Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

14 Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. In *Proc. 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 111–119. IEEE, 2012.

15 Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *Proc. 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 356–364, 1994.

16 Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: derandomizing the XOR lemma. In *Proc. 29th Annual ACM Symposium on Theory of Computing (STOC)*, pages 220–229, 1997.

17 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

18 Chin Ho Lee, Edward Pyne, and Salil P. Vadhan. Fourier growth of regular branching programs. In Amit Chakrabarti and Chaitanya Swamy, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022, September 19-21, 2022, University of Illinois, Urbana-Champaign, USA (Virtual Conference)*, volume 245 of *LIPIcs*, pages 2:1–2:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

19 Chin Ho Lee and Emanuele Viola. More on bounded independence plus noise: Pseudorandom generators for read-once polynomials. *Theory of Computing*, 16:1–50, 2020.

20 Raghu Meka, Omer Reingold, and Avishay Tal. Pseudorandom generators for width-3 branching programs. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 626–637. ACM, 2019.

21 Joseph Naor and Moni Naor. Small-bias probability spaces: efficient constructions and applications. *SIAM Journal of Computing*, 22(4):838–856, 1993.

22 Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

23 Noam Nisan and Avi Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

24 Edward Pyne and Salil P. Vadhan. Limitations of the impagliazzo-nisan-wigderson pseudorandom generator against permutation branching programs. In Chi-Yeh Chen, Wing-Kai Hon, Ling-Ju Hung, and Chia-Wei Lee, editors, *Computing and Combinatorics – 27th International Conference, COCOON 2021, Tainan, Taiwan, October 24-26, 2021, Proceedings*, volume 13025 of *Lecture Notes in Computer Science*, pages 3–12. Springer, 2021.

25 Edward Pyne and Salil P. Vadhan. Pseudodistributions that beat all pseudorandom generators (extended abstract). In Valentine Kabanets, editor, *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPIcs*, pages 33:1–33:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

26 Omer Reingold, Thomas Steinke, and Salil P. Vadhan. Pseudorandomness for regular branching programs via fourier analysis. In Prasad Raghavendra, Sofya Raskhodnikova, Klaus Jansen, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques – 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*, volume 8096 of *Lecture Notes in Computer Science*, pages 655–670. Springer, 2013.

27 Michael E. Saks and Shiyu Zhou. BP $_{\mathrm{h}}$space(s) subseteq dspace($s^{3/2}$). *Journal of Computer and System Sciences*, 58(2):376–403, 1999.

28 Thomas Steinke, Salil P. Vadhan, and Andrew Wan. Pseudorandomness and fourier-growth bounds for width-3 branching programs. *Theory Comput.*, 13(1):1–50, 2017.

29 Yoav Tzur. Notions of weak pseudorandomness and gf (2n)-polynomials. *Master's thesis, Weizmann Institute of Science*, 2009.

30 Salil P. Vadhan. *Pseudorandomness*. Foundations and Trends in Theoretical Computer Science. Now Publishers, 2012.

A Fourier Growth of Constant-Width Adaptive roBP

In this appendix, we show that the Fourier growth of width-w adaptive roBP is upper bounded by that of width-$2w$ oblivious roBP. As a corollary, we can use almost k-wise independent primitives in the construction of Forbes-Kelley PRG, which saves the seed length from $O(\log^3(n/\varepsilon))$ to $\widetilde{O}(w\log^2(n/\varepsilon))$ when w is small.

A.1 Reducing Adaptive roBP to Oblivious roBP

We start by proving the following lemma.

▶ **Lemma 19.** *Suppose $B : \{0,1\}^n \to \{0,1\}$ is computed by a width-w adaptive roBP. Then there is a width-$2w$ oblivious roBP $B' : \{0,1\}^{n^2} \to \{0,1\}$ such that the following inequality holds for every $L \geq 1$,*

$$\sum_{\alpha\subseteq[n]:|\alpha|=L} |\widehat{B}(\alpha)| \leq \sum_{\alpha\subseteq[n^2]:|\alpha|=L} |\widehat{B'}(\alpha)|.$$

Proof. For an input $x \in \{0,1\}^{n^2}$ to B', we partition the bits into chunks of length n. Namely,

$$x = ((x_1^1,\ldots,x_1^n),(x_2^1,\ldots,x_2^n),\ldots,(x_n^1,\ldots,x_n^n)).$$

For each $i \in [n]$, we will think of $(x_j^i)_{j\in[n]}$ as n duplicate bits that equal to the i-th input bit to the original program B. Namely, consider a mapping $\sigma\colon \{0,1\}^n \to \{0,1\}^{n^2}$ as

$$\sigma(z) = ((z_1,z_2,\ldots,z_n),\ldots,(z_1,z_2,\ldots,z_n)).$$

Constructing the oblivious program. We construct a width-$2w$ oblivious roBP B' such that $B(x) = B'(\sigma(x))$. To illustrate, in the following, we use $x = (x_1, \ldots, x_n)$ to denote the input of B, and $z = (z_1^1, \ldots, z_n^n)$ to denote the input of B'. Note that if $z = \sigma(x)$, then $z_i^j = x_j$ for every i, j,

We describe the construction now. Note that B' involves $n^2 + 1$ layers and n^2 transitions. For each $i \in [n]$, We use the $((i-1)n+1)$-th to the (in)-th transitions of B' to implement the i-th transition of B.

Recall that the $(i-1)$-th (resp. i-th) layer of B contains states V_{i-1} (resp. V_i). Write $V_{i-1} = \{v_1, \ldots, v_w\}$ and $V_i = \{u_1, \ldots, u_w\}$. We construct $\overline{V}_{i-1} = \{v'_1, \ldots, v'_w, u'_1, \ldots, u'_w\}$. Identify $v'_1, \ldots, v'_w$ with $v_1, \ldots, v_w$, and $u'_1, \ldots, u'_w$ with $u_1, \ldots, u_w$. Recall that each vertex v_j reads one input bit $x_{\mathsf{pos}_{v_j}}$ from x.

We make $n+1$ copies of $\overline{V}_{i-1}$, denoted by $\overline{V}^0_{i-1}, \ldots, \overline{V}^n_{i-1}$. Next, we build a sub-program from $\overline{V}^0_{i-1}$ to $\overline{V}^n_{i-1}$, using inputs $z_i^1, \ldots, z_i^n$. For each $t \in [n]$, we add edges from $\overline{V}^{t-1}_{i-1}$ to $\overline{V}^t_{i-1}$. We let all states of $\overline{V}^{t-1}_{i-1}$ read the variable z_i^t (which is supposed to be x_t if $\sigma(x) = z$). For every v_j such that $\mathsf{pos}_{v_j} = t$, suppose v_j has two out edges to u_{j_0}, u_{j_1} with label 0 and 1. We add two edges from v'_j (in $\overline{V}^{t-1}_{i-1}$) to u'_{j_0} and $u'_{j,1}$ (in $\overline{V}^t_{i-1}$) with label 0 and 1. For every v_j where $\mathsf{pos}_{v_j} \neq t$ and every u_j, the state reads the input and simply ignores it. (operationally, this means we add two edges from the current state to the corresponding state in the next layer.)

Now we have n sub-programs: for each $i \in [n]$, we have a subprogram from $\overline{V}^0_{i-1}$ to $\overline{V}^n_{i-1}$. For each $i \in [n]$, observe that both the "u"-states of $\overline{V}^n_{i-1}$ and the "v"-states of $\overline{V}^0_i$ are identified with states in V_i. We naturally glue each pair of corresponding states together. We also glue "v"-states of $\overline{V}^n_{i-1}$ and "u"-states of $\overline{V}^0_i$ arbitrarily. This way, we construct a larger branching program of length n^2 (from $\overline{V}^0_1$ to $\overline{V}^n_n$) and width $2w$. It is straightforward to verify that $B(x) = B'(\sigma(x))$.

Calculating Fourier weights. Now we verify that B' satisfies the lemma statement. Consider the Fourier spectrum of B':

$$B'(z) = \sum_{\alpha \subseteq [n^2]} \widehat{B'}(\alpha)\chi_\alpha(z).$$

We claim that, for every $\alpha \subseteq [n^2]$ such that there exists $\{k_1 n + i, k_2 n + i\} \subseteq \alpha$ for some $k_1 \neq k_2$ and i, it must be the case that $|\widehat{B'}(\alpha)| = 0$. Indeed, we have

$$\widehat{B'}(\alpha) = \mathbf{E}_{z \sim U_{n^2}}[\chi_\alpha(z) \cdot B'(z)]. \tag{5}$$

We observe that

$$B'(z) = \sum_{\pi : \text{accepting computation path}} \mathbf{1}[B' \text{ on input } z \text{ follows } \pi].$$

Let $z^i_{k_1}, z^i_{k_2}$ be the two variables associated with indices $\{k_1 n + i, k_2 n + i\}$. By the promise that B is read-once, in any computation path π of B', it cannot be the case that both $z^i_{k_1}$ and $z^i_{k_2}$ are used (i.e., at least one of them is ignored in the path). It follows that each path contributes zero to (5). Consequently, $\widehat{B'}(\alpha) = 0$.

Next, we have

$$B(x) = B'(\sigma(x)) = \sum_{\alpha \subseteq [n^2]} \widehat{B'}(\alpha)\chi_\alpha(\sigma(x)).$$

As we have shown, $\widehat{B'}(\alpha)$ is non-zero only when α does not contain two variables $z^i_{k_1}, z^i_{k_2}$ in the same group k. For every such α, $\chi_\alpha(\sigma(x)) = \chi_{\Pi(\alpha)}(x)$ where Π denotes the projection of α onto $[n]$. Namely, $\Pi(\alpha)_i = 1$ if and only if $\alpha_{kn+i} = 1$ for some $k \in [n]$. It follows that $|\alpha| = |\Pi(\alpha)|$. Finally, applying the triangle inequality gives

$$\sum_{\beta \subseteq [n]:|\beta|=L} |\widehat{B}(\beta)| \leq \sum_{\alpha \subseteq [n^2]:|\alpha|=L} |\widehat{B'}(\alpha)|,$$

as desired. ◀

A.2 Fourier Growth and Pseudorandomness

Chattopadhyay, Hatami, Reingold and Tal [7] proved the following Fourier growth bound for width-w oblivious roBP.

▶ **Theorem 20** ([7]). *Suppose $B : \{0,1\}^n \to \{0,1\}$ is computed by a width-w oblivious roBP. Then, for every $k \geq 1$, it holds that*

$$\sum_{\alpha:|\alpha|=k} |\widehat{B}(\alpha)| \leq O(\log(nw))^{wk}.$$

As a direct corollary from Lemma 19 and Theorem 20, we obtain the following Fourier growth bound for width-w adaptive roBP.

▶ **Corollary 21.** *Suppose $B : \{0,1\}^n \to \{0,1\}$ is computed by a width-w adaptive roBP. Then, for every $k \geq 1$, it holds that*

$$\sum_{\alpha:|\alpha|=k} |\widehat{B}(\alpha)| \leq O(\log(nw))^{2wk}.$$

Similarly as done by Forbes and Kelley [8], one can use the Fourier growth bound to improve the seed length for small-width adaptive roBP, and obtain the following corollary.

▶ **Corollary 22** (Restating Theorem 8). *For every $n, w \geq 1$ and $\varepsilon > 0$, there is an explicit ε-PRG $G : \{0,1\}^s \to \{0,1\}^n$ fooling width-w adaptive roBPs with seed length $s = \widetilde{O}(w \log^2(n/\varepsilon))$.*

Approximate Nearest Neighbor for Polygonal Curves Under Fréchet Distance

Siu-Wing Cheng
Department of Computer Science and Engineering,
Hong Kong University of Science and Technology, Hong Kong, China

Haoqiang Huang
Department of Computer Science and Engineering,
Hong Kong University of Science and Technology, Hong Kong, China

Abstract

We propose κ-approximate nearest neighbor (ANN) data structures for n polygonal curves under the Fréchet distance in $\mathbb{R}^d$, where $\kappa \in \{1+\varepsilon, 3+\varepsilon\}$ and $d \geq 2$. We assume that every input curve has at most m vertices, every query curve has at most k vertices, $k \ll m$, and k is given for preprocessing. The query times are $\tilde{O}(k(mn)^{0.5+\varepsilon}/\varepsilon^d + k(d/\varepsilon)^{O(dk)})$ for $(1+\varepsilon)$-ANN and $\tilde{O}(k(mn)^{0.5+\varepsilon}/\varepsilon^d)$ for $(3+\varepsilon)$-ANN. The space and expected preprocessing time are $\tilde{O}(k(mnd^d/\varepsilon^d)^{O(k+1/\varepsilon^2)})$ in both cases. In two and three dimensions, we improve the query times to $O(1/\varepsilon)^{O(k)} \cdot \tilde{O}(k)$ for $(1+\varepsilon)$-ANN and $\tilde{O}(k)$ for $(3+\varepsilon)$-ANN. The space and expected preprocessing time improve to $O(mn/\varepsilon)^{O(k)} \cdot \tilde{O}(k)$ in both cases. For ease of presentation, we treat factors in our bounds that depend purely on d as $O(1)$. The hidden polylog factors in the big-$\tilde{O}$ notation have powers dependent on d.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Polygonal curves, Fréchet distance, approximate nearest neighbor

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.40

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2304.14643

1 Introduction

Given a set of trajectories, the *nearest neighbor* problem is to efficiently report the one most similar to a query trajectory. Trajectories are often represented as polygonal curves, and the nearest neighbor problem is encountered frequently in applications [19, 20, 21].

Various similarity metrics have been proposed for polygonal curves. We are interested in the *Fréchet distance* [3] which has attracted much attention in recent years. It is defined as follows. A parameterization of a curve τ is a function $\rho : [0,1] \to \mathbb{R}^d$ such that, as t increases from 0 to 1, the point $\rho(t)$ moves monotonically from the beginning of τ to its end. We may have $\rho(t_1) = \rho(t_2)$ for two distinct values t_1 and t_2. Two parameterizations ρ and ϱ for curves τ and σ, respectively, induce a *matching* $\mathcal{M}$: for all $t \in [0,1]$, $\mathcal{M}$ matches $\rho(t)$ with $\varrho(t)$. A point can be matched with multiple partners. The distance between τ and σ under $\mathcal{M}$ is $d_{\mathcal{M}}(\tau,\sigma) = \max_{t\in[0,1]} d(\rho(t), \varrho(t))$, where $d(\cdot,\cdot)$ denotes the Euclidean distance. The Fréchet distance is $d_F(\tau,\sigma) = \min_{\mathcal{M}} d_{\mathcal{M}}(\tau,\sigma)$. We call a minimizing matching a *Fréchet matching*.

Let $T = \{\tau_1, \ldots, \tau_n\}$ be a set of n polygonal curves with at most m vertices each. Given any value $\kappa \geq 1$, the *κ-approximate nearest neighbor (ANN) problem* is to construct a data structure so that for any query curve σ, we can quickly report a curve $\tau_l \in T$ with $d_F(\sigma,\tau_l) \leq \kappa \cdot \min_{\tau_i \in T} d_F(\sigma,\tau_i)$. We assume that every query curve has at most k vertices, and k is given for preprocessing. In the literature, if $k = m$, it is called the *symmetric* version; if $k < m$, it is called the *asymmetric version*. If the query curve is sketched by the user, it

is likely that $k \ll m$ and this is the scenario for which we design our data structures. We define the (κ, δ)-*ANN problem* as follows: for any query curve, we report "no" or a curve $\tau_l \in T$ with $d_F(\sigma, \tau_l) \leq \kappa\delta$; if we report "no", it must be the case that $\min_{\tau_i \in T} d_F(\sigma, \tau_i) > \delta$.

There have been many results on the ANN problem under the *discrete* Fréchet distance $\tilde{d}_F$, which restricts the definition of d_F to parameterizations ρ and ϱ that match each vertex of τ with at least one vertex of σ, and vice versa. As a result, $d_F(\tau, \sigma) \leq \tilde{d}_F(\tau, \sigma)$. It is possible that $d_F(\tau, \sigma) \ll \tilde{d}_F(\tau, \sigma)$; for example, σ is a long horizontal line segment, and τ is a parallel copy near σ with an extra vertex in the middle. The advantage of $\tilde{d}_F$ is that it can be computed using a simple dynamic programming algorithm [11].

Indyk and Motwani [17] and Har-Peled [14] proved that a solution for the (κ, δ)-ANN problem for points in a metric space gives a solution for the $\kappa(1 + O(\varepsilon))$-ANN problem. The result has been simplified in the journal version [15]. The method is general enough that it works for polygonal curves under d_F and $\tilde{d}_F$. Theorem 1 in Section 2 states the deterministic result in our context; the reduction increases the space and query time by polylogarithmic factors. If a probabilistic (κ, δ)-ANN solution with failure probability f is used, the bounds in Theorem 1 also hold, and the ANN solution has an $O(f \log n)$ failure probability.

Indyk [16] proposed the first (κ, δ)-ANN solution under $\tilde{d}_F$, where $\kappa = O(\log m + \log\log n)$, for the case that $k = m$ and the vertices come from a discrete point set X. It uses $O\big(|X|^{\sqrt{m}}(m^{\sqrt{m}}n)^2\big)$ space and answers a query in $O\big(m^{O(1)} \log n\big)$ time.[1] Driemel and Silverstri [10] developed probabilistic (κ, δ)-ANN solutions under $\tilde{d}_F$ with a failure probability $1/n$; they achieve the following combinations of (κ, query time, space) for the case of $k = m$: $\big(4d^{3/2}m, O(m), O(n \log n + mn)\big)$, $\big(4d^{3/2}, O(2^{4dm}m \log n), O(2^{4md}n \log n + mn)\big)$, and $\big(4d^{3/2}m/t, O(2^{2t}m^t \log n), O(2^{2t}m^{t-1}n \log n + mn)\big)$ for any integer $t \geq 1$. The approximation ratio has been reduced to $1 + \varepsilon$ by two research groups later. Filtser et. al. [13] proposed two deterministic $(1+\varepsilon, \delta)$-ANN data structures under $\tilde{d}_F$; one answers a query in $O(kd)$ time and uses $n \cdot O(\frac{1}{\varepsilon})^{kd}$ space and $O(mn(d \log m + O(\frac{1}{\varepsilon})^{kd}))$ expected preprocessing time; the other answers a query in $O(kd \log \frac{dkn}{\varepsilon})$ time and uses $n \cdot O(\frac{1}{\varepsilon})^{kd}$ space and $O(mn \log \frac{n}{\varepsilon} \cdot (d \log m + O(\frac{1}{\varepsilon})^{kd}))$ worst-case preprocessing time. Emiris and Psarros [12] obtained probabilistic $(1+\varepsilon)$-ANN and $(1 + \varepsilon, \delta)$-ANN data structures under $\tilde{d}_F$ with failure probabilities $1/2$ for the case of $k = m$. The $(1+\varepsilon)$-ANN data structure answers a query in $\tilde{O}(d2^{4m}m^{O(1/\varepsilon)})$ time and uses $\tilde{O}(dm^2n) \cdot (2 + d/\log m)^{O(dm^{O(1/\varepsilon)} \log(1/\varepsilon))}$ space and preprocessing time. The $(1 + \varepsilon, \delta)$-ANN data structure answers a query in $O(d2^{4m} \log n)$ time and uses $O(dn) + (mn)^{O(m/\varepsilon^2)}$ space and preprocessing time. The failure probabilities can be reduced to $1/n$ with an increase in the query time, space, and preprocessing time by an $O(\log n)$ factor.

Most known results under d_F are for $\mathbb{R}$. For curves in $\mathbb{R}$ (time series), Driemel and Psarros [9] developed the first (κ, δ)-ANN data structures under d_F with the following combinations of (κ, query time, space): $\big(5+\varepsilon, O(k), O(mn)+n\cdot O(\frac{1}{\varepsilon})^k\big)$, $\big(2+\varepsilon, O(2^k k), O(mn)+n\cdot O(\frac{m}{k\varepsilon})^k\big)$, and $\big(24k + 1, O(k \log n), O(n \log n + mn)\big)$. The last one is probabilistic, and the failure probability is $1/\text{poly}(n)$. Later, Bringman et al. [5] obtained improved solutions with the following combinations of (κ, query time, space): $\big(1 + \varepsilon, O(2^k k), n \cdot O(\frac{m}{k\varepsilon})^k\big)$, $\big(2 + \varepsilon, O(k), n \cdot O(\frac{m}{k\varepsilon})^k\big)$, $\big(2 + \varepsilon, O(2^k k), O(mn) + n \cdot O(\frac{1}{\varepsilon})^k\big)$, $\big(2 + \varepsilon, O(\frac{1}{\varepsilon})^{k+2}, O(mn)\big)$, and $\big(3 + \varepsilon, O(k), O(mn) + n \cdot O(\frac{1}{\varepsilon})^k\big)$. They also obtained lower bounds that are conditioned on the Orthogonal Vectors Hypothesis: for all $\varepsilon, \varepsilon' \in (0, 1)$, it is impossible to achieve the combination $\big(2 - \varepsilon, O(n^{1-\varepsilon'}), \text{poly}(n)\big)$ in $\mathbb{R}$ when $1 \ll k \ll \log n$ and $m = kn^{\Theta(1/k)}$, or $\big(3 - \varepsilon, O(n^{1-\varepsilon'}), \text{poly}(n)\big)$ in $\mathbb{R}$ when $m = k = \Theta(\log n)$, or $\big(3 - \varepsilon, O(n^{1-\varepsilon'}), \text{poly}(n)\big)$ in $\mathbb{R}^2$ when $1 \ll k \ll \log n$ and $m = kn^{\Theta(1/k)}$. Mirzanezhad [18] described a $(1 + \varepsilon, \delta)$-ANN data structure for $\mathbb{R}^d$ that answer a query in $O(kd)$ time and uses $O(n \cdot \max\{\sqrt{d}/\varepsilon, \sqrt{d}D/\varepsilon^2\}^{dk})$

[1] A tradeoff is also presented in [16].

space, where D is the diameter of the set of input curves. If k is not given, the approximation ratio and space increase to $5+\varepsilon$ and $n \cdot O(\frac{1}{\varepsilon})^{dm}$, respectively. There is no bound on D in the space complexity of the first solution. We summarize all these previous results in Table 1 for easier comparison.

Table 1 Comparison of our data structures to the previous results.

Distance	Space	Query time	Approximation
Continuous Fréchet, $\mathbb{R}$	$O(mn) + n \cdot O\left(\frac{1}{\varepsilon}\right)^k$	$O(k)$	$(5+\epsilon,\delta)$-ANN [9]
	$O(mn) + n \cdot O\left(\frac{m}{k\varepsilon}\right)^k$	$O(2^k k)$	$(2+\epsilon,\delta)$-ANN [9]
	$O(n \log n + mn)$	$O(k \log n)$	$(24k+1,\delta)$-ANN [9][a]
	$n \cdot O\left(\frac{m}{k\varepsilon}\right)^k$	$O(2^k k)$	$(1+\varepsilon,\delta)$-ANN [5]
	$n \cdot O\left(\frac{m}{k\varepsilon}\right)^k$	$O(k)$	$(2+\varepsilon,\delta)$-ANN [5]
	$O(mn) + n \cdot O\left(\frac{1}{\varepsilon}\right)^k$	$O(2^k k)$	$(2+\varepsilon,\delta)$-ANN [5]
	$O(mn)$	$O\left(\frac{1}{\varepsilon}\right)^{k+2}$	$(2+\varepsilon,\delta)$-ANN [5]
	$O(mn) + n \cdot O\left(\frac{1}{\varepsilon}\right)^k$	$O(k)$	$(3+\varepsilon,\delta)$-ANN [5]
Continuous Fréchet, $\mathbb{R}^d$	$O\left(n \cdot \max\{\sqrt{d}/\varepsilon, \sqrt{d}D/\varepsilon^2\}^{dk}\right)$	$O(kd)$	$(1+\varepsilon,\delta)$-ANN [18]
	$n \cdot O\left(\frac{1}{\varepsilon}\right)^{dm}$	$O(kd)$	$(5+\varepsilon,\delta)$-ANN [18]
	$\tilde{O}\left(k(mnd^d/\varepsilon^d)^{O(k+1/\varepsilon^2)}\right)$	$\tilde{O}\left(k(mn)^{0.5+\varepsilon}/\varepsilon^d + k(d/\varepsilon)^{O(dk)}\right)$	$(1+\varepsilon,\delta)$-ANN, Theorem 9
	$\tilde{O}\left(k(mnd^d/\varepsilon^d)^{O(k+1/\varepsilon^2)}\right)$	$\tilde{O}\left(k(mn)^{0.5+\varepsilon}/\varepsilon^d\right)$	$(3+\varepsilon,\delta)$-ANN, Theorem 11
Continuous Fréchet, $\mathbb{R}^2$ and $\mathbb{R}^3$	$O\left(\frac{1}{\varepsilon}\right)^{4d(k-1)+1} (mn)^{4(k-1)} k \log^2 n$	$O\left(\frac{1}{\varepsilon}^{2d(k-2)}\right) k \log \frac{mn}{\varepsilon} \log n$	$(1+\varepsilon,\delta)$-ANN, Theorem 9
	$O\left(\frac{1}{\varepsilon}\right)^{2d(k-1)+1} (mn)^{2(k-1)} k \log^2 n$	$O\left(k \log \frac{mn}{\varepsilon} \log n\right)$	$(3+\varepsilon,\delta)$-ANN, Theorem 11
Discrete Fréchet, $\mathbb{R}^d$	$O\left(\lvert X\rvert^{\sqrt{m}}(m^{\sqrt{m}}n)^2\right)$	$O\left(m^{O(1)} \log n\right)$	$(O(\log m + \log\log n),\delta)$-ANN [16]
	$O(n \log n + mn)$	$O(m)$	$(4d^{3/2}m,\delta)$-ANN [10]
	$O(2^{4md} n \log n + mn)$	$O(2^{4dm} m \log n)$	$(4d^{3/2},\delta)$-ANN [10]
	$O(2^{2t} m^{t-1} n \log n + mn)$	$O(2^{2t} m^t \log n)$	$(4d^{3/2}m/t,\delta)$-ANN [10]
	$n \cdot O\left(\frac{1}{\varepsilon}\right)^{kd}$	$O(kd)^b$	$(1+\varepsilon,\delta)$-ANN [13]
	$O(dn) + (mn)^{O(m/\varepsilon^2)}$	$O\left(d2^{4m} \log n\right)$	$(1+\varepsilon,\delta)$-ANN[c] [12]

[a] A randomized data structure with a failure probability of $1/\text{poly}(n)$.
[b] The query time is achieved by implementing the dictionary with a hash table. The query time is $O(kd \log \frac{dkn}{\varepsilon})$ when implementing the dictionary with a trie.
[c] A randomized data structure with a failure probability of $\frac{1}{2}$.

We develop (κ,δ)-ANN data structures under d_F in $\mathbb{R}^d$ for $\kappa \in \{1+\varepsilon, 3+\varepsilon\}$ and $d \geq 2$. We assume that every query curve has at most k vertices, $k \ll m$, and k is given for preprocessing. To simplify the bounds, we assume that $k \geq 3$ throughout this paper. There are three design goals. First, the query times are sublinear in mn. Second, the space complexities depend only on the input parameters. Third, the space complexities are neither proportional to $\min\{m^{\Omega(d)}, n^{\Omega(d)}\}$ nor exponential in $\min\{m,n\}$. It would be desirable to remove all exponential dependencies on d, but we are not there yet.

We achieve a query time of $\tilde{O}(k(mn)^{0.5+\varepsilon}/\varepsilon^d + k(d/\varepsilon)^{O(dk)})$ for $(1+\varepsilon,\delta)$-ANN. We remove the exponential dependence on k for $(3+\varepsilon,\delta)$-ANN and obtain an $\tilde{O}(k(mn)^{0.5+\varepsilon}/\varepsilon^d)$ query time. The space and expected preprocessing time are $\tilde{O}(k(mnd^d/\varepsilon^d)^{O(k+1/\varepsilon^2)})$ in both cases. For ease of presentation, we treat any factor in our bounds that depends only on d as $O(1)$. The hidden polylog factors in the big-$\tilde{O}$ notation have powers dependent on d. In two and three dimensions, we improve the query times to $O(1/\varepsilon)^{O(k)} \cdot \tilde{O}(k)$ for $(1+\varepsilon,\delta)$-ANN and $\tilde{O}(k)$ for $(3+\varepsilon,\delta)$-ANN. The space and expected preprocessing time improve to $O(mn/\varepsilon)^{O(k)} \cdot \tilde{O}(k)$ in both cases. Using the reduction in [15] (Theorem 1 in Section 2), we obtain $(1+\varepsilon)$-ANN and $(3+\varepsilon)$-ANN data structures by increasing the query time and space by an $O(\log n)$ and an $O(\frac{1}{\varepsilon}\log^2 n)$ factors, respectively. More precise bounds are stated in Theorems 9 and 11.

Our $(1+\varepsilon,\delta)$-ANN result is based on two new ideas. First, we develop a novel encoding of query curves that are based on local grids in the input vertex neighborhoods. Second, we draw a connection to an approximate segment shooting problem which we solve efficiently. We present these ideas in Sections 2 and 4. Our $(3+\varepsilon)$-ANN result is obtained by simplifying the encoding. We present this result in Section 3.

We work in the word RAM model. We use $(v_{i,1}, \ldots, v_{i,m})$ to denote the sequence of vertices of τ_i from beginning to end – τ_i is oriented from $v_{i,1}$ to $v_{i,m}$. We use $\tau_{i,a}$ to denote the edge $v_{i,a}v_{i,a+1}$. For any two points $x, y \in \tau_i$, we say that $x \leq_{\tau_i} y$ if x does not appear behind y along τ_i, and $\tau_i[x, y]$ denotes the subcurve between x and y. Given two subsets $X, Y \subseteq \tau_i$, $X \leq_{\tau_i} Y$ if and only if for every point $x \in X$ and every point $y \in Y$, $x \leq_{\tau_i} y$. A ball centered at the origin with radius r is denoted by B_r. Given two subsets $X, Y \subset \mathbb{R}^d$, $d(X, Y) = \min_{x \in X, y \in Y} d(x, y)$; their *Minkowski sum* is $X \oplus Y = \{x + y : x \in X, y \in Y\}$; if $X = \{p\}$, we write $p \oplus Y$ for simplicity. For any $x, y \in \mathbb{R}^d$, xy denotes the *oriented segment from x to y*, and $\mathrm{aff}(xy)$ is the *oriented support line of xy* that shares the orientation of xy.

2 $(1 + O(\varepsilon), \delta)$-ANN

Har-Peled et al. [15, Theorem 2.10] proved a reduction from the $(1 + \varepsilon)$-ANN problem to the $(1 + \varepsilon, \delta)$-ANN problem. Although the result is described for points in a metric space with a probabilistic data structure for the $(1 + \varepsilon, \delta)$-ANN problem, the method is general enough to work for polygonal curves under d_F or $\tilde{d}_F$ in $\mathbb{R}^d$ and any deterministic solution for the $(1 + \varepsilon, \delta)$-ANN problem. We rephrase their result in our context below.

▶ **Theorem 1** ([15]). *Let T be a set of n polygonal curves in $\mathbb{R}^d$. If there is a data structure for the (κ, δ)-ANN problem for T under d_F or $\tilde{d}_F$ that has space complexity S, query time Q, deletion time D, and preprocessing time P, then there is a $\kappa(1 + O(\varepsilon))$-ANN data structure for T under d_F or $\tilde{d}_F$ that has space complexity $O(\frac{1}{\varepsilon} S \log^2 n)$, query time $O(Q \log n)$, and expected preprocessing time $O\big(\frac{1}{\varepsilon \log^2 n} P + (Q + D) n \log n\big)$.*

By Theorem 1, we can focus on the $(1 + \varepsilon, \delta)$-ANN problem. Without loss of generality, we assume that each curve in T has exactly m vertices, and every query curve has exactly k vertices. If necessary, extra vertices can be added in an arbitrary manner to enforce this assumption without affecting the Fréchet distance.

The high level idea of our preprocessing is to identify all query curves that are within a Fréchet distance $(1 + O(\varepsilon))\delta$ from each $\tau_i \in T$, group the curves that share similar structural characteristics, assign each group a unique key value, and store these key values in a trie $\mathcal{D}$. It is possible for a query curve to belong to multiple groups. Each key value is associated with the subset of curves in T that induce that key value. Correspondingly, given a query curve σ, we generate all possible key values for σ and search $\mathcal{D}$ with them. If some curve in T is retrieved, it is the desired answer; otherwise, we report "no".

There are two challenges to overcome. First, it is impossible to examine all possible query curves. We can only check some space discretization in order to obtain a finite running time. To control the discretization error, it is easy to cover the input curves by a grid with an appropriate cell width; however, the grid size and hence the data structure size would then depend on some non-combinatorial parameters. We propose *coarse encodings* of query curves so that there are $O(\sqrt{d}/\varepsilon)^{4d(k-1)}(mn)^{4(k-1)}$ of them. A query curve may have $O(\sqrt{d}/\varepsilon)^{2d(k-2)}$ coarse encodings. The second challenge is to efficiently generate all possible coarse encodings of a query curve at query time. We reduce the coarse encoding generation to an approximate segment shooting problem. This step turns out to be the bottleneck in four and higher dimensions as we aim to avoid any factor of the form $m^{\Omega(d)}$ or $n^{\Omega(d)}$ in the space complexity. It is the reason for the $(mn)^{0.5+\varepsilon}$ term in the query time. In two and three dimensions, the approximate segment shooting problem can be solved more efficiently.

In the rest of this section, we present the coarse encoding and a $(1 + O(\varepsilon), \delta)$-ANN data structure, using an approximate segment shooting oracle. The approximate segment shooting problem can be solved by the results in [8] in two and three dimensions. We solve the approximate segment shooting problem in four and higher dimensions in Section 4.

2.1 Coarse encodings of query curves

Imagine an infinite grid in $\mathbb{R}^d$ of cell width $\varepsilon\delta/\sqrt{d}$. For any subset $R \subset \mathbb{R}^d$, we use $G(R)$ to denote the set of grid cells that intersect R. Let $\mathcal{G}_1 = \bigcup_{i\in[n],a\in[m]} G(v_{i,a} \oplus B_\delta)$. Let $\mathcal{G}_2 = \bigcup_{i\in[n],a\in[m]} G(v_{i,a} \oplus B_{(2+12\varepsilon)\delta})$. Both $\mathcal{G}_1$ and $\mathcal{G}_2$ have $O(mn/\varepsilon^d)$ size.

The coarse encoding of a curve $\sigma = (w_1, w_2, \ldots, w_k)$ is a 3-tuple $\mathcal{F} = (\mathcal{A}, \mathcal{B}, \mathcal{C})$. The component $\mathcal{C}$ is sequence of pairs of grid cells $((c_{j,1}, c_{j,2}))_{j\in[k-1]}$ such that $(c_{j,1}, c_{j,2}) \in (\mathcal{G}_1 \times \mathcal{G}_1) \cup \{\text{null}\}$. Both $\mathcal{A}$ and $\mathcal{B}$ are arrays of length $k-1$, and every element of $\mathcal{A}$ and $\mathcal{B}$ belongs to $\mathcal{G}_2 \cup \{\text{null}\}$. We first provide the intuition behind the design of $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ before describing the constraints that realize the intuition.

Imagine that a curve $\tau_i \in T$ is a $(1 + O(\varepsilon))$-ANN of σ. The data structure needs to cater for the preprocessing, during which the query curve σ is not available; it also needs to cater for the query procedure, during which we do not want to directly consult the input curves in T in order to avoid a linear dependence in mn.

In preprocessing, we use pairs of grid cells as surrogates of the possible query curve edges. The advantage is that we can enumerate all possible pairs of grid cells and hence cater for all possible query curve edges. Specifically, for $j \in [k-1]$, if $(c_{j,1}, c_{j,2}) \neq$ null, it is the surrogate of w_jw_{j+1}, so w_jw_{j+1} should pass near $c_{j,1}$ and $c_{j,2}$. Each non-null $(c_{j,1}, c_{j,2})$ corresponds to a contiguous subsequence $v_{i,a}, \ldots, v_{i,b}$ of vertices of τ_i that are matched to points in w_jw_{j+1} in a Fréchet matching. Of course, we do not know the Fréchet matching, so we will need to enumerate and handle all possibilities. Also, since w_jw_{j+1} is unknown in preprocessing, $v_{i,a}, \ldots, v_{i,b}$ can only be matched to a segment joining a vertex x_j of $c_{j,1}$ to a vertex y_j of $c_{j,2}$ so that $d_F(x'_jy'_j, \tau_i[v_{i,a}, v_{i,b}]) \leq (1 + O(\varepsilon))\delta$ for some subsegment $x'_jy'_j \subseteq x_jy_j$. This property will be enforced in the data structure construction later.

At query time, given $\sigma = (w_1, \ldots, w_k)$, we will make approximate segment shooting queries to determine a sequence of cell pairs $((c_{j,1}, c_{j,2}))_{j\in[k-1]}$. We do not always use $(c_{j,1}, c_{j,2})$ as a surrogate for the edge w_jw_{j+1} though. As mentioned in the previous paragraph, a non-null $(c_{j,1}, c_{j,2})$ denotes the matching of a contiguous subsequence of input curve vertices to w_jw_{j+1}; however, we must also allow the matching of a contiguous subsequence of vertices of σ to a single input edge. Therefore, after determining $((c_{j,1}, c_{j,2}))_{j\in[k-1]}$, we still have the choice of using $(c_{j,1}, c_{j,2})$ as is or substituting it by the null value. For a technical reason, $(c_{1,1}, c_{1,2})$ and $(c_{k-1,1}, c_{k-1,2})$ are always kept non-null, so we have 2^{k-3} possible sequences of pairs of cells. Take one of these sequences. If $(c_{r,1}, c_{r,2})$ and $(c_{s,1}, c_{s,2})$ are two non-null pairs such that $(c_{j,1}, c_{j,2}) =$ null for $j \in [r+1, s-1]$, it means that no vertex of τ_i is matched to w_jw_{j+1} for $j \in [r+1, s-1]$. As a result, the vertices $w_{r+1}, \ldots, w_s$ of σ are matched to the edge $v_{i,b}v_{i,b+1}$ of τ_i, where $v_{i,b}$ is the last vertex of τ_i matched to w_rw_{r+1} in the current enumeration. We use the pair of cells $\mathcal{A}[r]$ and $\mathcal{B}[s]$ as the surrogate of the edge $v_{i,b}v_{i,b+1}$. So we require $\mathcal{A}[r]$ and $\mathcal{B}[s]$ to be near $c_{r,2}$ and $c_{s,1}$, respectively, because $(c_{r,1}, c_{r,2})$ is the surrogate of w_rw_{r+1}, and $(c_{s,1}, c_{s,2})$ is the surrogate of w_sw_{s+1}. We have to try all possible locations of $\mathcal{A}[r]$ and $\mathcal{B}[s]$ in the vicinity of $c_{r,2}$ and $c_{s,1}$. $\mathcal{A}[r]$ and $\mathcal{B}[s]$ can be the surrogate for edges of multiple curves in T, which allows us to compare σ with multiple input curves simultaneously at query time. The constraint to be enforced is that $w_{r+1}, \ldots, w_s$ can be matched to a segment joining a vertex x_r of $\mathcal{A}[r]$ and a vertex x_s of $\mathcal{B}[s]$ so that $d_F(x'_rx'_s, \sigma[w_{r+1}, w_s]) \leq (1 + O(\varepsilon))\delta$ for some subsegment $x'_rx'_s \subseteq x_rx_s$. Figure 1 shows a illustration for the intuition above.

We present the constraints for $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ that realize the intuition above. When $(c_{j,1}, c_{j,2}) \neq$ null, a natural choice of $c_{j,1}$ is the first grid cell in $\mathcal{G}_1$ that we hit when walking from w_j to w_{j+1}, i.e., segment shooting. In $\mathbb{R}^d$ where $d \in \{2, 3\}$, there are ray shooting data structures for boxes [8]. In higher dimensions, ray shooting results are known for a single convex

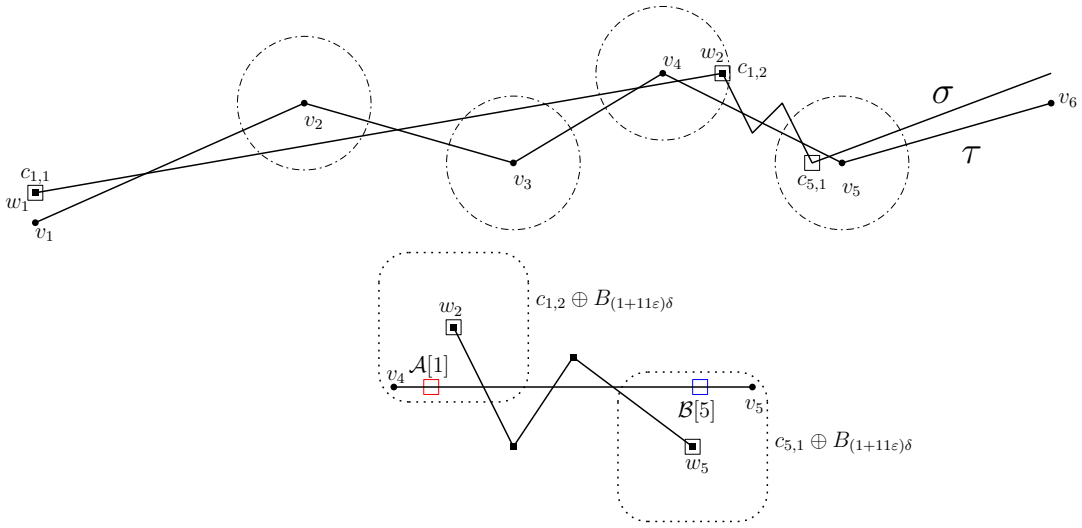


Figure 1 The underlying intuition for deriving the coarse encoding of σ. We use τ instead of τ_i for ease of notation. Assume that $d_F(\tau, \sigma) \leq \delta$ and the vertices v_1, v_2, v_3, v_4 are matched to the segment w_1w_2 by the Fréchet matching. w_1w_2 must intersect some balls centered at τ's vertices, which means that w_1w_2 intersects $\mathcal{G}_1$. Let $c_{1,1}$ and $c_{1,2}$ be the first and the last cells in $\mathcal{G}_1$ that intersect w_1w_2 along the direction of w_1w_2. $(c_{1,1}, c_{1,2})$ can serve as a surrogate of the edge w_1w_2 in a sense that we can verify whether v_1, v_2, v_3, v_4 can be matched to w_1w_2 properly by checking whether they can be matched to a segment that joins vertices of $c_{1,1}$ and $c_{1,2}$ properly. This idea can be generalized to all edges of σ with vertices of τ matched to them. The subcurve $\sigma[w_2, w_5]$ is matched to an edge v_4v_5 of τ. We introduce $\mathcal{A}[1] \subset G(c_{1,2} \oplus B_{(1+11\varepsilon)\delta})$ and $\mathcal{B}[5] \subset G(c_{5,1} \oplus B_{(1+11\varepsilon)\delta})$. $(\mathcal{A}[1], \mathcal{B}[5])$ can serve as a surrogate of v_4v_5. $(\mathcal{A}[1], \mathcal{B}[5])$ can encode $\sigma[w_2, w_5]$ sufficiently because for every segment x_1x_5 that joins a vertex x_1 of $\mathcal{A}[1]$ and a vertex x_5 of $\mathcal{B}[5]$, there exists a subsegment $x_1'x_5' \subset x_1x_5$ such that $d_F(x_1'x_5', \sigma[w_2, w_5]) \leq (1 + O(\varepsilon))\delta$.

polytope and an arrangement of hyperplanes [1]; even in such cases, the query time is substantially sublinear only if the space complexity is at least the input size raised to a power of $\Omega(d)$. It would be $(mn)^{\Omega(d)}$ in our case. We define a λ-segment query problem below that approximates the ray shooting problem, and we will present an efficient solution for $\lambda = 11\varepsilon\delta$ in Section 4 that avoids an $(mn)^{\Omega(d)}$ term in the space complexity. As mentioned before, the ray shooting result in [8] suffices in two and three dimensions.

λ-segment query. A set O of objects in $\mathbb{R}^d$ is preprocessed into a data structure so that for any oriented query segment pq, the *λ-segment query with pq on O* returns one of the following answers:

- If pq intersects an object in O, let x be the first intersection point with an object in O as we walk from p to q. In this case, the query returns an object $o \in O$ such that px intersects $o \oplus B_\lambda$. Figure 2 shows an illustration.
- Otherwise, the query returns null or an object $o \in O$ such that $d(o, pq) \leq \lambda$.

We are now ready to state the three constraints on $(\mathcal{A}, \mathcal{B}, \mathcal{C})$.

- **Constraint 1:**

(a) Both $(c_{1,1}, c_{1,2})$ and $(c_{k-1,1}, c_{k-1,2})$ belong to $\mathcal{G}_1 \times \mathcal{G}_1$.

(b) For $j \in [k-1]$, if $(c_{j,1}, c_{j,2}) \neq$ null, then:

(i) $c_{j,1}$ and $c_{j,2}$ are the grid cells returned by the $(11\varepsilon\delta)$-segment queries with w_jw_{j+1} and $w_{j+1}w_j$ on $\mathcal{G}_1$, respectively;

(ii) the minimum point in $w_jw_{j+1} \cap (c_{j,1} \oplus B_{11\varepsilon\delta})$ lies in front of the maximum point in $w_jw_{j+1} \cap (c_{j,2} \oplus B_{11\varepsilon\delta})$ with respect to $\leq_{w_jw_{j+1}}$.

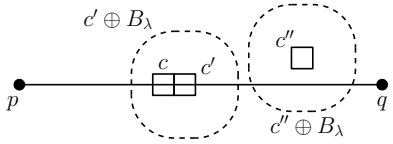


Figure 2 The λ-segment query with pq on the boxes $\{c, c', c''\}$ can return c or c' but not c''.

- **Constraint 2:**
  - **(a)** $\mathcal{B}[1]$ and $\mathcal{A}[k-1]$ belong to $\mathcal{G}_2$.
  - **(b)** $w_1 \in \mathcal{B}[1]$ and $w_k \in \mathcal{A}[k-1]$.
- **Constraint 3:**
  - **(a)** For $j \in [2, k-2]$, if $(c_{j,1}, c_{j,2}) = \text{null}$, then $\mathcal{A}[j]$ and $\mathcal{B}[j]$ are null.
  - **(b)** For $j \in [k-1]$, if $(c_{j,1}, c_{j,2}) \neq \text{null}$, then $\mathcal{A}[j]$ and $\mathcal{B}[j]$ belong to $\mathcal{G}_2$, $d(c_{j,1}, \mathcal{B}[j]) \leq (1+11\varepsilon)\delta$, and $d(c_{j,2}, \mathcal{A}[j]) \leq (1+11\varepsilon)\delta$.
  - **(c)** Let $\mathcal{J}$ be the set of $(r, s) \in [k-1] \times [k-1]$ such that $r < s$, $(c_{r,1}, c_{r,2}) \neq \text{null}$, $(c_{s,1}, c_{s,2}) \neq \text{null}$, and $(c_{j,1}, c_{j,2}) = \text{null}$ for $j \in [r+1, s-1]$. For every $(r, s) \in \mathcal{J}$, let x_r and x_s be the smallest vertices of $\mathcal{A}[r]$ and $\mathcal{B}[s]$ according to the lexicographical order of their coordinates, there exist $x'_r, x'_s \in x_r x_s$ such that $x'_r \leq_{x_r x_s} x'_s$ and $d_F(x'_r x'_s, \sigma[w_{r+1}, w_s]) \leq (1+\varepsilon)\delta$.

We remark that if $w_j w_{j+1}$ intersects the interior of the union of cells in $\mathcal{G}_1$, constraint 1(b)(ii) is satisfied automatically for $(c_{j,1}, c_{j,2})$ given constraint 1(b)(i). When $w_j w_{j+1}$ does not intersect the interior of the union of cells in $\mathcal{G}_1$, it is possible that the $(11\varepsilon\delta)$-segment queries return two cells that violate constraint 1(b)(ii). In this case, the input vertices are too far from $w_j w_{j+1}$ to be matched to any point in $w_j w_{j+1}$ within a distance δ, so we can set $(c_{j,1}, c_{j,2})$ to be null.

The next result shows that any query curve σ near a curve $\tau_i \in T$ has a coarse encoding with some additional properties. These properties will be useful in the analysis. Let $\mathcal{M}$ denote a matching between σ and some $\tau_i \in T$. For any subcurve $\sigma' \subseteq \sigma$, we use $\mathcal{M}(\sigma')$ to denote the subcurve of τ_i matched to σ' by $\mathcal{M}$.

▶ **Lemma 2.** *Let $\sigma = (w_1, \ldots, w_k)$ be a curve of k vertices. Let $\mathcal{M}$ be a matching between σ and $\tau_i \in T$ such that $d_{\mathcal{M}}(\tau_i, \sigma) \leq \delta$. Let $\tilde{\pi}_j = \{v_{i,a} : a \in [m-1], v_{i,a} \in \mathcal{M}(w_j w_{j+1}) \setminus \mathcal{M}(w_j)\}$ for all $j \in [k-1]$. Define $\pi_j = \tilde{\pi}_j$ for all $j \in [k-2]$, $\pi_{k-1} = \{v_{i,m}\} \cup \tilde{\pi}_{k-1}$, and $\pi_0 = \{v_{i,1} \ldots, v_{i,m}\} \setminus \bigcup_{j=1}^{k-1} \pi_j$. There is a coarse encoding $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ for σ that satisfies the following properties.*

- **(i)** *For $j \in [2, k-1]$, $\pi_j = \emptyset$ if and only if $(c_{j,1}, c_{j,2}) = \text{null}$.*
- **(ii)** *For all $(r, s) \in \mathcal{J}$, if $r = 1$ and $\pi_1 = \emptyset$, let $b_1 = 1$; otherwise, let $b_r = \max\{b : v_{i,b} \in \pi_r\}$. For all $(r, s) \in \mathcal{J}$, there exist a point $z \in \mathcal{A}[r] \cap \tau_{i,b_r}$ and another point $z' \in \mathcal{B}[s] \cap \tau_{i,b_r}$ such that $z \leq_{\tau_{i,b_r}} z'$.*

Proof. We define the component $\mathcal{C}$ as follows. Given that $v_{i,1} \in \mathcal{M}(w_1)$, $w_1 w_2$ intersects the interior of the union of cells in $\mathcal{G}_1$, so the $(11\varepsilon\delta)$-segment query with $w_1 w_2$ on $\mathcal{G}_1$ must return some cell; we define it to be $c_{1,1}$. Similarly, the $(11\varepsilon\delta)$-segment query with $w_2 w_1$ on $\mathcal{G}_1$ must return some cell; we define it to be $c_{1,2}$. The pair $(c_{k-1,1}, c_{k-1,2})$ are also defined in a similar way as $v_{i,m} \in \mathcal{M}(w_k)$. Consider any $j \in [2, k-1]$. If $v_{i,a} \in \pi_j$ for some $a \in [m]$, then $v_{i,a} \in \mathcal{M}(w_j w_{j+1})$, which implies that $w_j w_{j+1}$ intersects $v_{i,a} \oplus B_\delta$ and hence the interior of the union of cells in $\mathcal{G}_1$. Thus, $(c_{j,1}, c_{j,2})$ can be defined using the $(11\varepsilon\delta)$-segment queries with $w_j w_{j+1}$ and $w_{j+1} w_j$ as before. On the other hand, if $\pi_j = \emptyset$, we define $(c_{j,1}, c_{j,2})$ to be null. Constraint 1 and property (i) in the lemma are thus satisfied.

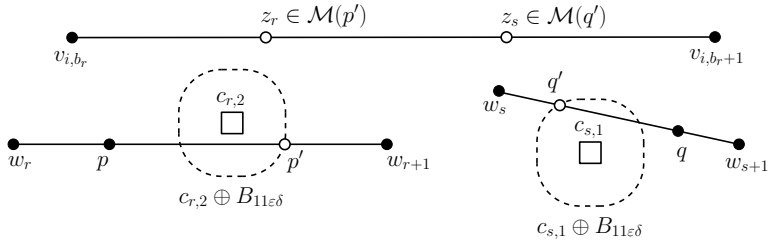


Figure 3 Illustration of p, p', q', q, z_r, and z_s.

Next, we define $\mathcal{A}$ and $\mathcal{B}$ to satisfy constraints 2 and 3.

As $v_{i,1} \in \mathcal{M}(w_1)$ and $v_{i,m} \in \mathcal{M}(w_k)$, both $d(w_1, v_{i,1})$ and $d(w_k, v_{i,m})$ are at most δ. So w_1 lies in a cell in $G(v_{i,1} \oplus B_\delta) \subset G(v_{i,1} \oplus B_{(2+12\varepsilon)\delta}) \subset \mathcal{G}_2$; we make this cell $\mathcal{B}[1]$. Similarly, we define $\mathcal{A}[k-1]$ to be the cell in $\mathcal{G}_2$ that contains w_k. Constraint 2 is thus enforced.

For $j \in [2, k-2]$, if $(c_{j,1}, c_{j,2}) = \text{null}$, let $\mathcal{A}[j]$ and $\mathcal{B}[j]$ be null, satisfying constraint 3(a). $\mathcal{B}[1]$ and $\mathcal{A}[k-1]$ have already been defined, and they belong to $\mathcal{G}_2$. Since w_1 lies in a cell in $G(v_{i,1} \oplus B_\delta) \subset \mathcal{G}_1$, we have $w_1 \in c_{1,1} \oplus B_{11\varepsilon\delta}$ by the $(11\varepsilon\delta)$-segment query. Then, $d(c_{1,1}, \mathcal{B}[1]) \le 11\varepsilon\delta$ as $w_1 \in \mathcal{B}[1]$. Similarly, $d(c_{k-1,2}, \mathcal{A}[k-1]) \le 11\varepsilon\delta$. So $\mathcal{B}[1]$ and $\mathcal{A}[k-1]$ satisfy constraint 3(b). It remains to discuss $\mathcal{A}[j]$ for $j \in [1, k-2]$ and $\mathcal{B}[j]$ for $j \in [2, k-1]$.

Consider an arbitrary $j_* \in [k-1]$ such that $(c_{j_*,1}, c_{j_*,2}) \neq \text{null}$. Recall that $\mathcal{J}$ is the set of $(r, s) \in [k-1] \times [k-1]$ such that $r < s$, $(c_{r,1}, c_{r,2}) \neq \text{null}$, $(c_{s,1}, c_{s,2}) \neq \text{null}$, and $(c_{j,1}, c_{j,2}) = \text{null}$ for $j \in [r+1, s-1]$. Thus, if $j_* \le k-2$, it must exist as the first value in exactly one element of $\mathcal{J}$, and if $j_* \ge 2$, it must also exist as the second value in exactly another element of $\mathcal{J}$. As a result, it suffices to define $\mathcal{A}[r]$ and $\mathcal{B}[s]$ for every $(r, s) \in \mathcal{J}$ and verify that constraints 3(b) and 3(c) are satisfied.

Take any $(r, s) \in \mathcal{J}$. If $\pi_r \neq \emptyset$, it is legal to define $b_r = \max\{b : v_{i,b} \in \pi_r\}$. If $\pi_r = \emptyset$, then $r = 1$ because for any $r > 1$, $\pi_r \neq \text{null}$ by (i) as $(c_{r,1}, c_{r,2}) \neq \text{null}$ by the definition of $\mathcal{J}$. In the case that $r = 1$ and $\pi_1 = \emptyset$, b_1 is defined to be 1. Therefore, b_r is well defined for all $(r, s) \in \mathcal{J}$. The definition of b_r implies that $b_r = \max\{b : v_{i,b} \in \mathcal{M}(w_r w_{r+1})\}$. Since $(c_{s,1}, c_{s,2}) \neq \text{null}$ and $(c_{j,1}, c_{j,2}) = \text{null}$ for $j \in [r+1, s-1]$, by (i), $\pi_s \neq \emptyset$ and $\pi_j = \emptyset$ for $j \in [r+1, s-1]$. It follows that $v_{i,b_r+1} \in \pi_s$ which is a subset of $\mathcal{M}(w_s w_{s+1})$. Pick any point $p \in w_r w_{r+1}$ and any point $q \in w_s w_{s+1}$ such that $v_{i,b_r} \in \mathcal{M}(p)$ and $v_{i,b_r+1} \in \mathcal{M}(q)$.

We claim that $p w_{r+1} \cap (c_{r,2} \oplus B_{11\varepsilon\delta})$ and $w_s q \cap (c_{s,1} \oplus B_{11\varepsilon\delta})$ are non-empty. Since $c_{r,2}$ is the cell in $\mathcal{G}_1$ returned by the $(11\varepsilon\delta)$-segment query with $w_{r+1} w_r$, for any intersection point x between $w_{r+1} w_r$ and any cell in $\mathcal{G}_1$, we have $x w_{r+1} \cap (c_{r,2} \oplus B_{11\varepsilon\delta}) \neq \emptyset$ by definition. We have $p \in w_r w_{r+1} \cap (v_{i,b_r} \oplus B_\delta)$ by our choice of p; it means that p is an intersection point between $w_r w_{r+1}$ and a cell in $G(v_{i,b_r} \oplus B_\delta) \subset \mathcal{G}_1$. We can thus substitute p for x and conclude that $p w_{r+1} \cap (c_{r,2} \oplus B_{11\varepsilon\delta}) \neq \emptyset$. Similarly, we get $w_s q \cap (c_{s,1} \oplus B_{11\varepsilon\delta}) \neq \emptyset$.

By our claim, when we walk from w_{r+1} to p, we hit $c_{r,2} \oplus B_{11\varepsilon\delta}$ at some point p', and when we walk from w_s to q, we hit $c_{s,1} \oplus B_{11\varepsilon\delta}$ at some point q'. Pick two points $z_r \in \mathcal{M}(p')$ and $z_s \in \mathcal{M}(q')$. By definition, $c_{r,2} \in G(v_{i_r,a_r} \oplus B_\delta)$ for some $\tau_{i_r} \in T$ and some index $a_r \in [m]$. The cell width of $c_{r,2}$ is $\varepsilon\delta/\sqrt{d}$, so $c_{r,2} \subset v_{i_r,a_r} \oplus B_{(1+\varepsilon)\delta}$. By triangle inequality, $p' \in v_{i_r,a_r} \oplus B_{(1+12\varepsilon)\delta}$ and hence $z_r \in v_{i_r,a_r} \oplus B_{(2+12\varepsilon)\delta}$, which implies that z_r is contained in a cell in $G(v_{i_r,a_r} \oplus B_{(2+12\varepsilon)\delta}) \subset \mathcal{G}_2$. By a similar reasoning, we can also deduce that z_s is contained in a cell in $\mathcal{G}_2$. We define $\mathcal{A}[r]$ and $\mathcal{B}[s]$ to be the cells in $\mathcal{G}_2$ that contain z_r and z_s, respectively. Figure 3 shows an illustration.

Since $d(c_{r,2}, z_r) \le d(c_{r,2}, p') + d(p', z_r) \le (1 + 11\varepsilon)\delta$, we get $d(c_{r,2}, \mathcal{A}[r]) \le (1 + 11\varepsilon)\delta$. Similarly, $d(c_{s,1}, \mathcal{B}[s]) \le (1 + 11\varepsilon)\delta$. This takes care of constraint 3(b).

As $v_{i,b_r} \in \mathcal{M}(p)$ and $p \leq_{w_r w_{r+1}} p'$, we have $v_{i,b_r} \leq_{\tau_i} \mathcal{M}(p')$. Similarly, we have $\mathcal{M}(q') \leq_{\tau_i} v_{i,b_{r+1}}$. Therefore, $v_{i,b_r} \leq_{\tau_i} \mathcal{M}(p') \leq_{\tau_i} \mathcal{M}(q') \leq_{\tau_i} v_{i,b_{r+1}}$. As $z_r \in \mathcal{M}(p')$ and $z_s \in \mathcal{M}(q')$, z_r and z_s satisfy property (ii) of the lemma. The distance between z_r and any vertex x_r of $\mathcal{A}[r]$ is at most $\varepsilon\delta$. So is the distance between z_s and any vertex x_s of $\mathcal{B}[s]$. Thus, we can use the linear interpolation $\mathcal{I}$ between $x_r x_s$ and $z_r z_s$ as a matching to get $d_{\mathcal{I}}(x_r x_s, z_r z_s) \leq \varepsilon\delta$. Combining $\mathcal{M}$ and $\mathcal{I}$ shows that there is a matching between $\sigma[p', q']$ and $x_r x_s$ within a distance of $(1+\varepsilon)\delta$. Since $\sigma[w_{r+1}, w_s] \subseteq \sigma[p', q']$, we have thus verified constraint 3(c). ◀

2.2 Data structure organization and construction

We construct $\mathcal{G}_1$ and $\mathcal{G}_2$ in $O(mn/\varepsilon^d)$ time and space. We need a point location data structure for $\mathcal{G}_2$ which is organized as a multi-level tree as follows. The top-level tree has leaves corresponding to the intervals of the cells on the first coordinate axis. Each leaf is associated with the cells that project to the interval of that leaf, and these cells are stored in a second-level tree with leaves corresponding to the intervals of these cells on the second coordinate axis. Continuing in this manner yields $d = O(1)$ levels, using $O(|\mathcal{G}_2|) = O(mn/\varepsilon^d)$ space and $O\big((mn/\varepsilon^d)\log\frac{mn}{\varepsilon}\big)$ preprocessing time. A point location takes $O(\log\frac{mn}{\varepsilon})$ time.

The $(1+O(\varepsilon), \delta)$-ANN data structure is a trie $\mathcal{D}$. Each key to be stored in $\mathcal{D}$ is a *candidate* coarse encoding, which is a 3-tuple $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ just like a coarse encoding. For a candidate coarse encoding, constraints 1(a), 2(a), 3(a), and 3(b) must be satisfied, but constraints 1(b), 2(b), and 3(c) are ignored. This difference is necessary because constraints 1(b), 2(b), and 3(c) require the query curve, which is not available in preprocessing. For each candidate coarse encoding E, let T_E be the subset of input curves that are within a Fréchet distance of $(1+O(\varepsilon))\delta$ from any query curve that has E as a coarse encoding, we will discuss shortly how to obtain the curves in T_E.

Each key E in $\mathcal{D}$ has $O(k)$ size because E stores $O(k)$ cells in $\mathcal{G}_1$ and $\mathcal{G}_2$. As a trie, $\mathcal{D}$ is a rooted tree with as many levels as the length of the key E. Searching in D boils down to visiting the appropriate child of an internal node of $\mathcal{D}$. Each component of the key E is an element of $\mathcal{G}_2 \cup \{\text{null}\}$ or $(\mathcal{G}_1 \times \mathcal{G}_1) \cup \{\text{null}\}$; there are $O(m^2n^2/\varepsilon^{2d})$ possibilities. We keep a dictionary at each internal node of $\mathcal{D}$ for finding the appropriate child to visit in $O(\log\frac{mn}{\varepsilon})$ time. Hence, the total search time of $\mathcal{D}$ is $O(k\log\frac{mn}{\varepsilon})$.

To bound the size of $\mathcal{D}$, observe that each key E at a leaf of $\mathcal{D}$ induces $O(k)$ entries in the dictionaries at the ancestors of that leaf. There are $O(\sqrt{d}/\varepsilon)^{4d(k-1)}(mn)^{4(k-1)}$ candidate coarse encodings. So the total space taken by the dictionaries at the internal nodes is $O(\sqrt{d}/\varepsilon)^{4d(k-1)}(mn)^{4(k-1)}k$. We will show that if a query curve has E as a coarse encoding, any curve in T_E is within a Fréchet distance of $(1+O(\varepsilon))\delta$ from that query curve. Therefore, we only need to store one of the curves in T_E at the leaf for E, and it suffices to store the index of this curve. Therefore, the total space complexity of $\mathcal{D}$ is $O(\sqrt{d}/\varepsilon)^{4d(k-1)}(mn)^{4(k-1)}k$.

The construction of $\mathcal{D}$ proceeds as follows. We initialize $\mathcal{D}$ to be empty. We enumerate all possible candidate coarse encodings based on constraints 1(a), 2(a), 3(a), and 3(b). Take a possible candidate coarse encoding E. The set T_E is initialized to be empty. We go through every input curve τ_i to determine whether to include τ_i in T_E. If $T_E \neq \emptyset$ in the end, we insert E together with one curve in T_E into $\mathcal{D}$. In the following, we discuss the checking of whether to include τ_i in T_E.

Let E be $(\mathcal{A}, \mathcal{B}, \mathcal{C})$. We generate all possible *partitions* of $\{v_{i,1}, \ldots, v_{i,m}\}$ that satisfy the following properties.

Partition: a sequence of k *disjoint* subsets $(\pi_0, \pi_1, \ldots, \pi_{k-1})$ such that $\bigcup_{j=0}^{k-1} \pi_j = \{v_{i,1}, \ldots, v_{i,m}\}$, $v_{i,1} \in \pi_0$, $v_{i,m} \in \pi_{k-1}$, π_j may be empty for some $j \in [k-2]$, and for any $v_{i,a} \in \pi_j$ and any $v_{i,b} \in \pi_l$, if $j < l$, then $a < b$.

There are fewer than m^{k-1} partitions. Given a partition $(\pi_0, \ldots, \pi_{k-1})$, the vertices in π_0 are to be matched with $v_{i,1}$; for $j \in [k-1]$, the vertices in π_j are to be matched with points in $w_j w_{j+1} \setminus \{w_j\}$, where $w_j w_{j+1}$ is the j-th edge of the query curve; $v_{i,m}$ and possibly other vertices of τ_i are matched with w_k. The reference to $w_j w_{j+1}$ is conceptual; we do not need to know the query curve in preprocessing.

We describe four tests for each partition below. As soon as we come across a partition that passes all four tests, we insert τ_i into T_E. If a partition fails any test, we move on to the next partition. If no partition can pass all four tests in the end, we do not include τ_i in T_E.

The first test is that for $j \in [2, k-1]$, $\pi_j = \emptyset$ if and only if $(c_{j,1}, c_{j,2}) = \text{null}$. This test takes $O(k)$ time. We exclude π_1 from this test because $(c_{1,1}, c_{1,2}) \neq \text{null}$ by constraint 1(a), whereas π_1 may be empty or not depending on the partition enumerated.

In the second test, for $j \in [k-1]$, if $\pi_j \neq \emptyset$, let $a_j, b_j \in [m]$ be the smallest and largest indices such that $v_{i,a_j}, v_{i,b_j} \in \pi_j$, the intuition is that $v_{i,a_j}, \ldots, v_{i,b_j}$ can be matched to the surrogate $(c_{j,1}, c_{j,2})$ of $w_j w_{j+1}$ within a distance of $(1 + O(\varepsilon))\delta$. The second test checks this property as follows. Observe that $(c_{j,1}, c_{j,2}) \neq \text{null}$: $(c_{1,1}, c_{1,2}) \neq \text{null}$ by constraint 1(a), and for $j \in [2, k-1]$, $(c_{j,1}, c_{j,2}) \neq \text{null}$ by the first test. Pick the smallest vertices x_j of $c_{j,1}$ and y_j of $c_{j,2}$ according to the lexicographical order of their coordinates. If $x_j y_j \cap (v_{i,a_j} \oplus B_{(1+12\varepsilon)\delta})$ or $x_j y_j \cap (v_{i,b_j} \oplus B_{(1+12\varepsilon)\delta})$ is empty, the test fails. Otherwise, compute the minimum point x'_j in $x_j y_j \cap (v_{i,a_j} \oplus B_{(1+12\varepsilon)\delta})$ and the maximum point y'_j in $x_j y_j \cap (v_{i,b_j} \oplus B_{(1+12\varepsilon)\delta})$ with respect to $\leq_{x_j y_j}$. If it is not the case that $x'_j \leq_{x_j y_j} y'_j$, the test fails. Suppose that $x'_j \leq_{x_j y_j} y'_j$. Compute $d_F(x'_j y'_j, \tau_i[v_{i,a_j}, v_{i,b_j}])$ and check whether it is $(1 + 12\varepsilon)\delta$ or less. If all of the above checks succeed for all $j \in [k-1]$, the second test succeeds; otherwise, the test fails. The test takes $O(m \log m)$ time, which is dominated by the computation of $d_F(x'_j y'_j, \tau_i[v_{i,a_j}, v_{i,b_j}])$ over all $j \in [k-1]$.

The third test is that $\mathcal{B}[1] \in G(v_{i,1} \oplus B_\delta)$ and $\mathcal{A}[k-1] \in G(v_{i,m} \oplus B_\delta)$, which boils down to checking whether $d(v_{i,1}, \mathcal{B}[1])$ and $d(v_{i,m}, \mathcal{A}[k-1])$ are at most δ.

The fourth test involves $\mathcal{J}$, the set of $(r, s) \in [k-1] \times [k-1]$ such that $r < s$, $(c_{r,1}, c_{r,2}) \neq \text{null}$, $(c_{s,1}, c_{s,2}) \neq \text{null}$, and $(c_{j,1}, c_{j,2}) = \text{null}$ for $j \in [r+1, s-1]$. Note that $|\mathcal{J}| \leq k-1$ and it can be constructed in $O(k)$ time. For every $(r, s) \in \mathcal{J}$, if $r = 1$ and $\pi_1 = \emptyset$, let $b_1 = 1$; otherwise, let $b_r = \max\{b : v_{i,b} \in \pi_r\}$. It follows that $b_r + 1 = \min\{a : v_{i,a} \in \pi_s\}$. We check if it is the case that $\tau_{i,b_r} \cap \mathcal{A}[r] \neq \emptyset$, $\tau_{i,b_r} \cap \mathcal{B}[s] \neq \emptyset$, and we hit $\mathcal{A}[r]$ no later than $\mathcal{B}[s]$ when we walk from v_{i,b_r} to v_{i,b_r+1}. (Recall the intuition that the pair $\mathcal{A}[r]$ and $\mathcal{B}[s]$ serve as the surrogate of the edge $\tau_{i,b_r} = v_{i,b_r} v_{i,b_r+1}$.) If check fails for any $(r, s) \in \mathcal{J}$, the test fails. Otherwise, the test succeeds. This test runs in $O(k)$ time.

The following result summarizes the construction of $\mathcal{D}$ and four properties of each candidate coarse encoding in $\mathcal{D}$.

▶ **Lemma 3.** *The trie $\mathcal{D}$ has $O(\sqrt{d}/\varepsilon)^{4d(k-1)}(mn)^{4(k-1)}k$ size and can be constructed in $O(\sqrt{d}/\varepsilon)^{4d(k-1)}(mn)^{4(k-1)}(k \log \frac{mn}{\varepsilon} + m^k \log m)$ time. We can search $\mathcal{D}$ with a coarse encoding in $O(k \log \frac{mn}{\varepsilon})$ time. For each candidate coarse encoding $E = (\mathcal{A}, \mathcal{B}, \mathcal{C})$, a curve $\tau_i \in T$ belongs to T_E if and only if there exists a partition $(\pi_0, \ldots, \pi_{k-1})$ of the vertices of τ_i that satisfy the following four properties. For $j \in [k-1]$, if $j = 1$ and $\pi_1 = \emptyset$, let $b_1 = 1$; otherwise, if $\pi_j \neq \emptyset$, let $a_j = \min\{a : v_{i,a} \in \pi_j\}$ and let $b_j = \max\{b : v_{i,b} \in \pi_j\}$.*

- **(i)** *For $j \in [2, k-1]$, $\pi_j = \emptyset$ if and only if $(c_{j,1}, c_{j,2}) = \text{null}$.*
- **(ii)** *For $j \in [k-1]$, if $\pi_j \neq \emptyset$, let x_j and y_j be the smallest vertices of $c_{j,1}$ and $c_{j,2}$ according to the lexicographical order of their coordinates, there exist $x''_j, y''_j \in x_j y_j$ such that $x''_j \leq_{x_j y_j} y''_j$ and $d_F(x''_j y''_j, \tau_i[v_{i,a_j}, v_{i,b_j}]) \leq (1 + 12\varepsilon)\delta$.*
- **(iii)** *$\mathcal{B}[1] \in G(v_{i,1} \oplus B_\delta)$ and $\mathcal{A}[k-1] \in G(v_{i,m} \oplus B_\delta)$.*
- **(iv)** *For every $(r, s) \in \mathcal{J}$, $\tau_{i,b_r} \cap \mathcal{A}[r] \neq \emptyset$, $\tau_{i,b_r} \cap \mathcal{B}[s] \neq \emptyset$, and we hit $\mathcal{A}[r]$ no later than $\mathcal{B}[s]$ when we walk from v_{i,b_r} to v_{i,b_r+1}.*

2.3 Querying

At query time, we are given a curve $\sigma = (w_1, ..., w_k)$. We enumerate all coarse encodings of σ; for each coarse encoding E enumerated, we search the trie $\mathcal{D}$ for E; if E is found, we return the curve in T_E stored with E as the answer of the query; if no coarse encoding of σ can be found in $\mathcal{D}$, we return "no".

Each search in $\mathcal{D}$ takes $O(k \log \frac{mn}{\varepsilon})$ time as stated in Lemma 3. The enumeration of the coarse encodings of σ require a solution for the $(11\varepsilon\delta)$-segment queries on $\mathcal{G}_1$ as stated in constraint 1(b)(i) in Section 2.1. We will discuss an efficient solution later.

For $j \in [k-1]$, we make two $(11\varepsilon\delta)$-segment queries with w_jw_{j+1} and $w_{j+1}w_j$ on $\mathcal{G}_1$ to obtain $u_{j,1}$ and $u_{j,2}$, respectively. If any of the two queries returns null, define $(u_{j,1}, u_{j,2})$ to be null. If $(u_{j,1}, u_{j,2}) \neq$ null and the minimum point in $w_jw_{j+1} \cap (u_{j,1} \oplus B_{11\varepsilon\delta})$ does not lie in front of the maximum point in $w_jw_{j+1} \cap (u_{j,2} \oplus B_{11\varepsilon\delta})$ with respect to $\leq_{w_jw_{j+1}}$, then constraint 1(b)(ii) is not satisfied. It must be the case that w_jw_{j+1} does not intersect the interior of the union of cells in $\mathcal{G}_1$, and the $(11\varepsilon\delta)$-segment queries just happen to return two cells that violate constraint 1(b)(ii). In this case, the input vertices are too far from w_jw_{j+1} to be matched to any point in w_jw_{j+1} within a distance δ, so we reset $(u_{j,1}, u_{j,2})$ to be null.

After defining $(u_{j,1}, u_{j,2})$ for $j \in [k-1]$, we generate the coarse encodings of σ as follows. The pairs $(c_{1,1}, c_{1,2})$ and $(c_{k-1,1}, c_{k-1,2})$ are defined to be $(u_{1,1}, u_{1,2})$ and $(u_{k-1,1}, u_{k-1,2})$, respectively. For $j \in [2, k-2]$, we enumerate all possible $\mathcal{C}$ by setting $(c_{j,1}, c_{j,2})$ to be $(u_{j,1}, u_{j,2})$ or null. This gives a total of 2^{k-3} possible $\mathcal{C}$'s. We query the point location data structure for $\mathcal{G}_2$ to find the cells $\mathcal{B}[1]$ and $\mathcal{A}[k-1]$ that contain w_1 and w_k, respectively. Then, for each $\mathcal{C}$ enumerated, we enumerate $\mathcal{A}[j]$ for $j \in [1, k-2]$ and $\mathcal{B}[j]$ for $j \in [2, k-1]$ according to constraints 3(a) and 3(b) in Section 2.1. This enumeration produces $O(\sqrt{d}/\varepsilon)^{2d(k-2)}$ tuples of $(\mathcal{A}, \mathcal{B}, \mathcal{C})$. For each $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ enumerated, we check whether it satisfies constraint 3(c), which can be done in $O(k \log k)$ time as implied by the following result.

▶ **Lemma 4.** *Take any $(r, s) \in \mathcal{J}$. Let x_r and x_s be the smallest vertices of $\mathcal{A}[r]$ and $\mathcal{B}[s]$ by the lexicographical order of their coordinates. We can check in $O((s-r)\log(s-r))$ time whether there are $x'_r, x'_s \in x_rx_s$ such that $x'_r \leq_{x_rx_s} x'_s$ and $d_F(x'_rx'_s, \sigma[w_{r+1}, w_s]) \leq (1+\varepsilon)\delta$.*

▶ **Lemma 5.** *The query time is $O(kQ_{\text{seg}}) + O(\sqrt{d}/\varepsilon)^{2d(k-2)} k \log \frac{mn}{\varepsilon}$, where Q_{seg} is the time to answer an $(11\varepsilon\delta)$-segment query.*

2.4 Approximation guarantee

First, we show that if σ is within a Fréchet distance δ from some input curve, there exists a coarse encoding E of σ such that $T_E \neq \emptyset$. Hence, E and a curve in T_E are stored in $\mathcal{D}$.

▶ **Lemma 6.** *If $d_F(\tau_i, \sigma) \leq \delta$, then $\tau_i \in T_E$ for some coarse encoding E of σ.*

Proof. Let $\mathcal{M}$ be a Fréchet matching between τ_i and σ. Let E be the coarse encoding specified for σ in Lemma 2. For any subcurve $\sigma' \subseteq \sigma$, we use $\mathcal{M}(\sigma')$ to denote the subcurve of τ_i matched to σ' by $\mathcal{M}$. For $j \in [k-1]$, let $\tilde{\pi}_j = \{v_{i,a} : a \in [m-1], v_{i,a} \in \mathcal{M}(w_jw_{j+1}) \setminus \mathcal{M}(w_j)\}$. Define $\pi_j = \tilde{\pi}_j$ for $j \in [k-2]$, $\pi_{k-1} = \{v_{i,m}\} \cup \tilde{\pi}_{k-1}$, and $\pi_0 = \{v_{i,1}, \ldots, v_{i,m}\} \setminus \bigcup_{j=1}^{k-1} \pi_j$. We obtain a partition $(\pi_0, ..., \pi_{k-1})$ of the vertices of τ_i.

We prove that E, τ_i, and $(\pi_0, ..., \pi_{k-1})$ satisfy Lemma 3(i)–(iv) which put τ_i in T_E. Lemma 3(i) follows directly from Lemma 2(i),

Take any $j \in [k-1]$ such that $\pi_j \neq \emptyset$. Let π_j be $\{v_{i,a}, v_{i,a+1}, \ldots, v_{i,b}\}$. By the definition of π_j, every vertex in π_j belongs to $\mathcal{M}(w_jw_{j+1})$, so $\tau_i[v_{i,a}, v_{i,b}] \subset \mathcal{M}(w_jw_{j+1})$. Then, there must exist two points $p, q \in w_jw_{j+1}$ such that $p \leq_{w_jw_{j+1}} q$ and $d_F(pq, \tau_i[v_{i,a}, v_{i,b}]) \leq \delta$. If $j = 1$, we

have $(c_{1,1}, c_{1,2}) \neq$ null by constraint 1(a); if $j \in [2, k-1]$, by Lemma 2(i), $(c_{j,1}, c_{j,2}) \neq$ null as $\pi_j \neq$ null. Therefore, $c_{j,1}$ and $c_{j,2}$ are cells in $\mathcal{G}_1$ returned by the $(11\varepsilon\delta)$-segment queries with $w_j w_{j+1}$ and $w_{j+1} w_j$, respectively. We have shown that $d_F(pq, \tau_i[v_{i,a}, v_{i,b}]) \leq \delta$; therefore, p is contained in a cell in $G(v_{i,a} \oplus B_\delta) \subset \mathcal{G}_1$. As $c_{j,1}$ is the cell returned by the $(11\varepsilon\delta)$-segment query with $w_j w_{j+1}$, there must be a point $z_p \in w_j w_{j+1} \cap (c_{j,1} \oplus B_{11\varepsilon\delta})$ such that $z_p \leq_{w_j w_{j+1}} p$. In a similar way, we can conclude that there must be a point $z_q \in w_j w_{j+1} \cap (c_{j,2} \oplus B_{11\varepsilon\delta})$ such that $q \leq_{w_j w_{j+1}} z_q$. That is, $z_p \leq_{w_j w_{j+1}} p \leq_{w_j w_{j+1}} q \leq_{w_j w_{j+1}} z_q$. Let x_j and y_j be the smallest vertices of $c_{j,1}$ and $c_{j,2}$ according to the lexicographical order of their coordinates. Both $d(z_p, x_j)$ and $d(z_q, y_j)$ are at most $12\varepsilon\delta$. A linear interpolation from $z_p z_q$ to $x_j y_j$ maps p and q to two points x''_j and y''_j on $x_j y_j$, respectively, such that $x''_j \leq_{x_j y_j} y''_j$. Also, the linear interpolation adds a distance $12\varepsilon\delta$ or less, which gives $d_F(x''_j y''_j, \tau_i[v_{i,a}, v_{i,b}]) \leq d_F(x''_j y''_j, pq) + d_F(pq, \tau_i[v_{i,a}, v_{i,b}]) \leq (1 + 12\varepsilon)\delta$. Hence, Lemma 3(ii) is satisfied.

The grid cells $\mathcal{B}[1]$ and $\mathcal{A}[k-1]$ are defined to contain w_1 and w_k, respectively. Also, $v_{i,1} \in \mathcal{M}(w_1)$ and $v_{i,m} \in \mathcal{M}(w_k)$. Hence, $\mathcal{B}[1] \in G(v_{i,1} \oplus B_\delta)$ and $\mathcal{A}[k-1] \in G(v_{i,m} \oplus B_\delta)$, satisfying Lemma 3(iii).

For any pair $(r, s) \in \mathcal{J}$, by Lemma 2(ii), there exist two points $z \in \mathcal{A}[r] \cap \tau_{i,b_r}$ and $z' \in \mathcal{B}[s] \cap \tau_{i,b_r}$ such that $z \leq_{\tau_{i,b_r}} z'$. Since $\mathcal{A}[r]$ and $\mathcal{B}[s]$ are interior-disjoint unless they are equal, we must hit $\mathcal{A}[r]$ no later than $\mathcal{B}[s]$ when we walk from v_{i,b_r} to v_{i,b_r+1}, satisfying Lemma 3(iv). ◀

We show that if E is a coarse encoding of σ, each curve in T_E is close to σ.

▶ **Lemma 7.** *Let E be a coarse encoding of σ. For every $\tau_i \in T_E$, $d_F(\tau_i, \sigma) \leq (1 + 24\varepsilon)\delta$.*

Proof. (Sketch) Suppose that $T_E \neq \emptyset$ as the lemma statement is vacuous otherwise. Take any $\tau_i \in T_E$. We construct a matching $\mathcal{M}$ between τ_i and σ such that $d_{\mathcal{M}}(\tau_i, \sigma) \leq (1+24\varepsilon)\delta$. Since $T_E \neq \emptyset$, there exists a partition $(\pi_0, \ldots, \pi_{k-1})$ of the vertices of τ_i that satisfy Lemma 3(i)–(iv). Using these properties, we can match the vertices of τ_i to points on σ and then the vertices of σ to points on τ_i. Afterwards, σ and τ_i divided into line segments by their vertices and images of their matching partners. We use linear interpolations to match the corresponding line segments. More details can be found in the appendix. ◀

In two and three dimensions, the ray shooting data structure for boxes in [8] can be used as the $(11\varepsilon\delta)$-segment query data structure. It has an $O(\log |\mathcal{G}_1|) = O(\log \frac{mn}{\varepsilon})$ query time and an $O(|\mathcal{G}_1|^{2+\mu}) = O((mn)^{2+\mu}/\varepsilon^{d(2+\mu)})$ space and preprocessing time for any fixed $\mu \in (0, 1)$. If the query segment does not intersect any cell in $\mathcal{G}_1$, we return null. In four and higher dimensions, we will prove the following result in Section 4.

▶ **Lemma 8.** *We can construct a data structure in $O(\sqrt{d}/\varepsilon)^{O(d/\varepsilon^2)} \cdot \tilde{O}((mn)^{O(1/\varepsilon^2)})$ space and preprocessing time such that given any oriented edge e of the query curve σ, the data structure either discovers that $\min_{\tau_i \in T} d_F(\sigma, \tau_i) > \delta$, or reports a correct answer for the $(11\varepsilon\delta)$-segment query with e on $\mathcal{G}_1$. The query time is $\tilde{O}((mn)^{0.5+\varepsilon}/\varepsilon^d)$.*

Combining the results in this section with the ray shooting result in [8], Lemma 8, and Theorem 1 gives $(1+\varepsilon)$-ANN data structures. Theorem 1 uses the deletion cost of a $(1+\varepsilon, \delta)$-ANN data structure. We perform each deletion by reconstructing the data structure from scratch because we do not have a more efficient solution.

▶ **Theorem 9.** *For any $\varepsilon \in (0, 0.5)$, there is a $(1 + O(\varepsilon))$-ANN data structure for T under the Fréchet distance with the following performance guarantees:*

- $d \in \{2,3\}$:
  query time $= O\big(\frac{1}{\varepsilon}\big)^{2d(k-2)} k \log \frac{mn}{\varepsilon} \log n$,
  space $= O\big(\frac{1}{\varepsilon}\big)^{4d(k-1)+1} (mn)^{4(k-1)} k \log^2 n$,
  expected preprocessing time $= O\big(\frac{1}{\varepsilon}\big)^{4d(k-1)} (mn)^{4(k-1)} \big(k \log \frac{mn}{\varepsilon} + m^k \log m\big) n \log n$.
- $d \geq 4$:
  query time $= \tilde{O}\big(\frac{1}{\varepsilon^d} k (mn)^{0.5+\varepsilon}\big) + O\big(\frac{\sqrt{d}}{\varepsilon}\big)^{2d(k-2)} k \log \frac{mn}{\varepsilon} \log n$,
  space $= O\big(\frac{\sqrt{d}}{\varepsilon}\big)^{4d(k-1)} (mn)^{4(k-1)} k \cdot \frac{1}{\varepsilon} \log^2 n + O\big(\frac{\sqrt{d}}{\varepsilon}\big)^{O(d/\varepsilon^2)} \cdot \tilde{O}((mn)^{O(1/\varepsilon^2)})$,
  expected preprocessing time $= O\big(\frac{\sqrt{d}}{\varepsilon}\big)^{4d(k-1)} (mn)^{4(k-1)} \big(k \log \frac{mn}{\varepsilon} + m^k \log m\big) n \log n +$
  $O\big(\frac{\sqrt{d}}{\varepsilon}\big)^{O(d/\varepsilon^2)} \cdot \tilde{O}((mn)^{O(1/\varepsilon^2)})$.

3 $(3 + O(\varepsilon), \delta)$-ANN

Given a query curve $\sigma = (w_1, w_2, \ldots, w_k)$, for $j \in [k-1]$, we solve the $(11\varepsilon\delta)$-segment queries with $w_j w_{j+1}$ and $w_{j+1} w_j$ on $\mathcal{G}_1$ as before. Let $((c_{j,1}, c_{j,2}))_{j \in [k-1]}$ denote the results of the queries. Recall that each $(c_{j,1}, c_{j,2})$ belongs to $(\mathcal{G}_1 \times \mathcal{G}_1) \cup \{\text{null}\}$.

Suppose that there are $k_0 \leq k-1$ non-null pairs in $((c_{j,1}, c_{j,2}))_{j \in [k-1]}$. Extract these non-null pairs to form the sequence $((c_{j_r,1}, c_{j_r,2}))_{r \in [k_0]}$. Note that $j_1 = 1$ and $j_{k_0} = k-1$ by constraint 1(a). We construct a polygonal curve σ_0 by connecting the centers of $c_{j_r,1}$ and $c_{j_r,2}$ for $r \in [k_0]$ and the centers of $c_{j_r,2}$ and $c_{j_{r+1},1}$ for $r \in [k_0 - 1]$. The polygonal curve σ_0 acts as a surrogate of σ. It has at most $2k-2$ vertices. We will use σ_0 as the key to search a trie at query time to obtain an answer for a $(3 + O(\varepsilon), \delta)$-ANN query. As a result, no enumeration is needed which avoids the exponential dependence of the query time on k.

In preprocessing, we enumerate all sequences of $2l$ cells in $\mathcal{G}_1$ for $l \in [2, k-1]$. For each sequence, we construct the polygonal curve σ' that connects the centers of the cells in the sequence, and we find the nearest input curve τ_i to σ'. If $d_F(\sigma', \tau_i) \leq (1 + 12\varepsilon)\delta$, we store (σ', i) in a trie $\mathcal{D}$. There are $O(\sqrt{d}/\varepsilon)^{2d(k-1)} (mn)^{2(k-1)}$ entries in $\mathcal{D}$. We organize the trie $\mathcal{D}$ in the same way as described in Section 2.2. The space required by $\mathcal{D}$ is $O(\sqrt{d}/\varepsilon)^{2d(k-1)} (mn)^{2(k-1)} k$. The search time of $\mathcal{D}$ is $O(k \log \frac{mn}{\varepsilon})$. The preprocessing time is $O(\sqrt{d}/\varepsilon)^{2d(k-1)} (mn)^{2(k-1)} (k \log \frac{mn}{\varepsilon} + kmn \log(km)) = O(\sqrt{d}/\varepsilon)^{2d(k-1)} (mn)^{2k-1} k \log \frac{mn}{\varepsilon}$ due to the computation of the nearest input curve for each sequence enumerated.

At query time, we construct σ_0 from σ in $O(kQ_{\text{seg}})$ time, where Q_{seg} is the time to answer a $(11\varepsilon\delta)$-segment query. We compute $d_F(\sigma, \sigma_0)$ in $O(k^2 \log k)$ time. If $d_F(\sigma, \sigma_0) > (2 + 12\varepsilon)\delta$, we report "no". Otherwise, we search $\mathcal{D}$ with σ_0 in $O(k \log \frac{mn}{\varepsilon})$ time. If the search fails, we report "no". Otherwise, the search returns (σ_0, i) for some $i \in [n]$.

▶ **Lemma 10.** *If $d_F(\sigma, \sigma_0) \leq (2 + 12\varepsilon)\delta$ and the search in $\mathcal{D}$ with σ_0 returns (σ_0, i), then $d_F(\sigma, \tau_i) \leq (3 + 24\varepsilon)\delta$. Otherwise, $\min_{\tau_i \in T} d_F(\sigma, \tau_i) > \delta$.*

Combining the results in this section with the ray shooting results in two and three dimensions [8], Lemma 8, and Theorem 1, we obtain the following theorem.

▶ **Theorem 11.** *For any $\varepsilon \in (0, 0.5)$, there is a $(3 + O(\varepsilon))$-ANN data structure for T under the Fréchet distance with the following performance guarantees:*

- $d \in \{2,3\}$:
  query time $= O(k \log \frac{mn}{\varepsilon} \log n)$,
  space $= O\big(\frac{1}{\varepsilon}\big)^{2d(k-1)+1} (mn)^{2(k-1)} k \log^2 n$,
  expected preprocessing time $= O\big(\frac{1}{\varepsilon}\big)^{2d(k-1)} (mn)^{2k-1} kn \log \frac{mn}{\varepsilon} \log n$.

- $d \geq 4$:
  query time $= \tilde{O}\big(\frac{1}{\varepsilon^d} k(mn)^{0.5+\varepsilon}\big)$,
  space $= O\big(\frac{\sqrt{d}}{\varepsilon}\big)^{2d(k-1)}(mn)^{2(k-1)}k \cdot \frac{1}{\varepsilon}\log^2 n + O\big(\frac{\sqrt{d}}{\varepsilon}\big)^{O(d/\varepsilon^2)} \cdot \tilde{O}((mn)^{O(1/\varepsilon^2)})$,
  expected preprocessing time $= O\big(\frac{\sqrt{d}}{\varepsilon}\big)^{2d(k-1)}(mn)^{2k-1}kn \log \frac{mn}{\varepsilon} \log n +$
  $O\big(\frac{\sqrt{d}}{\varepsilon}\big)^{O(d/\varepsilon^2)} \cdot \tilde{O}((mn)^{O(1/\varepsilon^2)})$.

4 ($11\varepsilon\delta$)-segment queries and proof of Lemma 8

We describe the $(11\varepsilon\delta)$-segment query data structure in Lemma 8. We first present the main ideas before giving the details. Let $w_j w_{j+1}$ be a query segment, which is unknown at preprocessing. There are three building blocks.

First, the intuition is to capture the support lines of all possible query segments using pairs of cells in $\mathcal{G}_1$. It would be ideal to retrieve a pair of cells intersected by $\text{aff}(w_j w_{j+1})$, but this seems to be as difficult as the ray shooting problem. For a technical reason, we need to use more grid cells in a larger neighborhood of the input vertices than in $\mathcal{G}_1$, so define $\mathcal{G}_3 = \bigcup_{i\in[n],a\in[m]} G(v_{i,a} \oplus B_{(1+6\varepsilon)\delta})$.

We find a grid vertex x of $\mathcal{G}_1$ that is a $(1+\varepsilon)$-approximate nearest grid vertex to $\text{aff}(w_j w_{j+1})$. We will show that if $d(x, \text{aff}(w_j w_{j+1})) > (1+\varepsilon)\varepsilon\delta$, the answer to the $(11\varepsilon\delta)$-segment query is null; otherwise, we can find a cell $\gamma \in \mathcal{G}_3$ near x that intersects $\text{aff}(w_j w_{j+1})$. We can use any other cell $c \in \mathcal{G}_1$ to form a pair with γ that acts as a surrogate for the support lines of query segments that pass near c and γ.

Second, given $w_j w_{j+1}$ at query time, among all possible choices of c, we need to find the right one(s) efficiently so that (c, γ) is a surrogate for $\text{aff}(w_j w_{j+1})$. We explain the ideas using the case that w_{j+1} lies between w_j and $\text{aff}(w_j w_{j+1}) \cap \gamma$. Note that w_j may not be near any cell in $\mathcal{G}_1$. In order that $\min_{\tau_i \in T} d(\sigma, \tau_i) \leq \delta$, w_j must be within a distance δ from some input edge $\tau_{i,a}$. We find a maximal packing of $\text{aff}(\tau_{i,a}) \oplus B_{O(\delta)}$ using lines that are parallel to $\tau_{i,a}$ and are at distance $\Theta(\varepsilon\delta)$ or more apart. There are $O(\varepsilon^{1-d})$ lines in the packing, and every point in $\text{aff}(\tau_{i,a}) \oplus B_\delta$ is within a distance $O(\varepsilon\delta)$ from some line in the packing. The projection of w_j to the approximately nearest line approximates the location of w_j. Hence, we should seek to divide the lines in the packing into appropriate segments so that, given w_j and its approximately nearest line in the packing, we can efficiently find the segment that contains the projection of w_j and retrieve some precomputed information for that segment.

Third, let ℓ be a line in the packing mentioned above, for each possible cell $c \in \mathcal{G}_1$, we use the geometric construct $F(c, \gamma) = \{x \in \mathbb{R}^d : \exists\ y \in \gamma \text{ s.t. } xy \cap c \neq \emptyset\}$ defined in [6] which can be computed in $O(1)$ time. The projection of $(\ell \oplus B_{2\varepsilon\delta}) \cap F(c, \gamma)$ in ℓ is the set of points on ℓ such that if the projection of w_j is in it, then (c, γ) is a surrogate for $\text{aff}(w_j w_{j+1})$. As a result, the endpoints of the projections of $(\ell \oplus B_{2\varepsilon\delta}) \cap F(c, \gamma)$ over all possible choices of c divide ℓ into segments that we desire. Each segment may stand for several choices of c's. For each segment, we store the cell c' close to that segment because the ideal choice is the cell that we hit first as we walk from w_j to w_{j+1}.

As described above, we use two approximate nearest neighbor data structures that involve lines. The first one is due to Andoni et al. [4] which stores a set of points P such that given a query line, the $(1+\varepsilon)$-approximate nearest point to the query line can be returned in $\tilde{O}(d^3|P|^{0.5+\varepsilon})$ time. It uses $\tilde{O}\big(d^2|P|^{O(1/\varepsilon^2)}\big)$ space and preprocessing time. The second result is due to Agarwal et al. [2] which stores a set L of lines such that given a query point, the 2-approximate nearest line to the query point can be returned in $\tilde{O}(1)$ time. It uses $\tilde{O}(|L|^2)$ space and expected preprocessing time.

4.1 Data structure organization

We restrict ε to be chosen from $(0, 0.5)$. We construct the data structure of Andoni et al. [4] for the grid vertices of $\mathcal{G}_1$ so that for any query line, the $(1+\varepsilon)$-approximate nearest grid vertex can be returned in $\tilde{O}((mn)^{0.5+\varepsilon}/\varepsilon^d)$ time. We denote this data structure by D_{anp}. It takes $O(\sqrt{d}/\varepsilon)^{O(d/\varepsilon^2)} \cdot \tilde{O}((mn)^{O(1/\varepsilon^2)})$ space and preprocessing time.

For each input edge $\tau_{i,a}$, define a set of lines $L_{i,a}$ as follows. Let H be the hyperplane through $v_{i,a}$ orthogonal to $\text{aff}(\tau_{i,a})$. Take a $(d-1)$-dimensional grid in H with $v_{i,a}$ as a grid vertex and cell width $\varepsilon\delta/\sqrt{d-1}$. The set $L_{i,a}$ includes every line that is orthogonal to H and passes through a vertex of this grid in H at distance within $(1+2\varepsilon)\delta$ from $v_{i,a}$. The set $L_{i,a}$ has $O(\varepsilon^{1-d})$ size, and it can be constructed in $O(\varepsilon^{1-d})$ time. Moreover, every point in the cylinder $\text{aff}(\tau_{i,a}) \oplus B_\delta$ is within a distance $\varepsilon\delta$ from some line in $L_{i,a}$.

Define $\mathcal{L} = \bigcup_{i\in[n],a\in[m-1]} L_{i,a}$. The size of $\mathcal{L}$ is $O(mn/\varepsilon^{d-1})$, and $\mathcal{L}$ can be constructed in $O(mn/\varepsilon^{d-1})$ time. We construct the data structure of Agarwal et al. [2] for $\mathcal{L}$ so that for any query point, a 2-approximate nearest line in $\mathcal{L}$ can be returned in $\tilde{O}(1)$ time. We denote this data structure by D_{anl}. It uses $\tilde{O}((mn)^2/\varepsilon^{2d-2})$ space and expected preprocessing time.

Recall that $\mathcal{G}_3 = \bigcup_{i\in[n],a\in[m]} G(v_{i,a} \oplus B_{(1+6\varepsilon)\delta})$.

For every $\gamma \in \mathcal{G}_3$ and every $c \in \mathcal{G}_1$, we construct $F(c,\gamma) = \{x \in \mathbb{R}^d : \exists\, y \in \gamma \text{ s.t. } xy \cap c \neq \emptyset\}$, which is empty or an unbounded convex polytope of $O(1)$ size that can be constructed in $O(1)$ time as a Minkowski sum [6]. The total time needed is $O((mn)^2/\varepsilon^{2d})$.

For every $\gamma \in \mathcal{G}_3$, every $c \in \mathcal{G}_1$, and every line $\ell \in \mathcal{L}$, compute the intersection $(\ell \oplus B_{2\varepsilon\delta}) \cap F(c,\gamma)$ and project it orthogonally to a segment in ℓ. Take any line $\ell \in \mathcal{L}$. The resulting segment endpoints in ℓ divide ℓ into *canonical segments*. There are $O((mn)^2/\varepsilon^{2d})$ canonical segments in ℓ. For every cell $\gamma \in \mathcal{G}_3$ and every canonical segment $\xi \subseteq \ell$, compute the set $C_{\gamma,\xi}$ of every cell $c \in \mathcal{G}_1$ such that ξ is contained in the projection of $(\ell \oplus B_{2\varepsilon\delta}) \cap F(c,\gamma)$ onto ℓ. Fix an arbitrary point in ξ and denote it by p_ξ. Each $C_{\gamma,\xi}$ has $O(mn/\varepsilon^d)$ size. The total time needed over all cells in $\mathcal{G}_3$ and all canonical segments in all lines in $\mathcal{L}$ is $\tilde{O}((mn)^5/\varepsilon^{5d-1})$.

Let p_γ be the center of the cell γ. Define $c_{\gamma,\xi}$ to be the cell in $C_{\gamma,\xi}$ such that $p_\xi p_\gamma \cap (c_{\gamma,\xi} \oplus B_{5\varepsilon\delta})$ is nearest to p_ξ among $\{p_\xi p_\gamma \cap (c \oplus B_{5\varepsilon\delta}) : c \in C_{\gamma,\xi}\}$. The total time to compute $c_{\gamma,\xi}$ over all cells in $\mathcal{G}_3$ and all canonical segments in all lines in $\mathcal{L}$ is $O((mn)^5/\varepsilon^{5d-1})$.

Finally, for every line $\ell \in \mathcal{L}$, we store the canonical segments in ℓ in an interval tree T_ℓ [7]. It uses linear space and preprocessing time. For any query point in ℓ, one can search T_ℓ in $O(\log \frac{mn}{\varepsilon})$ time to find the canonical segment in ℓ that contains the query point. For each canonical segment ξ stored in T_ℓ, we keep a dictionary T_ξ that stores the set $\{(\gamma, c_{\gamma,\xi}) : \gamma \in \mathcal{G}_3\}$ with γ as the key. For any cell $\gamma \in \mathcal{G}_3$, we can search T_ξ in $O(\log \frac{mn}{\varepsilon})$ time to report $c_{\gamma,\xi}$. These interval trees and dictionaries have a total size of $O((mn)^4/\varepsilon^{4d-1})$, and they can be constructed in $\tilde{O}((mn)^4/\varepsilon^{4d-1})$ time.

The data structures D_{anp}, D_{anl}, T_ℓ for $\ell \in \mathcal{L}$, and T_ξ for all canonical segments ξ's are what we need to support the $(11\varepsilon\delta)$-segment queries on $\mathcal{G}_1$.

▶ **Lemma 12.** *We can construct D_{anp}, D_{anl}, T_ℓ for $\ell \in \mathcal{L}$, and T_ξ for every $\ell \in \mathcal{L}$ and every canonical segment $\xi \subset \ell$ in $O(\sqrt{d}/\varepsilon)^{O(d/\varepsilon^2)} \cdot \tilde{O}((mn)^{O(1/\varepsilon^2)})$ space and preprocessing time.*

In the definition of $c_{\gamma,\xi}$, one may ask what if $p_\xi p_\gamma$ does not intersect $c \oplus B_{5\varepsilon\delta}$ for some $c \in C_{\gamma,\xi}$. We prove that this cannot happen. We also establish some other properties.

▶ **Lemma 13.** *Let γ be a cell in $\mathcal{G}_3$. Let ξ be a canonical segment. Let L_ξ be the cylinder with ξ as the axis and radius $2\varepsilon\delta$.*

(i) *For every cell $c \in \mathcal{G}_1$, if $c \cap xy \neq \emptyset$ for some points $x \in L_\xi$ and $y \in \gamma$, then $c \in C_{\gamma,\xi}$.*

(ii) *For every point $x \in L_\xi$, every point $y \in \gamma$ and every cell $c \in C_{\gamma,\xi}$, $xy \cap (c \oplus B_{5\varepsilon\delta}) \neq \emptyset$.*

(iii) *Let λ be any value greater than or equal to $11\varepsilon\delta$. When we walk from a point $x \in L_\xi$ to a point $y \in \gamma$, we cannot hit any $c \in C_{\gamma,\xi}$ earlier than $c_{\gamma,\xi} \oplus B_\lambda$ irrespective of the choices of x and y.*

4.2 Answering a query

Given an oriented segment w_jw_{j+1} of the query curve σ, we answer the $(11\varepsilon\delta)$-segment query with w_jw_{j+1} on $\mathcal{G}_1$ by the following steps.

- Step 1: We query D_{anp} with $\text{aff}(w_jw_{j+1})$ to report a grid vertex x of $\mathcal{G}_1$. This takes $\tilde{O}((mn)^{0.5+\varepsilon}/\varepsilon^d)$ time.

- Step 2: We check the distance $d(x, \text{aff}(w_jw_{j+1}))$. If $d(x, \text{aff}(w_jw_{j+1})) > (1+\varepsilon)\varepsilon\delta$, then $\text{aff}(w_jw_{j+1})$ is at distance more than $\varepsilon\delta$ from the closest grid vertex of $\mathcal{G}_1$, which implies that $\text{aff}(w_jw_{j+1})$ does not intersect any cell in $\mathcal{G}_1$. In this case, we report null.
  We also check the distances $d(w_j, \mathcal{L})$ and $d(w_{j+1}, \mathcal{L})$. We query D_{anl} with w_j in $\tilde{O}(1)$ time to find a line $\ell_j \in \mathcal{L}$. If $d(w_j, \ell_j) > 2\varepsilon\delta$, then $d(w_j, \mathcal{L}) > \varepsilon\delta$, which implies that w_j is at distance farther than δ from $\text{aff}(\tau_{i,a})$ for any $\tau_i \in T$ and any $a \in [m-1]$. As a result, $d_F(\sigma, \tau_i) > \delta$ for all $\tau_i \in T$, so we report "no" for the (κ, δ)-ANN query. Analogously, we query D_{anl} with w_{j+1} in $\tilde{O}(1)$ time to find a line $\ell_{j+1} \in \mathcal{L}$. If $d(w_{j+1}, \ell_{j+1}) > 2\varepsilon\delta$, we report "no" for the (κ, δ)-ANN query.

- Step 3: Suppose that $d(x, \text{aff}(w_jw_{j+1})) \leq (1+\varepsilon)\varepsilon\delta$, $d(w_j, \ell_j) \leq 2\varepsilon\delta$, and $d(w_{j+1}, \ell_{j+1}) \leq 2\varepsilon\delta$. Then, we check the cells in $G(x \oplus B_{2\varepsilon\delta})$ in $O(\varepsilon^{-d})$ time to find one that intersects $\text{aff}(w_jw_{j+1})$. Let γ be this cell. We do not know if γ belongs to $\mathcal{G}_1$ or not. Nevertheless, since x is a grid vertex of $\mathcal{G}_1$, γ is within a distance $(1+3\varepsilon)\delta$ from some input curve vertex. Therefore, γ must be a cell in $\mathcal{G}_3$. There are three cases depending on the relative positions of w_j and γ.

  - Step 3(a): $w_j \in \gamma \cap \text{aff}(w_jw_{j+1})$. We claim that $\mathcal{G}_1 \cap G(w_j \oplus B_{7\varepsilon\delta})$ is non-empty, and we report an arbitrary cell in it as the answer for the $(11\varepsilon\delta)$-segment query. This step takes $O(\varepsilon^{-d})$ time.
  - Step 3(b): w_j precedes $\gamma \cap \text{aff}(w_jw_{j+1})$ along $\text{aff}(w_jw_{j+1})$ oriented from w_j to w_{j+1}. We query T_{ℓ_j} to find the canonical segment $\xi \subset \ell_j$ that contains the projection of w_j in ℓ_j. Then, we query T_ξ with γ to return $c_{\gamma,\xi}$ as the answer for the $(11\varepsilon\delta)$-segment query. The time needed is $O(\log \frac{mn}{\varepsilon})$.
  - Step 3(c): $\gamma \cap \text{aff}(w_jw_{j+1})$ precedes w_j along $\text{aff}(w_jw_{j+1})$ oriented from w_j to w_{j+1}. We query $T_{\ell_{j+1}}$ to find the canonical segment $\xi \subset \ell_{j+1}$ that contains the projection of w_{j+1} in ℓ_{j+1}. Then, we query T_ξ with γ to obtain $c_{\gamma,\xi}$. We claim that $G(c_{\gamma,\xi} \oplus B_{5\varepsilon\delta}) \subset \mathcal{G}_3$ and some cell in $G(c_{\gamma,\xi} \oplus B_{5\varepsilon\delta})$ intersects w_jw_{j+1}. Pick one such cell $\hat{\gamma}$ in $O(\varepsilon^{-d})$ time. Either step 3(a) or 3(b) is applicable with γ replaced by $\hat{\gamma}$. Whichever case is applicable, we jump to that case with γ replaced by $\hat{\gamma}$ to return an answer for the $(11\varepsilon\delta)$-segment query. The time needed is $\tilde{O}(\varepsilon^{-d})$.

▶ **Lemma 14.** *It takes $\tilde{O}((mn)^{0.5+\varepsilon}/\varepsilon^d)$ time to answer a $(11\varepsilon\delta)$-segment query.*

Lemmas 12 and 14 gives the performance of the $(11\varepsilon\delta)$-segment query data structure in Lemma 8. The proof of the query output correctness in Lemma 8 can be found in the full version.

5 Conclusion

We present $(1+\varepsilon)$-ANN and $(3+\varepsilon)$-ANN data structures that achieve sublinear query times without having space complexities that are proportion to $\min\{m^{\Omega(d)}, n^{\Omega(d)}\}$ or exponential in $\min\{m, n\}$. The query times are $\tilde{O}(k(mn)^{0.5+\varepsilon}/\varepsilon^{O(d)} + k(d/\varepsilon)^{O(dk)})$ for $(1+\varepsilon)$-ANN and $\tilde{O}(k(mn)^{0.5+\varepsilon}/\varepsilon^{O(d)})$ for $(3+\varepsilon)$-ANN. In two and three dimensions, the query times can be improved to $\tilde{O}(k/\varepsilon^{O(k)})$ for $(1+\varepsilon)$-ANN and $\tilde{O}(k)$ for $(3+\varepsilon)$-ANN. It is an open problem is to lower the exponential dependence on d and k.

References

1 P.K. Agarwal and J. Matoušek. Ray shooting and parametric search. *SIAM Journal on Computing*, 22(4):794–806, 1993.

2 P.K. Agarwal, N. Rubin, and M. Sharir. Approximate nearest neighbor search amid higher-dimensional flats. In *Proceedings of the European Symposium on Algorithms*, pages 4:1–4:13, 2017.

3 H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry and Applications*, 5:75–91, 1995.

4 A. Andoni, P. Indyk, R. Krauthgamer, and H.L. Nguyen. Approximate line nearest neighbor in high dimensions. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 293–301, 2009.

5 K. Bringmann, A. Driemel, A. Nusser, and I. Psarros. Tight bounds for approximate near neighbor searching for time series under Fréchet distance. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 517–550, 2022.

6 S.-W. Cheng and H. Huang. Curve simplification and clustering under Fréchet distance. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 1414–1432, 2023.

7 T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, second edition, 2001.

8 M. de Berg, D. Halperin, M. Overmars, J. Snoeyink, and M. van Kreveld. Efficient ray shooting and hidden surface removal. *Algorithmica*, 12:30–53, 1994.

9 A. Driemel and I. Psarros. $(2+\epsilon)$-ANN for time series under the Fréchet distance. *arXiv preprint v5*, 2021. `arXiv:2008.09406`.

10 A. Driemel and F. Silvestri. Locality-sensitive hashing of curves. In *Proceedings of the International Symposium on Computational Geometry*, pages 37:1–37:16, 2017.

11 T. Eiter and H. Mannila. Computing discrete Fréchet distance. Technical Report CD-TR 94/64, Christian Doppler Laboratory for Expert Systems, TU Vienna, Austria, 1994.

12 I.Z. Emiris and I. Psarros. Products of Euclidean metrics, applied to proximity problems among curves: Unified treatment of discrete fréchet and dynamic time warping distances. *ACM Transactions on Spatial Algorithms and Systems*, 6(4):1–20, 2020.

13 A. Filtser, O. Filtser, and M.J. Katz. Approximate nearest neighbor for curves: simple, efficient, and deterministic. In *Proceedings of the International Colloquium on Automata, Languages, and Programming*, pages 48:1–48:19, 2020.

14 S. Har-Peled. A replacement for Voronoi diagrams of near linear size. In *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science*, pages 94–103, 2001.

15 S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbor: towards removing the curse of dimensionality. *Theory of Computing*, 8:321–350, 2012.

16 P. Indyk. Approximate nearest neighbor algorithms for Fréchet distance via product metrics. In *Proceedings of the Annual Symposium on Computational Geometry*, pages 102–106, 2002.

17 P. Indyk and R. Motwani. Approximate nearest neighbor: towards removing the curse of dimensionality. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 604–613, 1998.

18 Mirzanezhad M. On the approximate nearest neighbor queries among curves under the Fréchet distance. *arXiv preprint*, 2020. `arXiv:2004.08444`.

19 C. Shahabi, M. Kolahdouzan, and M. Sharifzadeh. A road network embedding technique for k-nearest neighbor search in moving object databases. *GeoInformatica*, 7:255–273, 2003.

20 Z. Song and N. Roussopoulos. K-nearest neighbor search for moving query point. In *Proceedings of the International Symposium on Spatial and Temporal Databases*, pages 79–96, 2001.

21 Y. Tao and D. Papadias. Parameterized queries in spatio-temporal databases. In *Proceedings of ACM International Conference on Management of Data*, pages 334–345, 2002.

Linear Insertion Deletion Codes in the High-Noise and High-Rate Regimes

Kuan Cheng
Department of Computer Science, Peking University, Beijing, China

Zhengzhong Jin
Massachusetts Institute of Technology, Cambridge, MA, USA

Xin Li
Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA

Zhide Wei
Department of Computer Science, Peking University, Beijing, China

Yu Zheng
Meta Platforms Inc

Abstract

This work continues the study of linear error correcting codes against adversarial insertion deletion errors (insdel errors). Previously, the work of Cheng, Guruswami, Haeupler, and Li [6] showed the existence of asymptotically good linear insdel codes that can correct arbitrarily close to 1 fraction of errors over some constant size alphabet, or achieve rate arbitrarily close to 1/2 even over the binary alphabet. As shown in [6], these bounds are also the best possible. However, known explicit constructions in [6], and subsequent improved constructions by Con, Shpilka, and Tamo [9] all fall short of meeting these bounds. Over any constant size alphabet, they can only achieve rate $< 1/8$ or correct $< 1/4$ fraction of errors; over the binary alphabet, they can only achieve rate $< 1/1216$ or correct $< 1/54$ fraction of errors. Apparently, previous techniques face inherent barriers to achieve rate better than 1/4 or correct more than 1/2 fraction of errors.

In this work we give new constructions of such codes that meet these bounds, namely, asymptotically good linear insdel codes that can correct arbitrarily close to 1 fraction of errors over some constant size alphabet, and binary asymptotically good linear insdel codes that can achieve rate arbitrarily close to 1/2. All our constructions are efficiently encodable and decodable. Our constructions are based on a novel approach of code concatenation, which embeds the index information implicitly into codewords. This significantly differs from previous techniques and may be of independent interest. Finally, we also prove the existence of linear concatenated insdel codes with parameters that match random linear codes, and propose a conjecture about linear insdel codes.

2012 ACM Subject Classification Theory of computation → Error-correcting codes

Keywords and phrases Error correcting code, Edit distance, Pseudorandomness, Derandomization

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.41

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2303.17370

Funding *Kuan Cheng*: Supported by a start-up fund of Peking University.
Zhengzhong Jin: Supported in part by DARPA under Agreement No. HR00112020023 and by an NSF grant CNS-2154149. Most of the work was done while the author was a PhD student at Johns Hopkins University, and supported by NSF CAREER Award CCF-1845349.
Xin Li: Supported by NSF CAREER Award CCF-1845349 and NSF Award CCF-2127575.
Zhide Wei: Supported by a start-up fund of Peking University.
Yu Zheng: Most of the work was done while the author was a PhD student at Johns Hopkins University, and partially supported by NSF Award CCF-2127575.

1 Introduction

Error correcting codes are fundamental objects in computer science and information theory. Starting from the seminal work of Shannon and Hamming, the study of error correcting codes has led to a deep understanding of how to ensure reliable communications in various noisy channels. Furthermore, error correcting codes have found rich applications in other seemingly unrelated areas such as complexity theory, learning theory, pseudorandomness and many more. Traditionally, the errors studied are either erasures (where a transmitted symbol is replaced by a '?') or symbol modifications (where a transmitted symbol is replaced by a different symbol), and they can be either random or adversarial. Through decades of effort, we now have an almost complete understanding of codes for such errors, and constructions with efficient encoding and decoding algorithms that match or are close to various known bounds.

An important and more general type of errors, known as *synchronization errors*, however, is much less understood. These errors include insertion and deletions (so we also call them insdel errors for short), which can cause the positions of received symbols to shift. On the other hand, they occur frequently in real world applications, including disk access, integrated circuits, communication networks and so on. They are also closely related to applications in computational biology and DNA-based storage systems [3, 25]. Although the study of codes for such errors started around the same time as Shannon's works, progress has historically been slow due to the apparent difficulty of handling the loss of index information with such errors. For example, many basic questions, such as the capacity of the binary deletion channel with deletion probability p is still wide open, and the first explicit construction that has a constant rate and can correct a constant fraction of adversarial errors is not known until 1999 [21].

From now on, we will focus exclusively on adversarial insdel errors. Over the past several years, with the development of new techniques such as *synchronization strings* [17], there has been a wave of new constructions of codes for these errors [17, 16, 22, 8, 4, 15, 19, 14, 5, 7, 18, 12, 20, 24, 14]. Some of them achieve excellent parameters, e.g., codes that approach the singleton bound over a large constant alphabet [17], codes with almost optimal redundancy over the binary alphabet [8, 15], list-decodable codes over large alphabets that can correct more errors than the length of the codeword [18], and list-decodable codes over any alphabet of positive rate for the information-theoretically largest possible combination of insertions and deletions [12, 20, 24, 14]. However, none of the above constructions gives a *linear* code, and the existence of asymptotically good linear codes for insdel errors over a constant size alphabet is not known until the work of Cheng, Guruswami, Haeupler, and Li [6].

The motivation of studying linear codes comes from several aspects. First, they have compact representations using either generator matrices or parity check matrices, which directly give efficient encoding and testing algorithms with running time $O(n^2)$. Second, such codes have simple structures, so they are often easier to analyze and allow one to use powerful techniques from linear algebra. Finally, linear codes have had great success in codes for erasures and symbol modifications, achieving some of the most well known constructions with (near) optimal parameters. Thus, one could ask if the same is true for insdel codes.

As is standard in the literature of error correcting codes, the two most important parameters of a linear insdel code are δ, the fraction of insdel errors the code can correct; and R, the rate of the code, defined as the message length divided by the codeword length. In [6], the authors established several bounds regarding the tradeoff between these two parameters for linear insdel codes. First, they showed that any linear code correcting δ fraction of insdel errors must have rate at most $\frac{1}{2}(1-\delta)$, regardless of the alphabet size. This is known as the *half-singleton bound* and generalizes a previous result in [1], which shows that any linear code

that can correct even a single deletion must have a rate of at most 1/2. This bound shows a severe limitation of linear codes for insdel errors, as general codes can correct δ fraction of errors with R approaching $1-\delta$. Taking into consideration the alphabet size q, this bound can be improved to $\frac{1}{2}(1-\frac{q}{q-1}\delta)+o(1)$, which is known as the *half-Plotkin bound*. On the other hand, the authors also showed that over the field F_q, for any $\delta>0$ there exists a linear code family that can correct δ fraction of insdel errors, with rate $(1-\delta)/2-\mathsf{H}(\delta)/\log_2 q$, where H is the binary entropy function. In particular, this implies the existence of binary linear codes with rate $1/2-\varepsilon$ capable of correcting $\Omega(\varepsilon\log^{-1}\frac{1}{\varepsilon})$ fraction of insdel errors for any $\varepsilon>0$; and linear insdel codes over F_q of rate $\frac{1}{2}(1-\delta)-\varepsilon$ capable of correcting any δ-fraction of insdel errors, for a large enough $q=2^{\Theta(\varepsilon^{-1})}$, which approaches the half-singleton bound. Hence, the rate can approach 1/2 even over the binary alphabet, and the fraction of errors corrected can approach 1 over a constant size alphabet, both of which are the best possible.

Going further, [6] also constructed explicit asymptotically good linear insdel codes. However, the fraction of errors the code can correct and the rate of the code are both quite small. [6] did not specify these constants, but a rough estimate shows that the code has $\delta<1/400$ and $R<2^{-80}$. Thus a natural question left in their work is to improve these parameters.

Recently, a subsequent work by Con, Shpilka, and Tamo [9] made progress in this direction. For a field F_q with $q=\mathsf{poly}(1/\varepsilon)$, they constructed explicit linear insdel codes that can correct δ fraction of errors with rate $R=(1-4\delta)/8-\varepsilon$. For the field F_2 their explicit linear code can correct δ fraction of errors with rate $R=(1-54\delta)/1216$. Hence, for a constant size alphabet their construction can achieve $\delta<1/4$ with a positive R, or $R<1/8$ with a positive δ. For the binary alphabet, their construction can achieve $\delta<1/54$ with a positive R, or $R<1/1216$ with a positive δ. One caveat is that their codes over the binary alphabet can only decode efficiently from deletions (although they can also decode from insertions inefficiently), while their codes over the large alphabet can decode efficiently from both deletions and insertions. In another work by the same authors [10], they also showed the existence of Reed-Solomon codes over a field of size $n^{O(k)}$ that have message length k, codeword length n, and can correct $n-2k+1$ insdel errors. This achieves the half-singleton bound. They complemented the existential result by providing a deterministic construction over a field of size $n^{k^{O(k)}}$, which runs in polynomial time for $k=O(\log n/\log\log n)$. Nevertheless, in this paper we only focus on the case of a constant alphabet size.

In summary, all known explicit constructions over constant size alphabets fall short of getting rate close to 1/2, or getting the fraction of errors correctable close to 1. In fact, previous techniques seem to face inherent barriers to achieve rate better than 1/4 or correct more than 1/2 fraction of errors, which we will talk about in more details when we give an overview of our techniques.

1.1 Our Results

In this paper we further improve the fraction of errors δ and the rate R that can be achieved by linear insdel codes with efficient encoding and decoding algorithms. In the case of high noise, we give explicit constructions of insdel codes with positive rate that can correct δ fraction of errors with δ arbitrarily close to 1, over a constant size alphabet. In the case of high rate, we give explicit constructions of insdel codes that can achieve rate arbitrarily close to 1/2 and correct a positive constant fraction of errors, over the binary alphabet.[1]

[1] It's also easy to generalize our constructions to larger alphabet size, but for clarity we omit the details in this version.

Specifically, we have the following theorems.

▶ **Theorem 1** (High noise). *For any constant $\varepsilon > 0$ there exists an efficient construction of linear insdel codes over an alphabet of size* $\mathsf{poly}(1/\varepsilon)$*, with rate $\Omega(\varepsilon^2)$ that can correct $1-\varepsilon$ fraction of insdel errors (possibly inefficiently).*

With efficient decoding, the rate becomes slightly worse.

▶ **Theorem 2** (High noise). *For any constant $\varepsilon > 0$, there is a family of linear codes with rate $\Omega(\varepsilon^4)$ and alphabet size* $\mathsf{poly}(1/\varepsilon)$*, that can be encoded in polynomial time and decoded from up to $1-\varepsilon$ fraction of insdel errors in polynomial time.*

▶ **Theorem 3** (High rate). *For any constant $\varepsilon > 0$, there is a family of binary linear codes with rate $1/2-\varepsilon$, that can be encoded in polynomial time and decoded from $\Omega(\varepsilon^3 \log^{-1} \frac{1}{\varepsilon})$ fraction of insdel errors in polynomial time.*

Our constructions are based on code concatenation. We complement our explicit constructions by showing that there exist linear concatenated codes that match the parameters of random linear codes. These constructions can be considered in a sense "semi-explicit" since the outer code is explicit, and we only need to find explicit inner codes.

▶ **Theorem 4.** *For any field F_{q_0} and any constant $\delta > 0$, there exists a family of linear concatenated code over F_{q_0} where the outer code is a Reed-Solomon code, such that the code has rate $\frac{1}{2}(1-\delta) - \mathsf{H}(\delta)/\log q_0 - o(1)$ and can correct δ fraction of insdel errors, where $\mathsf{H}()$ is the binary entropy function.*

We emphasize that the inner codes here may be different for different positions. So if one wants to use brute force to search for a sequence of proper inner codes, then this may take time at least $2^{n \log^2 n}$ where n is the length of the outer codewords.

This theorem implies the following corollaries.

▶ **Corollary 5.** *For any constant $\delta > 0$, there exists a family of binary linear concatenated code where the outer code is a Reed-Solomon code, such that the code has rate $\frac{1}{2}(1-\delta)$ and can correct $\Omega(\delta \log^{-1} \frac{1}{\delta})$ fraction of insdel errors.*

▶ **Corollary 6.** *For any constants $\delta, \varepsilon > 0$ there exists a family of linear concatenated code over an alphabet of size $q = 2^{\Theta(\varepsilon^{-1})}$ where the outer code is a Reed-Solomon code, such that the code has rate $\frac{1}{2}(1-\delta) - \epsilon$ and can correct δ fraction of insdel errors.*

Finally, we study the question of whether binary linear insdel codes can achieve δ arbitrarily close to $1/2$ with a positive rate R. Notice that even for general binary codes, it is well known that the maximum fraction of deletions that any non-trivial binary code of size $\geqslant 3$ can correct is below $1/2$ since any 3 different n-bit binary strings must contain two strings with the same majority bit, and thus their longest common subsequence is at least $n/2$. For binary linear codes this can also be seen from the half-Plokin bound. A recent work by Guruswami, He, and Li [13] in fact already provided a negative answer to this question even for general binary codes. In particular, they showed that there exists an absolute constant $\alpha > 0$ such that any binary code $\mathbb{C} \subseteq \{0,1\}^n$ with $|\mathbb{C}| \geqslant 2^{\mathsf{polylog} n}$ must have two strings whose longest common subsequence has length at least $(1/2+\alpha)n$. Thus $\mathbb{C}$ cannot correct more than $1/2-\alpha$ fraction of insdel errors. Since linear codes are more restricted, one may expect that a stronger result can be proved for binary linear codes. Specifically, we have the following conjecture:

▶ **Conjecture 7.** *There exists an absolute constant $\alpha > 0$ such that any linear subspace $\mathbb{C} \subseteq \mathsf{F}_2^n$ with dimension $\geqslant 3$ must have two strings (vectors) whose longest common subsequence has length at least $(1/2 + \alpha)n$.*

However, we are not able to prove this conjecture. Instead, we can prove a weaker result.

▶ **Theorem 8.** *There exists an absolute constant $\alpha > 0$ such that any linear subspace $\mathbb{C} \subseteq \mathsf{F}_2^n$ with dimension $\geqslant 3$ must have two strings (vectors) whose longest common subsequence has length at least $(\frac{1}{2} + \frac{\alpha}{\log n})n$.*

1.2 Overview of the Techniques

There have been only two previous works on explicit constructions of asymptotically good linear insdel codes over fields of constant size, i.e., [6] and [9]. The apparent difficulty of constructing such codes comes from the following aspects: First, many of the previous constructions of (non-linear) insdel codes are based on adding index information to the codewords, either in the form of direct encoding of indices, or more sophisticated objects such as synchronization strings. Since all of these result in fixed strings, adding such information in any naive way will lead to non-linear codes. Indeed, both [6] and [9] have to find alternative ways to "embed" synchronization strings into a linear code. Specifically, [6] uses what is called a *synchronization sequence*, which is a sequence of 0's added in between each pair of adjacent symbols in a codeword. This preserves the linearity if the original code is linear. [9], on the other hand, embeds the synchronization string by combining a codeword symbol x and a synchronization string symbol a into a pair $(x, a \cdot x)$, where $\cdot$ is the multiplication over the corresponding field F_q. This also preserves the linearity over F_q, but now the symbols from the synchronization strings are mixed with symbols from the codeword, and it is not easy to tell them apart. Note that for decoding, one needs to first use the synchronization string to recover the positions of the codeword symbols. To solve this problem, [9] also needs to add buffers of 0's between adjacent pairs, where the length of a buffer is at least as long as the pair $(x, a \cdot x)$.

It can be seen that the added 0's in the above two approaches form an inherent barrier to achieving high rate or high fraction of correctable errors. In [6], a constant number of 0's are added in between each pair of adjacent symbols in a codeword, which already decreases the rate and the possible decoding radius to a small constant. In [9], the operation of converting a codeword symbol x and a synchronization string symbol a into a pair $(x, a \cdot x)$ already decreases the rate of the code to below $1/2$, while adding 0's as buffers decreases the rate even more to below $1/4$. Similarly, add 0's as buffers also decreases the possible decoding radius to below $1/2$. For binary codes, [9] needs to use another layer of code concatenation, which further decreases the rate and decoding radius.

The key idea in all our constructions is to eliminate the use of 0's as buffers or synchronization sequences. Instead, we embed synchronization information directly into the codewords. To achieve this, we also use code concatenation, where for the outer code we choose a suitable Reed-Solomon code. On the other hand, the key difference between our constructions and standard concatenated codes is that we choose a *different* inner code for *every position* of the outer code. This way, we can make sure that the inner codewords corresponding to outer codeword symbols at different positions are far enough from each other, and thus we can roughly tell them apart by just looking at the received codeword. By using linear inner codes for all positions, this preserves the linearity of the code, and at the same time eliminates the use of 0's. On a high level, this is why our constructions can achieve either high rate (arbitrarily close to $1/2$) or high fraction of correctable errors (arbitrarily close to 1). We now discuss our techniques in more details for the two cases.

Constructions for high error. Note that to correct $1-\varepsilon$ fraction of insdel errors, a linear code must have alphbet size at least $1/\varepsilon$ by the half-Plotkin bound. Here we use an alphabet of size $\mathsf{poly}(1/\varepsilon)$. With an appropriately chosen parameter $\gamma = \Omega(\varepsilon)$, after picking an outer Reed-Solomon code with codeword length n, rate γ and relative distance $1-\gamma$, our strategy is to design n different inner codes $\mathbb{C}^1_{\mathsf{in}}, \cdots, \mathbb{C}^n_{\mathsf{in}}$. The goal is to ensure that codewords in different inner codes have large edit distance, or equivalently, the length of their longest common subsequence (LCS for short) is at most $\gamma n'$ where $n' = O(\log n)$ is the block length of the inner code. However, since all these codes are linear, 0 is a codeword of each inner code, and two 0's (even from different inner codes) are guaranteed to have 0 edit distance. We design the inner codes to ensure this is the only bad case.

More specifically, we ensure that for any two inner codewords x, y, unless they are both 0 or they correspond to the same message in one inner code $\mathbb{C}^i_{\mathsf{in}}$, their edit distance is large. We show that if we pick n random linear codes for $\mathbb{C}^1_{\mathsf{in}}, \cdots, \mathbb{C}^n_{\mathsf{in}}$, then this property holds with high probability. Furthermore, we can derandomize this by using a small biased sample space to generate the n generator matrices of $\mathbb{C}^1_{\mathsf{in}}, \cdots, \mathbb{C}^n_{\mathsf{in}}$. Roughly, this is because the property we want is *local* – it only looks at any two inner codewords x, y. By using a small biased sample space, we can show that (roughly) under the above conditions, any non-trivial parity of the bits (we treat a symbol in the alphabet of size $\mathsf{poly}(1/\varepsilon)$ as a binary string of length $O(\log(1/\varepsilon))$) of (x, y) has a small bias. Hence a standard XOR lemma implies the joint distribution of (x, y) is close to uniform. Since $n' = O(\log n)$, we only need to look at $\mathsf{poly}(n)$ such pairs of (x, y). Thus it suffices to choose the error in the small biased sample space to be $1/\mathsf{poly}(n)$. This gives us a sample space of size $\mathsf{poly}(n)$ and we can exhaustively search for a good construction. This gives us n different inner codes with rate $\Omega(\gamma)$.

Using these inner codes, it is now relatively straightforward to argue about the parameters of the concatenated code. The rate is $\Omega(\gamma^2) = \Omega(\varepsilon^2)$. To argue about the distance, we consider the LCS between any two different codewords C_1, C_2, and divide it sequentially into blocks according to the inner codewords of C_1. Each block now covers a substring of C_2. Intuitively, by the property of our inner codes, each block contains only a small number of matches compared to the total size of this block in C_1 and the substring covered in C_2, unless it is a 0 inner codeword in C_1 and is matched to another 0 inner codeword in C_2, or it is a match between the same inner codeword in a single inner code $\mathbb{C}^i_{\mathsf{in}}$. However our outer code guarantees that the latter cannot happen too many times (i.e., at most $O(\gamma n)$ times). Therefore the LCS has length at most $O(\gamma n n')$. By choosing γ appropriately, the code can correct $1-\varepsilon$ fraction of insdel errors.

We present a simple polynomial time decoding algorithm. Given any received string y, we consider the partition of y into n substrings $y_1, \cdots, y_n$ such that $y = y_1 \circ y_2 \circ \cdots \circ y_n$, where each y_i can be the empty string. For each y_i, we find the closest codeword $x_i \in \mathbb{C}^i_{\mathsf{in}}$ in edit distance and record their edit distance Δ_i. We then minimize $\Delta = \sum_{i\in[n]} \Delta_i$, by using a simple dynamic programming. We show that as long as there are not too many errors, by using the optimal partition returned from the dynamic programming, one can correctly recover a small fraction of the outer codewords. Intuitively, this is because if the partition results in too many errors in the recovered outer codewords, then again by the property of our inner codes, the quantity Δ will be very large, unless there are a lot of errors. We then use a list decoding algorithm for the Reed-Solomon code to get a list of candidate codewords, and search the list to find the correct codeword, which is the one closest to y in edit distance. For technical reasons, this decreases the rate of the code to $\Omega(\varepsilon^4)$.

Constructions with high rate. Now we explain our construction with high rate and polynomial time encoding and decoding. We first exhibit a warm-up construction achieving rate $1/3 - \gamma$. Then we improve the rate to $1/2 - \gamma$, while the construction will be significantly more involved due to additional issues arising in the analysis.

Inheriting the structure of the general construction, our first construction is as the following. The outer code is a Reed-Solomon code with block length n, alphabet size n, relative distance δ and rate $(1 - \delta)$, where $\delta = \gamma/2$. To achieve a high rate, we will design the inner codes to have a large rate, ideally close to $1/2$. At the same time, we also need to ensure that the code can correct a positive constant fraction of errors, thus we want to make sure that the LCS between any two different codewords is not too large.

As before, we will design the inner codes such that ideally, codewords from different inner codes are far away from each other (or equivalently, have small LCS). However, there are additional issues in the analysis of the LCS. First, the 0 codewords from different inner codes are always the same. This is inevitable since we are dealing with linear codes. Second, in a matching between two different codewords C_1, C_2, some inner codeword of C_1 may be matched to a substring of the concatenation of two adjacent inner codewords of C_2. Thus it is not enough to just ensure that codewords from different inner codes are far away from each other. We note that this issue also occurs in our constructions for high noise. However, there we designed the inner codes to have small rate but large distance, so the LCS between different inner codewords is quite small. When some inner codeword of C_1 is matched to a substring of the concatenation of two adjacent codewords of C_2, the size of the matching in this part at most doubles the size of the LCS between two different inner codewords, and is affordable in that case. Here however, since we are trying to achieve a high rate, the distance between two different inner codewords becomes quite small, and the LCS becomes relatively large (e.g., larger than $1/2$ fraction). Hence, we cannot afford to double this size.

On a high level, we resolve the second issue by strengthening our local property of inner codes, while our analysis will show that the first issue can also be resolved as a consequence. We begin by discussing the local property we need to achieve rate $1/3 - \gamma$. The distinct binary inner codes $\mathbb{C}^1_{\mathsf{in}}, \mathbb{C}^2_{\mathsf{in}}, \ldots, \mathbb{C}^n_{\mathsf{in}}$ are constructed to have block length n', message length $k' = (1/3 - \gamma/2)n'$, with the following property: for every $i, j \in [n]$, for every codeword w in $\mathbb{C}^i_{\mathsf{in}}$, for every two codewords u, v from two adjacent inner codes $\mathbb{C}^j_{\mathsf{in}}, \mathbb{C}^{j+1}_{\mathsf{in}}$, unless $w = u$ or $w = v$, the distance between w and any substring of $u \circ v$ is at least $d' = \Omega(n')$. We first explain why this property implies a good decoding radius and then explain how to construct these inner codes.

We show the decoding radius by directly providing the following decoding algorithm. On an input y which is a corrupted version of a codeword z, the algorithm first finds a string $\tilde{z} \in \{0,1\}^{nn'}$ which has a maximum block matching with y. A block matching is defined to be a set of matches where each match, denoted as $(i, [\alpha, \beta])$, consists of a non-zero inner codeword $u \in \mathbb{C}^i_{\mathsf{in}}$ and a substring $y_{[\alpha,\beta]}$, such that their edit distance is at most $d'/2$. Furthermore, the matching is monotone in the sense that the substrings of y involved in the matching do not overlap and the matches cannot cross. We call u a candidate string for the i-th block. We give a simple dynamic programming algorithm to find a maximum block matching together with a corresponding sequence of candidates. To construct $\tilde{z}$, we first fill these candidates to their corresponding blocks and then set all the other blocks to be 0.

Now we show that as long as there are at most $\rho nn'$ errors for some small constant $\rho > 0$, $\tilde{z}$ agrees with z in most of the blocks (inner codewords). To show this, divide z into blocks $z^1 \circ z^2 \circ \cdots \circ z^n$ such that each z^i corresponds to an inner codeword. Similarly, divide y into blocks $y^1 \circ y^2 \circ \cdots \circ y^n$ such that each y^i is the corrupted version of z^i. Notice that there can

be at most $\frac{\rho n n'}{d'/2} = (c\gamma)n$ blocks with at least $d'/2$ errors, for some constant $c = c(\rho)$. So the maximum block matching has size at least $\hat{n} - c\gamma n$ where $\hat{n}$ is the number of non-zero blocks in z. Now consider a maximum block matching and the sequence of candidates returned by the algorithm. We show that there are at most $c\gamma n$ candidates that are not equal to the corresponding blocks of z, by using the local property. As we fill all the other blocks to be 0, this also implies there are at most $c\gamma n$ zero-blocks being incorrectly recovered. Hence the algorithm correctly recovers $1 - O(c\gamma)$ fraction of blocks in z. By taking c (and thus also ρ) to be a small enough constant, one can use the list-decoding algorithm of Reed-Solomon codes to recover z.

Next, we explain how to construct the inner codes. We start by considering a random construction, that is, all the inner codes are independent random linear codes. We show the local property holds with high probability. Consider arbitrary codewords $w \in \mathbb{C}_{\mathsf{in}}^i \backslash \{0\}, u \in \mathbb{C}_{\mathsf{in}}^j, v \in \mathbb{C}_{\mathsf{in}}^{j+1}$ for some $i, j \in [n]$, where $w \neq u$ and $w \neq v$. Here the inequality means the two codewords are either from different inner codes or they correspond to different messages in one inner code. Suppose there is a substring w' of $u \circ v$, which has distance $< d'$ to w. So the LCS between w and w' should be $\ell \geqslant \frac{|w|+|w'|-d'}{2}$. Notice that $\ell \leqslant |w| \leqslant n'$. Consider any monotone alignment between w and w'. Because $w \neq u, w \neq v$ and the inner codes are all independent and generated randomly, by a similar argument as in [6], the event that the alignment is indeed a matching of bits happens with probability at most $2^{-\ell}$. We then apply a union bound over all possible alignments of size ℓ and all possible codewords w, u, v. A key observation is that the number of all possible codewords w, u, v is $2^{3k'}$ since we have three different codewords here. However, we have $\ell \leqslant n'$. Therefore for the union bound to work, we have to set $k' < n'/3$. This is the reason that we can only achieve rate close to $1/3$ with this construction.

Next, we derandomize the construction by replacing the uniform randomness used with an ε-biased distribution. Here, as before, we crucially use the fact that our property for the inner codes is local: the only place where we use randomness is when we bound the probability that an alignment is a valid matching, and it only involves three codewords. Since $n' = O(\log n)$, by using a standard XOR Lemma and taking $\varepsilon = 1/\mathsf{poly}(n)$, we can argue that when restricted to any three codewords, the ε-biased distribution is $1/\mathsf{poly}(n)$ close to the uniform distribution in statistical distance. This is enough for the union bound since there are at most $\mathsf{poly}(n)$ such triples w, u, v.

Since we only need $O(\log n)$ random bits to generate the above ε-biased distribution, one can exhaustively search for a good construction that satisfies our local property. This also takes polynomial time since one only needs to check every triple of inner codewords.

In our improved construction, we add new ideas to bypass the rate $1/3$ barrier in the above construction, by giving a new local property of the inner codes. Recall that the reason we need to choose $k' < n'/3$ in the above construction is that the alignment we consider in the local property consists of matches that involve three different codewords, which results in a $2^{3k'}$ term in the union bound, but the alignment has size at most n'. In the new local property, we generalize this by considering alignments that involve $2s + 1$ different codewords for some integer s. In a simplified version, consider any two different codewords C_1, C_2 of the concatenated codes and an LCS between them, we analyze any s consecutive inner codewords in C_1, and how they can be matched to a substring in C_2. Note that the s consecutive inner codewords cannot be matched to a substring with length much larger than sn', or there are already many unmatched bits in C_2. So we can imagine a new local property like the following: let w be the concatenation of any s adjacent inner codewords, and u be the concatenation of any $s + 1$ adjacent inner codewords. As long as the codewords in w and

u are sufficiently different, the distance between w and any substring of u is at least $\Omega(n')$. The idea is that an alignment between w and u can have size up to sn', while the union bound gives a $2^{(2s+1)k'}$ term. Thus we can potentially achieve $k' < \frac{s}{2s+1}n'$, and if s is large enough, the rate is close to $1/2$. Note that the warm-up construction i corresponds to the case of $s = 1$.

However, it is not straightforward to make this idea work. The main issue is that unlike the simple case of $s = 1$, when we consider s consecutive inner codewords for $s > 1$, there can be multiple 0 codewords in them, which can potentially be matched to the 0 codewords in u. Furthermore, there can be inner codewords in w and u that correspond to the same message in a single inner code. These issues will increase the probability that the alignment is a valid matching and can cause the union bound to fail. To fix this, we require the "unique" blocks in w to be dense. Specifically, we define a unique block of w (or u) to be a non-zero inner codeword such that either no block of u (or w) is in the same inner code with it, or any block of u (or w) in the same inner code with it corresponds to a different message. Now we define the following new local property:

For every w which is a sequence of $t = O(\frac{\log \frac{1}{\gamma}}{\gamma^2})$ consecutive inner codewords, every u which is a sequence of $t + 1$ consecutive inner codewords, and every w' which is a substring of u, the distance between w and w' is at least $d' = \Omega(\gamma n')$, as long as the number of unique blocks in w or u is at least $s = \Omega(\gamma t)$. By the distance property of the outer code, for any two different concatenated codewords, in at least one of them, the fraction of such t consecutive inner codewords with at least s unique blocks is a constant.

Using this new property, we can design a similar decoding algorithm as that of the first construction, and with a similar analysis, achieve decoding radius $\Omega(\gamma^3 n/\log \frac{1}{\gamma})$.

We defer these details to the technical part and mainly explain here how to construct the inner codes with the new property and why this indeed gives a rate of $1/2 - \gamma$.

Similar to before, we start with a construction where all inner codes are independent random linear codes, and later derandomize it with an ε-biased space. As long as the parameters s, t are constants, it is easy to see that the derandomization step still works. Therefore, now we only focus on the random construction and argue that the new local property holds with high probability. For this, we use a delicate combinatorial and probabilistic argument.

Suppose the property is not satisfied with some concatenated codewords C_1, C_2. Then there exists a w' such that the edit distance between w, w' is less than d', which implies the LCS between w and w' is $\ell > (|w| + |w'| - d')/2$. Consider an arbitrary monotone alignment M between w and w' of size ℓ. We have two cases. The first case is that there is a pair of indices (i, j) in M such that $|i - j| \geqslant d'$. This implies that there cannot be any pair of indices (i', j') in M such that $i' = j'$, for otherwise there are already at least d' bits in C_1 or C_2 that are not matched. Let $\hat{t}$ be the larger number of non-zero blocks in w and w'. Note that $\hat{t} \geqslant s$. Since all inner codes are independent and random, and every pair of indices (i, j) in M has $i \neq j$, the probability that M is a matching is at most $2^{-((\hat{t}-1)n'-O(d'))}$ (w' can have length as small as $(t + 1)n - n - d'$). Now if we apply the union bound, the main term is actually the total number of possible tuples of the non-zero inner codewords. Since there are at most $2\hat{t}$ non zero blocks in w and u, this number is at most $2^{2\hat{t}k'}$. Thus as long as s is a large enough integer, the rate of the code can approach $1/2$.

The second case is that every pair of indices (i, j) in M has $|i - j| < d'$. In this case, we focus on the unique blocks. Let s' be the larger number of unique blocks in w and u, and for simplicity assume u has more unique blocks. We delete all matches where the endpoint in u is not in a unique block, or the endpoint in w falls out of the block at the same position as the block in u which contains the endpoint in u. Thus we attain a trimmed alignment

M'. Under the assumed condition in this case, we don't lose too many matches. Indeed the number of matches left is at least $\ell' = s'(n' - d') - n' - d'$. We now upper bound the probability that there exists such an M' which is a valid matching. Since this event is implied by the original event, this also provides an upper bound of the original event.

The probability that any M' is a valid matching is $2^{\ell'}$, by our definition of unique blocks. Now in the union bound, the main term turns out to be the total number of possible tuples of the inner codewords corresponding to the s' unique blocks in u and the other s' blocks at the same positions in w, which is roughly $2^{(2s')k'}$. Notice that $s' \geqslant s$. Thus in this case, as long as s is large enough, the rate of the code can also approach $1/2$.

The existence of linear concatenated codes matching random linear codes. This part is similar in spirit to Thommesen's work [23], which shows the existence of binary linear concatenated codes with Reed-Solomon outer codes that asymptotically meet the Gilbert-Varshamov bound. In particular, we also take a Reed-Solomon code as the outer code, and use an independent random linear inner code for every symbol of the outer codeword. Interestingly, here we take the outer code to be a $[n, k = (1-\gamma)n/2, d = (1+\gamma)n/2]_q$ Reed-Solomon code with $q = \Theta(n)$, i.e., the rate of the outer code is less than $1/2$. On the other hand, we take all inner codes to have rate 1. Using a careful probabilistic counting argument together with an estimate of the number of Reed-Solomon codewords with a specific weight (as done in [23]), we can prove the existence of linear concatenated insdel codes with parameters as in Theorem 4.

The choice of the parameters of the outer code is different from our explicit constructions, suggesting that maybe different constructions based on these parameters can lead to better explicit linear insdel codes.

Organization of the paper. Our general construction is exhibited in Section 3. The high error construction and its analysis are given in Section 4. We put the technical details of the rest of our results in the full version.

2 Preliminaries

Notation. Let Σ be an alphabet. For a string $x \in \Sigma^*$,

1. $|x|$ denotes the length of the string.
2. $x[i, j]$ denotes the substring of x from position i to position j (both endpoints included).
3. $x[i]$ denotes the i-th symbol of x.
4. $x \circ x'$ denotes the concatenation of x and some other string $x' \in \Sigma^*$.
5. For a string s which is a concatenation of shorter strings $s_1, s_2, \ldots, s_t$, the i-th block of s refers to s_i.

▶ **Definition 9** (Edit distance and Longest Common Subsequence). *For any two strings $x, y \in \Sigma^n$, the edit distance $\Delta_E(x, y)$ is the minimum number of edit operations (insertions and deletions) required to transform x into y.*[2] *A longest common subsequence of x and y is a longest pair of subsequences of x and y that are equal as strings. We use $\mathsf{LCS}(x, y)$ to denote the length of a longest common subsequence between x and y.*

[2] The standard definition of edit distance also allows substitution, but for simplicity we only consider insertions and deletions here, as a substitution can be replaced by a deletion followed by an insertion.

Note that $\Delta_E(x,y) = |x| + |y| - 2 \cdot \mathsf{LCS}(x,y)$. We use $\Delta_H(x,y)$ to denote the Hamming distance between two strings x and y.

▶ **Definition 10.** *An (n,m,d)-code C is an error-correcting code (for Hamming errors) with codeword length n, message length m, such that the Hamming distance between every pair of codewords in C is at least d.*

▶ **Definition 11.** *Fix an alphabet Σ, an error-correcting code $C \subseteq \Sigma^n$ for edit errors with message length m and codeword length n consists of an encoding function $Enc : \Sigma^m \to \Sigma^n$ and a decoding function $Dec : \Sigma^* \to \Sigma^m$. The code can correct k edit errors if for every y, s. t. $\Delta_E(y, Enc(x)) \leqslant k$, we have $Dec(y) = x$. The rate of the code is defined as $\frac{m}{n}$.*

We say C is a linear code if the alphabet Σ is a finite field F_q and the encoding function $\mathrm{Enc} : \mathsf{F}_q^m \to \mathsf{F}_q^n$ is a F_q-linear map.

We use the following list decoding algorithm for Reed-Solomon codes due to Guruswami and Sudan [11].

▶ **Theorem 12.** *Given a family of Reed-Solomon codes of message rate γ, an error rate of $\varepsilon = 1 - \sqrt{\gamma}$ can be list-decoded in polynomial time.*

We use U_n to denote the uniform distribution on $\{0,1\}^n$.

▶ **Definition 13.** *An ε-biased distribution X over $\{0,1\}^n$ is such that for any $S \subseteq [n]$, $|\Pr[\bigoplus_{i \in S} X_i = 1] - 1/2| \leqslant \varepsilon$. A function $g : \{0,1\}^s \to \{0,1\}^n$ is an ε-biased generator if $g(U_s)$ is an ε-biased distribution.*

The following ε-biased generator is used.

▶ **Theorem 14** ([2])**.** *For every $n \in \mathbb{N}$, every $\varepsilon \in (0,1)$, there exists an explicit ε-biased generator $\{0,1\}^s \to \{0,1\}^n$ with $s = O(\log n + \log(1/\varepsilon))$.*

We also need the following XOR lemma.

▶ **Lemma 15** (XOR Lemma)**.** *The statistical distance between an ε-biased distribution and a uniform distribution, both over $\{0,1\}^n$, is at most $\varepsilon\sqrt{2^n}$.*

3 General Construction of Our Codes

All our codes follow the general strategy of code concatenation, which we describe below.

The outer code $\mathbb{C}_{\mathsf{out}}$ with encoding function : $\mathsf{Enc}_{out} : \Sigma_{\mathrm{out}}^k \to \Sigma_{\mathrm{out}}^n$ is an $[n,k,d]$ Reed Solomon Code for Hamming errors. We then use n different inner codes $\mathbb{C}_{\mathsf{in}}^1, \ldots, \mathbb{C}_{\mathsf{in}}^n$, such that for any $i \in [n]$, $\mathbb{C}_{\mathsf{in}}^i$ is a linear code $\mathsf{Enc}_{in}^i : \Sigma_{\mathrm{out}} \to \Sigma_{\mathrm{in}}^{n'}$, where n' is the block length of the inner code. In this paper Σ_{in} always has constant size and we let $n' = \Theta(\log n)$. For different applications, we will need the inner codes to have slightly different properties.

Our final code $\mathbb{C}$ works naturally by first encoding the message using the outer code, then encoding each symbol of the outer code using the inner codes. This gives a codeword over Σ_{in} with length $N = n \cdot n'$. If the outer code and all the inner codes are linear, the concatenated code is also linear.

4 Constructions For High Noise

In this section we give our linear codes that can correct $1 - \varepsilon$ fraction of insdel errors, for any constant $\varepsilon > 0$. Our codes can still achieve a constant rate.

The construction. Following our general construction, we take $\mathbb{C}_{\mathsf{out}}$ to be an $[n, k, d]_n$ Reed-Solomon code with $|\Sigma_{\text{out}}| = n$, $k = \gamma n$ and $d = (1-\gamma)n$ for some constant $\gamma > 0$ to be chosen later. We construct n different inner codes $\mathbb{C}^1_{\mathsf{in}}, \ldots, \mathbb{C}^n_{\mathsf{in}}$ with alphabet size $|\Sigma_{\text{in}}| = \mathsf{poly}(1/\gamma)$, message length $k' = \Theta(\log n)$, and codeword length $n' = \Theta(\log n)$, with the following property.

▶ **Property 1.** *For any two codewords $x \in \mathbb{C}^i_{in}, y \in \mathbb{C}^j_{in}$, if either of the following two conditions holds:*
1. *$i \neq j$, and $x \neq 0^{n'}$ or $y \neq 0^{n'}$.*
2. *$i = j$ and $x \neq y$.*

Then we have $\mathsf{LCS}(x, y) \leqslant \gamma n'$.

▶ **Lemma 16.** *There exists an efficient construction of n inner codes $\mathbb{C}^1_{in}, \ldots, \mathbb{C}^n_{in}$, where each $\mathbb{C}^i_{in}$ has alphabet size $|\Sigma_{in}| = \mathsf{poly}(1/\gamma)$ and rate $\Omega(\gamma)$.*

Proof. We first show that if we pick n independent random linear inner codes $\mathbb{C}^1_{\mathsf{in}}, \ldots, \mathbb{C}^n_{\mathsf{in}}$ over an alphabet size $|\Sigma_{\text{in}}| = \mathsf{poly}(1/\gamma)$, then they satisfy Property 1 with high probability. We then show how to derandomize the construction using a small biased sample space.

Fix a field F_q. For each $\mathbb{C}^i_{\mathsf{in}}$ we independently pick $\log n$ uniformly random vectors in $\mathsf{F}_q^{n'}$ with $n' = \Theta(\log n/\gamma)$ as the basis for $\mathbb{C}^i_{\mathsf{in}}$, or equivalently, the rows in the generating matrix of $\mathbb{C}^i_{\mathsf{in}}$. We bound the probability that there exist two codewords $x \in \mathbb{C}^i_{\mathsf{in}}, y \in \mathbb{C}^j_{\mathsf{in}}$ that satisfy the conditions of Lemma 16 but $\mathsf{LCS}(x, y) > \gamma n'$.

▷ Claim 17. Consider any fixed common subsequence between x and y of length t, where the corresponding indices in x are $\{s_1, \cdots, s_t\}$ and the corresponding indices in y are $\{r_1, \cdots, r_t\}$. Then

$$\Pr[\forall k \in [t], x_{s_k} = y_{r_k}] \leqslant q^{-t}.$$

To prove the claim we have two cases.

Case 1: $i \neq j$, and $x \neq 0^{n'}$ or $y \neq 0^{n'}$. This is the easy case. Since $i \neq j$, and all the entries in the generating matrices of $\mathbb{C}^i_{\mathsf{in}}$ and $\mathbb{C}^j_{\mathsf{in}}$ are chosen independently uniformly from F_q, we know that the events $x_{s_k} = y_{r_k}$ are all independent, even if $x = 0^{n'}$ or $y = 0^{n'}$. Furthermore, the probability of each such event is $1/q$. Hence the claim follows.

Case 2: $i = j$. In this case, the events $x_{s_k} = y_{r_k}$ are not necessarily all independent. However, the claim still follows from the following claim in [6], which deals exactly with this situation.

▷ Claim 18. [Claim 4.2 of [6]] Let G be a random generating matrix for a linear code over F_q. For any two different messages x^i, x^j and codewords $C^i = x^i G, C^j = x^j G$, consider any fixed common subsequence between C^i and C^j of length t, where the corresponding indices in C^i are $\{s_1, \cdots, s_t\}$ and the corresponding indices in C^j are $\{r_1, \cdots, r_t\}$. Then

$$\Pr[\forall k \in [t], C^i_{s_k} = C^j_{r_k}] \leqslant q^{-t}.$$

Now by a union bound, and noticing that the total number of possible cases where two strings of length n' have a common subsequence of length $\gamma n'$ is at most $\binom{n'}{\gamma n'}^2$, we have

$$\Pr[\text{Property 1 does not hold}] \leqslant n^2 q^{2\log n} \binom{n'}{\gamma n'}^2 q^{-\gamma n'} \leqslant \left(\frac{e}{\gamma}\right)^{2\gamma n'} n^2 q^{2\log n - \gamma n'}.$$

Therefore, one can set $q = (\frac{e}{\gamma})^3$ and $n' = \Theta(\log n/\gamma)$ so that the above probability is $q^{-\Omega(\log n)} = 1/\mathsf{poly}(n)$.

Next we show how to derandomize the above construction using a small biased space. Without loss of generality we assume the field we use is F_q with $q = 2^\ell$. Thus, by choosing an arbitrary basis $b_1, \cdots, b_\ell$ in F_q we can identify the field with the vector space F_2^ℓ, such that any $a \in \mathsf{F}_q$ can be expressed as $a = \sum_{i\in[\ell]} a_i b_i$, where $\forall i, a_i \in \mathsf{F}_2$. In this way, the generating matrix of each $\mathbb{C}_{\mathsf{in}}^i$ can be viewed as consisting of $\ell n' \log n = \Theta(\ell \log^2 n)$ bits.

We pick a τ-biased sample space with $n\ell n' \log n$ bits for some $\tau = 1/\mathsf{poly}(n)$ to be chosen later. Note that by Theorem 14 this can be generated by $O(\log n)$ uniform random bits.

Given ℓ bits $a_1, \cdots, a_\ell$ which defines the field element $a = \sum_{i\in[\ell]} a_i b_i$, and any $p \in \mathsf{F}_q$, consider the operation $p \cdot a$ and the corresponding coefficient in the basis b_1. It's not hard to see that this is a F_2-linear function (i.e., a parity) of $a_1, \cdots, a_\ell$. Call this parity $L_p(a_1, \cdots, a_\ell)$. We have the following claim.

▷ Claim 19. $L_p(a_1, \cdots, a_\ell) \equiv 0$ if and only if $p = 0$.

Proof of the claim. The "if" part is trivially true. For the other part, note that if $p \neq 0$ then $pb_1, \cdots, pb_\ell$ must also be linearly independent and thus form a basis of F_q. Therefore, some pb_i must have a non-zero coefficient in b_1 and thus $L_p(a_1, \cdots, a_\ell)$ has a term a_i in the parity, therefore it cannot be the 0 function. ◁

Note that there are altogether 2^ℓ different parity functions involving $a_1, \cdots, a_\ell$, and $q = 2^\ell$ elements in F_q. Thus the previous claim immediately implies the following claim.

▷ Claim 20. Any parity function involving $a_1, \cdots, a_\ell$ is equivalent to $L_p(a_1, \cdots, a_\ell)$ for some $p \in \mathsf{F}_q$.

Now consider the two codewords $x \in \mathbb{C}_{\mathsf{in}}^i, y \in \mathbb{C}_{\mathsf{in}}^j$. Let x_0 and y_0 be the corresponding messages for x and y respectively. We now have the following claim.

▷ Claim 21. Unless $i = j$ and $y_0 = p \cdot x_0$ or $x_0 = p \cdot y_0$ for some $p \in \mathsf{F}_q$, under the τ-biased sample space, the joint distribution of (x, y) is $q^{n'}\tau$-close to the uniform distribution over $\mathsf{F}_q^{2n'}$.

Proof of the claim. Let $x = (x_1, \cdots, x_{n'}) \in \mathsf{F}_q^{n'} = \mathsf{F}_2^{\ell n'}$ and $y = (y_1, \cdots, y_{n'}) \in \mathsf{F}_q^{n'} = \mathsf{F}_2^{\ell n'}$. Consider any non-trivial parity of the $2\ell n'$ bits, which by Claim 20 corresponds to the coefficient of b_1 under some function $\sum_{k\in[n']}(p_k^x x_k + p_k^y y_k)$, where $\forall k, p_k^x, p_k^y \in \mathsf{F}_q$, and they are not all 0.

If $i \neq j$, then $\sum_{k\in[n']} p_k^x x_k$ and $\sum_{k\in[n']} p_k^y y_k$ use different bits in the τ-biased sample space. Since x, y are not both $0^{n'}$, the resulted parity is a non-trivial parity of the bits in the sample space, which by definition has bias at most τ.

Otherwise we have $i = j$. Let G be the generating matrix for $\mathbb{C}_{\mathsf{in}}^i$, thus $x = x_0 G$ and $y = y_0 G$. For any $k \in [n']$, let G_k be the k'th column of G. We have

$$\sum_{k\in[n']}(p_k^x x_k + p_k^y y_k) = \sum_{k\in[n']}(p_k^x x_0 G_k + p_k^y y_0 G_k) = \sum_{k\in[n']}(p_k^x x_0 + p_k^y y_0) G_k.$$

Notice that each entry in each G_k is independently uniformly chosen from $\mathsf{F}_q = \mathsf{F}_2^\ell$. Thus by Claim 19 if the coefficient of the above sum in b_1 is the trivial parity 0, then we must have $\forall k \in [n'], p_k^x x_0 + p_k^y y_0 = 0$. This implies that either $y_0 = p \cdot x_0$ or $x_0 = p \cdot y_0$ for some $p \in \mathsf{F}_q$.

Otherwise, the parity is a non-trivial parity of the bits in the sample space, which by definition has bias at most τ. Now, by Lemma 15, the joint distribution of (x, y) is $q^{n'}\tau$-close to the uniform distribution over $\mathsf{F}_q^{2n'}$. ◁

Back to the proof of our lemma. If the conditions of the above claim hold, then the joint distribution of (x, y) is $q^{n'}\tau$-close to the uniform distribution. Hence, the probability that there exists any common subsequence of length $\gamma n'$ between x and y is at most $\binom{n'}{\gamma n'}^2 q^{-\gamma n'} + q^{n'}\tau$.

On the other hand, if the conditions of the above claim do not hold, then without loss of generality assume that $y_0 = p \cdot x_0$ for some $p \in \mathsf{F}_q$. Hence $p \neq 1$. In this case, notice that we also have $y = p \cdot x$, and thus the probability that there exists any common subsequence of length $\gamma n'$ between x and y is completely determined by the random variables in x. Note that any non-trivial parity of the bits in x is also a non-trivial parity of the bits of the τ-biased sample space, which has bias at most τ. By Lemma 15, the distribution of x is $q^{n'/2}\tau$-close to being uniform on $\mathsf{F}_q^{n'}$.

We have the following claim.

▷ Claim 22. Let x be a uniformly random vector in $\mathsf{F}_q^{n'}$, and $y = p \cdot x$. Then

$$\Pr[\exists \text{a common subsequence of length } t \text{ between } x \text{ and } y] \leqslant \binom{n'}{t}^2 q^{-t}.$$

Proof of the claim. Consider any fixed common subsequence of length t between x and y. Assume where the corresponding indices in x are $\{s_1, \cdots, s_t\}$ and the corresponding indices in y are $\{r_1, \cdots, r_t\}$, such that $s_1 < s_2 < \cdots < s_t$ and $r_1 < r_2 < \cdots < r_t$. For any $k \in [t]$, let $m_k = \max(s_k, r_k)$. Notice that $m_1 < m_2 < \cdots < m_t$. Define E_k to be the event $x_{s_k} = y_{r_k}$.

For each $k \in [t]$, if $s_k = r_k$, then

$$\Pr[E_k] = \Pr[x_{s_k} = p \cdot x_{s_k}] = \Pr[x_{s_k} = 0] = \frac{1}{q}.$$

Furthermore, since $s_k = r_k = m_k$ is larger than all $\{s_{k'}, r_{k'}, k' < k\}$, the event E_k is independent of all $\{E_{k'}, k' < k\}$. Thus

$$\Pr[E_k | \{E_{k'}, k' < k\}] = \frac{1}{q}.$$

Otherwise, $s_k \neq r_k$ and without loss of generality assume $s_k > r_k$. This means $s_k = m_k$ and is larger than all $\{s_{k'}, r_{k'}, k' < k\}$. We can now first fix all $\{x_{s_{k'}}, y_{r_{k'}}, k' < k\}$ and y_{r_k}, and conditioned on this fixing x_{s_k} is still uniform over F_q. Thus

$$\Pr[E_k] = \Pr[x_{s_k} = p \cdot x_{r_k}] = \frac{1}{q}.$$

Note that any such fixing also fixes the outcomes of all $\{E_{k'}, k' < k\}$. Hence we also have

$$\Pr[E_k | \{E_{k'}, k' < k\}] = \frac{1}{q}.$$

Therefore, the above equation holds in all cases, and for all k. This gives

$$\Pr[\bigcap_{k \in [t]} E_k] \leqslant q^{-t},$$

and the claim follows from a union bound. ◁

Since x is $q^{n'/2}\tau$-close to being uniform on $\mathsf{F}_q^{n'}$, the probability that there exists any common subsequence of length $\gamma n'$ between x and y is at most $\binom{n'}{\gamma n'}^2 q^{-\gamma n'} + q^{n'/2}\tau$ in this case.

To summarize, using the τ-biased sample space we always have that the probability that there exists any common subsequence of length $\gamma n'$ between x and y is at most $\binom{n'}{\gamma n'}^2 q^{-\gamma n'} + q^{n'}\tau$. By another union bound, we have

$$\begin{aligned}\Pr[\text{Property 1 does not hold}] &\leqslant n^2 q^{2\log n}\left(\binom{n'}{\gamma n'}^2 q^{-\gamma n'} + q^{n'}\tau\right)\\ &\leqslant \left(\frac{e}{\gamma}\right)^{2\gamma n'} n^2 q^{2\log n - \gamma n'} + n^2 q^{n'+2\log n}\tau.\end{aligned}$$

Therefore, one can still set $q = (\frac{e}{\gamma})^3$, $n' = \Theta(\log n/\gamma)$, and $\tau = q^{-\Omega(\log n/\gamma)} = 1/\mathsf{poly}(n)$ so that the above probability is $q^{-\Omega(\log n)} = 1/\mathsf{poly}(n)$.

Once we know this, we can exhaustively search the τ-biased sample space and find a sample point which gives us a construction that satisfies Property 1. Since we only have $\mathsf{poly}(n)$ sample points and checking each sample point takes polynomial time, altogether this takes polynomial time. ◀

Note that our concatenated code $\mathbb{C}$ now has rate $\Omega(\gamma^2)$. Further, Property 1 implies the following property:

▶ **Property 2.**
1. *$\forall i \neq j$, we have $\mathbb{C}_{in}^i \bigcap \mathbb{C}_{in}^j = \{0^{n'}\}$.*
2. *For any $i, j \in [n]$ and any two codewords $x \in \mathbb{C}_{in}^i, y \in \mathbb{C}_{in}^j$, if $x \neq y$ then $\mathsf{LCS}(x, y) \leqslant \gamma n'$.*

Let z be any substring of a codeword from the concatenated code $\mathbb{C}$, and assume z is a substring of $z_j \circ z_{j+1} \circ \cdots \circ z_{j+\ell}$, where $\forall t$, z_{j+t} is a codeword in $\mathbb{C}_{\mathsf{in}}^{j+t}$. We say the codewords $\{z_{j+t}, t = 0, \cdots, \ell\}$ contribute to the string z.

We now show that Property 1 and Property 2 give us the following lemma.

▶ **Lemma 23.** *Let x be a codeword from the code $\mathbb{C}_{in}^i$. Let z be any substring of a codeword from the concatenated code $\mathbb{C}$, and $\{z_{j+t}, t = 0, \cdots, \ell\}$ are the inner codewords contributing to z. If $\forall t$, $z_{j+t} \neq x$, then we have $\mathsf{LCS}(x, z) < 2\gamma(|x| + |z|)$.*

Proof. By Property 2, the longest common subsequence between x and any z_{j+t} has length at most $\gamma n'$. If $\ell = 1$, then we have

$$\mathsf{LCS}(x, z) \leqslant \gamma n' < 2\gamma(|x| + |z|).$$

Otherwise we have $\ell \geqslant 2$. Notice that $|z| > (\ell - 2)n'$. Thus we have

$$\mathsf{LCS}(x, z) \leqslant \ell\gamma n' \leqslant 2\gamma(\ell - 1)n' < 2\gamma(|x| + |z|).$$ ◀

We can now prove the following lemma.

▶ **Lemma 24.** *For any two different codewords $C_1, C_2 \in \mathbb{C}$, we have $\Delta_E(C_1, C_2) > 2(1-6\gamma)N$.*

Proof. We upper bound $\mathsf{LCS}(C_1, C_2)$ as follows. Consider a particular longest common subsequence and divide it sequentially according to the n inner codewords in C_1. Let the codewords in C_1 be $x_1, \cdots, x_n$ and the corresponding substrings in C_2 under the LCS be $z_1, \cdots, z_n$.

By Lemma 23, for any $i \in [n]$, we must have $\mathsf{LCS}(x_i, z_i) \leqslant 2\gamma(|x_i| + |z_i|)$, unless some inner codeword in z_i is equal to x_i. This could happen either because $x_i = 0^{n'}$ or because z_i contains part of x_i from exactly the i'th inner code. In the latter two cases, we call such an index i *bad*. Note that for a bad i we have $\mathsf{LCS}(x_i, z_i) \leqslant n'$, and there are at most γn such bad indices for either case, by our choice of the outer code. Let t be the number of bad indices, thus $t \leqslant 2\gamma n$. Therefore,

$$\begin{aligned}\mathsf{LCS}(x, z) &= \sum_{i \text{ is not bad}} \mathsf{LCS}(x_i, z_i) + \sum_{i \text{ is bad}} \mathsf{LCS}(x_i, z_i) \\ &\leqslant 2\gamma \sum_{i \text{ is not bad}} (|x_i| + |z_i|) + tn' \\ &< 2\gamma(2n'n) + 2\gamma nn' = 6\gamma N,\end{aligned}$$

where the last inequality follows from the fact that if the number of bad indices is larger than 0, then $\sum_{i \text{ is not bad}}(|x_i| + |z_i|) < 2n'n$. Therefore $\Delta_E(C_1, C_2) > 2N - 12\gamma N = 2(1 - 6\gamma)N$. ◀

Setting $\gamma = \varepsilon/6$, this gives the following theorem.

▶ **Theorem 25.** *For any constant $\varepsilon > 0$ there exists an efficient construction of linear insdel codes over an alphabet of size $\mathsf{poly}(1/\varepsilon)$, with rate $\Omega(\varepsilon^2)$ that can correct $1 - \varepsilon$ fraction of insdel errors (possibly inefficiently).*

References

1. Khaled A.S. Abdel-Ghaffar, Hendrik C. Ferreira, and Ling Cheng. On linear and cyclic codes for correcting deletions. In *2007 IEEE International Symposium on Information Theory (ISIT)*, pages 851–855, 2007.
2. Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992.
3. J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss. A dna-based archival storage system. *ACM SIGARCH Comput. Archit. News*, 44:637–649, 2016.
4. Joshua Brakensiek, Venkatesan Guruswami, and Samuel Zbarsky. Efficient low-redundancy codes for correcting multiple deletions. *IEEE Transactions on Information Theory*, 64(5):3403–3410, 2018. Preliminary version in SODA 2016.
5. Boris Bukh, Venkatesan Guruswami, and Johan Håstad. An improved bound on the fraction of correctable deletions. *IEEE Trans. Information Theory*, 63(1):93–103, 2017. Preliminary version in SODA 2016. doi:10.1109/TIT.2016.2621044.
6. Kuan Cheng, Venkatesan Guruswami, Bernhard Haeupler, and Xin Li. Efficient linear and affine codes for correcting insertions/deletions. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1–20, 2021. doi:10.1137/1.9781611976465.1.
7. Kuan Cheng, Zhengzhong Jin, Xin Li, and Ke Wu. Block Edit Errors with Transpositions: Deterministic Document Exchange Protocols and Almost Optimal Binary Codes. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 37:1–37:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2019.37.
8. Kuan Cheng, Zhengzhong Jin, Xin Li, and Ke Wu. Deterministic document exchange protocols, and almost optimal binary codes for edit errors. *Journal of the ACM (JACM)*, 69(6):1–39, 2022. Preliminary version in 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS).
9. Roni Con, Amir Shpilka, and Itzhak Tamo. Explicit and efficient constructions of linear codes against adversarial insertions and deletions. *IEEE Transactions on Information Theory*, 68(10):6516–6526, 2022. doi:10.1109/TIT.2022.3173185.

10 Roni Con, Amir Shpilka, and Itzhak Tamo. Reed solomon codes against adversarial insertions and deletions. In *2022 IEEE International Symposium on Information Theory (ISIT)*, pages 2940–2945, 2022. doi:10.1109/ISIT50566.2022.9834672.

11 V. Guruswami and M. Sudan. Improved decoding of reed-solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999. doi:10.1109/18.782097.

12 Venkatesan Guruswami, Bernhard Haeupler, and Amirbehshad Shahrasbi. Optimally resilient codes for list-decoding from insertions and deletions. In *Proccedings of the 52nd Annual ACM Symposium on Theory of Computing*, pages 524–537, 2020. doi:10.1145/3357713.3384262.

13 Venkatesan Guruswami, Xiaoyu He, and Ray Li. The zero-rate threshold for adversarial bit-deletions is less than 1/2. *IEEE Transactions on Information Theory*, pages 1–1, 2022. doi:10.1109/TIT.2022.3223023.

14 Venkatesan Guruswami and Carol Wang. Deletion codes in the high-noise and high-rate regimes. *IEEE Trans. Information Theory*, 63(4):1961–1970, 2017.

15 Bernhard Haeupler. Optimal document exchange and new codes for insertions and deletions. In *60th IEEE Annual Symposium on Foundations of Computer Science*, pages 334–347, 2019.

16 Bernhard Haeupler, Aviad Rubinstein, and Amirbehshad Shahrasbi. Near-Linear Time Insertion-Deletion Codes and (1+eps)-Approximating Edit Distance via Indexing. *Proceeding of the ACM Symposium on Theory of Computing (STOC)*, pages 697–708, 2019.

17 Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization strings: codes for insertions and deletions approaching the singleton bound. *Journal of the ACM (JACM)*, 68(5):1–39, 2021. Preliminary version in 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC).

18 Bernhard Haeupler, Amirbehshad Shahrasbi, and Madhu Sudan. Synchronization strings: List decoding for insertions and deletions. *Proceeding of the International Colloquium on Automata, Languages and Programming (ICALP)*, pages 76:1–76:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.76.

19 Bernhard Haeupler, Amirbehshad Shahrasbi, and Ellen Vitercik. Synchronization strings: Channel simulations and interactive coding for insertions and deletions. *Proceeding of the International Colloquium on Automata, Languages and Programming (ICALP)*, pages 75:1–75:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.75.

20 Tomohiro Hayashi and Kenji Yasunaga. On the list decodability of insertions and deletions. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 86–90. IEEE, 2018.

21 Leonard J. Schulman and David Zuckerman. Asymptotically good codes correcting insertions, deletions, and transpositions. *IEEE Trans. Inf. Theory*, 45(7):2552–2557, 1999. Preliminary version in SODA 1997. doi:10.1109/18.796406.

22 Jin Sima and Jehoshua Bruck. Optimal k-deletion correcting codes. In *IEEE International Symposium on Information Theory*, pages 847–851, 2019. doi:10.1109/ISIT.2019.8849750.

23 C. Thommesen. The existence of binary linear concatenated codes with reed - solomon outer codes which asymptotically meet the gilbert- varshamov bound. *IEEE Transactions on Information Theory*, 29(6):850–853, 1983. doi:10.1109/TIT.1983.1056765.

24 Antonia Wachter-Zeh. List decoding of insertions and deletions. *IEEE Transactions on Information Theory*, 64(9):6297–6304, 2017.

25 S. M. Hossein Tabatabaei Yazdi, Ryan Gabrys, and Olgica Milenkovic. Portable and error-free dna-based data storage. *Scientific Reports*, 7:2045–2322, 2017. doi:10.1038/s41598-017-05188-1.

Online Learning and Disambiguations of Partial Concept Classes

Tsun-Ming Cheung
McGill University, Montreal, Canada

Hamed Hatami
McGill University, Montreal, Canada

Pooya Hatami
Ohio State University, Columbus, OH, USA

Kaave Hosseini
University of Rochester, NY, USA

Abstract

In a recent article, Alon, Hanneke, Holzman, and Moran (FOCS '21) introduced a unifying framework to study the learnability of classes of *partial* concepts. One of the central questions studied in their work is whether the learnability of a partial concept class is always inherited from the learnability of some "extension" of it to a total concept class.

They showed this is not the case for PAC learning but left the problem open for the stronger notion of online learnability.

We resolve this problem by constructing a class of partial concepts that is online learnable, but no extension of it to a class of total concepts is online learnable (or even PAC learnable).

2012 ACM Subject Classification Theory of computation → Online learning theory

Keywords and phrases Online learning, Littlestone dimension, VC dimension, partial concept class, clique vs independent set, Alon-Saks-Seymour conjecture, Standard Optimal Algorithm, PAC learning

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.42

Category Track A: Algorithms, Complexity and Games

Related Version *arXiv*: `https://arxiv.org/abs/2303.17578`

Funding *Hamed Hatami*: Supported by an NSERC grant.
Pooya Hatami: Supported by NSF grant CCF-1947546.

Acknowledgements We wish to thank Mika Göös for clarifying the reductions in [3, 4, 5, 2].

1 Introduction

In many practical learning problems, the learning task is tractable because we are only required to predict the labels of the data points that satisfy specific properties. In the setting of binary classification problems, instead of learning a total concept $h : \mathcal{X} \to \{0, 1\}$, we are often content with learning a partial version of it $\widetilde{h} : \mathcal{X} \to \{0, 1, \star\}$, where $\widetilde{h}(x) = \star$ means that both 0 and 1 are acceptable predictions. This relaxation of allowing unspecified predictions renders a wider range of learning tasks tractable.

Consider, for example, predicting whether a person approves or disapproves of various political stances by observing their previous voting pattern. This person might not hold a strong opinion about particular political sentiments, and it might be impossible to predict their vote on those issues based on their previous history. However, the learning task might become possible if we allow both "approve" and "disapprove" as acceptable predictions in those cases where a firm conviction is lacking.

A well-studied example of this phenomenon is learning half-spaces with a large margin. In this problem, the domain is the set of points in a bounded region in an arbitrary Euclidean space, and the concepts are half-spaces that map each point to 1 or 0 depending on whether they belong to the half-space or not. It is well-known that when the dimension of the underlying Euclidean space is large, one needs many samples to learn a half-space. However, in the large margin setting, we are only required to correctly predict the label of a point if its distance from the defining hyperplane is bounded from below by some margin. Standard learning algorithms for this task, such as the classical Perceptron algorithm, due to Rosenblatt [9], show that this relaxation of the learning requirement makes the problem tractable even for high-dimensional Euclidean spaces. Motivated by such examples, Alon, Hanneke, Holzman, and Moran [1] initiated a systematic study of the learnability of partial concept classes $\mathbb{H} \subseteq \{0,1,\star\}^{\mathcal{X}}$. They focused on the two frameworks of *probably approximately correct (PAC) learning* and *online learning*. We refer to [1] for the definition of PAC learnability of partial concept classes. We define online learnability in Definition 4.

PAC learning is an elegant theoretical framework characterized by the combinatorial parameter of the Vapnik–Chervonenkis (VC) dimension. The fundamental theorem of PAC learning states that a total binary concept class is PAC learnable if and only if its VC dimension is finite. Similarly, online learnability of total concept classes is characterized by a combinatorial parameter called the Littlestone dimension (LD). We formally define the VC dimension and the Littlestone dimension in Definitions 14 and 15 respectively. Alon, Hanneke, Holzman, and Moran [1] proved that these characterizations of PAC and online learnability extend to the setting of partial concept classes.

▶ **Theorem 1** ([1, Theorems 1 and 15]). *Let $\mathbb{H} \subseteq \{0,1,\star\}^{\mathcal{X}}$ be a partial concept class.*

- *$\mathbb{H}$ is PAC learnable if and only if $\mathrm{VC}(\mathbb{H}) < \infty$.*
- *$\mathbb{H}$ is online learnable if and only if $\mathrm{LD}(\mathbb{H}) < \infty$.*

It follows from the definitions of VC and LD dimensions that for every partial concept class $\mathbb{H} \subseteq \{0,1,\star\}^{\mathcal{X}}$, we have $\mathrm{VC}(\mathbb{H}) \leq \mathrm{LD}(\mathbb{H})$. In particular, online learnability always implies PAC learnability.

One of the central questions studied in [1] is whether the learnability of a partial concept class is always inherited from the learnability of some total concept class. To make this question precise, we need to define the notion of disambiguation of a partial concept class. While we defer the formal definitions to Section 2.2, one may understand a *strong disambiguation* of a partial class as simply an assignment of each $\star$ to either 1 or 0 for each partial concept in the class. When $\mathcal{X}$ is infinite, it is more natural to consider the weaker notion of *disambiguation* that we shall define in Definition 17. When $\mathcal{X}$ is finite, the notions of disambiguation and strong disambiguation coincide.

Consider the problem of learning the partial concept class $\mathbb{H} \subseteq \{0,1,\star\}^{\mathcal{X}}$ in PAC learning or online learning. If the partial concept class $\mathbb{H}$ has a disambiguation $\overline{\mathbb{H}} \subseteq \{0,1\}^{\mathcal{X}}$ that is PAC learnable, then $\mathbb{H}$ is PAC learnable. This follows from $\mathrm{VC}(\mathbb{H}) \leq \mathrm{VC}(\overline{\mathbb{H}})$, or simply by running the PAC learning algorithm of $\overline{\mathbb{H}}$ on $\mathbb{H}$. Similarly, if a disambiguation $\overline{\mathbb{H}}$ of $\mathbb{H}$ is online learnable, then $\mathbb{H}$ is online learnable.

Is the learnability of every partial concept class inherited from the learnability of some disambiguation to a total concept class?

▶ **Question 2** (Informal [1]). *Does every learnable partial class have a learnable disambiguation?*

Equipped with the VC dimension characterization of Theorem 1, [1] proved that for PAC learning, the answer to Question 2 is *negative*.

▶ **Theorem 3** ([1, Theorem 11]). *For every $n \in \mathbb{N}$, there exists a partial concept class $\mathbb{H}_n \subseteq \{0,1,\star\}^{[n]}$ with $\mathrm{VC}(\mathbb{H}_n) = 1$ such that any disambiguation $\overline{\mathbb{H}}$ of $\mathbb{H}_n$ has $\mathrm{VC}(\overline{\mathbb{H}}) \geq (\log n)^{1-o(1)}$. Moreover, for $\mathcal{X} = \mathbb{N}$, there exists $\mathbb{H}_\infty \subseteq \{0,1,\star\}^{\mathcal{X}}$ with $\mathrm{VC}(\mathbb{H}_\infty) = 1$ such that $\mathrm{VC}(\overline{\mathbb{H}}) = \infty$ for every disambiguation $\overline{\mathbb{H}}$ of $\mathbb{H}_\infty$.*

While Theorem 3 gives a strong negative answer to Question 2 in the case of PAC learning, the question was left open for online learning. Roughly speaking, this question strengthens the bounded-VC assumption on $\mathbb{H}$ to bounded *Littlestone dimension* (LD), which pertains to *online learnability* of $\mathbb{H}$.

The authors in [1] also proposed a second open problem that replaces the bounded-VC dimension assumption by the assumption of *polynomial growth*. This assumption is weaker than bounded LD dimension but stronger than bounded VC dimension.

As we discuss below, our main result resolves these two open problems.

Online learnability

Online learning is performed in a sequence of consecutive rounds, where at round t, the learner is presented with an instance $x_t \in \mathcal{X}$ and is required to predict its label. After predicting the label, the correct label $y_t \in \{0,1\}$ is revealed to the learner. Note that even for partial concept classes, we require that the correct label is 0 or 1. The learner's goal is to make as few prediction mistakes as possible during this process. We assume that the true labels are always *realizable*, i.e. there is a partial concept $h \in \mathbb{H}$ with $h(x_i) = y_i$ for all $i = 1, \ldots, t$.

▶ **Definition 4** (Online Learnability). *A partial concept class $\mathbb{H} \subseteq \{0,1,\star\}^{\mathcal{X}}$ is online learnable if there is a* mistake bound *$m := m(\mathbb{H}) \in \mathbb{N}$ such that for every $T \in \mathbb{N}$, there exists a learning algorithm that on every realizable sequence $(x_i, y_i)_{i=1,\ldots,T}$ makes at most m mistakes.*

Online learnability for total classes is equivalent to the bounded Littlestone dimension. In Theorem 1, Alon, Hanneke, Holzman, and Moran [1] showed that the same equivalence carries out in the setting of partial classes. They asked the following formulation of Question 2.

If a partial class is online learnable, is there a disambiguation of it that is online learnable?

More precisely, they pose the following question:

▶ **Problem 5** ([1]). *Let $\mathbb{H}$ be a partial class with $\mathrm{LD}(\mathbb{H}) < \infty$. Does there exist a disambiguation $\overline{\mathbb{H}}$ of $\mathbb{H}$ with $\mathrm{LD}(\overline{\mathbb{H}}) < \infty$? Is there one with $\mathrm{VC}(\overline{\mathbb{H}}) < \infty$?*

We give a negative answer to Problem 5:

▶ **Theorem 6** (Main Theorem). *For every $n \in \mathbb{N}$, there exists a partial concept class $\mathbb{H}_n \subseteq \{0,1,\star\}^{[n]}$ with $\mathrm{LD}(\mathbb{H}_n) \leq 2$ such that every disambiguation $\overline{\mathbb{H}}$ of $\mathbb{H}_n$ satisfies $\mathrm{LD}(\overline{\mathbb{H}}) \geq \mathrm{VC}(\overline{\mathbb{H}}) = \Omega(\log\log n)$. Consequently, for $\mathcal{X} = \mathbb{N}$, there exists $\mathbb{H}_\infty \subseteq \{0,1,\star\}^{\mathcal{X}}$ with $\mathrm{LD}(\mathbb{H}_\infty) \leq 2$ and $\mathrm{LD}(\overline{\mathbb{H}}) \geq \mathrm{VC}(\overline{\mathbb{H}}) = \infty$ for every disambiguation $\overline{\mathbb{H}}$ of $\mathbb{H}_\infty$.*

Polynomial growth

A general strategy to prove a super-constant lower bound on the VC dimension of a total concept class $\mathbb{H} \subseteq \{0,1\}^n$ is to show that the class is of super-polynomial size. This is the approach utilized in Theorem 3 and Theorem 6. For a total concept class $\mathbb{H} \subseteq \{0,1\}^n$ with VC dimension d, one has $2^d \leq |\mathbb{H}| \leq O(n^d)$: the lower bound is immediate from the definition of VC dimension, and the upper bound is the consequence of the celebrated Sauer-Shelah-Perles (SSP) lemma.

▶ **Theorem 7** (Sauer-Shelah-Perles lemma [10]). *Let* $\mathbb{H} \subseteq \{0,1\}^n$ *and* $\mathrm{VC}(\mathbb{H}) = d$. *Then*

$$|\mathbb{H}| \leq \binom{n}{\leq d} \coloneqq \sum_{i=0}^{d} \binom{n}{i} = O(n^d).$$

The direct analog of the SSP lemma is not true for partial concept classes: [1] proved that there exists $\mathbb{H} \subseteq \{0,1,\star\}^{[n]}$ with $\mathrm{VC}(\mathbb{H}) = 1$ such that every disambiguation $\overline{\mathbb{H}}$ has size $|\overline{\mathbb{H}}| \geq n^{\Omega(\log n)}$. This result, combined with the SSP lemma for total classes, immediately implies Theorem 3.

Interestingly, under the stronger assumption of the bounded Littlestone dimension, the polynomial growth behavior of the original SSP lemma remains valid.

▶ **Theorem 8** ([1]). *Every partial concept class* $\mathbb{H} \subseteq \{0,1,\star\}^{[n]}$ *with* $\mathrm{LD}(\mathbb{H}) \leq d$ *has a disambiguation* $\overline{\mathbb{H}}$ *with* $|\overline{\mathbb{H}}| \leq O(n^d)$.

We say that a partial concept class $\mathbb{H} \subseteq \{0,1,\star\}^{\mathcal{X}}$ has *polynomial growth with parameter* $d \in \mathbb{N}$ if for every finite $\mathcal{X}' \subseteq \mathcal{X}$, there is a disambiguation $\overline{\mathbb{H}|_{\mathcal{X}'}}$ of $\mathbb{H}|_{\mathcal{X}'}$ of size at most $O(|\mathcal{X}'|^d)$. Note that by Theorem 8, every partial concept class with Littlestone dimension d has polynomial growth with parameter d.

Alon, Hanneke, Holzman, and Moran asked the following question:

▶ **Problem 9** ([1]). *Let* $\mathbb{H} \subseteq \{0,1,\star\}^{\mathcal{X}}$ *be a partial concept class with polynomial growth. Does there exist a disambiguation* $\overline{\mathbb{H}}$ *of* $\mathbb{H}$ *such that* $\mathrm{VC}(\overline{\mathbb{H}}) < \infty$*?*

Note that Problem 9 cannot be resolved (in the negative) by a naive application of the SSP lemma to disambiguations of $\mathbb{H}$ or its restrictions. However, Theorem 6 combined with Theorem 8 refutes Problem 9 as well.

▶ **Theorem 10.** *For every* $n \in \mathbb{N}$, *there is* $\mathbb{H} \subseteq \{0,1,\star\}^{[n]}$ *with polynomial growth with parameter* 2 *such that every disambiguation* $\overline{\mathbb{H}}$ *of* $\mathbb{H}$ *has* $\mathrm{VC}(\overline{\mathbb{H}}) = \Omega(\log\log n)$.

Consequently, for $\mathcal{X} = \mathbb{N}$, *there exists* $\mathbb{H}_\infty \subseteq \{0,1,\star\}^{\mathcal{X}}$ *with polynomial growth with parameter* 2 *such that every disambiguation* $\overline{\mathbb{H}_\infty}$ *of* $\mathbb{H}_\infty$ *has* $\mathrm{VC}(\overline{\mathbb{H}_\infty}) = \infty$.

The Alon-Saks-Seymour Problem

The proof of Theorem 3 in [1] hinges on the breakthrough result of Göös [4] and its subsequent improvements [2] that led to almost optimal super-polynomial bounds on the "biclique partition number versus chromatic number" problem of Alon, Saks, and Seymour. The *biclique partition number* of a graph G, denoted by $\mathrm{bp}(G)$, is the smallest number of complete bipartite graphs (bicliques) that partition the edge set of G. Alon, Saks, and Seymour conjectured that the chromatic number of a graph with biclique partition number k is at most $k+1$. Huang and Sudakov refuted the Alon-Saks-Seymour conjecture in [6] by establishing a superlinear gap between the two parameters. Later in a breakthrough, Göös [4] proved a superpolynomial separation.

Our main result, Theorem 6, also builds on the aforementioned graph constructions. However, unlike previous works, our theorem demands a reasonable upper bound on the number of vertices. Since the constructions result from a complex sequence of reductions involving query complexity, communication complexity, and graph theory [3, 4, 5, 2], it is necessary to scrutinize them to ensure that the required parameters are met. We present a reorganized and partly simplified sequence of constructions in Section 3.3 that establishes the following theorem.

▶ **Theorem 11** (Small-size refutation of the Alon-Saks-Seymour conjecture). *There exists a graph G on $2^{\Theta(k^4 \log^3 k)}$ vertices that admits a biclique partition of size $2^{O(k \log^4 k)}$ but its chromatic number is at least $2^{\Omega(k^2)}$.*

Theorem 11 is essentially due to [2]. Our contribution to this theorem is obtaining an explicit and optimized bound on the size of G.

Standard Optimal Algorithm

Theorem 6 provides an example partial class with Littlestone dimension ≤ 2, such that the VC dimension of every disambiguation is $\Omega(\log \log n)$. Whether one can improve the $\Omega(\log \log n)$ lower bound is unclear. In particular, it is an interesting question whether every disambiguation of a partial class of Littlestone dimension at most 2 has VC dimension $O(\log \log n)$. One natural candidate approach for obtaining such an upper bound would be to utilize the Standard Optimal Algorithm (SOA).

SOA is an online learning algorithm devised by Littlestone [7] that can learn classes with bounded Littlestone dimensions. Alon, Hanneke, Holzman, and Moran, in their proof of Theorem 8, showed that applying SOA to a partial concept class $\mathbb{H}$ with Littlestone dimension d yields a disambiguation of size $|\overline{\mathbb{H}}| \leq O(n^d)$ and consequently VC dimension $O(d \log n)$. This shows that the lower bound of Theorem 6 on VC dimension of disambiguations cannot be improved beyond $O(\log n)$. It is hence natural to ask whether it is possible to obtain an improved upper bound on the VC dimension of the SOA-based disambiguation.

We answer this question in the negative by constructing a family of partial concept classes $\mathbb{H}$ of Littlestone dimension d where the disambiguation obtained by the SOA algorithm has VC dimension $\Omega(d \log(n/d))$.

▶ **Theorem 12.** *For every natural numbers $d \leq n$, there exists a partial concept class $\mathbb{H}_{n,d} \subseteq \{0,1,\star\}^{[n]}$ with $d \leq \mathrm{LD}(\mathbb{H}_{n,d}) \leq d+1$ such that the SOA disambiguation of $\mathbb{H}_{n,d}$ has* VC *dimension $\Omega(d \log(n/d))$.*

2 Preliminaries and Background

For a positive integer k, we denote $[k] \coloneqq \{1, \ldots, k\}$. We adopt the convention that $\{0,1\}^0$ or $\{0,1,\star\}^0$ contains the empty string only, which we denote by $()$.

We adopt the standard computer science asymptotic notations, such as Big-O, and use the asymptotic tilde notations to hide poly-logarithmic factors.

2.1 VC Dimension and Littlestone Dimension

Let $\mathbb{H} \subseteq \{0,1,\star\}^{\mathcal{X}}$ be a partial concept class. When the domain $\mathcal{X}$ is finite, we sometimes view $\mathbb{H}$ as a partial matrix $\mathbf{M}_{\mathcal{X} \times \mathbb{H}}$, where each row corresponds to a point $x \in \mathcal{X}$ and each column corresponds to a concept $h \in \mathbb{H}$, and the entries are defined as $\mathbf{M}(x,h) = h(x)$.

Next, we define the VC dimension and the Littlestone dimension of partial classes, which generalize the definitions of these notions for total classes. As shown in [1], the VC and Littlestone dimensions for partial classes capture PAC and online learnability, respectively.

▶ **Definition 13** (Shattered set). *A finite* set *of points $C = \{x_1, \ldots, x_n\} \subseteq \mathcal{X}$ is* shattered *by a partial concept class $\mathbb{H} \subseteq \{0,1,\star\}^{\mathcal{X}}$ if for every pattern $y \in \{0,1\}^n$, there exists $h \in \mathbb{H}$ with $h(x_i) = y_i$ for all $i \in [n]$.*

▶ **Definition 14** (VC dimension). *The* VC *dimension of a partial class* $\mathbb{H}$*, denoted by* $\mathrm{VC}(\mathbb{H})$*, is the maximum* d *such that there exists a size-*d *subset of* $\mathcal{X}$ *that is shattered by* $\mathbb{H}$*. If no such largest* d *exists, define* $\mathrm{VC}(\mathbb{H}) = \infty$*.*

Viewed as a matrix, the VC dimension of $\mathbb{H}$ is the maximum d such that the associated partial matrix $\mathbf{M}_{\mathcal{X}\times\mathbb{H}}$ contains a zero/one submatrix of dimensions $d \times 2^d$, where the columns enumerate all d-bit zero/one patterns.

The Littlestone dimension is defined through the shattering of decision trees instead of sets. Consider a full binary decision tree of height d where every non-leaf v is labelled with an element $x_v \in \mathcal{X}$. We identify every node of this tree by the string $v \in \bigcup_{k=0}^{d}\{0,1\}^k$ that corresponds to the path from the root to the node. That is, the root is the empty string, its children are the two elements in $\{0,1\}$, and more generally, the children of a node $\vec{v} \in \{0,1\}^k$ are the two strings $\vec{v}0$ and $\vec{v}1$ in $\{0,1\}^{k+1}$.

We say that such a tree is *shattered* by a partial concept class $\mathbb{H}$ if for every leaf $y \in \{0,1\}^d$, there exists $h \in \mathbb{H}$ such that $h(x_{y[<i]}) = y_i$ for each $i \in [d]$, where $y[< i]$ is the first $(i-1)$-th bits of y. In other words, applying the decision tree to h will result in the leaf y.

▶ **Definition 15** (Littlestone dimension). *The* Littlestone dimension *of a partial concept class* $\mathbb{H}$*, denoted by* $\mathrm{LD}(\mathbb{H})$*, is the maximum* d *such that there is an* $\mathcal{X}$*-labelled height-*d *full binary decision tree that is shattered by* $\mathbb{H}$*. If no such largest* d *exists, define* $\mathrm{LD}(\mathbb{H}) = \infty$*.*

The *dual* of a concept class $\mathbb{H}$ is the concept class with the roles of points and concepts exchanged. Concretely, the dual class of $\mathbb{H} \in \{0,1,\star\}^{\mathcal{X}}$, denoted by $\mathbb{H}^\top$, is the collection of functions $f_x : \mathbb{H} \to \{0,1,\star\}$ for every $x \in \mathcal{X}$, which is defined by $f_x(h) = h(x)$ for each $h \in \mathbb{H}$. When $\mathcal{X}$ is finite, taking the dual corresponds to transposing the matrix of the concept class. The VC-dimension of the dual-class is related to that of the primal class by the inequality

$$\mathrm{VC}(\mathbb{H}^\top) \le 2^{\mathrm{VC}(\mathbb{H})+1} - 1$$

(see [8]), which translates to a lower bound of the VC-dimension of the primal class.

2.2 Disambiguations

We start by formally defining *strong disambiguation* and *disambiguation*. As mentioned earlier, the two notions coincide when the domain $\mathcal{X}$ is finite.

▶ **Definition 16** (Strong Disambiguation). *A* strong disambiguation *of a partial concept class* $\mathbb{H} \subseteq \{0,1,\star\}^{\mathcal{X}}$ *is a total concept class* $\overline{\mathbb{H}} \subseteq \{0,1\}^{\mathcal{X}}$ *such that for every* $h \in \mathbb{H}$*, there exists a* $\bar{h} \in \overline{\mathbb{H}}$ *that is consistent with* h *on the points* $h^{-1}(\{0,1\})$*.*

▶ **Definition 17** (Disambiguation). *A* disambiguation *of a partial concept class* $\mathbb{H} \subseteq \{0,1,\star\}^{\mathcal{X}}$ *is a total concept class* $\overline{\mathbb{H}} \subseteq \{0,1\}^{\mathcal{X}}$ *such that for every* $h \in \mathbb{H}$ *and every finite* $S \subseteq h^{-1}(\{0,1\})$*, there exists* $\bar{h} \in \overline{\mathbb{H}}$ *that is consistent with* h *on* S*.*

A learning algorithm can often provide a disambiguation of a partial concept class by assigning the prediction of the algorithm to unspecified values. Relevant to our work is the disambiguation by the Standard Optimal Algorithm of Littlestone. It was observed in [1] that this algorithm can provide "efficient" disambiguations of partial classes with bounded Littlestone dimensions. We describe this disambiguation next.

Consider a partial concept class $\mathbb{H} \subseteq \{0,1,\star\}^{\mathcal{X}}$ with a countable domain $\mathcal{X}$ and an ordering $x_1, x_2, \ldots$ of $\mathcal{X}$. Given $\vec{b} \in \{0,1,\star\}^k$, let $\mathbb{H}|_{\vec{b}}$ be the set of concepts h where $h(x_i) = b_i$ for every $i \in [k]$. For convenience, we identify $\mathbb{H}|_{()} = \mathbb{H}$. For the purpose of the algorithm, we adopt the convention $\mathrm{LD}(\emptyset) = -1$.

The SOA obtains a disambiguation iteratively and assigns a 0/1 value to each $\star$ in $\mathbb{H}$: for each $k \in \mathbb{N}$, consider $\mathbb{H}|_{\vec{b}}$ for every $\vec{b} \in \{0,1\}^{k-1}$. Pick $c \in \{0,1\}$ which maximizes $\mathrm{LD}(\mathbb{H}|_{\vec{b}c})$, breaking ties by favoring $c = 0$, and assign c to $h(x_k) = \star$ for every $h \in \mathbb{H}|_{\vec{b}\star}$.

We use the notation $\overline{\mathbb{H}}^{\mathrm{SOA}}$ for the SOA disambiguation of a partial concept class $\mathbb{H}$. As mentioned earlier, for a partial class with Littlestone dimension d, Theorem 8 gives an upper bound of $\binom{n}{\leq d} = O(n^d)$ on $\left|\overline{\mathbb{H}}^{\mathrm{SOA}}\right|$. The theorem follows from the mistake bound of SOA for online learning, which relies on the crucial property that at least one choice of $c \in \{0,1\}$ satisfies $\mathrm{LD}(\mathbb{H}|_{\vec{b}c}) \leq \mathrm{LD}(\mathbb{H}|_{\vec{b}}) - 1$ whenever $\mathbb{H}|_{\vec{b}} \neq \emptyset$.

3 Proofs

In this section, we present the proofs of Theorems 6, 10, 11, and 12.

3.1 Proofs of Theorems 6 and 10

As mentioned earlier, Theorem 10 is an immediate corollary of Theorem 6 and Theorem 8. We focus on proving Theorem 6.

Suppose $G = (V, E)$ is the graph supplied by Theorem 11 on $|V| = n = 2^{\Theta(k^4 \log^3 k)}$ vertices with a biclique partition of size $m = 2^{O(k \log^4 k)}$. We will use G to build a partial concept class $\mathbb{G} \subseteq \{0,1,\star\}^V$. This construction is simply the dual of the partial concept class of [1] in their proof of Theorem 6.

Let $\{B_1, \ldots, B_m\}$ be the size-m biclique partition of the edges of G. We fix an orientation $B_i = L_i \times R_i$ for each biclique. Define $\mathbb{G} \subseteq \{0,1,\star\}^V$ as follows. For each $i \in [m]$, associate a concept $h_i : V \to \{0,1,\star\}$ to the biclique B_i, defined by

$$h_i(v) = \begin{cases} 0 & \text{if } v \in L_i \\ 1 & \text{if } v \in R_i \\ \star & \text{otherwise} \end{cases}.$$

We first observe that the Littlestone dimension of this concept class is at most 2.

▷ Claim 18. $\mathrm{LD}(\mathbb{G}) \leq 2$.

Proof. We show that $\mathbb{G}$, viewed as a matrix, does not contain $\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$ as a submatrix and then show that the existence of this submatrix is necessary for having a Littlestone dimension greater than 2.

If $\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$ appears in $\mathbb{G}$ as a submatrix, then there exist $i \neq j$ and $u \neq v \in V(G)$ such that $h_i(v) = h_j(v) = 1$ and $h_i(u) = h_j(u) = 0$. However, this means that $v \in R_i \cap R_j$ and $u \in L_i \cap L_j$, which in turn implies that the edge $\{u, v\}$ is covered by both B_i and B_j, contradicting the assumption that each edge is covered exactly once.

On the other hand, for a class $\mathbb{H} \subseteq \{0,1,\star\}^{\mathcal{X}}$ with Littlestone dimension greater than 2, there exists a shattered $\mathcal{X}$-labelled height-3 full binary tree. In particular, there exists $h, h' \in \mathbb{H}$ and points $x_{()}, x_1, x_{10}$ such that

$$\begin{aligned} h(x_{()}) &= 1, \quad h(x_1) = 0, \quad h(x_{10}) = 0, \\ h'(x_{()}) &= 1, \quad h'(x_1) = 0, \quad h'(x_{10}) = 1. \end{aligned}$$

This means that the submatrix restricted to the columns $\{x_{()}, x_1\}$ and the rows $\{h, h'\}$ is $\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$. We conclude that $\mathrm{LD}(\mathbb{G}) \leq 2$. ◁

Proof of Theorem 6. Consider the partial concept class $\mathbb{G} \subseteq \{0,1,\star\}^V$ above. By Claim 18, we have $\mathrm{LD}(\mathbb{G}) \leq 2$. We show that for every disambiguation $\overline{\mathbb{G}}$ of $\mathbb{G}$, we have $\mathrm{VC}(\overline{\mathbb{G}}) \geq \Omega(\log\log n)$. The argument here is similar to the proof of Theorem 3.

Consider a disambiguation $\overline{\mathbb{G}}$ of $\mathbb{G}$. Note that if two columns u and v are identical in $\overline{\mathbb{G}}$, then there is no edge between u and v, as otherwise, some h_i would have assigned 0 to one of u and v and 1 to the other. Therefore, if two columns u and v are identical, we can color the corresponding vertices with the same color. Consequently, the number of distinct columns in $\overline{\mathbb{G}}$ is at least the chromatic number $\chi(G) \geq 2^{\Omega(k^2)}$. By the SSP lemma (Theorem 7), if $\mathrm{VC}(\overline{\mathbb{G}}^\top) \leq d$, then $\overline{\mathbb{G}}$ must have at most $O(m^d)$ distinct columns. Therefore,

$$2^{\Omega(k^2)} \leq O(m^d).$$

Substituting $m = 2^{\tilde{O}(k)}$ shows that $d = \tilde{\Omega}(k)$. Finally,

$$\mathrm{VC}(\overline{\mathbb{G}}) \geq \Omega(\log \mathrm{VC}(\overline{\mathbb{G}}^\top)) \geq \Omega(\log k) \geq \Omega(\log\log n).$$

This completes the proof of the first part of Theorem 6.

For the second part, we adopt the same construction in the proof of [1, Theorem 11]. Let $\mathbb{H}_\infty$ be a union of disjoint copies of $\mathbb{H}_n$ over $n \in \mathbb{N}$, each supported on a domain $\mathcal{X}_n$ mutually disjoint from others and the partial concepts of $\mathbb{H}_n$ extend outside of its domain by $\star$. Since any disambiguation $\mathbb{H}$ of $\mathbb{H}_\infty$ simultaneously disambiguates all $\mathbb{H}_n$, the Sauer-Shelah-Perles lemma implies that $\mathrm{VC}(\mathbb{H})$ must be infinite. ◀

3.2 Disambiguations via the SOA algorithm (Theorem 12)

This section is dedicated to the proof of Theorem 12.

Proof of Theorem 12. We prove the statement by showing that for every $r, d \in \mathbb{N}$, there exists a partial concept class $\mathbb{H}_{r,d}$ on $[n]$, where $n = d(2^r + r)$, such that $d \leq \mathrm{LD}(\mathbb{H}_{r,d}) \leq d+1$ and the SOA disambiguation has VC dimension $\geq dr$ and at least 2^{dr} distinct rows. The other cases of n follow by trivially extending the domain.

For any $r, d \in \mathbb{N}$, define

$$\mathcal{F}_{r,d} = \{F \subseteq [d2^r] : |F| = d\}.$$

Note that $|\mathcal{F}_{r,d}| = \binom{d2^r}{d} \geq 2^{dr}$. We enumerate the sets in $\mathcal{F}_{r,d}$ as $F_1, \ldots, F_{\binom{d2^r}{d}}$ in the natural order.

Next, we define the partial concept class $\mathbb{H}_{r,d}$ on domain $[d(2^r + r)]$. The class consists of the partial concepts $h_{i,j}$ for $i \in [\binom{d2^r}{d}]$ and $j \in [dr]$ defined as follows:

$$h_{i,j}(x) = \begin{cases} 1 & \text{if } x \in F_i \\ 0 & \text{if } x \in [d2^r] \setminus F_i \\ \beta(i,j) & \text{if } x = d2^r + j \\ \star & \text{otherwise} \end{cases},$$

where $\beta(i,j)$ denotes j-th bit of the dr-bit binary representation of i if $i \in [2^{dr}]$, and $\beta(i,j) = \star$ otherwise.

We first prove that $d \leq \mathrm{LD}(\mathbb{H}_{r,d}) \leq d+1$. Note that there is a set of 2^d indices $I \subseteq [d2^r]$ which

$$\{F_i \cap [d] : i \in I\} = \mathcal{P}([d]),$$

therefore $[d]$ can be shattered by $\{h_{i,1} : i \in I\}$ and hence $\mathrm{LD}(\mathbb{H}_{r,d}) \geq \mathrm{VC}(\mathbb{H}_{r,d}) \geq d$. On the other hand, note that $|f^{-1}(1)| \leq d+1$ for any $f \in \mathbb{H}_{r,d}$, which implies that $\mathrm{LD}(\mathbb{H}_{r,d}) \leq d+1$.

Next, we consider the SOA disambiguation. We claim that $\{d2^r+1, \ldots, d(2^r+r)\}$ is shattered by $\{h_{i,1} : i \in [2^{dr}]\}$. There are no disambiguations for $x \in [d2^r]$. For $x > d2^r$, note that for any $\vec{b} \in \{0,1\}^{x-1}$, either $\mathbb{H}_{r,d}|_{\vec{b}} = \emptyset$ or

$$\mathbb{H}_{r,d}|_{\vec{b}} = \{h_{i,j} : j \in [dr]\},$$

where $i \in [d2^r]$ such that $F_i = \{k \in [d2^r] : b_k = 1\}$. We focus on the latter case and restrict to $i \in [2^{dr}]$. There is exactly one $c \in \{0,1\}$ such that $\mathbb{H}_{r,d}|_{\vec{b}c} \neq \emptyset$, namely $c = \beta(i, x - d2^r)$ and in this case $\mathbb{H}_{r,d}|_{\vec{b}c} = \{h_{i,c}\}$. This forces the algorithm to disambiguate every function f with $\vec{b} \in \{0,1\}^{x-1}$ by setting $f(x) = h_{i,c}(x) = \beta(i, x - d2^r)$. In this manner, every $h_{i,j}$ is eventually disambiguated into the same total function:

$$\overline{h_{i,j}}(x) = \begin{cases} 1 & \text{if } x \in F_i \\ 0 & \text{if } x \in [d2^r] \setminus F_i \\ \beta(i, x - d2^r) & \text{if } x > d2^r \end{cases}.$$

In particular, for every $i \in [2^{dr}]$, the bit string $(\overline{h_{i,1}}(d2^r+1), \ldots, \overline{h_{i,1}}(d2^r+dr))$ is the dr-bit binary representation of i. This provides a witness for which $\mathrm{VC}(\overline{\mathbb{H}_{r,d}}^{\mathrm{SOA}}) \geq dr$. ◀

As an illustration, we provide the matrix representation of $\mathbb{H}_{1,2}$ and some essential steps of the SOA disambiguation below in Figure 1.

$$\begin{pmatrix}
1 & 1 & 0 & 0 & 0 & \\
1 & 1 & 0 & 0 & & 0 \\
1 & 0 & 1 & 0 & 0 & \\
1 & 0 & 1 & 0 & & 1 \\
1 & 0 & 0 & 1 & 1 & \\
1 & 0 & 0 & 1 & & 0 \\
0 & 1 & 1 & 0 & 1 & \\
0 & 1 & 1 & 0 & & 1 \\
0 & 1 & 0 & 1 & & \\
0 & 1 & 0 & 1 & & \\
0 & 0 & 1 & 1 & & \\
0 & 0 & 1 & 1 & &
\end{pmatrix}$$

(a) Matrix representation of $\mathbb{H}_{1,2}$: all empty spaces are filled with stars.

$$\begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & & \\
0 & 1 & 0 & 1 & & \\
0 & 0 & 1 & 1 & & \\
0 & 0 & 1 & 1 & &
\end{pmatrix}$$

(b) The SOA disambiguation of $\mathbb{H}_{1,2}$: the shaded entries indicate where the shattering occurs.

Figure 1 $\mathbb{H}_{1,2}$ and its SOA disambiguation.

3.3 Small-size refutation of the Alon-Saks-Seymour conjecture (Theorem 11)

In this section, we present the construction of Theorem 11 in detail. The starting point is constructing a Boolean function due to [2] in query complexity. This Boolean function then goes through several reductions to be converted into a graph, as described below.

We first introduce some basic definitions related to the notion of *certificate complexity*. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. For $b \in \{0,1\}$ and an input $x \in f^{-1}(b)$, a partial input $\rho \in \{0,1,\star\}^n$ is called a b-certificate if x is consistent with ρ and for every

$x' \in \{0,1\}^n$ consistent with ρ, we have $f(x') = b$. The size of ρ is the number of non-$\star$ entries of ρ. Define $\mathrm{C}_b(f,x)$ as the smallest size of a b-certificate for x. The b-certificate complexity of f, denoted $\mathrm{C}_b(f)$, is the maximum of $\mathrm{C}_b(f,x)$ over all $x \in f^{-1}(b)$.

The *unambiguous* b-certificate complexity of f, denoted $\mathrm{UC}_b(f)$, is the smallest k such that

1. Every input $x \in f^{-1}(b)$ has a b-certificate ρ_x of size at most k;
2. For every $x \neq y$ in $f^{-1}(b)$, we have $\rho_x \neq \rho_y$.

The main result of [2] is the following separation between UC_1 and C_0.

▶ **Theorem 19** ([2, Theorem 1]). *There is a function* $f : \{0,1\}^{12n^4 \log^2 n} \to \{0,1\}$ *such that* $\mathrm{UC}_1(f) = O(n \log^3 n)$ *and* $\mathrm{C}_0(f) = \Omega(n^2)$.

The next step of the construction is to transform the function separating the certificate complexities UC_1 and C_0 into a communication problem. This is achieved by the "lifting" trick: given a function $f : \{0,1\}^n \to \{0,1\}$ and a "gadget" function $g : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}$, we define $f \circ g^n : \{0,1\}^{nk} \times \{0,1\}^{nk} \to \{0,1\}$ as

$$f \circ g^n([x_1,\dots,x_n],[y_1,\dots,y_n]) = f(g(x_1,y_1),\dots,g(x_n,y_n)).$$

For a communication problem $f : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ and $b \in \{0,1\}$, let $\mathrm{Cov}_b(f)$ denote the minimum number of b-monochromatic rectangles required to cover all the b-entries of f. We denote by $\mathrm{UCov}_b(f)$ the minimum number of b-monochromatic rectangles required to *partition* all the b-entries of f. The following theorem provides a connection between the communication complexity parameters and the certificate complexity parameters.

▶ **Theorem 20** ([5, Theorem 33]). *There exists a gadget* $g : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}$ *with* $k = \Omega(\log n)$ *such that for every* $f : \{0,1\}^n \to \{0,1\}$*, we have*

$$\log \mathrm{Cov}_b(f \circ g^n) = \Omega(k\, \mathrm{C}_b(f)).$$

Note that for every $b \in \{0,1\}$, we have $\log \mathrm{UCov}_b(f \circ g^n) \leq 2k\, \mathrm{UC}_b(f)$. This combined with Theorem 20 allows one to "lift" the UC_1 vs C_0 separation of Theorem 19 into a UCov_1 vs Cov_0 separation.

▶ **Corollary 21.** *There exists a function* $f : \{0,1\}^{O(n^4 \log^3 n)} \times \{0,1\}^{O(n^4 \log^3 n)} \to \{0,1\}$ *such that*

$$\log \mathrm{Cov}_0(f) = \Omega(n^2) \qquad \textit{and} \qquad \log \mathrm{UCov}_1(f) = n \log^4 n.$$

Next, we show how to convert these communication parameters to graph parameters of the biclique partition number and chromatic number.

▶ **Lemma 22.** *Let* $h : \{0,1\}^t \times \{0,1\}^t \to \{0,1\}$ *be a Boolean function with* $\mathrm{Cov}_0(h) = c$ *and* $\mathrm{UCov}_1(h) = m$*. There exists a graph* $G = (V,E)$ *on at most* 2^{2t} *vertices with* $\mathrm{bp}(G) \leq m^2$ *and* $\chi(G) \geq \sqrt{c}$.

Proof. Define the graph G with $V \coloneqq h^{-1}(0)$ as follows. Two vertices $(x,y),(x',y') \in V$ are adjacent in G iff $h(x,y') = 1$ or $h(x',y) = 1$. By construction, if $\{(x_1,y_1),\dots,(x_\ell,y_\ell)\} \subseteq V$ is an independent set, then $\{x_1,\dots,x_\ell\} \times \{y_1,\dots,y_\ell\}$ is a 0-monochromatic rectangle for h. Thus every proper vertex coloring of G with $\chi(G)$ colors corresponds to a 0-cover of h with $\chi(G)$ many 0-monochromatic rectangles. Therefore, $\chi(G) \geq c$.

We next show that there exists a small set of bicliques such that every edge of E is covered at least once and at most twice by these bicliques. Let $h^{-1}(1) = \bigcup_{i=1}^{m}(A_i \times B_i)$ be a partition of $h^{-1}(1)$ into m many 1-monochromatic rectangles. Note that every 1-monochromatic rectangle $A_i \times B_i$ corresponds to a biclique $Q_i := S_i^- \times S_i^+$ in G, where

$$S_i^- := \{(x,y) \in V(G) : \ x \in A_i\} \text{ and } S_i^+ = \{(x,y) \in V(G) : \ y \in B_i\}.$$

Notice that each edge $\{(x,y),(x',y')\}$ of G is covered at least once by $Q_1, \ldots, Q_m$, and it is covered at most twice, the latter happening when $h(x,y') = h(x',y) = 1$.

We have thus constructed a graph G on at most 2^{2t} vertices such that $\chi(G) \geq c$, and there are at most m bicliques where every edge in G appears in at least one and at most two bicliques.

Define H_2 as the subgraph of G that consists of all the edges covered by exactly two bicliques among $Q_1, \ldots, Q_m$. For every $i, j \in [m]$, define $Q_{ij} = (S_i^- \cap S_j^+) \times (S_i^+ \cap S_j^-)$. Note that each Q_{ij} is a biclique of H_2, and moreover, each edge of H_2 appears in exactly one Q_{ij}. Hence, the biclique partition number of H_2 is at most m^2. Now, if $\chi(H_2) \geq \sqrt{c}$, we obtain H_2 as the desired graph. Suppose otherwise that $\chi(H_2) < \sqrt{c}$, and consider a proper vertex coloring of H_2 with $\sqrt{c}$ colors with color classes $V_1, \ldots, V_{\sqrt{c}}$. Since $\chi(G) \geq c$, there must exist i such that the induced subgraph of G on V_i, denoted by $G[V_i]$, satisfies $\chi(G[V_i]) \geq \sqrt{c}$. Since V_i is an independent set of H_2, thus the restrictions of bicliques $Q_1, \ldots, Q_m$ to V_i form a biclique partition of $G[V_i]$. ◀

Lemma 22 and Corollary 21 together imply Theorem 11.

▶ Remark 23. In addition to providing effective bounds on the size of the graph, Lemma 22 also simplifies the original chain of reductions utilized in prior work [2, 4, 3, 11] toward achieving a super-polynomial separation between the biclique partition and chromatic numbers. We will briefly describe the original proof below and highlight the differences.

(i) Similar to our proof of Theorem 11, the chain of reduction begins with the function f provided by Corollary 21, such that

$$\log \mathrm{Cov}_0(f) = \Omega(n^2) \qquad \text{and} \qquad \log \mathrm{UCov}_1(f) = n \log^4 n.$$

(ii) Yannakasis [11] (see also [4, Figure 1]) showed how to use f to construct a graph F on $\mathrm{UCov}_1(f) = 2^{O(n \log^4 n)}$ vertices such that every Clique-Stable set separator of F is of size at least $\mathrm{Cov}_0(f) = 2^{\Omega(n^2)}$. Here, a Clique-Stable set separator is a collection of cuts in F such that for every disjoint pair (C, I) of a clique C and a stable set I in F, there is a cut (A, B) in the collection with $C \subseteq A$ and $I \subseteq B$.

(iii) Bousquet et. al., [3, Lemma 23] show how to use F to construct a new graph G with the so-called oriented biclique packing number at most $2^{n \log^4 n}$ and chromatic number $\chi(G) \geq 2^{\Omega(n^2)}$.

(iv) The graph G is then turned into a separation between the biclique partition number and chromatic number in a different graph H via a final reduction in [3].

The above chain of reductions is not sufficient for our application because the graph G of Step (iii) has a vertex for each pair (C, I) of a clique C and a stable set I of F, and as a result, there are no effective upper-bounds on the number of vertices of G. Our proof of Theorem 11 bypasses Step (ii) and employs a more direct approach to construct a small-size graph G that has similar properties to the graph G of Step (iii).

4 Concluding remarks

A few natural questions remain unanswered. The first question is whether a similar example $\mathbb{H}$ for Theorem 6 with the stronger assumption $\mathrm{LD}(\mathbb{H}) = 1$ exists.

▶ **Problem 24.** *Let $\mathbb{H}$ be a partial class with $\mathrm{LD}(\mathbb{H}) = 1$. Does there exist a disambiguation of $\mathbb{H}$ by a total class $\overline{\mathbb{H}}$ such that $\mathrm{LD}(\overline{\mathbb{H}}) < \infty$? Is there one with $\mathrm{VC}(\overline{\mathbb{H}}) < \infty$?*

Theorem 10 shows that for partial classes, having polynomial growth is not a sufficient condition for PAC learnability. A natural candidate reinstatement of the theorem is to work with the more restrictive assumption of linear growth.

▶ **Problem 25.** *Let $\mathbb{H} \subseteq \{0, 1, \star\}^{\mathcal{X}}$ have polynomial growth with parameter 1. Does there exist a disambiguation $\overline{\mathbb{H}}$ of $\mathbb{H}$ with $\mathrm{VC}(\overline{\mathbb{H}}) < \infty$?*

Another question is whether one can improve the lower bound of $\Omega(\log \log n)$ in Theorem 6 to $\Omega(\log n)$.

▶ **Problem 26.** *Can the lower bound in Theorem 6 be improved to $\mathrm{VC}(\overline{\mathbb{H}}) \geq \Omega(\log n)$?*

Forbidding combinatorial patterns

A natural method to prove upper bounds on the VC dimension of a concept class is establishing that it does not contain a specific combinatorial pattern. For example, the construction for Theorem 3 in [1] utilized the fact that the concept class (viewed as a matrix) does not contain the combinatorial patterns $\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$ and $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, which are patterns that are in any concept class $\mathbb{H}$ with $\mathrm{VC}(\mathbb{H}) \geq 2$. Similarly, the dual construction in Theorem 6 forbids the pattern $\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$, a compulsory pattern for any concept class $\mathbb{H}$ with $\mathrm{LD}(\mathbb{H}) \geq 3$.

▶ **Problem 27.** *Suppose $\mathbb{H} \subseteq \{0, 1, \star\}^{[n]}$ does not contain the pattern $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. Does every disambiguation $\overline{\mathbb{H}}$ of $\mathbb{H}$ satisfy $\mathrm{VC}(\overline{\mathbb{H}}) = O(1)$?*

References

1. Noga Alon, Steve Hanneke, Ron Holzman, and Shay Moran. A theory of PAC learnability of partial concept classes. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 658–671. IEEE, 2022.
2. Kaspars Balodis, Shalev Ben-David, Mika Göös, Siddhartha Jain, and Robin Kothari. Unambiguous DNFs and Alon-Saks-Seymour. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science – FOCS 2021*, pages 116–124. IEEE Computer Soc., Los Alamitos, CA, [2022] ©2022. doi:10.1109/FOCS52979.2021.00020.
3. N. Bousquet, A. Lagoutte, and S. Thomassé. Clique versus independent set. *European J. Combin.*, 40:73–92, 2014. doi:10.1016/j.ejc.2014.02.003.
4. Mika Göös. Lower bounds for clique vs. independent set. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science—FOCS 2015*, pages 1066–1076. IEEE Computer Soc., Los Alamitos, CA, 2015. doi:10.1109/FOCS.2015.69.
5. Mika Göös, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles are nonnegative juntas. *SIAM J. Comput.*, 45(5):1835–1869, 2016.
6. Hao Huang and Benny Sudakov. A counterexample to the Alon-Saks-Seymour conjecture and related problems. *Combinatorica*, 32(2):205–219, 2012.

7 Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988. doi:10.1023/a:1022869011914.

8 Jiří Matoušek, editor. *Lectures on Discrete Geometry*. Springer New York, 2002. doi:10.1007/978-1-4613-0039-7.

9 Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

10 Norbert Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1):145–147, 1972.

11 Mihalis Yannakakis. Expressing combinatorial optimization problems by linear programs. *J. Comput. System Sci.*, 43(3):441–466, 1991. doi:10.1016/0022-0000(91)90024-Y.

A General Framework for Learning-Augmented Online Allocation

Ilan Reuven Cohen
Faculty of Engineering, Bar-Ilan University, Ramat Gan, Israel

Debmalya Panigrahi
Department of Computer Science, Duke University, Durham, NC, USA

Abstract

Online allocation is a broad class of problems where items arriving online have to be allocated to agents who have a fixed utility/cost for each assigned item so to maximize/minimize some objective. This framework captures a broad range of fundamental problems such as the Santa Claus problem (maximizing minimum utility), Nash welfare maximization (maximizing geometric mean of utilities), makespan minimization (minimizing maximum cost), minimization of ℓ_p-norms, and so on. We focus on divisible items (i.e., fractional allocations) in this paper. Even for divisible items, these problems are characterized by strong super-constant lower bounds in the classical worst-case online model.

In this paper, we study online allocations in the *learning-augmented* setting, i.e., where the algorithm has access to some additional (machine-learned) information about the problem instance. We introduce a *general* algorithmic framework for learning-augmented online allocation that produces nearly optimal solutions for this broad range of maximization and minimization objectives using only a single learned parameter for every agent. As corollaries of our general framework, we improve prior results of Lattanzi et al. (SODA 2020) and Li and Xian (ICML 2021) for learning-augmented makespan minimization, and obtain the first learning-augmented nearly-optimal algorithms for the other objectives such as Santa Claus, Nash welfare, ℓ_p-minimization, etc. We also give tight bounds on the resilience of our algorithms to errors in the learned parameters, and study the learnability of these parameters.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases Algorithms with predictions, Scheduling algorithms, Online algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.43

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2305.18861`

Funding *Ilan Reuven Cohen*: This research was supported by the Israel Science Foundation grant No. 1737/21.
Debmalya Panigrahi: This research was supported in part by NSF grants CCF-1750140 (CAREER) and CCF-1955703.

1 Introduction

Recent research has focused on obtaining learning-augmented algorithms for many online problems to overcome pessimistic lower bounds in competitive analysis. In this paper, we consider the *online allocation* framework in the learning-augmented setting. In this framework, a set of (divisible) items have to be allocated online among a set of agents, where each agent has a non-negative utility/cost for each item. This framework captures a broad range of classic problems depending on the objective one seeks to optimize. In load balancing (also called *makespan minimization*), the goal is to *minimize the maximum* (MinMax) cost of any agent. A more general goal is to minimize the ℓ_p-norm of the cost vector defined on the agents, for some $p \geq 1$. Both makespan minimization (which is ℓ_∞-minimization) and ℓ_p-minimization are classic problems in scheduling theory and have been extensively studied in competitive analysis. In a different vein, the online allocation framework also applies to

maximization problems, where the allocation of an item obtains some utility for the receiving agent. This includes the famous Santa Claus problem, where the goal is to *maximize the minimum* (MaxMin) utility of any agent, or the maximization of *Nash welfare* which is defined as the geometric mean of the agents' utilities. These maximization objectives have also been been extensively studied, particularly because of their connection to *fairness* in allocations.

Learning-Augmented Online Allocation. In this paper, we consider the online allocation framework in the *learning-augmented* setting. Typically, online allocation problems are characterized by strong super-constant lower bounds in competitive analysis, e.g., $\Omega(\log m)$ for load balancing [7], $\Omega(p)$ for ℓ_p-minimization [4] and $\Omega(m)$ for both Santa Claus (folklore) and Nash welfare [9]. A natural question, then, is whether some additional (machine-learned) information about the problem instance (we call these *learned parameters*) can help overcome these lower bounds and obtain a near-optimal solution. In this paper, we answer this question in the affirmative. In particular, we give a simple, unified framework for obtaining near-optimal (fractional) allocations *using a single learned parameter for every agent.* Our result holds for both maximization and minimization problems, and applies to all objective functions that satisfy two mild technical conditions that we define below. Indeed, the most interesting aspect of our techniques and results is this generality: prior work for online allocation problems, both in *competitive analysis* and *beyond worst-case algorithms*, has typically been specific to the objective at hand, and the techniques for maximization and minimization objectives bear no similarity. In contrast, our techniques surprisingly handles not only a broad range of objectives but applies both to maximization and minimization problems simultaneously. We hope that the generality of our methods will cast a new light on what is one of the most important classes of problems in combinatorial optimization.

Before proceeding further, we define the two technical conditions that the objective function of the online allocation problem needs to satisfy for our results to apply. Let $f : \mathbb{R}^m_{>0} \to \mathbb{R}_{>0}$ be the objective function defined on the vector of costs/utilities of the agents. Then, the conditions are:

- *Monotonicity:* f is said to be *monotone* if the following holds: for any $\ell, \ell' \in \mathbb{R}^m_{>0}$ such that $\ell_i \geq \ell'_i$ for all $i \in [m]$, we have $f(\ell) \geq f(\ell')$.
- *Homogeneity:* f is said to be *homogeneous* if the following holds: for any $\ell, \ell' \in \mathbb{R}^m_{>0}$ such that $\ell'_i = \alpha \cdot \ell_i$ for all $i \in [m]$, then we have $f(\ell') = \alpha \cdot f(\ell)$.

We say an objective function is *well-behaved* if it is both monotone and homogeneous. All online allocation objectives studied previously that we are aware of are well-behaved, including the examples given above.

1.1 Our Results

We now state our main result below:

▶ **Theorem 1** (Informal). *Fix any $\epsilon > 0$. For any online allocation problem with a well-behaved objective, there is an algorithm that achieves a competitive ratio of $1 - \epsilon$ for maximization problems or $1 + \epsilon$ for minimization problems using a single learned parameter for every agent.*

We remark that the role of ϵ in the above theorem is to ensure that the learned parameter vector is of bounded precision.

Comparison to Prior Work. Lattanzi et al. [17] were the first to consider online allocation in a learning-augmented setting. They considered a special case of the load balancing problem called restricted assignment, and showed the surprising result that a single (learned)

parameter for each agent is sufficient to bypass the lower bound and obtain a nearly optimal (fractional) allocation. This result was further generalized by Li and Xian [20] to the full generality of the load balancing problem, but instead of a single parameter, they now required two parameters for every agent. At a high level, their algorithm first uses one set of parameters to restrict the set of agents who can receive an item, and then solves the resulting restricted assignment problem using the second set of parameters. A a corollary of Theorem 1, we improve this result by obtaining a near-optimal solution using a single learned parameter for every agent. In both these papers, as well as in our paper, the (fractional) allocation uses *proportional allocation*. In the setting of online optimization, proportional allocations were used earlier by Agrawal et al. [1] for the (weighted) b-matching problem. As in our paper, they also gave an iterative algorithm for computing the parameters of the allocation. However, because the two problems are structurally very different (e.g., matching is a packing problem while our allocation problems are covering problems), the iterative algorithm in the Agrawal et al. paper is different from ours. To the best of our knowledge, our results for the other problems, namely Santa Claus, Nash welfare maximization, ℓ_p-norm minimization, and other objectives that can be defined in the online allocation framework are the first results in learning-augmented algorithms for these problems.

We now state our additional results.

Resilience to Prediction Error. A key desiderata of learning-augmented online algorithms is resilience to errors in the learned parameters. In other words, one desires that the competitive ratio of the algorithm should gracefully degrade when the learned parameters used in the algorithm deviate from their optimal values. For well-behaved objectives for both minimization and maximization problems, we give an error-resilient algorithm whose competitive ratio degrades gracefully with prediction error:

▶ **Theorem 2** (Informal). *For any online allocation problem with a well-behaved objective, there is an (learning-augmented) algorithm that achieves a competitive ratio of $O(\alpha)$ when the learned parameter input to the algorithm is within a multiplicative factor of α of the optimal learned parameter for every agent. This holds for both minimization and maximization objectives.*

The above theorem is asymptotically tight for the MaxMin objective. But, interestingly, for the MinMax objective we can do better:

▶ **Theorem 3** (Informal). *For the load balancing problem (*MinMax *objective), there is an (learning-augmented) algorithm that achieves a competitive ratio of $O(\log \alpha)$ when the learned parameter input to the algorithm is within a multiplicative factor of α of the optimal learned parameter for every agent. Moreover, the dependence $O(\log \alpha)$ in the above statement is asymptotically tight.*

An analogous statement was previously known only in the special case of restricted assignment [17].

▶ Remark 4. We use a multiplicative measure of error α similar to [17]. For both MinMax and MaxMin objectives, we may assume w.l.o.g. that $\alpha \leq m$. This is because by standard techniques, it is possible to achieve $O(\min(\alpha, m))$ and $O(\log \min(\alpha, m))$ competitiveness for the MaxMin and MinMax objectives respectively. We also show that our bounds are asymptotically tight as a function of α, in addition to matching existing lower bounds for the two problems as a function of m.

Learnability of Parameters. We also study the learnability of the parameters used in our algorithm. Following [20] and [18], we adopt the PAC framework. We assume that each item is drawn independently (but not necessarily identically) from a distribution, and show a bound on the sample complexity of approximately learning the parameter vector under this setting. For the MaxMin and MinMax objectives, we show the following:

▶ **Theorem 5** (Informal). *Fix any $\epsilon > 0$. For the online allocation problem with* MaxMin *or* MinMax *objectives, the sample complexity of learning a parameter vector that gives a $1-\epsilon$ (for* MaxMin*) or $1+\epsilon$ (for* MinMax*) approximation is $O(\frac{m}{\log m} \cdot \log \frac{m}{\epsilon})$.*

We note that a similar result was previously known for the MinMax objective (Li and Xian [20]). We also generalize this result to all well-behaved objectives subject to a technical condition of *superadditivity* for maximization or *subadditivity* for minimization. All the objectives described earlier in the introduction satisfy these conditions.

1.2 Our Techniques

Our learning-augmented online algorithms for both minimization and maximization objectives follow from a single, unified algorithmic framework that we develop in this paper. This is quite surprising because in the worst-case setting, the online algorithms for the different objectives do not share any similarity (indeed have different competitive ratios), particularly between maximization and minimization problems. First, let us first consider the MinMax and MaxMin objectives. To use common terminology across these problems, let us call the cost/utility of an item j to an agent i the *weight* of item j for agent i and denote it $p_{i,j}$. Our common algorithmic framework uses proportional allocation according to the learned parameters of the agents. Let w_i denote the parameter for agent i. Normally, proportional allocation would entail that we allocate a fraction $x_{i,j}$ of item j to agent i where $x_{i,j} = \frac{w_i p_{i,j}}{\sum_{i'} w_{i'} p_{i',j}}$. But, this is clearly not adequate, since it would produce the same allocation for both the MaxMin and MinMax objectives. Specifically, if $p_{i,j}$ is *large* for a pair i, j, then $x_{i,j}$ should be large for the MaxMin objective and small for the MinMax objective respectively. To implement this intuition, we exponentiate the weight $p_{i,j}$ by a fixed value α that depends on the objective (i.e., is different for MaxMin and MinMax) and then allocate using fractions $x_{i,j} = \frac{w_i p_{i,j}^{\alpha}}{\sum_{i'} w_{i'} p_{i',j}^{\alpha}}$. We call this an *exponentiated proportional* allocation (or EP-allocation in short), and call α the *exponentiation constant*.

Let us fix any value of α. It is clear that for both the MinMax and MaxMin objectives, an optimal allocation has *uniform* cumulative fractional weights (called *load*) across all agents. (Note that otherwise, an infinitesimal fraction of an item can be repeatedly moved from the most loaded to the least loaded agent to eventually improve the competitive ratio.) Following this intuition, we define a *canonical allocation* as one that sets learned parameters on the agents in a way that equalizes the loads on all agents. We show that the canonical allocation always exists and is *unique*. Indeed, this is true not only for all EP-allocation algorithms, but for a much broader class of proportional allocation schemes that we called *generalized proportional* allocations (or GP-allocations). In the latter class, we allow any transformation of the weights $p_{i,j}$ before applying proportional allocation. Thus, EP-allocations represent the subclass of GP-allocations where the transformation is exponentiation by the fixed value α. We also give a simple iterative (Sinkhorn-like) algorithm for computing the optimal learned parameters, and establish its convergence properties, for GP-allocations. GP-allocations give an even larger palette of proportional allocation schemes to choose from than EP-allocations, and we hope it will be useful in future work for problem settings that are not covered in this paper (e.g., non-linear utilities).

Finally, we need to set the value of α specifically for the MinMax and MaxMin objectives. Intuitively, it is clear that we need to set α to a large *positive* value for the MaxMin objective and a large *negative* value for the MinMax objective. Indeed, we show that in the limit of $\alpha \to \infty$ and $\alpha \to -\infty$, the canonical allocation defined above recovers optimal allocations for the MaxMin and MinMax objectives respectively. We also show a monotonicity property of the optimal objective (with the value of α) that can be used to set α to a finite value (function of ϵ) and obtain a $1-\epsilon$ (resp., $1+\epsilon$) approximation for the MaxMin (resp., MinMax) objective, for any $\epsilon > 0$.

Now that we have described the EP-allocation scheme for obtaining nearly optimal algorithms for the MinMax and MaxMin objectives, we generalize to all well-behaved objective functions. This is quite simple. The main advantage of the MinMax and MaxMin objectives that is not shared by other objectives is the property that the optimal solution has uniform load across all agents. Now, suppose for a maximization objective, the load of agent i in an optimal solution is s_i (we call this the *scaling parameter* for agent i). For now, suppose these values s_i are also provided offline as a second set of parameters. Then, we can first scale the weights $p_{i,j}$ using these parameters to obtain a new instance $q_{i,j} = \frac{p_{i,j}}{s_i}$. Clearly, the optimal solution for the original instance has uniform load across all agents for the transformed instance. Indeed, by the monotonicity of the maximization objective, this solution for the transformed instance is also optimal for the MaxMin objective. Using the above analysis for the MaxMin objective, we can now claim that there exist learned parameters w_i for $i \in [m]$ such that setting $x_{i,j} = \frac{w_i q_{i,j}^\alpha}{\sum_{i'} w_{i'} q_{i',j}^\alpha}$ gives an optimal solution to the original instance of the problem. Now, note that

$$x_{i,j} = \frac{w_i q_{i,j}^\alpha}{\sum_{i'} w_{i'} q_{i',j}^\alpha} = \frac{(w_i/s_i^\alpha) p_{i,j}^\alpha}{\sum_{i'} (w_{i'}/s_{i'}^\alpha) p_{i',j}^\alpha} = \frac{w_i' p_{i,j}^\alpha}{\sum_{i'} w_{i'}' p_{i',j}^\alpha} \text{ for } w_i' = w_i/s_i^\alpha.$$

It follows that by using learned parameters w_i' in an EP-allocation on the original instance, we can obtain an optimal solution for the original maximization objective. (The case for a minimization objective is identical to the above argument, with the MaxMin objective being replaced by the MinMax objective.) Finally, using the homogeneity of the objective function, we can also set α to a finite value (function of ϵ) and obtain a $1-\epsilon$ (resp., $1+\epsilon$) approximation for the maximization (resp., minimization) objective, for any $\epsilon > 0$.

1.3 Related Work

Learning-augmented online algorithms were pioneered by the work of Lykouris and Vassilvikskii [21] for the caching problem, and has become a very popular research area in the last few years. The basic idea of this framework is to augment an online algorithm with (machine-learned) predictions about the future, which helps overcome pessimistic worst case lower bounds in competitive analysis. Many online allocation problems have been considered in this framework in scheduling [27, 5, 6, 8, 15, 24], online matching [2, 13, 16], ad delivery [22, 19], etc. The reader is referred to the survey by Mitzenmacher and Vassilvitskii [25, 26] for further examples of online learning-augmented algorithms. The papers specifically related to our work are those of Lattanzi et al. [17] and Li and Xian [20] that we described above, and that of Lavastida et al. [18] that focuses on the learnability of the parameters for the same problem. As mentioned earlier, Agrawal et al. [1] used the proportional allocation framework earlier for the online (weighted) b-matching problem, and gave an iterative algorithm for computing the parameters of the allocation.

We now give a brief summary of online allocation in the worst-case model. For minimization problems, two classic objectives are makespan (i.e., ℓ_∞ norm) and ℓ_p norm minimization for $p > 1$. The former was studied in several works (e.g., [7, 3]), eventually

leading to an asymptotically tight bound of $\Theta(\log m)$. This was later generalized to arbitrary ℓ_p norms, and a tight bound of $\Theta(p)$ was obtained for this case [4, 12]. For maximization objectives, there are $\Omega(m)$ lower bounds for many natural objectives such as MaxMin (see, e.g., [14]) and Nash welfare [9]. Some recent work has focused on overcoming these lower bounds using additional information such as monopolist values for the agents [9, 10]. While this improves the competitive ratio to sub-linear in m, lower bounds continue to rule out near-optimal solutions (or even constant factor approximations) that we seek in this paper.

Organization. For most of the paper, we only consider the MinMax and MaxMin objectives. We establish the notation in Section 2 and give an overview of the results. Then, we prove these results by showing properties of GP-allocations in Section 3 and of EP-allocations in Section 4. Next, we give noise resilient algorithms in Section 5 and discuss learnability of the parameters in Section 6. Finally, in Section 7, we extend our results to all well-behaved objective functions via simple reductions to the MaxMin and MinMax objectives.

2 Preliminaries and Results

2.1 Problem Definition

We have n (divisible) items that arrive online and have to be (fractionally) allocated to m agents. The weight of item $j \in [n]$ for agent $i \in [m]$ is denoted $p_{i,j}$ and is revealed when item j arrives. We denote the *weight matrix*

$$P = \begin{bmatrix} p_{1,1} & \cdots & p_{1,n} \\ \vdots & \ddots & \vdots \\ p_{m,1} & \cdots & p_{m,n} \end{bmatrix} \text{ where all } p_{i,j} > 0 \text{ for all } i \in [m], j \in [n].^1$$

A feasible allocation is given by an *assignment matrix*

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix} \text{ where } x_{i,j} \in [0,1] \text{ for all } i \in [m], j \in [n] \text{ and } \sum_{i=1}^{m} x_{i,j} = 1 \text{ for all } j \in [n].$$

Note that every item has to be fully allocated among all the agents. We use $\mathcal{X}$ to denote the set of feasible solutions. The total weight of an agent i corresponding to an allocation X (we call this the *load* of i) is given by

$$\ell_i(P, X) = \sum_{j \in [n]} x_{i,j} \cdot p_{i,j},$$

and the vector of loads of all the agents is denoted $\boldsymbol{\ell}(P, X)$.

The load balancing problem is now defined as

$$\min_{X \in \mathcal{X}} \Big\{ T : \ell_i(P, X) \leq T \text{ for all } i \in [m] \Big\},$$

while the Santa Claus problem is defined as

$$\max_{X \in \mathcal{X}} \Big\{ T : \ell_i(P, X) \geq T \text{ for all } i \in [m] \Big\}.$$

2.2 Exponentiated and Generalized Proportional Allocations

Our algorithmic framework is simple: when allocating item j, we first exponentiate the weights $p_{i,j}$ to $p_{i,j}^{\alpha}$ for some fixed α (called the *exponentiation constant*) that only depends on the objective being optimized. Next, we perform proportional allocation weighted by the learned parameters w_i for agents $i \in [m]$:

$$x_{i,j} = \frac{p_{i,j}^{\alpha} \cdot w_i}{\sum_{i' \in [m]} p_{i',j}^{\alpha} \cdot w_{i'}}.$$

We call this an *exponentiated proportional* allocation or EP-allocation in short.

Our main theorem is the following:

▶ **Theorem 6.** *For the load balancing and Santa Claus problems, there are* EP*-allocations that achieve a competitive ratio of* $1+\epsilon$ *and* $1-\epsilon$ *respectively, for any* $\epsilon > 0$.

The Canonical Allocation. In order to define an EP-allocation and establish Theorem 6, we need to specify two things: the vector of learned parameters $\mathbf{w} \in \mathbb{R}^m_{>0}$ and the exponentiation constant α. First, we focus on the learned parameters. For any fixed α and a weight matrix P, we use learned parameters $\mathbf{w} \in \mathbb{R}^m_{>0}$ that result in *equal load* for every agent. We call this the *canonical allocation*. The corresponding learned parameters and the load of every agent are respectively called the *canonical parameters* (denoted $\mathbf{w}^*$) and the *canonical load* (denoted ℓ^*).

Apriori, it is not clear that a canonical allocation should even exist, and even if it does, that it is unique. Interestingly, we show this existence and uniqueness not just from EP-allocations but for the much broader class of proportional allocations where *any* function $f : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ (called the *transformation function*) can be used to transform the weights rather than just an exponential function. I.e.,

$$x_{i,j} = \frac{f(p_{i,j}) \cdot w_i}{\sum_{i' \in [m]} f(p_{i',j}) \cdot w_{i'}}.$$

We call this a *generalized proportional* allocation or GP-allocation in short.

We show the following theorem for GP-allocations:

▶ **Theorem 7.** *For any weight matrix* $P \in \mathbb{R}^{m \times n}_{>0}$ *and any transformation function* $f : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$*, the canonical load for a* GP*-allocation exists and is unique. Moreover, it is attained by a unique (up to scaling) set of canonical parameters.*

We prove Theorem 7 algorithmically by giving a simple iterative (offline) algorithm that converges to the set of canonical parameters (see Algorithm 1). We will show later that the canonical allocations produced by appropriately setting the value of the exponentiation constant α are respectively optimal (fractional) solutions for the Santa Claus and the load balancing problems. Therefore, an interesting consequence of the iterative convergence of this algorithm to the canonical allocation is that it gives a simple alternative *offline* algorithm for computing an optimal fractional solution for these two problems. To the best of our knowledge, this was not explicitly known before our work.

An interesting direction for future research would be to explore other natural classes of transformation functions, other than the exponential functions considered in this paper. Since Theorem 7 holds for any transformation function, they also admit a canonical allocation,

and it is conceivable that such canonical allocations would optimize objective functions other than the MinMax and MaxMin functions considered here. For example, one natural open problem is following: are there a transformation functions whose canonical allocations correspond to maximizing Nash Social Welfare or minimizing p-norms of loads?

Monotonicity and Convergence of EP-allocations. Now that we have defined the learned parameters in Theorem 6 as the corresponding canonical parameters, we are left to define the values of the exponentiation constant α for the MaxMin and MinMax problems respectively. We show two key properties of canonical loads of EP-allocations. First, we show that the canonical load is monotone nondecreasing with the value of α. This immediately suggests that we should choose the largest possible value of α for the MaxMin problem since it is a maximization problem, and the smallest possible value of α for the MinMax problem since it is a minimization problem. Indeed, the second property that we show is that in the limit of $\alpha \to \infty$, the canonical load converges to the optimal objective for the Santa Claus problem (we denote this optimal value $\ell^{\mathbf{SNT}}$) and in the limit of $\alpha \to -\infty$, the canonical load converges to the optimal objective for the load balancing problem (we denote this optimal value $\ell^{\mathbf{MKS}}$).

For a fixed α, let $X(P, \alpha, \mathbf{w})$ denote the assignment matrix and $\boldsymbol{\ell}(P, \alpha, \mathbf{w})$ the load vector for a learned parameter vector $\mathbf{w}$. Let $\boldsymbol{\ell}^*(P, \alpha)$ denote the corresponding canonical load. We show the following properties of canonical EP-allocations:

▶ **Theorem 8.** *For any weight matrix $P \in \mathbb{R}^{m \times n}_{>0}$, the following properties hold for canonical* EP*-allocations:*

- *The monotonicity property: For $\alpha_1, \alpha_2 \in \mathbb{R}$ such that $\alpha_1 \geq \alpha_2$, we have $\ell^*(P, \alpha_1) \geq \ell^*(P, \alpha_2)$.*
- *The convergence property: $\lim_{\alpha \to \infty} \ell^*(P, \alpha) = \ell^{\boldsymbol{SNT}}(P)$ and $\lim_{\alpha \to -\infty} \ell^*(P, \alpha) = \ell^{\boldsymbol{MKS}}(P)$.*

Clearly, Theorem 8 implies Theorem 6 as a corollary when α is set sufficiently large for the Santa Claus problem and sufficiently small for the load balancing problem.

In the rest of the paper, we will prove Theorem 7 and Theorem 8.

3 Canonical Properties of Generalized Proportional Allocations

In this section, we prove Theorem 7. For notational convenience, we define a transformation matrix $G \in \mathbb{R}^{m \times n}_{>0}$ where $G(i, j) = f(p_{i,j})$ for the transformation function f. Using this notation, we denote by $x_{i,j}(G, \mathbf{w})$ the fractional allocation of item j to agent i, and by $\ell_i(P, G, \mathbf{w})$ the load of agent i (we use $\boldsymbol{\ell}(P, G, \mathbf{w})$ to denote the vector of agent loads) under the GP-allocation corresponding to the transformation matrix G and learned parameters $\mathbf{w}$.

We say two sets of learned parameters $\mathbf{w}, \mathbf{w}'$ are *equivalent* (denoted $\mathbf{w} \equiv \mathbf{w}'$) if there exists some constant $c > 0$ such that $w'_i = c \cdot w_i$ for every agent $i \in [m]$. The following is a simple observation from the GP-allocation scheme that two equivalent sets of learned parameters produce the same allocation:

▶ **Observation 9.** *For any $G \in \mathbb{R}^{m \times n}_{>0}$, if $\mathbf{w} \equiv \mathbf{w}' \in \mathbb{R}^m_{>0}$, then $x_{i,j}(G, \mathbf{w}) = x_{i,j}(G, \mathbf{w}')$ for all i, j.*

We also note that GP-allocations are monotone in the sense that if one agent's parameter decreases while the rest increase, then the allocation on this agent decreases as well.

▶ **Observation 10.** *Consider any $G \in \mathbb{R}^{m\times n}_{>0}$ and any nonzero vector $\boldsymbol{\epsilon} \in \mathbb{R}^m_{\geq 0}$ such that $-w_k < \epsilon_k \leq 0$ for some $k \in [m]$ and $\epsilon_i \geq 0$ for all $i \neq k$. Then, $x_{k,j}(G, \mathbf{w}') < x_{k,j}(G, \mathbf{w})$ for all $j \in [n]$, where $\mathbf{w}' = \mathbf{w} + \boldsymbol{\epsilon}$ and $\mathbf{w}' \neq \mathbf{w}$.*

Our first nontrivial property is that the load vector uniquely determines the learned parameters up to equivalence of the parameters.

▶ **Lemma 11.** *For any $P, G \in \mathbb{R}^{m\times n}_{>0}$, $\ell_i(P, G, \mathbf{w}) = \ell_i(P, G, \mathbf{w}')$ for all $i \in [m]$ if and only if $\mathbf{w} \equiv \mathbf{w}'$.*

Proof. In one direction, if $\mathbf{w} \equiv \mathbf{w}'$, the loads are identical because the allocations are identical (by Observation 9).

We now show the lemma in the opposite direction. Let $k = \arg\min_i \frac{w_i}{w'_i}$ and $c = \frac{w_k}{w'_k}$. Let us define $\hat{\mathbf{w}} = c \cdot \mathbf{w}'$. Then, $\hat{w}_k = w_k$, and $\hat{w}_{i'} = \left(\min_i \frac{w_i}{w'_i}\right) \cdot w'_{i'} \leq w_{i'}$ for all $i' \neq k$. Now, if $\mathbf{w}$ and $\mathbf{w}'$ are not equivalent, then there exists some $i' \in [m]$ such that $\hat{w}_{i'} < w_{i'}$. Therefore, by Observation 10, $x_{k,j}(G, \hat{\mathbf{w}}) > x_{k,j}(G, \mathbf{w})$ for all $j \in [n]$. But, by Observation 9, $x_{k,j}(G, \hat{\mathbf{w}}) = x_{k,j}(G, \mathbf{w}')$ for all $j \in [n]$. Thus, $x_{k,j}(G, \mathbf{w}') > x_{k,j}(G, \mathbf{w})$ for all $j \in [n]$, which contradicts $\ell_k(P, G, \mathbf{w}') = \ell_k(P, G, \mathbf{w})$. ◀

Similarly, we show that if the canonical load exists (i.e., a load vector where all loads are identical), it must be unique.

▶ **Lemma 12.** *For any $P, G \in \mathbb{R}^{m\times n}_{>0}$, if there exist $\mathbf{w}, \mathbf{w}' \in \mathbb{R}^m_{>0}$ such that $\ell_i(P, G, \mathbf{w}) = \ell$ and $\ell_i(P, G, \mathbf{w}') = \ell'$ for all $i \in [m]$, then $\ell = \ell'$.*

Proof. Assume for the purpose of contradiction that there exist $\mathbf{w}, \mathbf{w}' \in \mathbb{R}^m_{>0}$ such that for all $i \in [m]$, $\ell_i(P, G, \mathbf{w}) = \ell$ and $\ell_i(P, G, \mathbf{w}') = \ell'$ but $\ell > \ell'$. Let $k = \arg\min_i \frac{w_i}{w'_i}$ and $c = \frac{w_k}{w'_k}$, and let $\hat{\mathbf{w}} = c \cdot \mathbf{w}'$. We have

$$\ell' = \ell_k(P, G, \mathbf{w}') = \ell_k(P, G, \hat{\mathbf{w}}) \geq \ell_k(P, G, \mathbf{w}) = \ell, \text{which is a contradiction.}$$

Here, the second equality is by Observation 9, and the inequality is by Observation 10, since $\hat{w}_k = w_k$, and $\hat{w}_i \leq w_i$ for $i \in [m]$. ◀

3.1 Convergence of Algorithm 1

The rest of this section focuses on showing the existence of a canonical allocation for GP-allocations. We do so by showing convergence of the following simple iterative algorithm (Algorithm 1):

Note that Algorithm 1 ensures that if the loads of all agents are uniform at any stage, then the iterative process has converged and the algorithm terminates. So, it remains to show that for any $P, G \in \mathbb{R}^{m\times n}_{>0}$, this iterative process reaches a set of parameters $\mathbf{w}^* \in \mathbb{R}^m_{>0}$ such that $\ell_i(P, G, \mathbf{w}^*) = \ell_{i'}(P, G, \mathbf{w}^*)$ for all $i, i' \in [m]$.

Our proof has two parts. The first part shows that the maximum and minimum loads are (weakly) monotone over the course of the iterative process. For this, we focus on a single iteration. For a vector $\boldsymbol{\ell} \in \mathbb{R}^m_{>0}$, let $\ell_{\max} = \max_{i\in[m]} \ell_i$ and $\ell_{\min} = \min_{i\in[m]} \ell_i$ be the maximum and minimum coordinates of $\boldsymbol{\ell}$. We will show that if $\ell^{(r)}_{\max}$ and $\ell^{(r)}_{\min}$ are not equal at the beginning of an iteration, then $\ell^{(r)}_{\max}$ can only decrease (or stay unchanged) and $\ell^{(r)}_{\min}$ can only increase (or stay unchanged) in a single iteration.

▶ **Lemma 13.** *Consider any $P, G \in \mathbb{R}^{m\times n}_{>0}$, $\gamma > 0$. Let $\mathbf{w}, \mathbf{w}', \boldsymbol{\ell}, \boldsymbol{\ell}' \in \mathbb{R}^m_{>0}$ such that $\ell_i = \ell_i(P, G, \mathbf{w})$, $\ell'_i = \ell_i(P, G, \mathbf{w}')$ and $w'_i = \frac{w_i}{\ell_i} \cdot \gamma$ and let $\tilde{p}_i = \sum_j p_{i,j}$. Then, we have $\ell'_i \geq \ell_{\min} / \left(1 - \frac{\ell_i - \ell_{\min}}{\tilde{p}_i}\right)$ and $\ell'_i \leq \ell_{\max} / \left(1 + \frac{\ell_{\max} - \ell_i}{\tilde{p}_i}\right)$.*

Algorithm 1 The iterative algorithm showing the existence of a canonical allocation for GP-allocations.

- Initialize: $\mathbf{w}^{(0)} \leftarrow \mathbf{1}^m$

Iteration r:

- Compute $\boldsymbol{\ell}^{(r)}$ as $\ell_i^{(r)} \leftarrow \ell_i(P, G, \mathbf{w}^{(r)})$, for all $i \in [m]$, where $\ell_i(P, G, \mathbf{w}^{(r)})$ is the load of agent i under the GP-allocation with transformation matrix G and learned parameters $\mathbf{w}^{(r)}$.
- Set $\mathbf{w}^{(r+1)}$ as $w_i^{(r+1)} \leftarrow \frac{w_i^{(r)}}{\ell_i^{(r)}} \cdot \gamma^{(r)}$, for all $i \in [m]$.

  Here, $\gamma^{(r)} \in \mathbb{R}_{>0}$ is a scaling factor whose value does not affect the load (by Observation 9). But, by using, e.g., $\gamma^{(r)} = \ell_1^{(r)}$, we can ensure that the algorithm terminates with a single set of learned parameters instead of repeatedly finding equivalent sets of parameters after it has converged.

In the second part, we show that the ratio $\frac{\ell_{\max}^{(r)}}{\ell_{\min}^{(r)}}$ is strictly decreasing after a finite number of iterations. The proof of this stronger property requires the per-iteration weak monotonicity property that we establish in the first part of the proof. The proof is deferred to the full version of the paper.

▶ **Lemma 14.** *Let $P, G \in \mathbb{R}_{>0}^{m \times n}$ be given fixed matrices. Fix an iteration r in Algorithm 1 where $\ell_{\max}^{(r)} > \ell_{\min}^{(r)}$. Let $\ell_{\max}^{(r)} \geq (1+\epsilon) \cdot \ell_{\min}^{(r)}$ for some $\epsilon \in (0, 1]$. Then, in the next iteration, we have $\ell_{\min}^{(r+1)} \geq (1 + c \cdot \epsilon) \cdot \ell_{\min}^{(r)}$ for some constant $c > 0$ that only depends on P and G.*

Using Lemma 13 and Lemma 14, we complete the proof of Theorem 7.

Proof of Theorem 7. We are given fixed matrices $P, G \in \mathbb{R}_{>0}^{m \times n}$. Let $\ell_{\max}^{(r)}, \ell_{\min}^{(r)}$ denote the maximum and the minimum load respectively in iteration r of Algorithm 1. Let $c > 0$ be the constant (that depends only on P, G) in Lemma 14.

For a non-negative integer a, let r_a be defined recursively as follows:

$$r_a = r_{a-1} + \left\lceil \frac{\log(1 + 2^{-a+1})}{\log(1 + c \cdot 2^{-a})} \right\rceil + 1, \text{ where } r_0 = \left\lceil \frac{\log\left(\ell_{\max}^{(0)} / \ell_{\min}^{(0)}\right)}{\log(1+c)} \right\rceil + 1.$$

We will show for any a, in any iteration $r \geq r_a$, we have $\ell_{\max}^{(r)} / \ell_{\min}^{(r)} \leq 1 + 2^{-a}$. First, we prove it for $a = 0$. If there exists some $r \leq r_0$ such that $\ell_{\max}^{(r)} / \ell_{\min}^{(r)} \leq 2$, then this also holds for $r \geq r_0$ by Lemma 13. Otherwise, for all $r \leq r_0$ we have $\ell_{\max}^{(r)} / \ell_{\min}^{(r)} > 2$. Then, using Lemma 14 with $\epsilon = 1$, we get $\ell_{\min}^{(r+1)} \geq (1+c) \cdot \ell_{\min}^{(r)}$. Therefore, $\ell_{\min}^{(r_0)} \geq (1+c)^{r_0} \cdot \ell_{\min}^{(0)} > \ell_{\max}^{(0)}$ by our choice of r_0. This contradicts Lemma 13, thereby showing that $\ell_{\max}^{(r)} / \ell_{\min}^{(r)} \leq 2$ for any $r \geq r_0$.

Now, we show the inductive case. Assume the inductive hypothesis that $\ell_{\max}^{(r_{a-1})} / \ell_{\min}^{(r_{a-1})} \leq 1 + 2^{-(a-1)}$. We will prove that $\ell_{\max}^{(r_a)} / \ell_{\min}^{(r_a)} \leq 1 + 2^{-(a)}$. The proof is similar to the base case of $a = 0$. If there exists some $r \leq r_a$ such that $\ell_{\max}^{(r)} / \ell_{\min}^{(r)} \leq 1 + 2^{-a}$, then this inequality also holds for any $r \geq r_a$ by Lemma 13. Otherwise, for all $r \leq r_a$ we have $\ell_{\max}^{(r)} / \ell_{\min}^{(r)} > 1 + 2^{-a}$. Then, for all $r_{a-1} \leq r \leq r_a$, using Lemma 14 with $\epsilon = 2^{-a}$, we have $\ell_{\min}^{(r+1)} \geq (1 + c \cdot 2^{-a}) \cdot \ell_{\min}^{(r)}$. Therefore, $\ell_{\min}^{(r_a)} \geq (1 + c \cdot 2^{-a})^{r_a - r_{a-1}} \cdot \ell_{\min}^{(r_{a-1})}$. By our choice of r_a, this implies $\ell_{\min}^{(r_a)} > (1 + 2^{-(a-1)}) \cdot \ell_{\min}^{(r_{a-1})}$. By the induction hypothesis, this implies $\ell_{\min}^{(r_a)} > \ell_{\max}^{(r_{a-1})}$. But, this implies $\ell_{\max}^{(r_a)} > \ell_{\max}^{(r_{a-1})}$, which contradicts Lemma 13. Therefore,

$$\lim_{r \to \infty} \ell_{\max}^{(r)} / \ell_{\min}^{(r)} = 1,$$

and $\ell^*(P, G) = \lim_{r \to \infty} \ell_{\max}^{(r)}$. Moreover, by Lemma 12 this value is uniquely defined and attained by a unique (up to scaling) set of learned parameters. ◀

3.2 Weak Monotonicity of the Maximum and Minimum Loads in Algorithm 1: Proof of Lemma 13

For ease of description, we assume that G and $\mathbf{w}$ are normalized in the following sense:

$$\mathbf{w} = \mathbf{1}^m \text{ and } \sum_j g_{i,j} = 1.$$

This transformation is local to the current iteration, and only for the purpose of this proof. First, we explain why this change of notation is w.l.o.g. Suppose $\hat{G}, \hat{\mathbf{w}}$ represent the actual transformation matrix and learned parameters respectively. Now, we define G as follows:

$$g_{i,j} = \frac{\hat{g}_{i,j} \cdot \hat{w}_i}{\sum_{i' \in [m]} \hat{g}_{i',j} \cdot \hat{w}_{i'}},$$

and our new learned parameters is given by $\mathbf{1}^m$.

Note that the fractional allocation remains unchanged, i.e., $x_{i,j}(\hat{G}, \hat{w}) = x_{i,j}(G, \mathbf{1}^m) = g_{i,j}$, and therefore the loads are also unchanged: $\ell_i = \ell_i(P, \hat{G}, \hat{\mathbf{w}}) = \ell_i(P, G, \mathbf{1}^m) = \sum_{j \in [n]} g_{i,j} \cdot p_{i,j}$. Assume w.l.o.g. (by Observation 9) that $\gamma = \ell_1$, so $\hat{w}'_i = \frac{\hat{w}_i}{\ell_i} \cdot \ell_1$. In the normalized notation, the new parameters are $w'_i = \frac{\ell_1}{\ell_i}$. Again, the allocation is unchanged whether we use the original notation or the normalized one:

$$x_{i,j}(\hat{G}, \hat{\mathbf{w}}') = x_{i,j}(G, \mathbf{w}') = \frac{g_{i,j} \cdot w'_i}{\sum_{i' \in [m]} g_{i',j} \cdot w'_{i'}},$$

and we have, $\ell'_i = \ell_i(P, \hat{G}, \hat{\mathbf{w}}') = \ell_i(P, G, \mathbf{w}')$.

The case of Two Agents. For brevity, we will only consider the case of two agents here, i.e., $m = 2$. The reduction from general m to $m = 2$ is deferred to the full version of the paper.

We have

$$\ell_1 = \sum_j g_{1,j} \cdot p_{1,j} \qquad \text{and} \qquad \ell_2 = \sum_j g_{2,j} \cdot p_{2,j},$$

and the parameter for the second agent after the update is given by: $w'_2 = \frac{\ell_1}{\ell_2}$ (note that $w'_1 = 1$).

Accordingly, the loads after the update are given by:

$$\ell'_1 = \sum_j p_{1,j} \cdot \frac{g_{1,j}}{g_{1,j} + w'_2 \cdot g_{2,j}} \qquad \text{and} \qquad \ell'_2 = \sum_j p_{2,j} \cdot \frac{w'_2 \cdot g_{2,j}}{g_{1,j} + w'_2 \cdot g_{2,j}}.$$

Assume w.l.o.g that $\ell_1 < \ell_2$. First, note that, from monotonicity (Observation 10) we have:

$$\ell'_2 \leq \ell_2 = \ell_{\max} / \left(1 + \frac{\ell_{\max} - \ell_2}{\bar{p}_1}\right).$$

Next, we have to show that

$$\ell'_1 \leq \ell_{\max} / \left(1 + \frac{\ell_{\max} - \ell_1}{\bar{p}_1}\right) = \ell_2 / \left(1 + \frac{\ell_2 - \ell_1}{\bar{p}_1}\right). \tag{1}$$

The proof of the lower bound on ℓ'_1 is similar and is omitted for brevity.

We use the following standard inequality:

▶ **Fact 15** (Milne's Inequality [23]). *For any $a, b \in \mathbb{R}^n$, we have*

$$\sum_{j\in[n]} \frac{a_j \cdot b_j}{a_j + b_j} \quad \leq \quad \frac{\sum_{j\in[n]} a_j \cdot \sum_{j\in[n]} b_j}{\sum_{j\in[n]} (a_j + b_j)}.$$

In using this inequality, we set for any $j \in [n]$,

$$a_j = p_{1,j} \text{ and } b_j = p_{1,j} \cdot \left(\frac{f_j}{w'_2} - 1\right) \text{ where } f_j = g_{1,j} + w'_2 \cdot g_{2,j} = g_{1,j} + w'_2 \cdot (1 - g_{1,j}).$$

First, we calculate each term in Milne's inequality separately:

$$\sum_{j\in[n]} \frac{a_j \cdot b_j}{a_j + b_j} = \sum_{j\in[n]} p_{1,j} \cdot \frac{f_j - w'_2}{f_j} = \sum_{j\in[n]} p_{1,j} \cdot \frac{g_{1,j} + w'_2 \cdot g_{2,j} - w'_2}{f_j} = \sum_{j\in[n]} p_{1,j} \cdot \frac{g_{1,j} - w'_2 \cdot (1 - g_{2,j})}{f_j}$$

$$= \sum_{j\in[n]} p_{1,j} \cdot \frac{g_{1,j} - w'_2 \cdot g_{1,j}}{f_j} = \sum_{j\in[n]} p_{1,j} \cdot g_{1,j} \cdot \frac{1 - w'_2}{f_j} = \ell'_1 \cdot (1 - w'_2).$$

$$\sum_{j\in[n]} a_j = \tilde{p}_1.$$

$$\sum_{j\in[n]} b_j = \sum_{j\in[n]} p_{1,j} \cdot g_{1,j} \cdot \left(\frac{1}{w'_2} - 1\right) = \frac{\ell_1}{w'_2} - \ell_1 = \ell_2 - \ell_1 = \ell_2 \cdot (1 - w'_2).$$

Using Fact 15, we get

$$\ell'_1 \cdot (1 - w'_2) \leq \frac{\tilde{p}_1 \cdot \ell_2}{\ell_2 - \ell_1 + \tilde{p}_1} \cdot (1 - w'_2)$$

By our assumption that $\ell_1 < \ell_2$, and therefore $w'_2 < 1$. We now get Equation (1) by rearranging terms. This completes the proof for the lemma for the case of two agents. As mentioned previously, the reduction from general m to $m = 2$ is deferred to the full version of the paper.

4 Monotonicity and Convergence of Exponentiated Proportional Allocations

In this section, we prove the monotonicity and convergence of EP-allocations (Theorem 8).

First, we establish monotonicity of EP-allocations (first part of Theorem 8). We compare two EP-allocations with arbitrary learned parameters but different exponential constants. We show that with a larger exponent, at least one agent's load will be higher, regardless of the parameters used.

▶ **Lemma 16.** *Fix a weight matrix $P \in \mathbb{R}^{m\times n}_{>0}$. Let $\alpha, \alpha' \in \mathbb{R}$ such that $\alpha > \alpha'$. Now, for any two sets of learned parameters $\mathbf{w}_\alpha, \mathbf{w}_{\alpha'} \in \mathbb{R}^m_{>0}$, there exists an agent $k \in [m]$ such that*

$$\ell_k(P, \alpha, \mathbf{w}_\alpha) \geq \ell_k(P, \alpha', \mathbf{w}_{\alpha'}).$$

Proof. Let Δ denote the vector of differences of loads of the machines in the two allocations, namely $\Delta_i = \ell_i(P, \alpha, \mathbf{w}_\alpha) - \ell_i(P, \alpha', \mathbf{w}_{\alpha'})$. Our goal is to show that Δ has at least one nonnegative coordinate.

To show this, we define a vector in the positive orthant $\mathbf{c} \in \mathbb{R}^m_{>0}$ as follows:

$$c_i = \left(\frac{w_{\alpha,i}}{w_{\alpha',i}}\right)^{\frac{1}{\rho}}, \text{ where } \rho = \alpha - \alpha' > 0$$

and show that this vector $\mathbf{c}$ has a nonnegative inner product with the vector Δ. Note that this suffices since the inner product of a vector with all positive coordinates and one with all negative coordinates cannot be nonnegative. In other words, we want to show the following:

$$\sum_{i\in[m]} c_i \cdot (\ell_i(P,\alpha,w_\alpha) - \ell_i(P,\alpha',w_{\alpha'})) \geq 0. \tag{2}$$

Let us denote the fractional allocation of an item j in the two cases by $x_{i,j}$ and $x'_{i,j}$ respectively. Then, Equation (2) can be rewritten as

$$\sum_{i\in[m]} c_i \cdot \sum_{j\in[n]} p_{i,j} \cdot (x_{i,j} - x'_{i,j}) \geq 0.$$

Changing the order of the two summations, we rewrite further as

$$\sum_{j\in[n]} \left(\sum_{i\in[m]} c_i \cdot p_{i,j} \cdot (x_{i,j} - x'_{i,j}) \right) \geq 0.$$

We will prove this inequality separately for each item $j \in [n]$. Namely, we will show that

$$\sum_{i\in[m]} c_i \cdot p_{i,j} \cdot (x_{i,j} - x'_{i,j}) \geq 0, \text{ for every } j \in [n]. \tag{3}$$

Fix an item j. Since the item is fixed, we will drop j from the notation and define $\mathbf{u} \in \mathbb{R}^m$ as

$$u_i = p_i \cdot (x_i - x'_i).$$

So, we need to show that

$$\mathbf{c} \cdot \mathbf{u} \geq 0, \text{ i.e., } \sum_{i\in[m]} c_i \cdot u_i \geq 0. \tag{4}$$

We have

$$\begin{aligned}
\sum_i c_i \cdot u_i &= \sum_i c_i \cdot p_i \cdot \left(\frac{p_i^\alpha \cdot w_{\alpha,i}}{\sum_{i'} p_{i'}^\alpha \cdot w_{\alpha,i'}} - \frac{p_i^{\alpha'} \cdot w_{\alpha',i}}{\sum_{i'} p_{i'}^{\alpha'} \cdot w_{\alpha',i'}} \right) \\
&= \frac{1}{T} \cdot \sum_i c_i \cdot p_i \cdot \left(p_i^\alpha \cdot w_{\alpha,i} \cdot \left(\sum_{i'} p_{i'}^{\alpha'} \cdot w_{\alpha',i'} \right) - p_i^{\alpha'} \cdot w_{\alpha',i} \cdot \left(\sum_{i'} p_{i'}^{\alpha} \cdot w_{\alpha,i'} \right) \right) \\
&\text{where } T = \left(\sum_{i'} p_{i'}^{\alpha'} \cdot w_{\alpha',i'} \right) \cdot \left(\sum_{i'} p_{i'}^{\alpha} \cdot w_{\alpha,i'} \right).
\end{aligned}$$

Now, on the right hand side of the above equation, we replace α by $\alpha' + \rho$ and $w_{\alpha,i}$ by $w_{\alpha',i} \cdot c_i^\rho$ for every $i \in [m]$. This gives us:

$$\sum_i c_i \cdot u_i =$$
$$\frac{1}{T}\sum_i c_i \cdot p_i \left(p_i^{\alpha'} \cdot p_i^{\rho} \cdot w_{\alpha',i} \cdot c_i^{\rho} \left(\sum_{i'} p_{i'}^{\alpha'} \cdot w_{\alpha',i'} \right) - p_i^{\alpha'} \cdot w_{\alpha',i} \left(\sum_{i'} p_{i'}^{\alpha'} \cdot p_{i'}^{\rho} \cdot w_{\alpha',i'} \cdot c_{i'}^{\rho} \right) \right)$$
$$= \frac{1}{T}\sum_i b_i \left(a_i \cdot b_i^{\rho} \left(\sum_{i'} a_{i'} \right) - a_i \left(\sum_{i'} a_{i'} \cdot b_{i'}^{\rho} \right) \right),$$
where $a_i = w_{\alpha',i} \cdot p_i^{\alpha'}$ and $b_i = p_i \cdot c_i$.

Rearranging the summations on the two terms on the right hand side, we get

$$\sum_i c_i \cdot u_i = \frac{1}{T} \cdot \left(\sum_{i'} a_{i'} \right) \cdot \sum_i a_i \cdot b_i^{\rho+1} - \frac{1}{T} \cdot \left(\sum_{i'} a_{i'} \cdot b_{i'}^{\rho} \right) \cdot \sum_i a_i \cdot b_i$$

Now, let $z_i = a_i^{1/2}$, and $y_i = a_i^{1/2} \cdot b_i^{\rho/2+1/2}$, and $\theta = \frac{|\rho-1|}{\rho+1}$. Then, we have

$$\begin{aligned} T \cdot \sum_i c_i \cdot u_i &= \left(\sum_{i'} a_{i'} \right) \cdot \left(\sum_i a_i \cdot b_i^{\rho+1} \right) - \left(\sum_{i'} a_{i'} \cdot b_{i'}^{\rho} \right) \cdot \left(\sum_i a_i \cdot b_i \right) \\ &= \left(\sum_{i'} z_{i'}^2 \right) \cdot \left(\sum_i y_i^2 \right) - \left(\sum_{i'} z_{i'}^{1+\theta} \cdot y_{i'}^{1-\theta} \right) \cdot \left(\sum_i z_i^{1-\theta} \cdot y_i^{1+\theta} \right). \end{aligned}$$

In the last equation, the first term follows directly from $a_{i'} = z_{i'}^2$ and $a_i \cdot b_i^{\rho+1} = y_i^2$. The second term is more complicated. There are two cases. If $\rho \le 1$, then $a_{i'} \cdot b_{i'}^{\rho} = z_{i'}^{1+\theta} \cdot y_{i'}^{1-\theta}$ and $a_i \cdot b_i = z_i^{1-\theta} \cdot y_i^{1+\theta}$ but if $\rho > 1$, then the roles get reversed and we get $a_{i'} \cdot b_{i'}^{\rho} = z_{i'}^{1-\theta} \cdot y_{i'}^{1+\theta}$ and $a_i \cdot b_i = z_i^{1+\theta} \cdot y_i^{1-\theta}$.

Now, note that $T \ge 0$. So, to establish $\sum_i c_i \cdot u_i \ge 0$, it suffices to show that the right hard side of the equation is nonnegative. We do so by employing Callebaut's inequality which we state below:

▶ **Fact 17** (Callebaut's Inequality [11]). *For any $y, z \in \mathbb{R}^n$ and $\theta \le 1$, we have*

$$\left(\sum_{i'} z_{i'}^2 \right) \cdot \left(\sum_i y_i^2 \right) \ge \left(\sum_{i'} z_{i'}^{1+\theta} \cdot y_{i'}^{1-\theta} \right) \cdot \left(\sum_i z_i^{1-\theta} \cdot y_i^{1+\theta} \right)$$

Note that we can apply Callebaut's inequality because $\rho \ge 0$ implies that $\theta \le 1$. This completes the proof of the lemma. ◀

We now state a lemma asserting the convergence property of EP-allocations. The proof of the lemma, which is constructive in the sense that it gives an algorithm to determine α and $\mathbf{w}_\alpha$ or α' and $\mathbf{w}_{\alpha'}$, is deferred to the full version of the paper.

▶ **Lemma 18.** *Given any weight matrix $P \in \mathbb{R}_{>0}^{m \times n}$ and any constant $\epsilon > 0$,*

(a) *there exists an α (think of α as a sufficiently large negative number) and a corresponding set of parameters $\mathbf{w}_\alpha$ such that $\ell_i(P, \alpha, w_\alpha) \le (1+\epsilon) \cdot \ell^{\boldsymbol{MKS}}(P)$ for all $i \in [m]$.*

(b) *there exists an α' (think of α' as a sufficiently large positive number) and a corresponding set of parameters $\mathbf{w}_{\alpha'}$ such that $\ell_i(P, \alpha', w_{\alpha'}) \ge (1-\epsilon) \cdot \ell^{\boldsymbol{SNT}}(P)$ for all $i \in [m]$.*

We are now ready to complete the proof of Theorem 8.

Proof of Theorem 8. First by Lemma 11, there exists $\mathbf{w}^*_\alpha$ and $\mathbf{w}^*_{\alpha'}$, such that, for all $i \in [m]$, $\ell_i(P, \alpha, \mathbf{w}^*_\alpha) = \ell^*(P, \alpha)$ and $\ell_i(P, \alpha', \mathbf{w}^*_{\alpha'}) = \ell^*(P, \alpha')$. Now, if $\ell^*(P, \alpha) < \ell^*(P, \alpha')$, it would contradict Lemma 16. And combining Lemma 16 and Lemma 18, we completed the proof the second part of Theorem 8. ◀

5 Noise Resilience: Handling Predictions with Error

In this section, we show the noise resilience of our algorithms, namely that we can handle errors in the learned parameters. First, we will show that for both objectives (MaxMin and MinMax), an η-approximate set of learned parameters yields an online algorithm with a competitive ratio of at least/at most η. Second, for the MinMax objective, we show that it is possible to improve the competitive ratio further in the following sense: using a set of learned parameters with a multiplicative error of η with respect to the optimal parameters, we can obtain a $O(\log \eta)$-competitive algorithm. (This was previously shown by Lattanzi et al. [17] but only for the special case of restricted assignment.) We also rule out a similar guarantee for the MaxMin objective, i.e., we show that using η-approximate learned parameters, an algorithm cannot hope to obtain a competitive ratio better than η/c for some constant c. Finally, we show that noise-resilient bounds can be obtained not just for the MinMax and MaxMin objectives but also for any homogeneous monotone minimization or maximization objective function.

Formally, a weight vector $\mathbf{w}$ is η-approximate with respect to a weight vector to $\mathbf{w}^*$, if for any two agents $i, i' \in [m]$, $\frac{w_{i'}}{w_i} \leq \eta \cdot \frac{w^*_{i'}}{w^*_i}$. First, we show a basic noise resilience property that holds for both the MinMax and MaxMin objectives:

▶ **Lemma 19.** *Fix a weight matrix $P \in \mathbb{R}^{m \times n}_{>0}$ and a transformation matrix $G \in \mathbb{R}^{m \times n}_{>0}$. For any two parameter vectors $\mathbf{w}^*, \mathbf{w} \in \mathbb{R}^m_{>0}$, such that $\mathbf{w}$ is η-approximate to $\mathbf{w}^*$, we have that for any agent k:*

$$\frac{\ell_k(P, G, \mathbf{w}^*)}{\eta} \leq \ell_k(P, G, \mathbf{w}) \leq \eta \cdot \ell_k(P, G, \mathbf{w}^*).$$

Proof. Let $y_{i,j} = x_{i,j}(G, \mathbf{w}^*)$ and $z_{i,j} = x_{i,j}(G, \mathbf{w})$ be the respective fractional allocations under proportional allocation using the transformation matrix G. For an agent i, let $\tau_i = w_i / w^*_i$. Then for any two agents i, k, we have that $1/\eta \leq \tau_k/\tau_i \leq \eta$. We have, $\frac{y_{i,j}}{z_{i,j}} = \sum_{i' \in [m]} \frac{\tau_{i'}}{\tau_i} \cdot y_{i',j}$. Therefore,

$$\frac{y_{i,j}}{z_{i,j}} = \sum_{i' \in [m]} \frac{\tau_{i'}}{\tau_i} \cdot y_{i',j} \geq \sum_{i' \in [m]} \frac{1}{\eta} \cdot y_{i',j} = \frac{1}{\eta} \cdot \sum_{i' \in [m]} y_{i',j} = \frac{1}{\eta}, \text{ and}$$

$$\frac{y_{i,j}}{z_{i,j}} = \sum_{i' \in [m]} \frac{\tau_{i'}}{\tau_i} \cdot y_{i',j} \leq \sum_{i' \in [m]} \eta \cdot y_{i',j} = \eta \cdot \sum_{i' \in [m]} y_{i',j} = \eta.$$

Hence, $y_{i,j}/\eta \leq z_{i,j} \leq y_{i,j} \cdot \eta$. Finally, the lemma hold by summing over all items. ◀

The next theorem follows immediately by using a proportional allocation according to the parameter vector $\tilde{\mathbf{w}}$:

▶ **Theorem 20.** *Fix any $P, G \in \mathbb{R}^{m \times n}_{>0}$. Let $\mathbf{w}$ be a learned parameter vector that gives a solution of value γ for the* MaxMin *(resp.,* MinMax*) objective using proportional allocation. Let $\tilde{\mathbf{w}}$ be η-approximate to $\mathbf{w}$ for some $\eta > 1$. Then, there exists an online algorithm that given $\tilde{\mathbf{w}}$ generates a solution with value at least $\Omega(\gamma/\eta)$ (resp., at most $O(\eta\gamma)$).*

Algorithm 2 The online algorithm with predictions.

- Let $\hat{\mathbf{w}}$ a prediction vector and T is the offline optimal objective for the MinMax problem.
- Initialize: $\ell_i \leftarrow 0$ and $\tilde{w}_i \leftarrow \hat{w}_i$, for all $i \in [m]$

For each item j:

- Compute $x_{i,j} = \dfrac{f(p_{i,j}) \cdot \tilde{w}_i}{\sum_{i' \in [m]} f(p_{i',j}) \cdot \tilde{w}_{i'}}$
- $\ell_i \leftarrow \ell_i + p_{i.j} \cdot x_{i,j}$, for all $i \in [m]$
- If exists $i \in [m]$, s.t. $\ell_i > 2 \cdot T$
  Set $\ell_i \leftarrow 0$
  Update $\tilde{w}_i \leftarrow \tilde{w}_i/2$

In particular, if $\mathbf{w}$ is the *optimal* learned parameter vector in the above theorem and $\tilde{\mathbf{w}}$ is an η-approximation to it, then we obtain a competitive ratio of $\Omega(1/\eta)$.

The rest of this section focuses on the MinMax objective for which we can obtain an improved bound. In the next lemma, we establish an upper bound on the load, using Lemma 19 and monotonicity.

▶ **Lemma 21.** *Fix a weight matrix $P \in \mathbb{R}_{>0}^{m \times n}$ and a transformation matrix $G \in \mathbb{R}_{>0}^{m \times n}$. For any two parameter vectors $\mathbf{w}^*, \mathbf{w} \in \mathbb{R}_{>0}^m$ such that there exists an agent $k \in [m]$ for which $w_k^*/2 \leq w_k \leq w_k^*$ and for all other agents $i \neq k$, we have $w_i \geq w_i^*/2$, then the following holds: $\ell_k(P, G, \mathbf{w}) \leq 2 \cdot \ell_k(P, G, \mathbf{w}^*)$.*

Proof. Define $\mathbf{w}'$ where $w'_k = w_k^*$ (i.e., the maximum in its allowed range) and $w'_i = w_i^*/2$ for all $i \neq k$ (i.e., the minimum in their allowed ranges). Now, by monotonicity (Observation 10), we have $x_{k,j}(G, \mathbf{w}) \leq x_{k,j}(G, \mathbf{w}')$, and therefore, $\ell_k(P, G, \mathbf{w}) \leq \ell_k(P, G, \mathbf{w}')$. Note that for $\mathbf{w}'$, for any two agents i_1, i_2, $\frac{w_{i_1}}{w_{i_2}} \leq 2 \cdot \frac{w_{i_1}^*}{w_{i_2}^*}$. Therefore, by Lemma 19, we have $\ell_k(P, G, \mathbf{w}') \leq 2 \cdot \ell_k(P, G, \mathbf{w}^*)$. By combining the two inequalities, we have $\ell_k(P, G, \mathbf{w}) \leq \ell_k(P, G, \mathbf{w}') \leq 2 \cdot \ell_k(P, G, \mathbf{w}^*)$, as required. ◀

Let us denote the predicted learned parameter vector that is given offline to the MinMax algorithm by $\hat{\mathbf{w}}$. We also assume that the algorithm knows the optimal objective value T. By scaling, we assume w.l.o.g that $\tilde{\mathbf{w}}$ is coordinate-wise larger than the optimal learned parameter vector $\mathbf{w}$. The algorithm uses a learned parameter vector $\hat{\mathbf{w}}$ that is iteratively refined, starting with $\hat{\mathbf{w}} = \tilde{\mathbf{w}}$ (see Algorithm 2). In each iteration, the current parameter vector $\hat{\mathbf{w}}$ is used to determine the assignment using proportional allocation until an agent's load in the current phase exceeds $2T$. If this happens for any agent i, then the algorithm halves the value of $\hat{w}_i$, starts a new phase for agent i, and continues doing proportional allocation with the updated learned parameter vector $\hat{\mathbf{w}}$.

▶ **Theorem 22.** *Fix any $P, G \in \mathbb{R}_{>0}^{m \times n}$. Let $\mathbf{w}$ be a learned parameter vector that gives a fractional solution with maximum load T using proportional allocation. Let $\tilde{\mathbf{w}}$ be an η-approximate prediction for $\mathbf{w}$. Then there exists an online algorithm that given $\tilde{\mathbf{w}}$ generates a fractional assignment of items to agents with maximum load at most $O(T \log \eta)$.*

Proof. By the algorithm's definition, an agent's total load is at most $2T$ times the number of phases for the agent. We show that for any agent i, the parameter $\tilde{w}_i$ is always at least $w_i/2$. This immediately implies that the number of phases for machine i is $O(\log \eta)$, which in turn establishes the theorem.

Suppose, for contradiction, in some phase for agent k, we have $\tilde{w}_k < w_k/2$. Moreover, assume w.l.o.g. that agent k is the first agent for which this happens. Clearly, by the algorithm definition, there is a preceding phase for agent k when $\tilde{w}_k < w_k$. Note that, in this entire preceding phase, we have $w_k > \tilde{w}_k \geq w_k/2$, and for all $i \neq k$, $\tilde{w}_i \geq w_i/2$ (by our assumption that k is the first agent to have a violation). However, by Lemma 21, the load of agent k in the preceding phase would be at most $2T$. This contradicts the fact that the algorithm started a new phase for agent k when its load exceeded $2T$ in the preceding phase. ◀

6 Learnability of the Parameters

We consider the learning model introduced by [18], and show that under this model, the parameter vector $\mathbf{w}$ can be learned efficiently from sampled instances. Specifically, we consider the following model: the jth item (i.e., the values of $\mathbf{p}_j = (p_{i,j} : i \in [m])$ is independently sampled from a (discrete) distribution $\mathcal{D}_j$. In other words, the matrix P of utilities is sampled from $\mathcal{D} = \times_j \mathcal{D}_j$.

We set up the model for the MaxMin objective; the setup for the MinMax objective is very similar and is omitted for brevity. Let $T = \mathbb{E}_{P\sim\mathcal{D}}[\ell^{\mathbf{SNT}}(P)]$ be the expected value of the MaxMin objective in the optimal solution for an instance $\ell^{\mathbf{SNT}}(P)$ drawn from $\mathcal{D}$. Morally, we would like to say that we can obtain a vector $\mathbf{w}$ that gives a nearly optimal solution (in expectation) using proportional allocation (i.e., a MaxMin objective of $(1-\epsilon)\cdot T$ in expectation for some error parameter ϵ) using a bounded (as a function of ϵ) number of samples. Similar to [18], we need the following assumption:

Small Items Assumption. Conceptually, this assumption states that each individual item has a small utility compared to the overall utility of any agent in an optimal solution. Precisely, we need $p_{i,j} \leq \frac{T}{\zeta}$ for every $i \in [m], j \in [n]$ for some value $\zeta = \Theta\left(\frac{\log m}{\epsilon^2}\right)$.

Our main theorem in this section for the MaxMin and MinMax objectives are:

▶ **Theorem 23.** *Fix an $\epsilon > 0$ for which the small items assumption holds. Then, there is an (learning) algorithm that samples $O(\frac{m}{\log m} \cdot \log \frac{m}{\epsilon})$ independent instances from $\mathcal{D}$ and outputs (with high probability) a prediction vector $\mathbf{w}$ such that using $\mathbf{w}$ in the proportional allocation scheme gives a* MaxMin *objective of at least $(1 - \Omega(\epsilon)) \cdot T$ in expectation over instances $P \sim \mathcal{D}$.*

▶ **Theorem 24.** *Fix an $\epsilon > 0$ for which the small items assumption holds. Then, there is an (learning) algorithm that samples $O(\frac{m}{\log m} \cdot \log \frac{m}{\epsilon})$ independent instances from $\mathcal{D}$ and outputs (with high probability) a prediction vector $\mathbf{w}$ such that using $\mathbf{w}$ in the proportional allocation scheme gives a* MinMax *objective of at most $(1 + O(\epsilon))T$ in expectation over instances $P \sim \mathcal{D}$.*

Importantly, the description of the entries of $\mathbf{w}$ in Theorem 23 and Theorem 24 are bounded. Specifically, let us define $\mathbf{NET}(m, \epsilon) \subseteq \mathbb{R}^m_{>0}$ as follows: (a) for the MaxMin objective, $\mathbf{w} \in \mathbf{NET}(m, \epsilon)$ if there exist vectors $\mathbf{u}, \delta \in \mathbb{R}^m_{>0}$ such that $w_i = \frac{\delta_i}{u_i^\alpha}$ and $u_i, \delta_i \in \left\{\left(\frac{1}{1-\epsilon}\right)^r : r \in [K]\right\}$ for some $K = O(\frac{m}{\epsilon}\log\frac{m}{\epsilon})$, and (b) for the MinMax objective, $\mathbf{w} \in \mathbf{NET}'(m, \epsilon)$ if there exist vectors $\mathbf{u}, \delta \in \mathbb{R}^m_{>0}$ such that $w_i = \frac{\delta_i}{u_i^\alpha}$ and $u_i, \delta_i \in \{(1+\epsilon)^r : r \in [K]\}$ for some $K = O(\frac{m}{\epsilon}\log\frac{m}{\epsilon})$. The vectors $\mathbf{w}$ produced by the learning algorithm in Theorem 23 and Theorem 24 will satisfy $\mathbf{w} \in \mathbf{NET}(m, \epsilon)$ and $\mathbf{w} \in \mathbf{NET}'(m, \epsilon)$ in the respective cases.

Proof Idea for Theorem 23 and Theorem 24. Recall that in PAC theory, the number of samples needed to learn a function from a family of N functions is about $O(\log N)$. Indeed, restricting $\mathbf{w}$ to be in the class $\mathbf{NET}(m,\epsilon)$ or $\mathbf{NET}'(m,\epsilon)$ serves this role of limiting the hypothesis class to a finite, bounded set since $|\mathbf{NET}(m,\epsilon)| = |\mathbf{NET}'(m,\epsilon)| = K^{2m}$ where $K = O(\frac{m}{\epsilon}\log\frac{m}{\epsilon})$. Using standard PAC theory, this implies that using about $O(m\log K) = O(m\cdot\log\frac{m}{\epsilon})$ samples, we can learn the "best" vector in $\mathbf{NET}(m,\epsilon)$ or $\mathbf{NET}'(m,\epsilon)$ depending on whether we have the MaxMin or MinMax objective. Our main technical work is to show that this "best" vector produces an approximately optimal solution when used in proportional allocation. We state this lemma next:

▶ **Lemma 25.** *Fix any P. For the* MaxMin *objective, there exists a learned parameter vector $\mathbf{w} \in \boldsymbol{NET}(m,\epsilon)$ which when used in* EP*-allocation gives a $1-\Omega(\epsilon)$ approximation. For the* MaxMin *objective, there exists a learned parameter vector $\mathbf{w}' \in \boldsymbol{NET}'(m,\epsilon)$ which when used in* EP*-allocation gives a $1+O(\epsilon)$ approximation.*

The proofs of this lemma and the preceding theorems are deferred to the full version of the paper.

7 Generalization to Well-Behaved Objectives

We first generalize Theorem 6 to all well-behaved functions (Proofs for this section are deferred to the full version).

▶ **Theorem 26.** *Fix any instance of an online allocation problem with divisible items where the goal is to maximize or minimize a monotone homogeneous objective function. Then, there exists an online algorithm and a learned parameter vector in $\mathbb{R}^m_{>0}$ that achieves a competitive ratio of $1-\epsilon$ (for maximization) or $1+\epsilon$ (for minimization).*

Proof. Fix an objection function f and a matrix $P \in \mathbb{R}^{m\times n}_{>0}$. Let ℓ_i^f denote the load of agent i in an optimal solution for objective function f. Also, let $x_{i,j}$ denote the fraction of item j assigned to agent i in this optimal solution. Now, consider the matrix $\tilde{P}$, where $\tilde{p}_{i,j} = \frac{p_{i,j}}{\ell_i^f}$. By the monotonicity property of f, the optimal objective value for $\tilde{P}$ is 1. Therefore, by Theorem 8, there exist α and $\tilde{\mathbf{w}}$, such that using an EP-allocation, we get $\ell^*(\tilde{P},\alpha,\tilde{\mathbf{w}}) \geq 1-\epsilon$ for maximization and $\ell^*(\tilde{P},\alpha,\tilde{\mathbf{w}}) \leq 1+\epsilon$ for minimization. Let $x^*_{i,j}$ be the fraction of item j assigned to agent i in this approximate solution. By the definition of EP-allocation, $x^*_{i,j}$ is proportional to $\tilde{p}^\alpha_{i,j}\cdot\tilde{w}_i = \left(\frac{p_{i,j}}{\ell_i^f}\right)^\alpha \cdot \tilde{w}_i = p^\alpha_{i,j}\cdot\frac{\tilde{w}_i}{(\ell_i^f)^\alpha}$. Thus, if we define $\mathbf{w}$ such that $w_i = \frac{\tilde{w}_i}{(\ell_i^f)^\alpha}$, then the corresponding EP-allocation gives a $(1-\epsilon)$-approximate solution for maximization and $(1+\epsilon)$-approximate solution for minimization. ◀

7.1 Noise Resilience

Next, we consider noise resilience for well-behaved functions, i.e., we generalize Theorem 20 to all well-behaved objective functions. This follows immediately from Lemma 19 and the observation that if all loads are scaled by η, then the objective value for a well-behaved objective is also scaled by η. We state this generalized theorem below:

▶ **Theorem 27.** *Fix any $P, G \in \mathbb{R}^{m\times n}_{>0}$ and any monotone, homogeneous function f. Let $\mathbf{w}$ be a learned parameter vector that gives a solution of objective value γ using* EP*-allocation. Let $\tilde{\mathbf{w}}$ be η-approximate to $\mathbf{w}$ for some $\eta > 1$. Then, the* EP*-allocation for $\tilde{\mathbf{w}}$ gives a solution with value at least γ/η for maximization and at most $\eta\gamma$ for minimization.*

7.2 Learnability

Finally, we consider learnability of parameters for well-behaved functions, i.e., we generalize Theorem 23 and using an by assuming additional property of the objective function:

- For a maximization objective f, we need *superadditivity*: $f(\sum_r \ell_r) \geq \sum_r f(\ell_r)$.
- For a minimization objective f, we need *subadditivity*: $f(\sum_r \ell_r) \leq \sum_r f(\ell_r)$.

▶ **Theorem 28.** *Let f be a well-behaved function. If f is superadditive, the following theorem holds for maximization of f, while if f is subadditive, the following theorem holds for minimization of f. Let T be the expectation of the maximum value of f over instances sampled from $\mathcal{D}$. Fix an $\epsilon > 0$ for which the small items assumption holds. Then, there is an (learning) algorithm that samples $O(\frac{m}{\log m} \cdot \log \frac{m}{\epsilon})$ independent instances from $\mathcal{D}$ and outputs (with high probability) a prediction vector $\mathbf{w}$ such that using $\mathbf{w}$ in the* EP*-allocation gives a value of f that is at least $(1 - \Omega(\epsilon)) \cdot T$ for maximization and at most $(1 + O(\epsilon)) \cdot T$ for minimization, in expectation over instances $P \sim \mathcal{D}$.*

8 Conclusion and Future Directions

In this paper, we gave a unifying framework for designing near-optimal algorithm for fractional allocation problems for essentially all well-studied minimization and maximization objectives in the literature. The existence of this overarching framework is rather surprising because the corresponding worst-case problems exhibit a wide range of behavior in terms of the best competitive ratio achievable, as well as the techniques required to achieve those bounds. It would be interesting to gain further understanding of the optimal learned parameters introduced in this paper. One natural conjecture is that these are optimal dual variables for a suitably defined convex program (for instance, such convex programs are known for restricted assignment and b-matching [1]). Another interesting direction of future work would be to explore other polytopes beyond the simple assignment polytope considered in this paper, such as that corresponding to congestion minimization problems.

References

1 Shipra Agrawal, Morteza Zadimoghaddam, and Vahab Mirrokni. Proportional allocation: Simple, distributed, and diverse matching with high entropy. In *International Conference on Machine Learning*, pages 99–108. PMLR, 2018.

2 Antonios Antoniadis, Themis Gouleakis, Pieter Kleer, and Pavel Kolev. Secretary and online matching problems with machine learned advice. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020. URL: `https://proceedings.neurips.cc/paper/2020/hash/5a378f8490c8d6af8647a753812f6e31-Abstract.html`.

3 James Aspnes, Yossi Azar, Amos Fiat, Serge A. Plotkin, and Orli Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *J. ACM*, 44(3):486–504, 1997. `doi:10.1145/258128.258201`.

4 Baruch Awerbuch, Yossi Azar, Edward F. Grove, Ming-Yang Kao, P. Krishnan, and Jeffrey Scott Vitter. Load balancing in the l_p norm. In *36th Annual Symposium on Foundations of Computer Science*, pages 383–391. IEEE Computer Society, 1995. `doi:10.1109/SFCS.1995.492494`.

5 Yossi Azar, Stefano Leonardi, and Noam Touitou. Flow time scheduling with uncertain processing time. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1070–1080. ACM, 2021. `doi:10.1145/3406325.3451023`.

6 Yossi Azar, Stefano Leonardi, and Noam Touitou. Distortion-oblivious algorithms for minimizing flow time. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 252–274. SIAM, 2022. doi:10.1137/1.9781611977073.13.

7 Yossi Azar, Joseph Naor, and Raphael Rom. The competitiveness of on-line assignments. *J. Algorithms*, 18(2):221–237, 1995. doi:10.1006/jagm.1995.1008.

8 Étienne Bamas, Andreas Maggiori, Lars Rohwedder, and Ola Svensson. Learning augmented energy minimization via speed scaling. In *Advances in Neural Information Processing Systems 33, NeurIPS 2020*, 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/af94ed0d6f5acc95f97170e3685f16c0-Abstract.html.

9 Siddhartha Banerjee, Vasilis Gkatzelis, Artur Gorokh, and Billy Jin. Online nash social welfare maximization with predictions. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 1–19. SIAM, 2022.

10 Siddharth Barman, Arindam Khan, and Arnab Maiti. Universal and tight online algorithms for generalized-mean welfare. In *Thirty-Sixth AAAI Conference on Artificial Intelligence*, pages 4793–4800. AAAI Press, 2022.

11 DK Callebaut. Generalization of the cauchy-schwarz inequality. *Journal of mathematical analysis and applications*, 12(3):491–494, 1965.

12 Ioannis Caragiannis. Better bounds for online load balancing on unrelated machines. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008*, pages 972–981. SIAM, 2008.

13 Justin Y. Chen and Piotr Indyk. Online bipartite matching with predicted degrees. *CoRR*, 2021. arXiv:2110.11439.

14 MohammadTaghi Hajiaghayi, MohammadReza Khani, Debmalya Panigrahi, and Max Springer. Online algorithms for the santa claus problem. In *Advances in Neural Information Processing Systems 35, NeurIPS 2022*, 2022.

15 Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. Non-clairvoyant scheduling with predictions. In *SPAA '21: 33rd ACM Symposium on Parallelism in Algorithms and Architectures, Virtual Event, USA, 6-8 July, 2021*, pages 285–294. ACM, 2021. doi:10.1145/3409964.3461790.

16 Ravi Kumar, Manish Purohit, Aaron Schild, Zoya Svitkina, and Erik Vee. Semi-online bipartite matching. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019*, volume 124 of *LIPIcs*, pages 50:1–50:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ITCS.2019.50.

17 Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020*, pages 1859–1877. SIAM, 2020. doi:10.1137/1.9781611975994.114.

18 Thomas Lavastida, Benjamin Moseley, R. Ravi, and Chenyang Xu. Learnable and instance-robust predictions for online matching, flows and load balancing. In *29th Annual European Symposium on Algorithms, ESA 2021*, volume 204 of *LIPIcs*, pages 59:1–59:17, 2021. doi:10.4230/LIPIcs.ESA.2021.59.

19 Thomas Lavastida, Benjamin Moseley, R. Ravi, and Chenyang Xu. Using predicted weights for ad delivery. In *Applied and Computational Discrete Algorithms, ACDA 2021*, 2021. doi:10.1137/1.9781611976830.3.

20 Shi Li and Jiayi Xian. Online unrelated machine load balancing with predictions revisited. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, 2021. URL: http://proceedings.mlr.press/v139/li21w.html.

21 Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *J. ACM*, 68(4):24:1–24:25, 2021. doi:10.1145/3447579.

22 Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Online optimization with uncertain information. *ACM Trans. Algorithms*, 8(1):2:1–2:29, 2012. doi:10.1145/2071379.2071381.

23 EA Milne. Note on rosseland's integral for the stellar absorption coefficient. *Monthly Notices of the Royal Astronomical Society*, 85:979–984, 1925.

24 Michael Mitzenmacher. Scheduling with predictions and the price of misprediction. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020*, volume 151 of *LIPIcs*, pages 14:1–14:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ITCS.2020.14.

25 Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. In *Beyond the Worst-Case Analysis of Algorithms*, pages 646–662. Cambridge University Press, 2020. doi:10.1017/9781108637435.037.

26 Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. *Commun. ACM*, 65(7):33–35, 2022. doi:10.1145/3528087.

27 Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ML predictions. In *Advances in Neural Information Processing Systems 31, NeurIPS 2018*, 2018. URL: https://proceedings.neurips.cc/paper/2018/hash/73a427badebe0e32caa2e1fc7530b7f3-Abstract.html.

Sample-Based Distance-Approximation for Subsequence-Freeness

Omer Cohen Sidon
Tel Aviv University, Israel

Dana Ron
Tel Aviv University, Israel

Abstract

In this work, we study the problem of approximating the distance to subsequence-freeness in the sample-based distribution-free model. For a given subsequence (word) $w = w_1 \dots w_k$, a sequence (text) $T = t_1 \dots t_n$ is said to contain w if there exist indices $1 \leq i_1 < \dots < i_k \leq n$ such that $t_{i_j} = w_j$ for every $1 \leq j \leq k$. Otherwise, T is w-free. Ron and Rosin (ACM TOCT 2022) showed that the number of samples both necessary and sufficient for one-sided error testing of subsequence-freeness in the sample-based distribution-free model is $\Theta(k/\epsilon)$.

Denoting by $\Delta(T, w, p)$ the distance of T to w-freeness under a distribution $p : [n] \to [0, 1]$, we are interested in obtaining an estimate $\widehat{\Delta}$, such that $|\widehat{\Delta} - \Delta(T, w, p)| \leq \delta$ with probability at least $2/3$, for a given distance parameter δ. Our main result is an algorithm whose sample complexity is $\tilde{O}(k^2/\delta^2)$. We first present an algorithm that works when the underlying distribution p is uniform, and then show how it can be modified to work for any (unknown) distribution p. We also show that a quadratic dependence on $1/\delta$ is necessary.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Property Testing, Distance Approximation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.44

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2305.01358

Funding *Dana Ron*: Supported by the Israel Science Foundation (grant number 1146/18) and the Kadar-family award.

1 Introduction

Distance approximation algorithms, as defined in [29], are sublinear algorithms that approximate (with constant success probability) the distance of objects from satisfying a prespecified property $\mathcal{P}$. Distance approximation (and the closely related notion of tolerant testing) is an extension of property testing [31, 20], where the goal is to distinguish between objects that satisfy a property $\mathcal{P}$ and those that are far from satisfying the property.[1] In this work we consider the property of subsequence-freeness. For a given subsequence (word) $w_1 \dots w_k$ over some alphabet Σ, a sequence (text) $T = t_1 \dots t_n$ over Σ is said to be w-free if there do not exist indices $1 \leq j_1 < \dots < j_k \leq n$ such that $t_{j_i} = w_i$ for every $i \in [k]$.[2]

In most previous works on property testing and distance approximation, the algorithm is allowed query access to the object, and distance to satisfying the property in question, $\mathcal{P}$, is defined as the minimum Hamming distance to an object that satisfies $\mathcal{P}$, normalized by

[1] Tolerant testing algorithms are required to distinguish between objects that are close to satisfying a property and those that are far from satisfying it.

[2] For an integer x, we use $[x]$ to denote the set of integers $\{1, \dots, x\}$

the size of the object. In this work we consider the more challenging, and sometimes more suitable, sample-based model in which the algorithm is only given a random sample from the object. In particular, when the object is a sequence $T = t_1 \dots t_n$, each element in the sample is a pair (j, t_j).

We study both the case in which the underlying distribution according to which each index j is selected (independently) is the uniform distribution over $[n]$, and the more general case in which the underlying distribution is some arbitrary unknown $p : [n] \to [0, 1]$. We refer to the former as the *uniform sample-based model*, and to the latter as the *distribution-free sample-based model*. The distance (to satisfying the property) is determined by the underlying distribution. Namely, it is the minimum total weight according to p of indices j such that t_j must be modified so as to make the sequence w-free. Hence, in the uniform sample-based model, the distance measure is simply the Hamming distance normalized by n.

The related problem of testing the property of subsequence-freeness in the distribution-free sample-based model was studied by Ron and Rosin [30]. They showed that the sample-complexity of one-sided error testing of subsequence-freeness in this model is $\Theta(k/\epsilon)$ (where ϵ is the given distance parameter). A natural question is whether we can design a sublinear algorithm, with small sample complexity, that actually approximates the distance of a text T to w-freeness. It is worth noting that, in general, tolerant testing (and hence distance-approximation) for a property may be much harder than testing the property [18, 3].

1.1 Our results

In what follows, when we say that a sample is selected uniformly from T, we mean that for each sample point (j, t_j), j is selected uniformly and independently from $[n]$. This generalizes to the case in which the underlying distribution is an arbitrary distribution p.

We start by designing a distance-approximation algorithm in the uniform sample-based model. Let $\Delta(T, w)$ denote the distance under the uniform distribution of T from being w-free (which equals the fraction of symbols in T that must be modified so as to obtain a w-free text), and let $\delta \in (0, 1)$ denote the error parameter given to the algorithm.

▶ **Theorem 1.** *There exists a sample-based distance-approximation algorithm for subsequence-freeness under the uniform distribution, that takes a sample of size* $\Theta\left(\frac{k^2}{\delta^2} \cdot \log\left(\frac{k}{\delta}\right)\right)$ *and outputs an estimate* $\widehat{\Delta}$ *such that* $|\widehat{\Delta} - \Delta(T, w)| \leq \delta$ *with probability at least* $2/3$.[3]

We then turn to extending this result to the distribution-free sample-based model. For a distribution $p : [n] \to [0, 1]$, we use $\Delta(T, w, p)$ to denote the distance of T from w-freeness under the distribution p (i.e., the minimum weight, according to p, of the symbols in T that must be modified so as to obtain a w-free text).

▶ **Theorem 2.** *There exists a sample-based distribution-free distance-approximation algorithm for subsequence-freeness, that takes a sample of size* $\Theta\left(\frac{k^2}{\delta^2} \cdot \log\left(\frac{k}{\delta}\right)\right)$ *from* T*, distributed according to an unknown distribution* p*, and outputs an estimate* $\widehat{\Delta}$ *such that* $|\widehat{\Delta} - \Delta(T, w, p)| \leq \delta$ *with probability at least* $\frac{2}{3}$.

Finally, we address the question of how tight is our upper bound. We show (using a fairly simple argument) that the quadratic dependence on $1/\delta$ is indeed necessary, even for the uniform distribution. To be precise, denoting by k_d the number of distinct symbols in w, we

[3] As usual, we can increase the success probability to $1 - \eta$, for any $\eta > 0$ at a multiplicative cost of $O(\log(1/\eta))$ in the sample complexity.

give a lower bound of $\Omega(1/(k_d\delta^2))$ under the uniform distribution (that holds for every w with k_d distinct symbols, sufficiently large n and sufficiently small δ – for a precise statement, see Theorem 27).

1.2 A high-level discussion of our algorithms

Our starting point is a structural characterization of the distance to w-freeness under the uniform distribution, which is proved in [30, Sec. 3.1].[4] In order to state their characterization, we introduce the notion of copies of w in T, and more specifically, role-disjoint copies.

A *copy* of $w = w_1 \dots w_k$ in $T = t_1 \dots t_n$ is a sequence of indices $(j_1, \dots, j_k)$ such that $1 \leq j_1 < \dots < j_k \leq n$ and $t_{j_1} \dots t_{j_k} = w$. It will be convenient to represent a copy as an array C of size k where $C[i] = j_i$. A set of copies $\{C_\ell\}$ is said to be *role-disjoint* if for every $i \in [k]$, the indices in $\{C_\ell[i]\}$ are distinct (though it is possible that $C_\ell[i] = C_{\ell'}[i']$ for $i \neq i'$ (and $\ell \neq \ell'$)). In the special case where the symbols of w are all different from each other, a set of copies is role disjoint simply if it consists of disjoint copies. Ron and Rosin prove [30, Theorem 3.4 + Claim 3.1] that $\Delta(T, w)$ equals the maximum number of role-disjoint copies of w in T, divided by n.

Note that the analysis of the sample complexity of one-sided error sample-based testing of subsequence-freeness translates to bounding the size of the sample that is sufficient and necessary for ensuring that the sample contains evidence that T is not w-free when $\Delta(T, w) > \epsilon$. Here evidence is in the form of a copy of w in the sample, so that the testing algorithm simply checks whether such a copy exists. On the other hand, the question of distance-approximation has a more algorithmic flavor, as it is not determined by the problem what must be done by the algorithm given a sample.

Focusing first on the uniform case, Ron and Rosin used their characterization (more precisely, the direction by which if $\Delta(T, w) > \epsilon$, then T contains more than ϵn role-disjoint copies of w), to prove that a sample of size $\Theta(k/\epsilon)$ contains at least one copy of w with probability at least $2/3$. In this work we go further by designing an algorithm that actually approximates the number of role-disjoint copies of w in T (and hence approximates $\Delta(T, w)$), given a uniformly selected sample from T. It is worth noting that the probability of obtaining a copy in the sample might be quite different for texts that have *exactly the same* number of role-disjoint copies of w (and hence the same distance to being w-free).[5]

In the next subsection we discuss the aforementioned algorithm (for the uniform case), and in the following one address the distribution-free case.

1.2.1 The uniform case

Let $R(T, w)$ denote the number of role-disjoint copies of w in T. In a nutshell, the algorithm works by computing estimates of the numbers of occurrences of symbols of w in a relatively small number of prefixes of T, and using them to derive an estimate of $R(T, w)$. The more precise description of the algorithm and its analysis are based on several combinatorial claims that we present and which we discuss shortly next.

Let $R_i^j(T, w)$ denote the number of role-disjoint copies of the length-i prefix of w, $w_1 \dots w_i$, in the length-j prefix of T, $t_1 \dots t_j$, and let $N_i^j(T, w)$ denote the number of occurrences of the symbol w_i in $t_1 \dots t_j$. In our first combinatorial claim, we show that for every $i \in [k]$

[4] Indeed, Ron and Rosin note that: "The characterization may be useful for proving further results regarding property testing of subsequence-freeness, as well as (sublinear) distance approximation."

[5] For example, consider $w = 1 \dots k$, $T_1 = (1 \dots k)^{n/k}$ and $T_2 = 1^{n/k} \dots k^{n/k}$.

and $j \in [n]$, the value of $R_i^j(T,w)$ can be expressed in terms of the values of $N_i^{j'}(T,w)$ for $j' \in [j]$ (in particular, $N_i^j(T,w)$) and the values of $R_{i-1}^{j'-1}(T,w)$ for $j' \in [j]$. In other words, we establish a recursive expression which implies that if we know what are $R_{i-1}^{j'-1}(T,w)$ and $N_i^{j'}(T,w)$ for every $j' \in [j]$, then we can compute $R_i^j(T,w)$ (and as an end result, compute $R(T,w) = R_k^n(T,w)$).

In our second combinatorial claim we show that if we only want an approximation of $R(T,w)$, then it suffices to define (also in a recursive manner) a measure that depends on the values of $N_i^j(T,w)$ for every $i \in [k]$ but only for a relatively small number of choices of j, which are evenly spaced. To be precise, each such j belongs to the set $J = \{r \cdot \gamma n\}_{r=1}^{1/\gamma}$ for $\gamma = \Theta(\delta/k)$. We prove that since each interval $[(r-1)\gamma n + 1, r\gamma n]$ is of size γn for this choice of γ, we can ensure that the aforementioned measure (which uses only $j \in J$) approximates $R(T,w)$ to within $O(\delta n)$.

We then prove that if we replace each $N_i^j(T,w)$ for these choices of j (and for every $i \in [k]$) by a sufficiently good estimate, then we incur a bounded error in the approximation of $R(T,w)$. Finally, such estimates are obtained using (uniform) sampling, with a sample of size $\tilde{O}(k^2/\delta^2)$.

1.2.2 The distribution-free case

In [30, Sec. 4] it is shown that, given a word w, a text T and a distribution p, it is possible to define a word $\widetilde{w}$ and a text $\widetilde{T}$ for which the following holds. First, $\Delta(T,w,p)$ is closely related to $\Delta(\widetilde{T},\widetilde{w})$. Second, the probability of observing a copy of w in a sample selected from T according to p is closely related to the probability of observing a copy of $\widetilde{w}$ in a sample selected uniformly from $\widetilde{T}$.

We use the first relation stated above (i.e., between $\Delta(T,w,p)$ and $\Delta(\widetilde{T},\widetilde{w})$). However, since we are interested in distance-approximation rather than one-sided error testing, the second relation stated above (between the probability of observing a copy of w in T and that of observing a copy of $\widetilde{w}$ in $\widetilde{T}$) is not sufficient for our needs, and we need to take a different (once again, more algorithmic) path, as we explain shortly next.

Ideally, we would have liked to sample uniformly from $\widetilde{T}$, and then run the algorithm discussed in the previous subsection using this sample (and $\widetilde{w}$). However, we only have sampling access to T according to the underlying distribution p, and we do not have direct sampling access to uniform samples from $\widetilde{T}$. Furthermore, since $\widetilde{T}$ is defined based on (the unknown) p, it is not clear how to determine the aforementioned subset of (evenly spaced) indices J.

For the sake of clarity, we continue the current exposition while making two assumptions. The first is that the distribution p is such that there exists a value β, such that p_j/β is an integer for every $j \in [n]$ (the value of β need not be known). The second is that in w there are no two consecutive symbols that are the same. Under these assumptions, $\widetilde{T} = t_1^{p_1/\beta} \dots t_n^{p_n/\beta}$, $\widetilde{w} = w$, and $\Delta(\widetilde{T},\widetilde{w}) = \Delta(T,w,p)$ (where t_j^x for an integer x is the subsequence that consists of x repetitions of t_j).

Our algorithm for the distribution-free case (working under the aforementioned assumptions), starts by taking a sample distributed according to p and using it to select a (relatively small) subset of indices in $[n]$. Denoting these indices by $b_0, b_1, \dots, b_\ell$, where $b_0 = 0 < b_1 < \dots < b_{\ell-1} < b_\ell = n$, we would have liked to ensure that the weight according to p of each interval $[b_{u-1}+1, b_u]$ is approximately the same (as is the case when considering the intervals defined by the subset J in the uniform case). To be precise, we would have liked each interval to have relatively small weight, while the total number of intervals is not

too large. However, since it is possible that for some single indices $j \in [n]$, the probability p_j is large, we also allow intervals with large weight, where these intervals consist of a single index (and there are few of them).

The algorithm next takes an additional sample, to approximate, for each $i \in [k]$ and $u \in [\ell]$, the weight, according to p, of the occurrences of the symbol w_i in the length-b_u prefix of T. Observe that prefixes of T correspond to prefixes of $\widetilde{T}$. Furthermore, the weight according to p of occurrences of symbols in such prefixes, translates to numbers of occurrences of symbols in the corresponding prefixes in $\widetilde{T}$, normalized by the length of $\widetilde{T}$. The algorithm then uses these approximations to obtain an estimate of $\Delta(\widetilde{T}, \widetilde{w})$.

We note that some pairs of consecutive prefixes in $\widetilde{T}$ might be far apart, as opposed to what we had in the algorithm for the uniform case described in Section 1.2.1. However, this is always due to single-index intervals in T (for j such that p_j is large). Each such interval corresponds to a consecutive subsequence in $\widetilde{T}$ with repetitions of the same symbol, and we show that no additional error is incurred because of such intervals.

1.3 Related results

As we have previously mentioned, the work most closely related to ours is that of Ron and Rosin on distribution-free sample-based testing of subsequence-freeness [30]. For other related results on property testing (e.g., testing other properties of sequences, sample-based testing of other types of properties and distribution-free testing (possibly with queries)), see the introduction of [30], and in particular Section 1.4. For another line of work, on sublinear approximation of the longest increasing subsequence, see [27] and references within. Here we shortly discuss related results on distance approximation / tolerant testing.

As already noted, distance approximation and tolerant testing were first formally defined in [29], and were shown to be significantly harder for some properties in [18, 3]. Almost all previous results are query-based, and where the distance measure is with respect to the uniform distribution. These include [21, 19, 1, 26, 16, 11, 23, 7, 25, 17, 28]. Kopparty and Saraf [24] present results for query-based tolerant testing of linearity under several families of distributions. Berman, Raskhodnikova and Yaroslavtsev [5] give tolerant (query based) L_p-testing algorithms for monotonicity. Berman, Murzbulatov and Raskhodnikova [4] give a sample-based distance-approximation algorithms for image properties that works under the uniform distribution.

Canonne et al. [12] study the property of k-monotonicity of Boolean functions over various posets. A Boolean function over a finite poset domain D is k-monotone if it alternates between the values 0 and 1 at most k times on any ascending chain in D. For the special case of $D = [n]$, the property of k-monotonicity is equivalent to being free of w of length $k+2$ where $w_1 \in \{0,1\}$ and $w_i = 1 - w_{i-1}$ for every $i \in [2, k+2]$. One of their results implies an upper bound of $\widetilde{O}\left(\frac{k}{\delta^3}\right)$ on the sample complexity of distance-approximation for k-monotonicity of functions $f : [n] \to \{0,1\}$ under the uniform distribution (and hence for w-freeness when w is a binary subsequence of a specific form). This result generalizes to k-monotonicity in higher dimensions (at an exponential cost in the dimension d).

Blum and Hu [9] study distance-approximation for k-interval (Boolean) functions over the line in the distribution-free active setting. In this setting, an algorithm gets an unlabeled sample and asks queries on a subset of sample points. Focusing on the sample complexity, they show that for any underlying distribution p on the line, a sample of size $\widetilde{O}\left(\frac{k}{\delta^2}\right)$ is sufficient for approximating the distance to being a k-interval function up to an additive error of δ. This implies a sample-based distribution-free distance-approximation algorithm with the same sample complexity for the special case of being free of the same pair of w's described in the previous paragraph, replacing $k+2$ by $k+1$.

Blais, Ferreira Pinto Jr. and Harms [8] introduce a variant of the VC-dimension and use it to prove lower and upper bounds on the sample complexity of distribution-free testing for a variety of properties. In particular, one of their results implies that the linear dependence on k in the result of [9] is essentially optimal.

Finally we mention that our procedure in the distribution-free case for constructing "almost-equal-weight" intervals by sampling is somewhat reminiscent of techniques used in other contexts of testing when dealing with non-uniform distributions [6, 22, 10].

1.4 Further research

The main open problem left by this work is closing the gap between the upper and lower bounds that we give, and in particular understanding the precise dependence on k, or possibly other parameters determined by w (such as k_d). One step in this direction can be found in the Master Thesis of the first author [13].

1.5 Organization

In Section 2 we present our algorithm for distance-approximation under the uniform distribution. Some of the main details of the distribution-free case appears in Section 3, and in Section 4 we prove our lower bound. All missing details and proofs can be found in the full version of this paper [14].

2 Distance approximation under the uniform distribution

In this section, we address the problem of distance approximation when the underlying distribution is the uniform distribution. As mentioned in the introduction, Ron and Rosin showed [30, Thm. 3.4] that $\Delta(T, w)$ (the distance of T from w-freeness under the uniform distribution), equals the number of role-disjoint copies of w in T, divided by $n = |T|$ (where role-disjoint copies are as defined in the introduction – see Section 1.2). We may use $T[j]$ to denote the j^{th} symbol of T (so that $T[j] = t_j$).

We start by introducing the following notations.

▶ **Definition 3.** *For every $i \in [k]$ and $j \in [n]$, let $N_i^j(T, w)$ denote the number of occurrences of the symbol w_i in the length j prefix of T, $T[1, j] = T[1] \dots T[j]$.*[6] *Let $R_i^j(T, w)$ denote the number of role-disjoint copies of the subsequence $w_1 \dots w_i$ in $T[1, j]$. When $i = k$ and $j = n$, we use the shorthand $R(T, w)$ for $R_k^n(T, w)$ (the total number of role-disjoint copies of w in T).*

Observe that $R_1^j(T, w)$ equals $N_1^j(T, w)$ for every $j \in [n]$.

Since, as noted above, $\Delta(T, w) = R(T, w)/n$, we would like to estimate $R(T, w)$. More precisely, given $\delta > 0$ we would like to obtain an estimate $\widehat{R}$, such that: $\left|\widehat{R} - R(T, w)\right| \leq \delta n$. To this end, we first establish two combinatorial claims. The first claim shows that the value of each $R_i^j(T, w)$ can be expressed in terms of the values of $N_i^{j'}(T, w)$ for $j' \in [j]$ (in particular, $N_i^j(T, w)$) and the values of $R_{i-1}^{j'-1}(T, w)$ for $j' \in [j]$. In other words, if we know what are $R_{i-1}^{j'-1}(T, w)$ and $N_i^{j'}(T, w)$ for every $j' \in [j]$, then we can compute $R_i^j(T, w)$.

▷ Claim 4. For every $i \in \{2, \dots, k\}$ and $j \in [n]$,

$$R_i^j(T, w) = N_i^j(T, w) - \max_{j' \in [j]} \left\{ N_i^{j'}(T, w) - R_{i-1}^{j'-1}(T, w) \right\} .$$

[6] Indeed, if $w_i = w_{i'}$ for $i \neq i'$, then $N_i^j(T, w) = N_{i'}^j(T, w)$ for every j.

Clearly, $R_i^j(T,w) \leq N_i^j(T,w)$ (for every $i \in \{2,\ldots,k\}$ and $j \in [n]$), since each role-disjoint copy of $w_1 \ldots w_i$ in $T[1,j]$ must end with a distinct occurrence of w_i in $T[1,j]$. Claim 4 states by exactly how much is $R_i^j(T,w)$ smaller than $N_i^j(T,w)$. Roughly speaking, the expression $\max_{j' \in [j]} \left\{ N_i^{j'}(T,w) - R_{i-1}^{j'-1}(T,w) \right\}$ accounts for the number of occurrences of w_i in $T[1,j]$ that cannot be used in role-disjoint copies of $w_1 \ldots w_i$ in $T[1,j]$.

Proof. For simplicity (in terms of notation), we prove the claim for the case that $i = k$ and $j = n$. The proof for general $i \in \{2,\ldots,k\}$ and $j \in [n]$ is essentially the same up to renaming of indices. Since T and w are fixed throughout the proof, we shall use the shorthand N_i^j for $N_i^j(T,w)$ and R_i^j for $R_i^j(T,w)$.

For the sake of the analysis, we start by describing a simple greedy procedure, that constructs $R = R_k^n$ role-disjoint copies of w in T. The correctness of this procedure follows from [30, Claim 3.5] and a simple inductive argument (details are provided in the full version of the paper [14]). Every copy C_m, for $m \in [R]$ is an array of size k whose values are monotonically increasing, where for every $i \in [k]$ we have that $C_m[i] \in [n]$, and $T[C_m[i]] = w_i$. Furthermore, for every $i \in [k]$ the indices $C_1[i], \ldots, C_R[i]$ are distinct. For every $m = 1,\ldots,R$ and $i = 1,\ldots,k$, the procedure scans T, starting from $T[C_m[i-1]+1]$ (where we define $C_m[0]$ to be 0) and ending at $T[n]$ until it finds the first index j such that $T[j] = w_i$ and $j \notin \{C_1[i], \ldots, C_{m-1}[i]\}$. It then sets $C_m[i] = j$. For $i > 1$ we say in such a case that the procedure *matches* j to the partial copy $C_m[1], \ldots, C_m[i-1]$.

For $i \in [k]$, define: $G_i = \{j \in [n] : T[j] = w_i\}$. Also define: $G_i^+ = \{j \in G_i : \exists m, C_m[i] = j\}$ and $G_i^- = \{j \in G_i : \nexists m, C_m[i] = j\}$ (recall that $C_m[i]$ is the i-th index in the m-th greedy copy).

It is easy to verify that $|G_i| = N_i^n$, $\left|G_i^+\right| = R_i^n$ and $|G_i| = \left|G_i^+\right| + \left|G_i^-\right|$. To complete the proof, we will show that $\left|G_i^-\right| = \max_{j \in [n]} \left\{ N_i^j - R_{i-1}^{j-1} \right\}$.

Let j^* be an index j that maximizes $\left\{ N_i^j - R_{i-1}^{j-1} \right\}$. In the interval $[j^*]$ we have $N_i^{j^*}$ occurrences of w_i, and in the interval $[j^*-1]$ we only have $R_{i-1}^{j^*-1}$ role-disjoint copies of $w_1 \ldots w_{i-1}$. This implies that in the interval $[j^*]$ there are at least $N_i^{j^*} - R_{i-1}^{j^*-1}$ occurrences of w_i that cannot be the i-th index of any greedy copy, and so we have

$$\left|G_i^-\right| \geq N_i^{j^*} - R_{i-1}^{j^*-1} = \max_{j \in [n]} \left\{ N_i^j - R_{i-1}^{j-1} \right\} . \tag{1}$$

On the other hand, denote by j^{**} the largest index in G_i^-. Since each index $j \in [j^{**}]$ such that $T[j] = w_i$ is either the i-th element of some copy or is not the i-th element of any copy, $N_i^{j^{**}} = R_i^{j^{**}-1} + \left|G_i^-\right|$. We claim that $R_i^{j^{**}-1} = R_{i-1}^{j^{**}-1}$. Otherwise, $R_i^{j^{**}-1} < R_{i-1}^{j^{**}-1}$, in which case the index j^{**} would have to be the the i-th element of a greedy copy. Hence,

$$\left|G_i^-\right| = N_i^{j^{**}} - R_{i-1}^{j^{**}-1} \leq \max_{j \in [n]} \left\{ N_i^j - R_{i-1}^{j-1} \right\} . \tag{2}$$

In conclusion,

$$\left|G_i^-\right| = \max_{j \in [n]} \left\{ N_i^j - R_{i-1}^{j-1} \right\} , \tag{3}$$

and the claim follows. ◀

In order to state our next combinatorial claim, we first introduce one more definition, which will play a central role in obtaining an estimate for $R(T,w)$.

▶ **Definition 5.** *For $\ell \leq n$, let $\mathcal{N}$ be a $k \times \ell$ matrix of non-negative numbers, where we shall use $\mathcal{N}_i^r$ to denote $\mathcal{N}[i][r]$. For every $r \in [\ell]$ let $M_1^r(\mathcal{N}) = \mathcal{N}_1^r$, and for every $i \in \{2, \ldots, k\}$, let*

$$M_i^r(\mathcal{N}) \stackrel{\text{def}}{=} \mathcal{N}_i^r - \max_{r' \leq r} \left\{ \mathcal{N}_i^{r'} - M_{i-1}^{r'}(\mathcal{N}) \right\} .$$

When $i = k$ and $r = \ell$ we use the shorthand $M(\mathcal{N})$ for $M_k^\ell(\mathcal{N})$.

In our second combinatorial claim we show that for an appropriate choice of a matrix $\mathcal{N}$, whose entries are a subset of all values in $\left\{ N_i^j(T,w) \right\}_{i \in [k]}^{j \in [n]}$, we can bound the difference between $M(\mathcal{N})$ and $R(T,w)$. We later use sampling to obtain an estimated version of $\mathcal{N}$.

▷ Claim 6. Let $J = \{j_0, j_1, \ldots, j_\ell\}$ be a set of indices satisfying $j_0 = 0 < j_1 < j_2 < \cdots < j_\ell = n$. Let $\mathcal{N} = \mathcal{N}(J,T,w)$ be the matrix whose entries are $\mathcal{N}_i^r = N_i^{j_r}(T,w)$, for every $i \in [k]$ and $r \in [\ell]$. Then we have

$$|M(\mathcal{N}) - R(T,w)| \leq (k-1) \cdot \max_{\tau \in [\ell]} \{ j_\tau - j_{\tau-1} \} .$$

Proof. Recall that $M(\mathcal{N}) = M_k^\ell(\mathcal{N})$ and $R(T,w) = R_k^{j_\ell}(T,w)$. We shall prove that for every $i \in [k]$ and for every $r \in [\ell]$, $\left| M_i^r(\mathcal{N}) - R_i^{j_r}(T,w) \right| \leq (i-1) \cdot \max_{\tau \in [r]} \{ j_\tau - j_{\tau-1} \}$. We prove this by induction on i.
For $i = 1$ and every $r \in [\ell]$,

$$\left| M_1^r(\mathcal{N}) - R_1^{j_r}(T,w) \right| = \left| N_1^{j_r}(T,w) - N_1^{j_r}(T,w) \right| = 0 \leq (1-1) \cdot \max_{\tau \in [1]} \{ j_\tau - j_{\tau-1} \} , \quad (4)$$

where the first equality follows from the setting of $\mathcal{N}$ and the definitions of $M_1^r(\mathcal{N})$ and $R_1^{j_r}(T,w)$.
For the induction step, we assume the claim holds for $i - 1 \geq 1$ (and every $r \in [\ell]$) and prove it for i. We have,

$$\begin{aligned}
& M_i^r(\mathcal{N}) - R_i^{j_r}(T,w) \\
&= N_i^{j_r}(T,w) - \max_{b \in [r]} \left\{ N_i^{j_b}(T,w) - M_{i-1}^b(\mathcal{N}) \right\} - R_i^{j_r}(T,w) && (5) \\
&= \max_{j \in [j_r]} \left\{ N_i^j(T,w) - R_{i-1}^{j-1}(T,w) \right\} - \max_{b \in [r]} \left\{ N_i^{j_b}(T,w) - M_{i-1}^b(\mathcal{N}) \right\} , && (6)
\end{aligned}$$

where Equation (5) follows from the setting of $\mathcal{N}$ and the definition of $M_i^r(\mathcal{N})$, and Equation (6) is implied by Claim 4. Denote by j^* an index $j \in [j_r]$ that maximizes the first max term and let b^* be the largest index such that $j_{b^*} \leq j^*$. We have:

$$\begin{aligned}
& \max_{j \in [j_r]} \left\{ N_i^j(T,w) - R_{i-1}^{j-1}(T,w) \right\} - \max_{b \in [r]} \left\{ N_i^{j_b}(T,w) - M_{i-1}^b(\mathcal{N}) \right\} \\
&\leq N_i^{j^*}(T,w) - R_{i-1}^{j^*-1}(T,w) - N_i^{j_{b^*}}(T,w) + M_{i-1}^{b^*}(\mathcal{N}) \\
&= N_i^{j^*}(T,w) + R_{i-1}^{j_{b^*}}(T,w) - R_{i-1}^{j_{b^*}}(T,w) - R_{i-1}^{j^*-1}(T,w) - N_i^{j_{b^*}}(T.w) + M_{i-1}^{b^*}(\mathcal{N}) \\
&\leq \left(M_{i-1}^{b^*}(\mathcal{N}) - R_{i-1}^{j_{b^*}}(T,w) \right) + \left(N_i^{j^*}(T,w) - N_i^{j_{b^*}}(T,w) \right) + \left(R_{i-1}^{j_{b^*}}(T,w) - R_{i-1}^{j^*-1}(T,w) \right) \\
&\leq (i-2) \max_{\tau \in [r]} \{ j_\tau - j_{\tau-1} \} + \left(j^* - j^{b^*} \right) + \left(j^{b^*} - (j^* - 1) \right) && (7) \\
&= (i-2) \max_{\tau \in [r]} \{ j_\tau - j_{\tau-1} \} + 1 \\
&\leq (i-2) \max_{\tau \in [r]} \{ j_\tau - j_{\tau-1} \} + \max_{\tau \in [r]} \{ j_\tau - j_{\tau-1} \} \\
&= (i-1) \max_{\tau \in [r]} \{ j_\tau - j_{\tau-1} \} , && (8)
\end{aligned}$$

where in Equation (7) we used the induction hypothesis. By combining Equations (6) and (8), we get that

$$M_i^r(\mathcal{N}) - R_i^{j_r}(T,w) \leq (i-1)\max_{\tau\in[r]}\{j_\tau - j_{\tau-1}\} \ . \tag{9}$$

Similarly to Equation (6),

$$R_i^{j_r}(T,w) - M_i^r(\mathcal{N}) = \max_{b\in[r]}\left\{N_i^{j_b}(T,w) - M_{i-1}^b(\mathcal{N})\right\} - \max_{j\in[j_r]}\left\{N_i^j(T,w) - R_{i-1}^{j-1}(T,w)\right\} \ . \tag{10}$$

Let b^{**} be the index $b\in[r]$ that maximizes the first max term. We have

$$\begin{aligned}
&\max_{b\in[r]}\left\{N_i^{j_b}(T,w) - M_{i-1}^b(\mathcal{N})\right\} - \max_{j\in[j_r]}\left\{N_i^j(T,w) - R_{i-1}^{j-1}(T,w)\right\} \\
&\quad\leq\ N_i^{j_{b^{**}}}(T,w) - M_{i-1}^{b^{**}}(\mathcal{N}) - N_i^{j_{b^{**}}}(T,w) + R_{i-1}^{j_{b^{**}}-1}(T,w) \\
&\quad\leq\ R_{i-1}^{j_{b^{**}}}(T,w) - M_{i-1}^{b^{**}}(\mathcal{N}) \ \leq\ \left|R_{i-1}^{j_{b^{**}}}(T,w) - M_{i-1}^{b^{**}}(\mathcal{N})\right| \\
&\quad\leq\ (i-2)\max_{\tau\in[r]}\{j_\tau - j_{\tau-1}\} \ \leq\ (i-1)\max_{\tau\in[r]}\{j_\tau - j_{\tau-1}\} \ .
\end{aligned} \tag{11}$$

Hence (combining Equations (10) and (11)),[7]

$$R_i^{j_r}(T,w) - M_i^r(\mathcal{N}) \leq (i-1)\max_{\tau\in[r]}\{j_\tau - j_{\tau-1}\} \ . \tag{12}$$

Together, Equations (9) and (12) give us that

$$\left|M_i^r(\mathcal{N}) - R_i^{j_r}(T,w)\right| \leq (i-1)\max_{\tau\in[r]}\{j_\tau - j_{\tau-1}\} \ , \tag{13}$$

and the proof is completed. ◁

In our next claim we bound the difference between $M(\widehat{\mathcal{N}}) - M(\widetilde{\mathcal{N}})$ for any two matrices (with dimensions $k\times\ell$), given a bound on the L_∞ distance between them. We later apply this claim with $\widetilde{\mathcal{N}} = \mathcal{N}$ for $\mathcal{N}$ as defined in Claim 6, and $\widehat{\mathcal{N}}$ being a matrix that contains estimates $\widehat{N}_i^r$ of $N_i^{j_r}(T,w)$ (respectively). We discuss how to obtain $\widehat{\mathcal{N}}$ in Claim 8.

▷ Claim 7. Let $\gamma\in(0,1)$, and let $\widehat{\mathcal{N}}$ and $\widetilde{\mathcal{N}}$ be two $k\times\ell$ matrices. If for every $i\in[t]$ and $r\in[\ell]$, $\left|\widehat{\mathcal{N}}_i^r - \widetilde{\mathcal{N}}_i^r\right| \leq \gamma n$, then $\left|M(\widehat{\mathcal{N}}) - M(\widetilde{\mathcal{N}})\right| \leq (2k-1)\gamma n$.

Proof. We shall prove that for every $t\in[k]$ and for every $r\in[\ell]$, $\left|M_t^r(\widehat{\mathcal{N}}) - M_t^r(\widetilde{\mathcal{N}})\right| \leq (2t-1)\gamma n$. We prove this by induction on t.
For $t=1$ and every $r\in[\ell]$, we have

$$\left|M_1^r(\widehat{\mathcal{N}}) - M_1^r(\widetilde{\mathcal{N}})\right| \ = \ \left|\widehat{\mathcal{N}}_1^r - \widetilde{\mathcal{N}}_1^r\right| \ \leq \ \gamma n \ . \tag{14}$$

Now assume the claim is true for $t-1\geq 1$ and for every $r\in[\ell]$, and we prove it for t. For any $r\in[\ell]$, by the definition of $M_t^r(\cdot)$,

$$\begin{aligned}
&\left|M_t^r(\widehat{\mathcal{N}}) - M_t^r(\widetilde{\mathcal{N}})\right| \\
&\quad=\ \left|\widehat{\mathcal{N}}_t^r - \max_{r''\in[r]}\left\{\widehat{\mathcal{N}}_t^{r''} - M_{t-1}^{r''}(\widehat{\mathcal{N}})\right\} - \widetilde{\mathcal{N}}_t^r + \max_{r'\in[r]}\left\{\widetilde{\mathcal{N}}_t^{r'} - M_{t-1}^{r'}(\widetilde{\mathcal{N}})\right\}\right| \\
&\quad\leq\ \gamma n + \left|\max_{r'\in[r]}\left\{\widetilde{\mathcal{N}}_t^{r'} - M_{t-1}^{r'}(\widetilde{\mathcal{N}})\right\} - \max_{r''\in[r]}\left\{\widehat{\mathcal{N}}_t^{r''} - M_{t-1}^{r''}(\widehat{\mathcal{N}})\right\}\right| \ ,
\end{aligned} \tag{15}$$

[7] It actually holds that $M_i^r(\mathcal{N}) \geq R_i^{j_r}(T,w)$, so that $R_i^{j_r}(T,w) - M_i^r(\mathcal{N}) \leq 0$, but for the sake of simplicity of the inductive argument, we prove the same upper bound on $R_i^{j_r}(T,w) - M_i^r(\mathcal{N})$ as on $M_i^r(\mathcal{N}) - R_i^{j_r}(T,w)$.

where in the last inequality we used the premise of the claim. Assume that the first max term in Equation (15) is at least as large as the second (the case that the second term is larger than the first is dealt with analogously), and let r^* be the index that maximizes the first max term. Then,

$$\begin{aligned}
&\left|\max_{r'\in[r]}\left\{\widetilde{\mathcal{N}}_t^{r'} - M_{t-1}^{r'}(\widetilde{\mathcal{N}})\right\} - \max_{r''\in[r]}\left\{\widehat{\mathcal{N}}_t^{r''} - M_{t-1}^{r''}(\widehat{\mathcal{N}})\right\}\right| \\
&\quad\leq \left|\left(\widetilde{\mathcal{N}}_t^{r^*} - \widehat{\mathcal{N}}_t^{r^*}\right) + \left(M_{t-1}^{r^*}(\widehat{\mathcal{N}}) - M_{t-1}^{r^*}(\widetilde{\mathcal{N}})\right)\right| \\
&\quad\leq \left|\widetilde{\mathcal{N}}_t^{r^*} - \widehat{\mathcal{N}}_t^{r^*}\right| + \left|M_{t-1}^{r^*}(\widehat{\mathcal{N}}) - M_{t-1}^{r^*}(\widetilde{\mathcal{N}})\right| \\
&\quad\leq \gamma n + (2t-3)\gamma n = (2t-2)\gamma n\ , \qquad (16)
\end{aligned}$$

where we used the premise of the claim once again, and the induction hypothesis. The claim follows by combining Equation (15) with Equation (16). ◁

The next claim states that we can obtain good estimates for all values in $\left\{N_i^{j_r}(T,w)\right\}_{i\in[k]}^{r\in[\ell]}$ (with a sufficiently large sample). Its (standard) proof is deferred to the full version of this paper [14].

▷ Claim 8. For any $\gamma \in (0,1)$ and $J = \{j_1, \ldots, j_\ell\}$ (such that $1 \leq j_1 < \cdots < j_\ell = n$), by taking a sample of size $\Theta\left(\frac{\log(k\cdot\ell)}{\gamma^2}\cdot\right)$ from T, we can obtain with probability at least $2/3$ estimates $\left\{\widehat{\mathcal{N}}_i^r\right\}_{i\in[k]}^{r\in[\ell]}$, such that

$$\left|\widehat{\mathcal{N}}_i^r - N_i^{j_r}(T,w)\right| \leq \gamma n\ , \qquad (17)$$

for every $i \in [k]$ and $r \in [\ell]$.

We can now restate and prove our main theorem for distance approximation under the uniform distribution.

▶ **Theorem 1**. *There exists a sample-based distance-approximation algorithm for subsequence-freeness under the uniform distribution, that takes a sample of size* $\Theta\left(\frac{k^2}{\delta^2}\cdot\log\left(\frac{k}{\delta}\right)\right)$ *and outputs an estimate* $\widehat{\Delta}$ *such that* $|\widehat{\Delta} - \Delta(T,w)| \leq \delta$ *with probability at least* $2/3$.[8]

While our focus is on the sample complexity of the algorithm, we note that its running time is linear in the size of the sample.

Proof. The algorithm sets $\gamma = \delta/(3k)$ and $J = \{\gamma n, 2\gamma n, \ldots, n\}$. It first applies Claim 8 with the above setting of γ to obtain the estimates $\left\{\widehat{\mathcal{N}}_i^r\right\}$ for every $i \in [k]$ and $r \in [\ell]$, which with probability at least $2/3$ are as stated in Equation (17). If we take $\widetilde{\mathcal{N}} = \mathcal{N}$ for $\mathcal{N}$ as defined in Claim 6, then the premise of Claim 7 holds. We can hence apply Claim 7, and combining with Claim 6 and the definition of J, we get that with probability at least $2/3$, for the matrix $\widehat{\mathcal{N}}$,

$$\left|M(\widehat{\mathcal{N}}) - R(T,w)\right| \leq (2k-1)\gamma n + (k-1)\gamma n = (3k-2)\gamma n \leq \delta n\ . \qquad (18)$$

The algorithm hence computes $M(\widehat{\mathcal{N}}) = M_k^\ell(\widehat{\mathcal{N}})$ in an iterative manner, based on Definition 5, and outputs $\widehat{\Delta} = M(\widehat{\mathcal{N}})/n$. Since $R(T,w)/n = \Delta(T,w)$, the theorem follows. ◀

[8] As usual, we can increase the success probability to $1-\eta$, for any $\eta > 0$ at a multiplicative cost of $O(\log(1/\eta))$ in the sample complexity.

3 Distribution-free distance approximation

As noted in the introduction, our algorithm for approximating the distance from subsequence-freeness under a general distribution p works by reducing the problem to approximating the distance from subsequence-freeness under the uniform distribution. However, we won't be able to use the algorithm presented in Section 2 as is. There are two main obstacles, explained shortly next. In the reduction, given a word w and access to samples from a text T, distributed according to p, we define a word $\widetilde{w}$ and a text $\widetilde{T}$ such that if we can obtain a good approximation of $\Delta(\widetilde{T}, \widetilde{w})$ then we get a good approximation of $\Delta(T, w, p)$. (Recall that $\Delta(T, w, p)$ denotes the distance of T from being w-free under the distribution p.) However, first, we don't actually have direct access to uniformly distributed samples from $\widetilde{T}$, and second, we cannot work with a set J of indices that induce equally sized intervals (of a bounded size), as we did in Section 2.

We address these challenges (as well as precisely define $\widetilde{T}$ and $\widetilde{w}$) in several stages. We start, in Sections 3.1 and 3.2, by using sampling according to p, in order to construct intervals in T that have certain properties (with sufficiently high probability). The role of these intervals will become clear as we proceed. Due to space constraints, several proofs are deferred to the full version of this paper [14].

3.1 Interval construction and classification

We begin this subsection by defining intervals in $[n]$ that are determined by p (which is unknown to the algorithm). We then construct intervals by sampling from p, where the latter intervals are in a sense approximations of the former (this will be formalized subsequently). Each constructed interval will be classified as either "heavy" or "light", depending on its (approximated) weight according to p. Ideally, we would have liked all intervals to be light, but not too light, so that their number won't be too large (as was the case when we worked under the uniform distribution and simply defined intervals of equal size). However, for a general distribution p we might have single indices $j \in [n]$ for which p_j is large, and hence we also need to allow heavy intervals (each consisting of a single index). We shall make use of the following two definitions.

▶ **Definition 9.** *For any two integers $j_1 \leq j_2$, let $[j_1, j_2]$ denote the interval $\{j_1, \ldots, j_2\}$. For every $j_1, j_2 \in [n]$, define $\mathrm{wt}_p([j_1, j_2]) \stackrel{\text{def}}{=} \sum_{j=j_1}^{j_2} p_j$ to be the weight of the interval $[j_1, j_2]$ according to p. We shall use the shorthand $\mathrm{wt}_p(j)$ for $\mathrm{wt}_p([j, j])$.*

▶ **Definition 10.** *Let S be a multiset of size s, with elements from $[n]$. For every $j \in [n]$, let $N_S(j)$ be the number of elements in S that equal j. For every $j_1, j_2 \in [n]$, define $\mathrm{wt}_S([j_1, j_2]) \stackrel{\text{def}}{=} \frac{1}{s} \sum_{j=j_1}^{j_2} N_S(j)$ to be the estimated weight of the interval $[j_1, j_2]$ according to S. We shall use the shorthand $\mathrm{wt}_S(j)$ for $\mathrm{wt}_S([j, j])$.*

In the next definition, and the remainder of this section, we shall use

$$z \;=\; c_z \frac{k}{\delta}\,, \tag{19}$$

where let $c_z = 100$.

We next define the aforementioned set of intervals, based on p. Roughly speaking, we try to make the intervals as equally weighted as possible, keeping in mind that some indices might have a large weight, so we assign each to an interval of its own.

▶ **Definition 11.** *Define a sequence of indices in the following iterative manner. Let* $h_0 = 0$ *and for* $\ell = 1, 2, \ldots,$ *as long as* $h_{\ell-1} < n$*, let* h_ℓ *be defined as follows. If* $\mathrm{wt}_p(h_{\ell-1}+1) > \frac{1}{8z}$*, then* $h_\ell = h_{\ell-1}+1$*. Otherwise, let* h_ℓ *be the maximum index* $h'_\ell \in [h_{\ell-1}+1, n]$ *such that* $\mathrm{wt}_p([h_{\ell-1}+1, h'_\ell]) \leq \frac{1}{4z}$ *and for every* $h''_\ell \in [h_{\ell-1}+1, h'_\ell]$, $\mathrm{wt}_p(h''_\ell) \leq \frac{1}{8z}$*. Let* L *be such that* $h_L = n$*.*

Based on the indices $\{h_\ell\}_{\ell=0}^{L}$ *defined above, for every* $\ell \in [L]$*, let* $H_\ell = [h_{\ell-1}+1, h_\ell]$ *and let* $\mathcal{H} = \{H_\ell\}_{\ell=1}^{L}$*. We partition* $\mathcal{H}$ *into three subsets as follows. Let* $\mathcal{H}_{sin}$ *be the subset of all* $H \in \mathcal{H}$ *such that* $|H| = 1$ *and* $\mathrm{wt}_p(H) > \frac{1}{8z}$*. Let* $\mathcal{H}_{med}$ *be the set of all* $H \in \mathcal{H}$ *such that* $|H| \neq 1$ *and* $\frac{1}{8z} \leq \mathrm{wt}_p(H) \leq \frac{1}{4z}$*. Let* $\mathcal{H}_{sml}$ *be the set of all* $H \in \mathcal{H}$ *such that* $\mathrm{wt}_p(H) < \frac{1}{8z}$*.*

Observe that since $\mathrm{wt}_p(T) = 1$, then $|\mathcal{H}_{sin} \cup \mathcal{H}_{med}| \leq 8z$. In addition, since between each $H', H'' \in \mathcal{H}_{sml}$ there has to be at least one $H \in \mathcal{H}_{sin}$, then we also have $|\mathcal{H}_{sml}| \leq 8z+1$.

By its definition, $\mathcal{H}$ is determined by p. We next construct a set of intervals $\mathcal{B}$ based on sampling according to p (in a similar, but not identical, fashion to Definition 11). Consider a sample S_1 of size s_1 selected according to p (with repetitions), where s_1 will be set subsequently.

▶ **Definition 12.** *Given a sample* S_1 *(multiset of elements in* $[n]$*) of size* s_1*, determine a sequence of indices in the following iterative manner. Let* $b_0 = 0$ *and for* $u = 1, 2, \ldots,$ *as long as* $b_{u-1} < n$*, let* b_u *be defined as follows. If* $\mathrm{wt}_{S_1}(b_{u-1}+1) > 1/z$*, then* $b_u = b_{u-1}+1$*. Otherwise, let* b_u *be the maximum index* $b'_u \in [b_{u-1}+1, n]$ *such that* $\mathrm{wt}_{S_1}([b_{u-1}+1, b'_u]) \leq \frac{1}{z}$*. Let* U *be such that* $b_U = n$*.*

Based on the indices $\{b_u\}_{u=0}^{U}$ *defined above, for every* $u \in [U]$*, let* $B_u = [b_{u-1}+1, b_u]$*, and let* $\mathcal{B} = \{B_u\}_{u=1}^{U}$*. For every* $u \in [U]$*, if* $\mathrm{wt}_{S_1}(B_u) > \frac{1}{z}$*, then we say that* B_u *is* heavy*, otherwise it is* light*.*

Observe that each heavy interval consists of a single element.

In order to relate between $\mathcal{H}$ and $\mathcal{B}$, we introduce the following event, based on the sample S_1.

▶ **Definition 13.** *Denote by* E_1 *the event where*

$$\forall H \in \mathcal{H}_{sin} \cup \mathcal{H}_{med}, \;\; \frac{1}{2}\mathrm{wt}_p(H) \leq \mathrm{wt}_{S_1}(H) \leq \frac{3}{2}\mathrm{wt}_p(H) \;, \tag{20}$$

$$\forall H \in \mathcal{H}_{sml}, \;\; \mathrm{wt}_{S_1}(H) \leq \frac{1}{2z} \;. \tag{21}$$

▷ Claim 14. If the size of the sample S_1 is $s_1 = 120z\log(240z)$, then $\Pr[E_1] \geq \frac{8}{10}$, where the probability is over the choice of S_1.

▷ Claim 15. Conditioned on the event E_1, for every $u \in [U]$ such that B_u is light, $\mathrm{wt}_p(B_u) < \frac{6}{z}$.

3.2 Estimation of symbol density and weight of intervals

In this subsection we estimate the weight, according to p, of every interval $[b_u]$ for $u \in U$, as well as its symbol density, focusing on symbols that occur in w. Note that $[b_u]$ is the union of the intervals $B_1, \ldots, B_u$. We first introduce some notations.

For any word w^*, text T^*, $i \in [|w^*|]$ and $j \in [|T^*|]$, let $I_i^j(T^*, w^*) = 1$ if $T^*[j] = w_i^*$ and 0 otherwise. We next set

$$\xi_i^u = \sum_{j \in [b_u]} I_i^j(T, w) p_j \;. \tag{22}$$

Consider a sample S_2 of size s_2 selected according to p (with repetitions), where s_2 will be set subsequently. For every $u \in [U]$ and $i \in [k]$, set

$$\breve{\xi}_i^u = \frac{1}{s_2} \sum_{j \in [b_u]} I_i^j(T, w) N_{S_2}(j) \ . \tag{23}$$

▶ **Definition 16.** *The event E_2 (based on S_2) is defined as follows. For every $i \in [k]$ and $u \in [U]$,*

$$\left| \breve{\xi}_i^u - \xi_i^u \right| \leq \frac{1}{z} \ , \tag{24}$$

and for every $u \in [U]$

$$|\mathrm{wt}_{S_2}([b_u]) - \mathrm{wt}_p([b_u])| \leq \frac{1}{z} \ . \tag{25}$$

▷ Claim 17. If the size of the sample S_2 is $s_2 = z^2 \log(40kU)$, then $\Pr[E_2] \geq \frac{9}{10}$, where the probability is over the choice of S_2.

3.3 Reducing from distribution-free to uniform

In this subsection we give the aforementioned reduction from the distribution-free case to the uniform case, using the intervals and estimators that were defined in the previous subsections. We start by providing three definitions, taken from [30], which will be used in the reduction. The first two definitions are for the notion of *splitting* (variants of this notion were also used in previous works, e.g., [15]).

▶ **Definition 18.** *For a text $T = t_1 \dots t_n$, a text $\widetilde{T}$ is said to be a splitting of T if $\widetilde{T} = t_1^{\alpha_1} \dots t_n^{\alpha_n}$ for some $\alpha_1 \dots \alpha_n \in \mathbb{N}^+$. We denote by ϕ the splitting map, which maps each (index of a) symbol of $\widetilde{T}$ to its origin in T. Formally, $\phi : [|\widetilde{T}|] \to [n]$ is defined as follows. For every $\ell \in [|\widetilde{T}|] = [\sum_{i=1}^{n} \alpha_i]$, let $\phi(\ell)$ be the unique $i \in [n]$ that satisfies $\sum_{r=1}^{i-1} \alpha_r < \ell < \sum_{r=1}^{i} \alpha_r$.*

Note that by this definition, ϕ is a non-decreasing surjective map, satisfying $\widetilde{T}[\ell] = T[\phi(\ell)]$ for every $\ell \in [|\widetilde{T}|]$. For a set $S \subseteq [|\widetilde{T}|]$ we let $\phi(S) = \{\phi(\ell) : \ell \in S\}$. With a slight abuse of notation, for any $i \in [n]$ we use $\phi^{-1}(i)$ to denote the set $\left\{\ell \in [|\widetilde{T}|] : \phi(\ell) = i\right\}$, and for a set $S \subseteq [n]$ we let $\phi^{-1}(S) = \left\{\ell \in [|\widetilde{T}|] : \phi(\ell) \in S\right\}$

▶ **Definition 19.** *Given text $T = t_1 \dots t_n$ and a corresponding probability distribution $p = (p_1, \dots, p_n)$, a splitting of (T, p) is a text $\widetilde{T}$ along with a corresponding probability distribution $\hat{p} = (\hat{p}_1, \dots, \hat{p}_{|\widetilde{T}|})$, such that $\widetilde{T}$ is a splitting of T and $\sum_{\ell \in \phi^{-1}(i)} \hat{p}_\ell = p_i$ for every $i \in [n]$.*

The third definition is of a set of words, where no two consecutive symbols are the same.

▶ **Definition 20.** *Let $\mathcal{W}_c = \{w : w_{j+1} \neq w_j, \forall j \in [k-1]\}$.*

3.3.1 A basis for reducing from distribution-free to uniform

Let $\widetilde{w}$ be a word of length $\widetilde{k}$ and $\widetilde{T}$ a text of length $\widetilde{n}$. In this subsection we establish a claim, which gives sufficient conditions on a (normalized version) of an estimation matrix $\widehat{\mathcal{N}}$, under which it can be used to obtain an estimate of $\Delta(\widetilde{T}, \widetilde{w})$ with a small additive error.

We first state a claim that is similar to Claim 6, with a small, but important difference, that takes into account intervals in $\widetilde{T}$ (determined by a set of indices J) that consist of repetitions of a single symbol. Recall that $M(\cdot)$ was defined in Definition 5, and that $R(\widetilde{T}, \widetilde{w})$ denotes the number of role-disjoint copies of $\widetilde{w}$ in $\widetilde{T}$.

▷ Claim 21. Let $J = \{j_0, j_1, \ldots, j_\ell\}$ be a set of indices satisfying $j_0 = 0 < j_1 < j_2 < \cdots < j_\ell = \widetilde{n}$. Let $\mathcal{N}$ be the matrix whose entries are $\mathcal{N}_i^r = N_i^{j_r}(\widetilde{T}, \widetilde{w})$ for every $i \in [\widetilde{k}]$ and $r \in [\ell]$. Let $J' = \{r \in [\ell] : \widetilde{T}[j_{r-1}+1] = \cdots = \widetilde{T}[j_r]\}$. Then

$$\left|M(\mathcal{N}) - R(\widetilde{T}, \widetilde{w})\right| \leq (\widetilde{k}-1) \cdot \max_{r \in [\ell] \setminus J'} \{(j_r - j_{r-1})\} \ .$$

The following observation can be easily proved by induction.

▶ **Observation 22.** *Let $\widehat{\mathcal{N}}$ be a matrix of size $\widetilde{k} \times \ell$. Then*

$$\frac{1}{\widetilde{n}} M(\widehat{\mathcal{N}}) = M\left(\frac{\widehat{\mathcal{N}}}{\widetilde{n}}\right) \ . \tag{26}$$

The next claim will serve as the basis for our reduction from the general, distribution-free case, to the uniform case.

▷ Claim 23. Let $\widehat{\mathcal{N}}$ be a $\widetilde{k} \times \ell$ matrix, $J = \{j_0, j_1, j_2, \ldots, j_\ell\}$ be a set of indices satisfying $j_0 = 0 < j_1 < j_2 < \cdots < j_\ell = \widetilde{n}$ and let c_1 and c_2 be constants. Suppose that the following conditions are satisfied.

1. For every $r \in [\ell]$, if $j_r - j_{r-1} > c_1 \cdot \frac{\delta\widetilde{n}}{\widetilde{k}}$, then $\widetilde{T}[j_{r-1}+1] = \cdots = \widetilde{T}[j_r]$.
2. For every $i \in [\widetilde{k}]$ and $r \in [\ell]$, $\left|\widehat{\mathcal{N}}_i^r - N_i^{j_r}(\widetilde{T}, \widetilde{w})\right| \leq c_2 \cdot \frac{\delta\widetilde{n}}{\widetilde{k}}$.

Then,

$$\left|M\left(\frac{\widehat{\mathcal{N}}}{\widetilde{n}}\right) - \Delta(\widetilde{T}, \widetilde{w})\right| \leq (c_1 + 2c_2)\delta \ .$$

3.3.2 Establishing the reduction for $w \in \mathcal{W}_c$ and quantized p

For the ease of readability, in this subsection we address the special case in which $w \in \mathcal{W}_c$ (recall Definition 20), and in the full version of this paper [14] we show how to deal with the general case.

For the case considered in this subsection, let $\widetilde{T} = t_1^{\alpha_1} \ldots t_n^{\alpha_n}$ where $\alpha_j = \frac{p_j}{\beta}$ for every $j \in [n]$, so that $|\widetilde{T}| = \frac{1}{\beta}$. Define $\widetilde{p}$ by $\widetilde{p}_j = \beta$ for every $j \in [|\widetilde{T}|]$, so that $\widetilde{p}$ is the uniform distribution. Since $p_j = \beta \cdot \alpha_j$, for every $j \in [n]$, we get that $(\widetilde{T}, \widetilde{p})$ is a splitting of (T, p) (recall Definition 19), and hence by [30, Clm. 4.4] (using the assumption that $w \in \mathcal{W}_c$),

$$\Delta(\widetilde{T}, w, \widetilde{p}) = \Delta(T, w, p) \ . \tag{27}$$

Denote $\widetilde{n} = |\widetilde{T}|$. We begin by defining a set of intervals of $[\widetilde{n}]$, where $\{b_0, \ldots, b_U\}$ and $\mathcal{B} = \{B_1, \ldots, B_U\}$ are as defined in Section 3.1, and ϕ is as in Definition 19.

▶ **Definition 24.** *Let $\widetilde{b}_0 = 0$, and for every $u \in [U]$, let $\widetilde{b}_u = \max\{h \in [\widetilde{n}] : \phi(h) = b_u\}$. For every $u \in [U]$ let $\widetilde{B}_u = [\widetilde{b}_{u-1}+1, \widetilde{b}_u]$, and define $\widetilde{\mathcal{B}} = \left\{\widetilde{B}_u\right\}_{u=1}^{U}$.*

We next introduce a notation for the weights, according to $\widetilde{p}$, of unions of these intervals. For every $i \in [k]$ and $u \in [U]$,

$$\widetilde{\xi}_i^u = \sum_{j \in [\widetilde{b}_u]} I_i^j(\widetilde{T}, w)\widetilde{p}_j \ . \tag{28}$$

Note that

$$\widetilde{\xi}_i^u = \frac{1}{\widetilde{n}} N_i^{b_u}(\widetilde{T}, w) \ . \tag{29}$$

▷ Claim 25. For every $i \in [k]$ and $u \in [U]$ $\widetilde{\xi}_i^u = \xi_i^u$, where ξ_i^u is as defined in Equation (22).

We can now state and prove the following lemma.

▶ **Lemma 26.** *Let w be a word of length k in $\mathcal{W}_c$, T a text of length n, and p a distribution over $[n]$ for which there exists $\beta \in (0,1)$ such that p_j/β is an integer for every $j \in [n]$. There exists an algorithm that, given a parameter $\delta \in (0,1)$, takes a sample of size $\Theta\left(\frac{k^2}{\delta^2} \cdot \log\left(\frac{k}{\delta}\right)\right)$ from T, distributed according to p, and outputs an estimate $\widehat{\Delta}$ such that $|\widehat{\Delta} - \Delta(T,w,p)| \leq \delta$ with probability at least $2/3$.*

As in the uniform case, the running time of the algorithm is linear in the size of the sample.

Proof. The algorithm first takes a sample S_1 of size $s_1 = 120z\log(240z)$ and constructs a set of intervals $\mathcal{B}$ as defined in Definition 12. Next the algorithm takes another sample, S_2, of size $s_2 = z^2\log(40kU)$ according to which it defines an estimation matrix $\widehat{\xi}$ of size $k \times U$ as follows. For every $i \in [k]$ and $u \in [U]$, it sets $\widehat{\xi}[i][u] = \breve{\xi}_i^u$, where $\breve{\xi}_i^u$ is as defined in Equation (23). Lastly the algorithm outputs $\widehat{\Delta} = M(\widehat{\xi})$, where M is as defined in Definition 5.

We would like to apply Claim 23 in order to show that $|\widehat{\Delta} - \Delta(\widetilde{T},w)| \leq \delta$ with probability of at least $\frac{2}{3}$. By the setting of s_1, applying Claim 14 gives us that with probability at least $\frac{8}{10}$, the event E_1, as defined in Definition 13, holds. By the setting of s_2, applying Claim 17 gives us that with probability at least $\frac{9}{10}$, the event E_2, as defined in Definition 16, holds. We henceforth condition on both events (where they hold together with probability at least $7/10$).

In order to apply Claim 23, we set $\widetilde{w} = w$, $J = \left\{\widetilde{b}_0, \widetilde{b}_1, \ldots, \widetilde{b}_U\right\}$ (recall Definition 24) and $\widehat{\mathcal{N}} = \widetilde{n}\widehat{\xi}$, for $\widehat{\xi}$ as defined above. Also, we set $c_1 = \frac{1}{2}$ and $c_2 = \frac{1}{4}$. We next show that both items in the premise of the claim are satisfied.

To show that Item 1 is satisfied, we first note that since $\widetilde{p}$ is uniform, then for every $u \in U$, $\mathrm{wt}_{\widetilde{p}}(b_u) = \frac{\widetilde{b}_u - \widetilde{b}_{u-1}}{\widetilde{n}}$. We use the consequence of Claim 15 (recall that we condition on E_1) by which for every u such that $\frac{\widetilde{b}_u - \widetilde{b}_{u-1}}{\widetilde{n}} \geq \frac{6}{z}$, B_u is heavy (since for every $u \in U$, $\mathrm{wt}_{\widetilde{p}}(\widetilde{B}_u) = \mathrm{wt}_p(B_u)$). By Definition 12 this implies that B_u contains only one index, and so $\widetilde{T}[\widetilde{b}_{u-1}+1] = \cdots = \widetilde{T}[\widetilde{b}_u]$. By the definition of z (Equation (19)) and the setting of c_1, the item is satisfied.

To show that Item 2 is satisfied, we use the definition of E_2 (Definition 16, Equation (24)) together with Claim 25, which give us $|\widehat{\xi}_i^u - \widetilde{\xi}_i^u| \leq \frac{1}{z}$ for every $i \in [k]$ and $u \in [U]$. By Equation (29), the definition of z and the setting of c_2, we get that the item is satisfied.

After applying Claim 23 we get that $|\widehat{\Delta} - \Delta(\widetilde{T},w)| \leq (c_1 + 2c_2)\delta$, which by the setting of c_1 and c_2 is at most δ. Since $\widetilde{p}$ is the uniform distribution, $\Delta(\widetilde{T},w) = \Delta(\widetilde{T},w,\widetilde{p})$ and since $\Delta(\widetilde{T},w,\widetilde{p}) = \Delta(T,w,p)$ (by Equation (27)), the lemma follows. ◀

In the full version of this paper [14] we address the general case where we do not necessarily have that $w \in \mathcal{W}_c$ or that there exists a value β such that for every $j \in [n]$, p_j/β is an integer.

4 A lower bound for distance approximation

In this section we give a lower bound for the number of samples required to perform distance-approximation from w-freeness of a text T. The lower bound holds when the underlying distribution is the uniform distribution.

▶ **Theorem 27.** *Let k_d be the number of distinct symbols in w. Any distance-approximation algorithm for w-freeness under the uniform distribution must take a sample of size $\Omega(\frac{1}{k_d\delta^2})$, conditioned on $\delta \leq \frac{1}{300k_d}$ and $n > \max\left\{\frac{8k}{\delta}, \frac{200}{k_d\delta^2}\right\}$.*

Note that if $\delta \geq 1/k_d$, then the algorithm can simply output 0. This is true since the number of role disjoint copies of w in T is at most the number of occurrences of the symbol in w that is least frequent in T. This number is upper bounded by $\frac{n}{k_d}$, and so the distance from w-freeness is at most $\frac{1}{k_d}$. In this case no sampling is needed, so only the trivial lower bound holds. The proof will deal with the case of $\delta \in (0, \frac{1}{300k_d}]$.

Proof. The proof is based on the difficulty of distinguishing between an unbiased coin and a coin with a small bias. Precise details follow.

Let $V = \{v_1, \ldots, v_{k_d}\}$ be the set of distinct symbols in w, and let 0 be a symbol that does not belong to V. We define two distributions over texts, $\mathcal{T}_1$ and $\mathcal{T}_2$ as follows. For each $\tau \in [\frac{n}{k_d}]$ and $\rho \in [0,1]$, let λ_ρ^τ be a random variable that equals 0 with probability ρ and equals v_1 with probability $1-\rho$. Let $\delta' = 3k_d\delta$ and consider the following two distributions over texts

$$\mathcal{T}_1 = \left[\lambda_{\frac{1}{2}}^1, v_2, v_3, \ldots, v_{k_d}, \lambda_{\frac{1}{2}}^2, v_2, v_3, \ldots, v_{k_d}, \ldots\ldots, \lambda_{\frac{1}{2}}^{n/k_d}, v_2, v_3, \ldots, v_{k_d}\right] , \tag{30}$$

$$\mathcal{T}_2 = \left[\lambda_{\frac{1}{2}+\delta'}^1, v_2, v_3, \ldots, v_{k_d}, \lambda_{\frac{1}{2}+\delta'}^2, v_2, v_3, \ldots, v_{k_d}, \ldots\ldots, \lambda_{\frac{1}{2}+\delta'}^{n/k_d}, v_2, v_3, \ldots, v_{k_d}\right] . \tag{31}$$

Namely, the supports of both distributions contain texts that consist of n/k_d blocks of size k_d each. For $i \in \{2, \ldots, k_d\}$, the i-th symbol in each block is v_i. The distributions differ only in the way the first symbol in each block is selected. In $\mathcal{T}_1$ it is 0 with probability $1/2$ and v_1 with probability $1/2$, while in $\mathcal{T}_2$ it is 0 with probability $1/2 + \delta' = 1/2 + 3\delta k_d$, and v_1 with probability $1/2 - \delta'$.

For $b \in \{1,2\}$, consider selecting a text T_b according to $\mathcal{T}_b$ (denoted by $T_b \sim \mathcal{T}_b$), and let O_b be the number of occurrences of v_1 in the text (so that O_b is a random variable). Observe that $\mathbb{E}[O_1] = \frac{n}{2k_d}$ and $\mathbb{E}[O_2] = \frac{n}{2k_d} - 3\delta n$. By applying the additive Chernoff bound (Theorem 28) and using the premise of the theorem regarding n,

$$\Pr_{T_1\sim\mathcal{T}_1}\left[O_1 < \mathbb{E}[O_1] - \delta n/8\right] \leq \exp(-2(k_d\delta/8)^2 \cdot n/k_d) \leq \frac{1}{100} , \tag{32}$$

and

$$\Pr_{T_2\sim\mathcal{T}_2}\left[O_2 < \mathbb{E}[O_2] + \delta n/8\right] \leq \exp(-2(k_d\delta/8)^2 \cdot n/k_d) \leq \frac{1}{100} . \tag{33}$$

For $b \in \{1,2\}$ let $R_b = R(T_b, w)$ (recall that $R(T_b, w)$ denotes the number of disjoint copies of w in T_b, and note that R_b is a random variable). Observe that $R_1 \geq O_1 - k + 1$, and $R_2 \leq O_2$.

Hence, by Equation (32), if we select T_1 according to $\mathcal{T}_1$ and use the premise that $n > \frac{8k}{\delta}$, then $R(T_1, w) \geq \frac{n}{2k_d} - \frac{1}{8}\delta n - k + 1 \geq \frac{n}{2k_d} - \frac{2}{8}\delta n$ with probability at least 99/100, and by Equation (33), if we select T_2 according to $\mathcal{T}_2$, then $R(T_2, w) \leq \frac{n}{2k_d} - 3\delta n + \frac{1}{8}\delta n = \frac{n}{2k_d} - \frac{23}{8}\delta n$ with probability at least 99/100.

Assume, contrary to the claim, that we have a sample-based distance-approximation algorithm for subsequence-freeness that takes a sample of size $Q(k_d, \delta) = 1/(ck_d\delta^2)$, for some sufficiently large constant c, and outputs an estimate of the distance to w-freeness that has additive error at most δ, with probability at least 2/3. Consider running the algorithm on either $T_1 \sim \mathcal{T}_1$ or $T_2 \sim \mathcal{T}_2$. Let L denote the number of times that the sample landed on an index of the form $j = \ell \cdot k_d + 1$ for an integer ℓ. By Markov's inequality, the probability that $L > 10 \cdot Q(k_d, \delta)/k_d = 10/(ck_d^2\delta^2)$ is at most 1/10.

By the above, if we run the algorithm on $T_1 \sim \mathcal{T}_1$, then with probability at least $2/3 - 1/100 - 1/10$ the algorithm outputs an estimate $\widehat{\Delta} \geq \frac{n}{2k_d} - \frac{10}{8}$ while $L \leq 10/(ck_d^2\delta^2)$. Similarly, if we run it on $T_2 \sim \mathcal{T}_2$, then with probability at least $2/3 - 1/100 - 1/10$ the algorithm outputs an estimate $\widehat{\Delta} \leq \frac{n}{2k_d} - \frac{15}{8}$ while $L \leq 10/(ck_d^2\delta^2)$. (In both cases the probability is taken over the selection of $T_b \sim \mathcal{T}_b$, the sample that the algorithm gets, and possibly additional internal randomness of the algorithm.) Based on the definitions of $\mathcal{T}_1$ and $\mathcal{T}_2$, this implies that it is possible to distinguish between an unbiased coin and a coin with bias $3k_d\delta$ with probability at least $2/3 - 1/100 - 1/10 > \frac{8}{15}$, using a sample of size $\frac{1}{c'k_d^2\delta^2}$ in contradiction to the result of Bar-Yosef [2, Thm. 8] (applied with $m = 2$, $\epsilon = 3k_d\delta$. Since we have $\delta < \frac{1}{300k_d}$, then $\epsilon < \frac{1}{96}$, as the cited theorem requires). ◀

References

1. Nir Ailon, Bernard Chazelle, Seshadhri Comandur, and Ding Liu. Estimating the distance to a monotone function. *Random Structures and Algorithms*, 31(3):371–383, 2007.
2. Ziv Bar-Yossef. Sampling lower bounds via information theory. In *Proceedings of the 35th Annual ACM Symposium on the Theory of Computing*, pages 335–344, 2003.
3. Omri Ben-Eliezer, Eldar Fischer, Amit Levi, and Ron D. Rothblum. Hard properties with (very) short PCPPs and their applications. In *Proceedings of the 11th Innovations in Theoretical Computer Science conference (ITCS)*, pages 9:1–9:27, 2020.
4. Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. Tolerant testers of image properties. *ACM Transactions on Algorithms*, 18(4):1–39, 2022. Article number 37.
5. Piotr Berman, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Lp-testing. In *Proceedings of the 46th Annual ACM Symposium on the Theory of Computing*, pages 164–173, 2014.
6. Hadley Black, Deeparnab Chakrabarty, and C. Seshadhri. Domain reduction for monotonicity testing: A $o(d)$ tester for boolean functions in d-dimensions. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1975–1994, 2020.
7. Eric Blais, Clément L Canonne, Talya Eden, Amit Levi, and Dana Ron. Tolerant junta testing and the connection to submodular optimization and function isomorphism. *ACM Transactions on Computation Theory*, 11(4):1–33, 2019.
8. Eric Blais, Renato Ferreira Pinto Jr., and Nathaniel Harms. VC dimension and distribution-free sample-based testing. In *Proceedings of the 53rd Annual ACM Symposium on the Theory of Computing*, pages 504–517, 2021.
9. Avrim Blum and Lunjia Hu. Active tolerant testing. In *Proceedings of the 31st Conference on Computational Learning Theory (COLT)*, pages 474–497, 2018.
10. Mark Braverman, Subhash Khot, Guy Kindler, and Dor Minzer. Improved monotonicity testers via hypercube embeddings. In *Proceedings of the 13th Innovations in Theoretical Computer Science conference (ITCS)*, pages 25:1–25:24, 2024.
11. Andrea Campagna, Alan Guo, and Ronitt Rubinfeld. Local reconstructors and tolerant testers for connectivity and diameter. In *Proceedings of the 17th International Workshop on Randomization and Computation*, pages 411–424, 2013.
12. Clément L Canonne, Elena Grigorescu, Siyao Guo, Akash Kumar, and Karl Wimmer. Testing k-monotonicity: The rise and fall of boolean functions. *Theory of Computing*, 15(1):1–55, 2019. This paper appeared in the proceedings of ITCS 2017.
13. Omer Cohen Sidon. Sample-based distance-approximation for subsequence-freeness. MSc thesis, Tel Aviv University, 2023.
14. Omer Cohen Sidon and Dana Ron. Sample-based distance-approximation for subsequence-freeness. *arXiv preprint*, 2023. `arXiv:2305.01358`.
15. Ilias Diakonikolas and Daniel Kane. A new approach for testing properties of discrete distributions. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science*, pages 685–694, 2016.

16 Shahar Fattal and Dana Ron. Approximating the distance to monotonicity in high dimensions. *ACM Transactions on Algorithms*, 6(3):1–37, 2010.

17 Nimrod Fiat and Dana Ron. On efficient distance approximation for graph properties. In *Proceedings of the 32nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1618–1637, 2021.

18 Eldar Fischer and Lance Fortnow. Tolerant versus intolerant testing for boolean properties. *Theory of Computing*, 2:173–183, 2006.

19 Eldar Fischer and Ilan Newman. Testing versus estimation of graph properties. *SIAM Journal on Computing*, 37(2):482–501, 2007.

20 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connections to learning and approximation. *Journal of the ACM*, 45:653–750, 1998.

21 Venkat Guruswami and Atri Rudra. Tolerant locally testable codes. In *Proceedings of the 9th International Workshop on Randomization and Computation*, pages 306–317, 2005.

22 Nathaniel Harms and Yuichi Yoshida. Downsampling for testing and learning in product distributions, 2022.

23 Carlos Hoppen, Yoshiharu Kohayakawa, Richard Lang, Hanno Lefmann, and Henrique Stagni. Estimating the distance to a hereditary graph property. *Electronic Notes in Discrete Mathematics*, 61:607–613, 2017.

24 Swastik Kopparty and Shubhangi Saraf. Tolerant linearity testing and locally testable codes. In *Proceedings of the 13th International Workshop on Randomization and Computation*, pages 601–614, 2009.

25 Amit Levi and Erik Waingarten. Lower bounds for tolerant junta and unateness testing via rejection sampling of graphs. In *Proceedings of the 10th Innovations in Theoretical Computer Science conference (ITCS)*, pages 52:1–52:20, 2019.

26 Sharon Marko and Dana Ron. Distance approximation in bounded-degree and general sparse graphs. *Transactions on Algorithms*, 5(2), 2009. Article number 22.

27 Ilan Newman and Nithin Varma. New sublinear algorithms and lower bounds for LIS estimation. In *Automata, Languages and Programming: 48th International Colloquium*, pages 100:1–100:20, 2021.

28 Ramesh Krishnan S Pallavoor, Sofya Raskhodnikova, and Erik Waingarten. Approximating the distance to monotonicity of boolean functions. *Random Structures & Algorithms*, 60(2):233–260, 2022.

29 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *Journal of Computer and System Sciences*, 72(6):1012–1042, 2006.

30 Dana Ron and Asaf Rosin. Optimal distribution-free sample-based testing of subsequence-freeness with one-sided error. *ACM Transactions on Computation Theory*, 14(4):1–31, 2022. An extended abstract of this work appeared in the proceedings of SODA 2021.

31 Ronitt Rubinfeld and Madhu Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

A Chernoff bounds

▶ **Theorem 28.** *Let $\chi_1, \ldots, \chi_m$ be m independent random variables where $\chi_i \in [0,1]$ for every $1 \leq i \leq m$. Let $p \stackrel{\text{def}}{=} \frac{1}{m}\sum_i \mathbb{E}[\chi_i]$. Then, for every $\gamma \in (0,1]$, the following bounds hold:*

- *(Additive Form)*

$$\Pr\left[\frac{1}{m}\sum_{i=1}^{m}\chi_i > p+\gamma\right] < \exp\left(-2\gamma^2 m\right) \tag{34}$$

$$\Pr\left[\frac{1}{m}\sum_{i=1}^{m}\chi_i < p-\gamma\right] < \exp\left(-2\gamma^2 m\right) \tag{35}$$

- *(Multiplicative Form)*

$$\Pr\left[\frac{1}{m}\sum_{i=1}^{m}\chi_i > (1+\gamma)p\right] < \exp\left(-\gamma^2 pm/3\right) \tag{36}$$

$$\Pr\left[\frac{1}{m}\sum_{i=1}^{m}\chi_i < (1-\gamma)p\right] < \exp\left(-\gamma^2 pm/2\right) \tag{37}$$

New Partitioning Techniques and Faster Algorithms for Approximate Interval Scheduling

Spencer Compton
Stanford University, CA, USA

Slobodan Mitrović
University of California Davis, CA, USA

Ronitt Rubinfeld
MIT, Cambridge, MA, USA

Abstract

Interval scheduling is a basic problem in the theory of algorithms and a classical task in combinatorial optimization. We develop a set of techniques for partitioning and grouping jobs based on their starting and ending times, that enable us to view an instance of interval scheduling on *many* jobs as a union of multiple interval scheduling instances, each containing only *a few* jobs. Instantiating these techniques in dynamic and local settings of computation leads to several new results.

For $(1+\varepsilon)$-approximation of job scheduling of n jobs on a single machine, we develop a fully dynamic algorithm with $O(^{\log n}/_{\varepsilon})$ update and $O(\log n)$ query worst-case time. Further, we design a local computation algorithm that uses only $O(^{\log N}/_{\varepsilon})$ queries when all jobs are length at least 1 and have starting/ending times within $[0, N]$. Our techniques are also applicable in a setting where jobs have rewards/weights. For this case we design a fully dynamic *deterministic* algorithm whose worst-case update and query time are $\text{poly}(\log n, \frac{1}{\varepsilon})$. Equivalently, this is *the first* algorithm that maintains a $(1+\varepsilon)$-approximation of the maximum independent set of a collection of weighted intervals in $\text{poly}(\log n, \frac{1}{\varepsilon})$ time updates/queries. This is an exponential improvement in $1/\varepsilon$ over the running time of a randomized algorithm of Henzinger, Neumann, and Wiese [SoCG, 2020], while also removing all dependence on the values of the jobs' starting/ending times and rewards, as well as removing the need for any randomness.

We also extend our approaches for interval scheduling on a single machine to examine the setting with M machines.

2012 ACM Subject Classification Theory of computation → Data structures design and analysis; Theory of computation → Approximation algorithms analysis

Keywords and phrases interval scheduling, dynamic algorithms, local computation algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.45

Category Track A: Algorithms, Complexity and Games

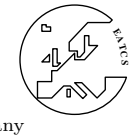
Related Version *Full Version*: https://arxiv.org/abs/2012.15002

Funding S. Compton was supported in part by the National Defense Science & Engineering Graduate (NDSEG) Fellowship Program. S. Mitrović was supported by the Swiss NSF grant No. P400P2_191122/1, NSF award CCF-1733808, and FinTech@CSAIL. R. Rubinfeld was supported by the NSF TRIPODS program (awards CCF-1740751 and DMS-2022448), NSF award CCF-2006664, and FinTech@CSAIL.

Acknowledgements We thank Benjamin Qi (MIT) for helpful discussions.

1 Introduction

Job scheduling is a fundamental task in optimization, with applications ranging from resource management in computing [21, 22] to operating transportation systems [14]. Given a collection of *machines* and a set of *jobs* (or tasks) to be processed, the goal of job scheduling is to assign those jobs to the machines while respecting certain constraints. Constraints set on



jobs may significantly vary. In some cases a job has to be scheduled, but the starting time of its processing is not pre-specified. In other scenarios a job can only be scheduled at a given time, but there is a flexibility on whether to process the job or not. Frequent objectives for this task can include either maximizing the number of scheduled jobs or minimizing needed time to process all the given jobs.

An important variant of job scheduling is the task of *interval scheduling*: here each job has a specified starting time and its length, but a job is not required to be scheduled. Given M machines, the goal is to schedule as many jobs as possible. More generally, each job is also assigned a *reward* or weight, which can be thought of as a payment received for processing the given job. If a job is not processed, the payment is zero, i.e., there is no penalty. We refer to this variant as *weighted interval scheduling*. This problem in a natural way captures real-life scenarios. For instance, consider an assignment of crew members to flights, where our goal is to assign (the minimum possible) crews to the specified flights. In the context of interval scheduling, flights can be seen as jobs and the crew members as machines [14, 17]. Interval scheduling also has applications in geometrical tasks – it can be seen as a task of finding a collection of non-overlapping geometric objects. In this context, its prominent applications are in VLSI design [13] and map labeling [1, 25].

The aforementioned scenarios are executed in different computational settings. For instance, some use-cases are dynamic in nature, e.g., a flight gets cancelled. Then, in certain cases we have to make online decisions, e.g., a customer must know immediately whether we are able to accept its request or not. While in some applications there might be so many requests that we would like to design extremely fast ways of deciding whether a given request/job can be scheduled or not, e.g., providing an immediate response to a user submitting a job for execution in a cloud. In this work, our aim is to develop methods for interval scheduling that can be turned into efficient algorithms across many computational settings:

Can we design unified techniques for approximating interval scheduling very fast?

In this paper we develop fast algorithms for the dynamic and local settings of computation. We also give a randomized black-box approach that reduces the task of interval scheduling on multiple machines to that of interval scheduling on a single machine by paying only $2 - 1/M$ in the approximation factor for unweighted jobs, where M is the number of machines, and e in approximation factor for weighted jobs. A common theme in our algorithms is partitioning jobs over dimensions (time and machines). It is well studied in the dynamic setting how to partition the time dimension to enable fast updates. It is also studied how to partition over the machines to enable strong approximation ratios for multiple-machine scheduling problems. We design new partitioning methods for the time dimension (starting and ending times of jobs), introduce a partitioning method over machines, and examine the relationship of partitioning over the time dimension and machines simultaneously in order to solve scheduling problems. We hope that, in addition to improving the best-known results, our work provides a new level of simplicity and cohesiveness for this style of approach.

1.1 Computation Models

In our work, we focus on the following two models of computation.

Dynamic setting. Our algorithms for the fully dynamic setting design data structures that maintain an approximately optimal solution to an instance of the interval scheduling problem while supporting insertions and deletions of jobs/intervals. The data structures also support queries of the maintained solution's total weight and whether or not a particular interval is used in the maintained solution.

Local computation algorithms (LCA). The LCA model was introduced by Rubinfeld et al. [20] and Alon et al. [2]. In this setting, for a given job J we would like to output whether J is scheduled or not, but we do not have a direct access to the entire list of input jobs. Rather, the LCA is given access to an oracle that returns answers to questions of the form: "*What is the input job with the earliest ending time among those jobs that start after time x?*" The goal of the LCA in this setting is to provide (yes/no) answers to user queries that ask "Is job i scheduled?" (and, if applicable, "On which machine?"), in such a manner that all answers should be consistent with the same valid solution, while using as few oracle-probes as possible.

1.2 Our Results

Our first result, given in Section 4, focuses on designing an efficient dynamic algorithm for unweighted interval scheduling on a single machine. Prior to our work, the state-of-the-art result for this unweighted interval scheduling problem was due to [4], who design an algorithm with $O(\log n/\varepsilon^2)$ update and query time. We provide an improvement in the dependence on ε.

▶ **Theorem 1** (Unweighted dynamic, single machine). *Let $\mathcal{J}$ be a set of n jobs. For any $\varepsilon > 0$, there exists a fully dynamic algorithm for $(1+\varepsilon)$-approximate unweighted interval scheduling for $\mathcal{J}$ on a single machine performing updates in $O\left(\frac{\log(n)}{\varepsilon}\right)$ and queries in $O(\log(n))$ worst-case time.*

Theorem 1 can be seen as a warm-up for our most challenging and technically involved result, which is an algorithm for the dynamic *weighted* interval scheduling problem on a single machine. We present our approach in detail in the full version. As a function of $1/\varepsilon$, our result constitutes an exponential improvement compared to the running times obtained in [12]. We also remove all use of randomness, remove all dependence on the job starting/ending times (previous work crucially used assumptions on the coordinates to bound the ratio of jobs' lengths by a parameter N), and remove all dependence on the value of the job rewards.

▶ **Theorem 2** (Weighted dynamic, single machine). *Let $\mathcal{J}$ be a set of n weighted jobs. For any $\varepsilon > 0$, there exists a fully dynamic algorithm for $(1+\varepsilon)$-approximate weighted interval scheduling for $\mathcal{J}$ on a single machine performing updates and queries in worst-case time $T \in \text{poly}(\log n, \frac{1}{\varepsilon})$. The exact complexity of T is given by*

$$O\left(\frac{\log^{12}(n)}{\varepsilon^7} + \frac{\log^{13}(n)}{\varepsilon^6}\right).$$

1.2.1 Implications in Other Settings

Local Computation Algorithms. We show that the ideas we developed to obtain Theorem 1 can also be efficiently implemented in the local setting, as we explain in detail in the full version and prove the following claim. This is the first non-trivial local computation algorithm for the interval scheduling problem.

▶ **Theorem 3** (Unweighted LCA, single machine). *Let $\mathcal{J}$ be a set of n jobs with length at least 1 and ending times upper-bounded by N. For any $\varepsilon > 0$, there exists a local computation algorithm for $(1+\varepsilon)$-approximate unweighted interval scheduling for $\mathcal{J}$ on a single machine using $O\left(\frac{\log N}{\varepsilon}\right)$ probes.*

Multiple machines. By building on techniques we introduced to prove Theorems 1 and 3, we show similar results in the full version in the case of interval scheduling on multiple machines at the expense of slower updates. To the best of our knowledge, these results initiate a study of dynamic and local interval scheduling in the general setting, i.e., in the setting of maximizing the total reward of jobs scheduled on multiple machines.

1.3 Related Work

The closest prior work to ours is that of Henzinger et al. [12] and of Bhore et al. [4]. [12] studies $(1+\varepsilon)$-approximate dynamic interval scheduling for one machine in both the weighted and unweighted setting. Unlike our main result in Theorem 2, they assume jobs have rewards within $[1, W]$, assume jobs have length at least 1, and assume all jobs start/end within times $[0, N]$. They obtain randomized algorithms with $O(\exp(1/\varepsilon)\log^2 n \cdot \log^2 N)$ update time for the unweighted and $O(\exp(1/\varepsilon)\log^2 n \cdot \log^5 N \cdot \log W)$ update time for the weighted case. They cast interval scheduling as the problem of finding a maximum independent set among a set of intervals lying on the x-axis. The authors extend this setting to multiple dimensions and design algorithms for approximating maximum independent set among a set of d-dimensional hypercubes, achieving a $(1+\varepsilon)2^d$-approximation in the unweighted and a $(4+\varepsilon)2^d$-approximation in the weighted regime.

The authors of [4] primarily focus on the unweighted case of approximating maximum independent set of a set of cubes. For the 1-dimensional case, which equals interval scheduling on one machine, they obtain $O(\log n/\varepsilon^2)$ update time, which is slower by a factor of $1/\varepsilon$ than our approach. They also show that their approach generalizes to the d-dimensional case, requiring $\operatorname{poly}\log n$ amortized update time and providing $O(4^d)$ approximation.

The problem of dynamically maintaining an exact solution to interval scheduling on one or multiple machines is studied by [11]. They attain a guarantee of $\tilde{O}(n^{1/3})$ update time for unweighted interval scheduling on $M = 1$ machine, and $\tilde{O}(n^{1-1/M})$ for $M \geq 2$. Moreover, they show an almost-linear time conditional hardness lower bound for dynamically maintaining an exact solution to the weighted interval scheduling problem on even just $M = 1$ machine. This further motivates work such as ours that dynamically maintains approximate solutions for weighted interval scheduling.

The authors of [9] consider dynamic interval scheduling on multiple machines in the setting in which all the jobs must be scheduled. The worst-case update time of their algorithm is $O(\log(n) + d)$, where d refers to the depth of what they call *idle intervals* (depth meaning the maximal number of intervals that contain a common point); they define an idle interval to be the period of time in a schedule between two consecutive jobs in a given machine. The same set of authors, in [10], study dynamic algorithms for the monotone case as well, in which no interval completely contains another one. For this setup they obtain an algorithm with $O(\log(n))$ update and query time.

In the standard model of computing (i.e. one processor, static), there exists an $O(n+m)$ running time algorithm for (exactly) solving the unweighted interval scheduling problem on a single machine with n jobs and integer coordinates bounded by m [8]. An algorithm with running time independent of m is described in [24], where it is shown how to solve this problem on M machines in $O(n\log(n))$ time. An algorithm is designed in [3] for weighted interval scheduling on M machines that runs in $O(n^2\log(n))$ time.

We refer a reader to [14] and references therein for additional applications of the interval scheduling problem.

Other related work. There has also been a significant interest in job scheduling problems in which our goal is to schedule *all* the given jobs across multiple machines, with the objective to minimize the total scheduling time. Several variants have been studied, including setups which allow preemptions, or setting where jobs have precedence constraints. We refer a reader to [15, 7, 19, 23, 5, 18, 16] and references therein for more details on these and additional variants of job scheduling. Beyond dynamic algorithms for approximating maximum independent sets of intervals or hypercubes, [6] show results for geometric objects such as disks, fat polygons, and higher-dimensional analogs. After we had published a preprint of this work, [6] proved a result that captures Theorem 1 with a more general class of fat objects.

2 Overview of Our Techniques

Our primary goal is to present unified techniques for approximating scheduling problems that can be turned into efficient algorithms for many settings. In this section, we discuss key insights of our techniques.

In the problems our work tackles, partitioning the problem instance into independent, manageable chunks is crucial. Doing so enables an LCA to determine information about a job of interest without computing an entire schedule, or enables a dynamic data structure to maintain a solution without restarting from scratch.

2.1 Unweighted Interval Scheduling – Partitioning Over Time (Section 4)

For simplicity of presentation, we begin by examining our method for partitioning over time for just the unweighted interval scheduling problem on one machine (i.e., $M = 1$). In particular, we first focus on doing so for the dynamic setting.

Recall that in this setting the primary motivation for partitioning over time, is to divide the problem into independent, manageable chunks that can be utilized by a data structure to quickly modify a solution while processing an update. In our work, we partition the time dimension by maintaining a set of *borders* that divide time into some number of contiguous regions. By doing so, we divide the problem into many *independent regions*, and we ignore jobs that intersect multiple regions; equivalently, we ignore jobs that contain a border. Our goal is then to dynamically maintain borders in a way such that we can quickly recompute the optimal solution completely within some region, and that the suboptimality introduced by these borders does not affect our solution much. In Section 4, we show that by maintaining borders where the optimal solution inside each region, i.e., a time-range between two borders, is of size $\Theta(\frac{1}{\varepsilon})$, we can maintain a $(1+\varepsilon)$-approximation of an optimal solution as long as we optimally compute the solution within each region.

Here, the underlying intuition is that because each region has a solution of size $\Omega(\frac{1}{\varepsilon})$, we can charge any suboptimality caused by a border against the selected jobs in an adjacent region. Likewise, because each region's solution has size $O(\frac{1}{\varepsilon})$, we are able to recompute the optimal solution within some region quickly using a balanced binary search tree. We dynamically maintain borders satisfying our desired properties by adding a new border when a region becomes too large, or merging with an adjacent region when a region becomes too small. As only $O(1)$ regions will require any modification when processing an update,

this method of partitioning time, while simple, enables us to improve the fastest known update/query time to $O(\log(n)/\varepsilon)$.[1] In Section 2.2 we build on these ideas to design an algorithm for the weighted interval scheduling problem.

2.2 Weighted Interval Scheduling

In our most technically involved result, we design the first deterministic $(1+\varepsilon)$ approximation algorithm for weighted interval scheduling that runs in $\text{poly}(\log n, \frac{1}{\varepsilon})$ time. In this section we give an outline of our techniques and discuss key insights. For full details we refer a reader to the full version.

2.2.1 Job data structure

Let $\mathcal{E}$ be the set of all the endpoints of given jobs, i.e., $\mathcal{E}$ contains s_i and f_i for each job $[s_i, f_i]$. We build a hierarchical data structure over $\mathcal{E}$ as follows. This structure is organized as a binary search tree T. Each node Q of T contains value $\text{KEY}(Q) \in \mathcal{E}$, with "1-1" mapping between $\mathcal{E}$ and the nodes of T. Each node Q is *responsible for a time range*. The root of T, that we denote by Q_{root}, is responsible for the entire time range $(-\infty, \infty)$. Each node Q has at most two children, that we denote by Q_L and Q_R. If Q is responsible for the time range $[X, Y]$, then Q_L is responsible for $[X, \text{KEY}(Q)]$, while Q_R is responsible for $[\text{KEY}(Q), Y]$.

Jobs are then assigned to nodes, where a job J is assigned to every node Q such that J is contained within the Q's responsible time range.

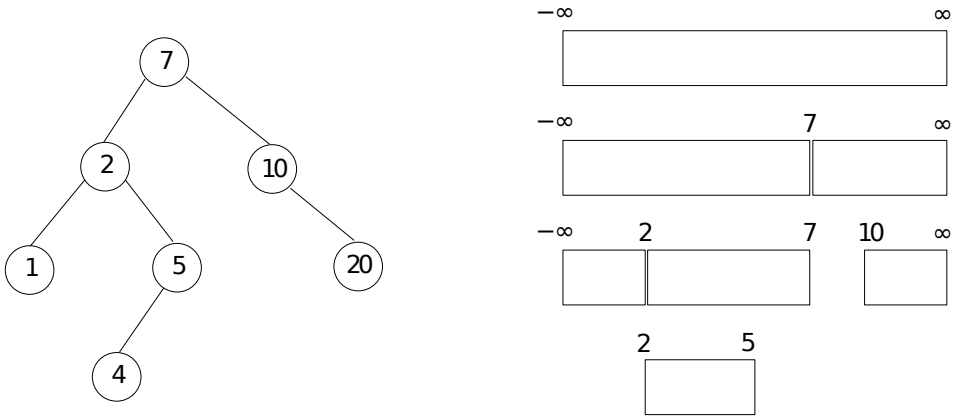


Figure 1 Visual example for hierarchical decomposition. Consider we are given jobs with the following ranges of $(1, 5), (2, 10), (7, 20), (4, 5)$. On the left is T, a balanced binary search tree over the set of all s_i and f_i. On the right is the hierarchical decomposition that corresponds to T. That is, in each row, the intervals on the right correspond to the $[l_Q, r_Q]$ for the nodes on the left. For instance, in the third row, $(-\infty, 2]$ corresponds to the node Q with $KEY(Q) = 1$.

[1] The main advantage of this techniques is that it leads to worst-case $O(\log(n)/\varepsilon)$ update time, as opposed to only an amortized one. We point out that it is not difficult to obtain such amortized guarantee in the following way: after each $\varepsilon \cdot OPT$ many updates, recompute the optimum solution from scratch. Given access to the balanced binary tree structure described above, this re-computation can be done in $O(OPT \cdot \log n)$ time.

2.2.2 Organizing computation

We now outline how the structure T is used in computation. As a reminder, our main goal is to compute a $(1+\varepsilon)$-approximate weighted interval scheduling. This task is performed by requesting Q_{root} to solve the problem for the range $(-\infty, \infty)$. However, instead of computing the answer for the entire range $(-\infty, \infty)$ directly, Q_{root} *partitions* the range $(-\infty, \infty)$ into:

- a number of ranges over which it is relatively easy to compute approximate solutions, such ones are called *sparse*, and
- the remaining ranges over which it is relatively hard to compute approximate solutions at the level of Q_{root}.

These hard-to-approximate ranges are deferred to the children of Q_{root}, and are hard to approximate because any near-optimal solution for the range contains many jobs. On the other hand, solutions in sparse ranges are of size $O(1/\varepsilon)$. As we discuss later, approximate optimal solutions within sparse ranges can be computed very efficiently; for details, see the paragraph *Approximate dynamic programming* below.

In general, a child Q_C of Q_{root} might receive *multiple* ranges from Q_{root} for which it is asked to find an approximately optimal solution. Q_C performs computation in the same manner as Q_{root} did – the cell Q_C partitions each range it receives into "easy" and "hard" to compute subranges. The first type of subranges is computed by Q_C, while the second type if deferred to the children of Q_C. The same as in Section 2.3, these "hard" ranges have large weight and allow for drawing a boundary and hence dividing a range into two or more *independent* ranges. We now discuss how the partitioning into ranges is undertaken.

2.2.3 Auxiliary data structure

To divide a range into "easy" and "hard" ranges at the level of a node Q, we design an auxiliary data structure, which relates to a rough approximation of the problem. This structure, called $Z(Q)$, maintains a set of points (we call these points *grid endpoints*) that partition Q into *slices of time*. We use slice to refer to a time range between two *consecutive* points of $Z(Q)$. Recall how for unweighted interval scheduling, we maintained a set of borders and ignored a job that crossed any border. In the weighted version, we will instead use $Z(Q)$ as a set of partitions from which we will use *some subset* to divide time. Our method of designing $Z(Q)$ reduces the task of finding a partitioning over time $Z(Q)$ within a cell for the $(1+\varepsilon)$-approximate weighted interval scheduling problem to finding multiple partitionings for the $(1+\varepsilon)$-approximate unweighted problem.

It is instructive to think of $Z(Q)$ in the following way. First, we view weighted interval scheduling as $O(\log n)$ independent instances of unweighted interval scheduling – instance i contains the jobs having weights in the interval $(w_{max}(Q)/2^{i+1}, w_{max}(Q)/2^i]$. Then, for each unweighted instance we compute borders as described in Section 2.1. $Z(Q)$ constitutes a subset of the union of those borders across all unweighted instances. We point out that the actual definition of $Z(Q)$ contains some additional points that are needed for technical reasons, but in this section we will adopt this simplified view. In particular, as we will see, $Z(Q)$ is designed such that the optimal solution within each slice has small total reward compared to the optimal solution over the entirety of Q. This enables us to partition the main problem into subproblems such that the suboptimality of discretizing the time towards slices, that we call *snapping*, is negligible.

However, a priori, it is not even clear that such structure $Z(Q)$ exists. So, one of the primary goals in our analysis is to show that there exists a near-optimal solution of a desirable structure that can be captured by $Z(Q)$. The main challenge here is to detect/localize sparse

and dense ranges efficiently and in a way that yields a fast dynamic algorithm. As an oversimplification, we define a solution as having *nearly-optimal sparse structure* if it can be generated with roughly the following process:

- Each cell Q receives a set of disjoint time ranges for which it is supposed to compute an approximately optimal solution using jobs assigned to Q or its descendants. Each received time range must have starting and ending time in $Z(Q)$.
- For each time range $\mathcal{R}$ that Q receives, the algorithm partitions $\mathcal{R}$ into disjoint time ranges of three types: sparse time ranges, time ranges to be sent to Q_L for processing, and time ranges to be sent to Q_R for processing. In particular, this means that subranges of $\mathcal{R}$ are deferred to the children of Q for processing.
- For every sparse time range, Q computes an optimal solution using at most $1/\varepsilon$ jobs.
- The union of the reward/solution of all sparse time ranges on all levels must be a $(1+\varepsilon)$-approximation of the globally optimal solution without any structural requirements.

Moreover, we develop a *charging method* that enables us to partition each cell with only $|Z(Q)| = \text{poly}(1/\varepsilon, \log(n))$ points and still have the property that it contains a $(1+\varepsilon)$-approximately optimal solution with nearly-optimal sparse structure. Then, we design an approximate dynamic programming approach to efficiently compute near-optimal solutions for sparse ranges. Combined, this enables a very efficient algorithm for weighted interval scheduling. On a high-level, $Z(Q)$ enables us to eventually decompose an entire solution into sparse regions.

2.2.4 The charging method

We now outline insights of our charging arguments that enable us to convert an optimal solution OPT into a near-optimal solution OPT' with nearly-optimal sparse structure while relaxing our partitioning to only need $|Z(Q)| = \text{poly}(1/\varepsilon, \log(N))$ points. For a visual aid, see Figure 2.

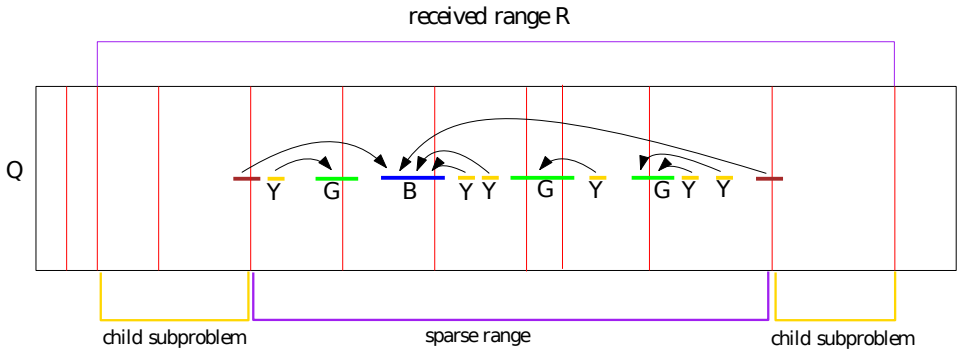


Figure 2 Visual example for charging argument.

As outlined in our overview of the nearly-optimal sparse structure, each cell Q receives a set of disjoint time ranges, with each time range having endpoints in $Z(Q)$, and must split them into three sets: sparse time ranges, time ranges for Q_L, and time ranges for Q_R. We will now modify OPT by deleting some jobs. This new solution will be denoted by OPT' and will have the following properties:

(1) OPT' exhibits nearly-optimal sparse structure; and

(2) OPT' is obtained from OPT by deleting jobs of total reward at most $O(\varepsilon \cdot w(OPT))$.

We outline an example of one such time range a cell Q may receive in Figure 2, annotated by "received range $\mathcal{R}$". We will color jobs in Figure 2 to illustrate aspects of our charging argument, but note that jobs do not actually have a color property beyond this illustration. Since our structure only allows a cell Q to use a job within its corresponding time range, any relatively valuable job that crosses between Q_L and Q_R must be used now by Q putting it in a sparse time range. One such valuable job in Figure 2 is in blue marked by "B". To have "B" belong to a sparse range, we must divide the time range $\mathcal{R}$ somewhere, as otherwise our solution in the received range will be dense. If we naively divide $\mathcal{R}$ at the partition of $Z(Q)$ to the left and right of the job "B", we might be forced to delete some valuable jobs; such jobs are pictured in green and marked by "G". Instead, we expand the division outwards in a more nuanced manner. Namely, we keep expanding outwards and looking at the job that contains the next partition point (if any). If the job's value exceeds a certain threshold, as those pictured as green and marked by "G" in Figure 2, we continue expanding. Otherwise, the job crossing a partition point is below a certain threshold, pictured as brown and not marked in Figure 2, and its deletion can be charged against the blue job. We delete such brown jobs and the corresponding partition points, i.e., the vertical red lines crossing those brown jobs, constitute the start and the end of the sparse range. By the end, we decided the starting and ending time of the sparse range, and what remains inside are blue job(s), green job(s), and yellow job(s) (also marked by "Y"). Note that yellow jobs must be completely within a partition slice of $Z(Q)$. Since we define $Z(Q)$ such that the optimal total reward within any grid slice is small, the yellow jobs have relatively small rewards compared to the total reward of green and blue jobs that we know must be large. Accordingly, we can delete the yellow jobs (to help make this time range's solution sparse) and charge their cost against a nearby green or blue job. In Figure 2, an arrow from one job to another represents a deleted job pointing towards the job who we charge its loss against. Finally, each sparse range contains only green job(s) and blue job(s). If there are more than $1/\varepsilon$ jobs in such a sparse range, we employ a simple sparsifying step detailed in the full proof.

It remains to handle the time ranges of the received range that were not put in sparse ranges. These will be time ranges that are sent to Q_L and Q_R. In Figure 2, these ranges are outlined in yellow and annotated by "child subproblem". However, the time ranges do not necessarily align with $Z(Q_L)$ or $Z(Q_R)$ as is required by nearly-optimal sparse structure. We need to adjust these ranges such that they align with $Z(Q_L)$ or $Z(Q_R)$ so we can send the ranges to the children. See Figure 3 for intuition on why we cannot just immediately "snap" these child subproblems to the partition points in $Z(Q_L)$ and $Z(Q_R)$. (We say that a range $\mathcal{R}$ is *snapped* inward (outward) within cell Q if $\mathcal{R}$ is shrunk (extended) on both sides to the closest points in $Z(Q)$. Inward snapping is illustrated in Figure 3.) Instead, we employ a similar charging argument to deal with snapping. As an analog to how we expanded outwards from the blue job for defining sparse ranges, we employ a charging argument where we contract inwards from the endpoints of the child subproblem. In summary, these charging arguments enabled us to show a solution of nearly-optimal sparse structure exists even when only partitioning each cell Q with $|Z(Q)| = \text{poly}(1/\varepsilon, \log(n))$ points.

2.2.5 Approximate dynamic programming

Now, we outline our key advance for more efficiently calculating the solution of nearly-optimal sparse structure. This structure allows us to partition time into ranges with sparse solutions. More formally, we are given a time range and we want to approximate an optimal solution within that range that uses at most $1/\varepsilon$ jobs. We outline an approximate dynamic programming approach that only requires polynomial time dependence on $1/\varepsilon$.

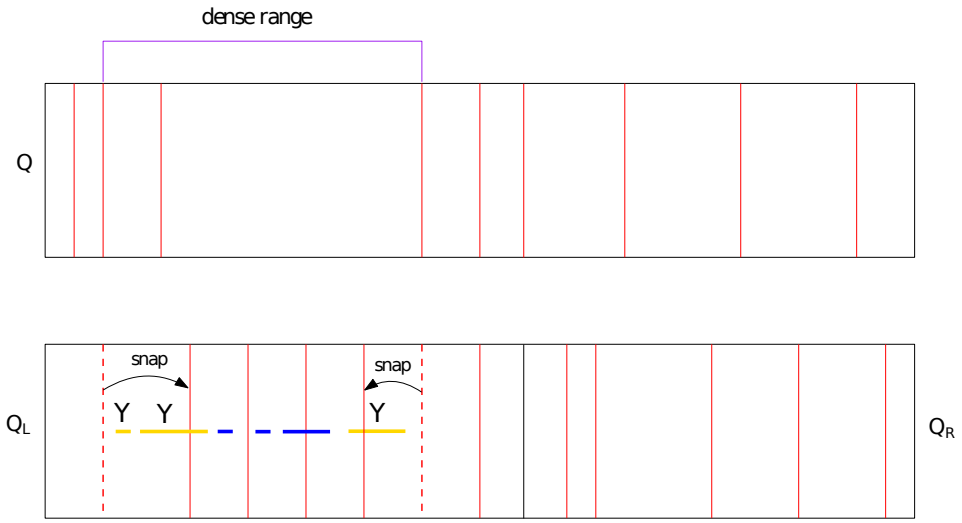


Figure 3 This example illustrates why the snapping we perform has to be done with care. The horizontal segments in this figure represent jobs. We show an initial dense range (outlined in purple) with endpoints in $Z(Q)$. With dashed vertical lines, we show where these endpoints are in Q_L. Importantly, they are not aligned with $Z(Q_L)$, i.e., the vertical dashed lines do not belong to $Z(Q_L)$. However, our structure *requires* that dense ranges align with $Z(Q_{child})$, so we must address this. If we were to naively snap the endpoints of the dense range inwards to the endpoints of $Z(Q_L)$, then we would need to delete some jobs (these deleted jobs are colored in yellow and marked by "Y"), while some other jobs would not be affected (like the remaining jobs in this example, those colored in blue). While this naive snapping may be fine in some cases, it will incur significant loss in cases in which the "Y" jobs have large weight. Notice that naively snapping outward to define a new region corresponding to the purple one is not a solution neither, as this could cause the dense time range to overlap with a previously selected sparse time range. Having overlapping ranges can cause us to choose intersecting jobs, and thus an invalid solution. Thus, we detail a more comprehensive manner of dealing with snapping.

The relatively well-known dynamic programming approach for computing weighted interval scheduling is to maintain a dynamic program where the state is a prefix range of time and the output is the maximum total reward that can be obtained in that prefix range of time. However, for our purposes, there are too many possibilities for prefix ranges of time to consider. Instead, we invert the dynamic programming approach, and have a state referencing some amount of reward, where the dynamic program returns the minimum length prefix range of time in which one can obtain a given reward. Unfortunately, there are also too many possible amounts of rewards. We observe that we do not actually need this exact state, but only an approximation. In particular, we show that one can round this state down to powers of $(1+\varepsilon^2)$ and hence significantly reduce the state-space. In the full version, we show how one can use this type of observation to quickly compute approximate dynamic programming for a near-optimal sparse solution inside any time range.

2.2.6 Comparison with Prior Work

The closest to our work is the one of [12]. In terms of improvements, we achieve the following: we remove the dependence on N and $w_{\max}$ in the running-time analysis; we obtain a deterministic approach; and, we design an algorithm with $\text{poly}(1/\varepsilon, \log n)$ update/query time, which is exponentially faster in $1/\varepsilon$ compared the prior work.

In this prior work, jobs are assumed to have length at least 1 and belong in the time-interval $[1, N]$. To remove the dependence on N and such assumptions, we designed a new way of bookkeeping jobs. Instead of using a complete binary tree on $[1, N]$ to organize jobs as done in the prior work, we employ binary balanced search tree on the endpoints of jobs. A complete binary tree on $[1, N]$ is oblivious to the density of jobs. On the other hand, and intuitively, our approach allows for "instance-based" bookkeeping: the jobs are in a natural way organized with respect to their density. Resorting to this approach incurs significant technical challenges. Namely, the structure of solution our tree maintains is hierarchically organized. However, each tree update, which requires node-rotations, breaks this structure which requires additional care in efficiently maintaining approximate solution after an update, as well as requiring an entirely different approach for maintaining a partitioning of time $Z(Q)$ within cells. Moreover, we show how to further leverage these ideas to obtain a deterministic approach.

In our work, we use borders to define the so-called sparse and dense ranges. This idea is inspired by the work of [12]. We emphasize, though, that one of our main contributions and arguably the most technically involved component is showing how to algorithmically employ those borders in running-time only polynomially dependent on $1/\varepsilon$, while [12] require exponential dependence on $1/\varepsilon$.

Our construction of auxiliary data structure $Z(Q)$ enables us to boost an $O(\log(n))$-approximate solution into a decomposition enabling a $(1+\varepsilon)$-approximate solution is inspired by the approach of [12]. They similarly develop $Z(Q)$ to boost an instead $O(1)$-approximation that fundamentally relies on the bounded coordinate assumptions of jobs being within $[1, N]$ and having length at least 1. Our different approach towards $Z(Q)$ enables simplification of some arguments as well as not relying on randomness, or on length or bounded coordinate assumptions. Further, we note that the dynamic programming approach for sparse regions that we develop is significantly faster than the enumerative approach used in the prior work, that eventually enables us to obtain a poly$(1/\varepsilon)$ dependence in the running time. The way we combine solutions over sparse regions is similar to the way it is done in the prior work.

2.3 Localizing the Time-Partitioning Method

We also show that this method of partitioning over time can be used to develop local algorithms for interval scheduling. Here, we desire to answer queries about whether a particular job is in our schedule. We hope to answer each of these queries consistently (i.e., they all agree with some approximately optimal schedule) and in less time than it would take to compute an entire schedule from scratch. Partitioning over time seems helpful for this setting, because this would enable us to focus on just the region of the job being queried. However, our previously mentioned method for maintaining borders does so in a sequential manner that we can no longer afford to do in this model of computation. Instead, we use a hierarchical approach to more easily compute the locations of borders that create regions with solutions not too big or too small.

For simplicity, we again focus on the unweighted setting with only one machine. In the standard greedy algorithm for computing unweighted interval scheduling on one machine, we repeatedly select the job *successor*(x): "*What is the interval with the earliest endpoint, of those that start after point* x*?*" (where x is the endpoint of the previously chosen job). As reading the entire problem instance would take longer than desired, an LCA requires some method of probing for information about the instance. Our LCA utilizes such successor probes to do so. For further motivation, see the full version. We outline a three-step approach towards designing an LCA that utilizes few probes:

Hierarchizing the greedy. Instead of just repeatedly using $successor(x)$ to compute the solution as the standard greedy does, we add hierarchical structure that adds no immediate value but serves as a helpful stepping stone. Consider a *binary search tree* (BST) like structure, where the root node corresponds to the entire time range $[0, N]$. Each node in the structure has a left-child and a right-child corresponding to the 1st and the 2nd half, respectively, of that node's range. Eventually, leaf nodes have no children and correspond to a time range of length one unit. At a high-level, we add hierarchical structure by considering jobs contained in some node's left-child, then considering jobs that go between the node's left-child and right-child, and then considering jobs contained in the node's right-child. This produces the same result as the standard greedy, but we do so with a hierarchical structure that will be easier to utilize.

Approximating the hierarchical greedy. Now, we modify the hierarchical greedy so that it is no longer exactly optimal but is instead an approximation. At first this will seem strictly worse, but it will yield an algorithm that is easier to localize. When processing each node, we will first check whether it is the case that both the left-child and the right-child have optimal solutions of size $> \frac{1}{\varepsilon}$. A key observation here is that checking whether a time range has an optimal solution of size $> \frac{1}{\varepsilon}$ can be done by making at most $1 + \frac{1}{\varepsilon}$ successor probes (i.e., one does not necessarily need to compute the entire optimal solution to check if it is larger than some relatively small threshold). If both the left-child and the right-child would have optimal solutions of size $> \frac{1}{\varepsilon}$, then we can afford to draw a border at the midpoint of our current node and solve the left-child and right-child independently. Jobs intersecting a border are *ignored*, and we charge the number of such ignored jobs, i.e., the number of drawn borders, to the size of solution in the corresponding left- and right-child. Ultimately, we show that the addition of these borders makes our algorithm $(1 + \varepsilon)$-approximate. Moreover, and importantly, these borders introduce *independence* between children with large solutions.

Localizing the approximate, hierarchical greedy. Finally, we localize the approximate, hierarchical greedy. To do so, we note that when some child of a node has a small optimal solution, then we can get all the information we need from that child in $O(\frac{1}{\varepsilon})$ probes. As such, if a node has a child with a small optimal solution, we can make the required probes from the small child and recurse to the large child. Otherwise, if both children have large solutions, we can draw a border at the midpoint of the current node and only need to recurse down the child which contains the job the LCA is being queried about.

With these insights, we have used our partitioning method over time for local algorithms to produce an LCA only requiring $O(\frac{\log(N)}{\varepsilon})$ successor probes.

3 Problem Setup

In the interval scheduling problem, we are given n jobs and M machines. With each job j are associated two numbers s_j and $l_j > 0$, referring to "start" and "length" respectively, meaning that the job j takes l_j time to be processed and its processing can only start at time s_j. While prior work such as [12] used assumptions such as $s_j \geq 0, l_j \geq 1$ and have an upper-bound N on $s_j + l_j$, we utilize such assumptions *only in our LCA results*. In addition, with each job j is associated weight/reward $w_j > 0$, that refers to the reward for processing the job j. The task of *interval scheduling* is to schedule jobs across machines while maximizing the total reward and respecting that each of the M machines can process at most one job at any point in time.

4 Dynamic Unweighted Interval Scheduling on a Single Machine

In this section we prove Theorem 1. As a reminder, Theorem 1 considers the case of interval scheduling in which $w_j = 1$ for each j and $M = 1$, i.e., the jobs have unit reward and there is only a single machine at our disposal. This case can also be seen as a task of finding a maximum independent set among intervals lying on the x-axis. The crux of our approach is in designing an algorithm that maintains the following invariant:

▶ **Invariant 1.** *The algorithm maintains a set of borders such that an optimal solution schedules between $1/\varepsilon$ and $2/\varepsilon$ intervals within each two consecutive borders.*

We will maintain this invariant unless the optimal solution has fewer than $1/\varepsilon$ intervals, in which case we are able to compute the solution from scratch in negligible time. We aim for our algorithm to maintain Invariant 1 while keeping track of the optimal solution between each pair of consecutive borders. The high level intuition for this is that if we do not maintain too many borders, then our solution must be very good (our solution decreases by size at most one every time we add a new border). Furthermore, if the optimal solution within borders is small, it is likely easier for us to maintain said solutions. We prove that this invariant enables a high-quality approximation:

▶ **Lemma 4.** *A solution that maintains an optimal solution within consecutive pairs of a set of borders, where the optimal solution within each pair of consecutive borders contains at least K intervals, maintains a $\frac{K+1}{K}$-approximation.*

Proof. For our analysis, suppose there are implicit borders at $-\infty$ and $+\infty$ so that all jobs are within the range of borders. Consider an optimal solution OPT. We will now design a K-approximate optimal solution OPT' as follows: given OPT, delete all intervals in OPT that overlap a drawn border. Fix an interval J appearing in OPT but not in OPT'. Assume that J intersects the i-th border. Recall that between the $(i-1)$-st and the i-th border there are at least K intervals in OPT'. Moreover, at most one interval from OPT intersects the i-th border. Hence, to show that OPT' is a $\frac{K+1}{K}$-approximation of OPT, we can charge the removal of J to the intervals appearing between the $(i-1)$-st and the i-th border in OPT'. ◀

Not only does Invariant 1 enable high-quality solutions, but it also assists us in quickly maintaining such a solution. We can maintain a data structure with $O(\frac{\log(n)}{\varepsilon})$ updates and $O(\log(n))$ queries that moves the borders to maintain the invariant and thus maintains an $(1+\varepsilon)$-approximation as implied by Lemma 4.

▶ **Theorem 1** (Unweighted dynamic, single machine)**.** *Let $\mathcal{J}$ be a set of n jobs. For any $\varepsilon > 0$, there exists a fully dynamic algorithm for $(1+\varepsilon)$-approximate unweighted interval scheduling for $\mathcal{J}$ on a single machine performing updates in $O\left(\frac{\log(n)}{\varepsilon}\right)$ and queries in $O(\log(n))$ worst-case time.*

Proof. Our goal now is to design an algorithm that maintains Invariant 1, which by Lemma 4 and for $K = 1/\varepsilon$ will result in a $(1+\varepsilon)$-approximation of Maximum-IS.

On a high-level, our algorithm will maintain a set of borders. When compiling a solution of intervals, the algorithm will not use any interval that contains any of the borders, but proceed by computing an optimal solution between each two consecutive borders. The union of those between-border solutions is the final solution. Moreover, we will maintain the invariant that the optimal solution for every contiguous region is of size within $[\frac{1}{\varepsilon}, \frac{2}{\varepsilon})$.

In the rest, we show how to implement these steps in the claimed running time.

Maintained data-structures. Our algorithm maintains a balanced binary search tree T_{all} of intervals sorted by their starting points. Each node of T_{all} will also maintain the end-point of the corresponding interval. It is well-known how to implement a balanced binary search tree with $O(\log n)$ worst-case running time per insertion, deletion and search query. Using such an implementation, the algorithm can in $O(\log n)$ time find the smallest ending-point in a prefix/suffix on the intervals sorted by their starting-points. That is, in $O(\log n)$ time we can find the interval that ends earliest, among those that start after a certain time.

In addition, the algorithm also maintains a balanced binary search tree T_{borders} of the borders currently drawn.

Also, we will maintain one more balanced binary search tree T_{sol} that will store the intervals that are in our current solution.

We will use that for any range with optimal solution of size S, we can make $O(S)$ queries to these data structures to obtain an optimal solution for the range in $O(S \cdot \log n)$ time.

Update after an insertion. Upon insertion of an interval J, we add J to T_{all}. We make a query to T_{borders} to check whether J overlaps a border. If it does, we need to do nothing; in this case, we ignore J even if it belongs to an optimal solution. If it does not, we recompute the optimal solution within the two borders adjacent to J. If after recomputing, the new solution between the two borders is too large, i.e, it has at least $\frac{2}{\varepsilon}$ intervals, then draw/add a border between the $\frac{1}{\varepsilon}$-th and the $(1+\frac{1}{\varepsilon})$-th of those intervals.

Update after a deletion. Upon deletion of an interval J, we delete J from T_{all}. If J was not in our solution, we do nothing else. Otherwise, we recompute the optimal solution within the borders adjacent to J and modify T_{sol} accordingly. Let those borders be the i-th and the $(i+1)$-st. If the new solution between borders i and $i+1$ now has size less than $1/\varepsilon$ (it would be size exactly $1/\varepsilon$), we delete an arbitrary one of the two borders (thus combining this region with an adjacent region). Then, we recompute the optimal solution within the (now larger) region J is in. If this results in a solution of size at least $2/\varepsilon$, we will need to split the newly created region by adding a border. Before splitting, the solution will have size upper-bounded by one more than the size of the solutions within the two regions before combining them as an interval may have overlapped the now deleted border (one region with size exactly $\frac{1}{\varepsilon}-1$ and the other upper-bounded by $\frac{2}{\varepsilon}-1$). Thus, the solution has size at in range $[2/\varepsilon, \frac{3}{\varepsilon})$. We can add a border between interval $1/\varepsilon$ and $1/\varepsilon+1$ of the optimal solution, and will have a region with exactly $1/\varepsilon$ intervals and another with $[1/\varepsilon, 2/\varepsilon)$ intervals, maintaining our invariant.

In all of these, the optimal solution for each region has size $O(1/\varepsilon)$, so recomputing takes $O(\log(n)/\varepsilon)$ time.

For queries, we will have maintained T_{sol} in our updates such that it contains exactly the intervals in our solution. So each query we just need to do a lookup to see if the interval is in T_{sol} in $O(\log n)$ time. ◀

This result improves the best-known time complexities [4, 12]. Unfortunately, it does not immediately generalize well to the weighted variant. In the full version, we show our more technically-challenging result for the weighted variant.

References

1 Pankaj K Agarwal and Marc J Van Kreveld. *Label placement by maximum independent set in rectangles*, volume 1998. Utrecht University: Information and Computing Sciences, 1998.

2 Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1132–1139. Society for Industrial and Applied Mathematics, 2012.

3 Esther M Arkin and Ellen B Silverberg. Scheduling jobs with fixed start and end times. *Discrete Applied Mathematics*, 18(1):1–8, 1987.

4 Sujoy Bhore, Jean Cardinal, John Iacono, and Grigorios Koumoutsos. Dynamic geometric independent set. *arXiv preprint*, 2020. `arXiv:2007.08643`.

5 Giorgio C Buttazzo, Marko Bertogna, and Gang Yao. Limited preemptive scheduling for real-time systems. a survey. *IEEE Transactions on Industrial Informatics*, 9(1):3–15, 2012.

6 Jean Cardinal, John Iacono, and Grigorios Koumoutsos. Worst-case efficient dynamic geometric independent set. In *29th Annual European Symposium on Algorithms (ESA 2021)*, volume 204, page 25. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

7 José R Correa and Andreas S Schulz. Single-machine scheduling with precedence constraints. *Mathematics of Operations Research*, 30(4):1005–1021, 2005.

8 A FRANK. Some polynomial algorithms for certain graphs and hypergraphs. In *Proceedings of the 5th British Combinatorial Conference, 1975*. Utilitas Mathematica, 1975.

9 Alexander Gavruskin, Bakhadyr Khoussainov, Mikhail Kokho, and Jiamou Liu. Dynamic interval scheduling for multiple machines. In *International Symposium on Algorithms and Computation*, pages 235–246. Springer, 2014.

10 Alexander Gavruskin, Bakhadyr Khoussainov, Mikhail Kokho, and Jiamou Liu. Dynamic algorithms for monotonic interval scheduling problem. *Theoretical Computer Science*, 562:227–242, 2015.

11 Paweł Gawrychowski and Karol Pokorski. Sublinear dynamic interval scheduling (on one or multiple machines). *arXiv preprint*, 2022. `arXiv:2203.14310`.

12 Monika Henzinger, Stefan Neumann, and Andreas Wiese. Dynamic approximate maximum independent set of intervals, hypercubes and hyperrectangles. In *36th International Symposium on Computational Geometry (SoCG 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

13 Dorit S Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and vlsi. *Journal of the ACM (JACM)*, 32(1):130–136, 1985.

14 Antoon WJ Kolen, Jan Karel Lenstra, Christos H Papadimitriou, and Frits CR Spieksma. Interval scheduling: A survey. *Naval Research Logistics (NRL)*, 54(5):530–543, 2007.

15 Jan Karel Lenstra and AHG Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26(1):22–35, 1978.

16 Elaine Levey and Thomas Rothvoss. A (1+ epsilon)-approximation for makespan scheduling with precedence constraints using lp hierarchies. *SIAM Journal on Computing*, pages STOC16–201, 2019.

17 Aristide Mingozzi, Marco A Boschetti, Salvatore Ricciardelli, and Lucio Bianco. A set partitioning approach to the crew scheduling problem. *Operations Research*, 47(6):873–888, 1999.

18 Michael Pinedo. *Scheduling*, volume 29. Springer, 2012.

19 Julien Robert and Nicolas Schabanel. Non-clairvoyant scheduling with precedence constraints. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 491–500, 2008.

20 Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. *arXiv preprint*, 2011. `arXiv:1104.1377`.

21 Pinal Salot. A survey of various scheduling algorithm in cloud computing environment. *International Journal of Research in Engineering and Technology*, 2(2):131–135, 2013.

22 Raksha Sharma, Vishnu Kant Soni, Manoj Kumar Mishra, and Prachet Bhuyan. A survey of job scheduling and resource management in grid computing. *world academy of science, engineering and technology*, 64:461–466, 2010.

23 Martin Skutella and Marc Uetz. Stochastic machine scheduling with precedence constraints. *SIAM Journal on Computing*, 34(4):788–802, 2005.

24 Eva Tardos and Jon Kleinberg. Algorithm design, 2005.

25 Bram Verweij and Karen Aardal. An optimisation algorithm for maximum independent set with applications in map labelling. In *European Symposium on Algorithms*, pages 426–437. Springer, 1999.

Optimal (Degree+1)-Coloring in Congested Clique

Sam Coy
University of Warwick, Coventry, UK

Artur Czumaj
University of Warwick, Coventry, UK

Peter Davies
Durham University, UK

Gopinath Mishra
University of Warwick, Coventry, UK

Abstract

We consider the distributed complexity of the (degree+1)-list coloring problem, in which each node u of degree $d(u)$ is assigned a palette of $d(u)+1$ colors, and the goal is to find a proper coloring using these color palettes. The (degree+1)-list coloring problem is a natural generalization of the classical $(\Delta+1)$-coloring and $(\Delta+1)$-list coloring problems, both being benchmark problems extensively studied in distributed and parallel computing.

In this paper we settle the complexity of the (degree+1)-list coloring problem in the Congested Clique model by showing that it can be solved deterministically in a constant number of rounds.

2012 ACM Subject Classification Theory of computation → Massively parallel algorithms; Theory of computation → Distributed algorithms; Theory of computation → Pseudorandomness and derandomization; Mathematics of computing → Graph algorithms

Keywords and phrases Distributed computing, graph coloring, parallel computing

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.46

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2306.12071

Funding *Sam Coy*: Research supported in part by the Centre for Discrete Mathematics and its Applications (DIMAP), by an EPSRC studentship, and by the Simons Foundation Award No. 663281 granted to the Institute of Mathematics of the Polish Academy of Sciences for the years 2021–2023.
Artur Czumaj: Research supported in part by the Centre for Discrete Mathematics and its Applications, by EPSRC award EP/V01305X/1, by a Weizmann-UK Making Connections Grant, by an IBM Award, and by the Simons Foundation Award No. 663281 granted to the Institute of Mathematics of the Polish Academy of Sciences for the years 2021–2023.
Gopinath Mishra: Research supported in part by the Centre for Discrete Mathematics and its Applications (DIMAP), by EPSRC award EP/V01305X/1, and by the Simons Foundation Award No. 663281 granted to the Institute of Mathematics of the Polish Academy of Sciences for the years 2021–2023.

1 Introduction

Graph coloring problems are among the most extensively studied problems in the area of distributed graph algorithms. In the distributed graph coloring problem, we are given an undirected graph $G = (V, E)$ and the goal is to properly color the nodes of G such that no edge in E is monochromatic. In the distributed setting, the nodes of G correspond to devices that interact by exchanging messages throughout some underlying communication network such that the nodes communicate with each other in synchronous rounds by exchanging

messages over the edges in the network. Initially, the nodes do not know anything about G (except possibly for some global parameters, e.g., the number of nodes n or the maximum degree Δ). At the end of computation, each node $v \in V$ should output its color (from a given palette) in the computed coloring. The *time* or *round complexity* of a distributed algorithm is the total number of rounds until all nodes terminate.

If adjacent nodes in G can exchange arbitrarily large messages in each communication round (and hence the underlying communication network is equal to the input graph G), this distributed model is known as the LOCAL model [30], and if messages are restricted to $O(\log n)$ bits per edge (limited bandwidth) in each round, the model is known as the CONGEST model [38]. If we allow all-to-all communication (i.e., the underlying network is a complete graph and thus the communication is independent of the input graph G) using messages of size $O(\log n)$ bits then the model is known as the CongestedClique model [31].

The most fundamental graph coloring problem in distributed computing (studied in the seminal paper by Linial [30] that introduced the LOCAL model) is $(\Delta+1)$-*coloring*: assuming that the input graph G is of maximum degree Δ, the objective is to color nodes of G using $\Delta+1$ colors from $\{1, 2, \ldots, \Delta+1\}$. The $(\Delta+1)$-coloring problem can be easily solved by a sequential greedy algorithm, but the interaction between local and global aspects of graph coloring create some non-trivial problems in a distributed setting. The problem has been used as a benchmark to study distributed symmetry breaking in graphs, and it is at the very core of the area of distributed graph algorithms. $(\Delta+1)$-*list coloring* is a natural generalization of $(\Delta+1)$-coloring: each node has a an arbitrary palette of $\Delta+1$ colors, and the goal is to compute a legal coloring in which each node is assigned a color from its own palette. A further generalization is the *(degree+1)-list coloring (D1LC)* problem, which is the same as the $(\Delta+1)$-list coloring problem except that the size of each node v's palette is $d(v)+1$, which might be much smaller than $\Delta+1$. These three problems always have a legal coloring (easily found sequentially using a greedy approach), and the main challenge in the distributed setting is to find the required coloring in as few rounds as possible.

These three graph coloring problems have been studied extensively in distributed computing, though $(\Delta+1)$-coloring, as the simplest, has attracted most attention. However, one can also argue that (degree+1)-list coloring, as the most versatile, is more algorithmically fundamental than $(\Delta+1)$-coloring. For example, given a partial solution to a $(\Delta+1)$-coloring problem, the remaining coloring problem on the uncolored nodes is an instance of the (degree+1)-list coloring problem. The (degree+1)-list coloring problem is self-reducible: after computing a partial solution to a (degree+1)-list coloring problem, the remaining problem is still a (degree+1)-list coloring problem. It also naturally appears as a subproblem in more constrained coloring problems: for example, it has been used as a subroutine in distributed Δ-coloring algorithms (see, e.g., [19]), in efficient $(\Delta+1)$-coloring and edge-coloring algorithms (see, e.g., [28]), and in other graph coloring applications (see, e.g. [4]).

Following an increasing interest in the distributed computing community for (degree+1)-list coloring, it is natural to formulate a central challenge relating it to $(\Delta+1)$-coloring:

> *Can we solve the (degree+1)-list coloring problem in asymptotically the same round complexity as the simpler* $(\Delta+1)$*-coloring problem?*

This challenge has been elusive for many years and only in the last year the affirmative answer was given for randomized algorithms in LOCAL and CONGEST. First, in a recent breakthrough, Halldórsson, Kuhn, Nolin, and Tonoyan [24] gave a randomized $O(\log^3 \log n)$-

round distributed algorithm for (degree+1)-list coloring in the LOCAL model, matching the state-of-the-art complexity for the $(\Delta+1)$-coloring problem [10, 40]. This has been later extended to the CONGEST model by Halldórsson, Nolin, and Tonoyan [25], who designed a randomized algorithm for (degree+1)-list coloring that runs in $O(\log^5 \log n)$-round, matching the state-of-the-art complexity for the $(\Delta+1)$-coloring problem in CONGEST [23].

The main contribution of our paper is a complete resolution of this challenge in the CongestedClique model, and in fact, even for deterministic algorithms. We settle the complexity of the (degree+1)-list coloring problem in CongestedClique by showing that it can be solved deterministically in a constant number of rounds.

▶ **Theorem 1.** *There is a deterministic* CongestedClique *algorithm which finds a (degree+1)-list coloring of any graph in a constant number of rounds.*

1.1 Background and Related Works

Distributed graph coloring problems have been extensively studied in the last three decades, starting with a seminal paper by Linial [30] that introduced the LOCAL model and originated the area of local graph algorithms. Since the $(\Delta+1)$-coloring problem can be solved by a simple sequential greedy algorithm, but it is challenging to be solved efficiently in distributed (and parallel) setting, the $(\Delta+1)$-coloring problem became a benchmark problem for distributed computing and a significant amount of research has been devoted to the study of these problems in all main distributed models: LOCAL, CONGEST, and CongestedClique. The monograph [6] gives a comprehensive description of many of the earlier results.

It is long known from research on parallel algorithms that $(\Delta+1)$-coloring can be computed in $O(\log n)$ rounds by randomized algorithms in the LOCAL model [2, 32]. Linial [30] observed that for smaller values of Δ, one can do better: he showed that it is possible to deterministically color arbitrary graphs of maximum degree Δ with $O(\Delta^2)$ colors in $O(\log^* n)$ rounds; this can be easily extended to obtain a deterministic LOCAL algorithm for $(\Delta+1)$-coloring that runs in $O(\Delta^2+\log^* n)$ rounds, and thus in bounded degree graphs, a $(\Delta+1)$-coloring can be computed in $O(\log^* n)$ rounds. Improve results have since been found for general values of Δ: the current state-of-the-art for the $(\Delta+1)$-coloring problem in LOCAL is $O(\log^3 \log n)$ rounds for randomized algorithms [10, 40] and $O(\log^2 \Delta \cdot \log n)$ for deterministic algorithms [22]. Furthermore, the fastest algorithms mentioned above can be modified to work also for the more general $(\Delta+1)$-list coloring problem in the LOCAL model. (In fact, many of those algorithms critically rely on this problem as a subroutine.)

For the CONGEST model, the parallel algorithms mentioned above [2, 32] can be implemented in the CONGEST model to obtain randomized algorithms for both the $(\Delta+1)$-coloring and $(\Delta+1)$-list coloring problems that run in $O(\log n)$ rounds. Only recently this bound has been improved for all values of Δ: In a seminal paper, Halldórsson et al. [23] designed a randomized CONGEST algorithm that solves the $(\Delta+1)$-coloring and $(\Delta+1)$-list coloring problems in $O(\log^5 \log n)$ rounds. For deterministic computation, the best LOCAL algorithm [22] works directly in CONGEST, running in $O(\log^2 \Delta \cdot \log n)$ rounds.

As for the lower bounds, one of the first results in distributed computing was a lower bound in LOCAL of $\Omega(\log^* n)$ rounds for computing an $O(1)$-coloring of a graph of maximum degree $\Delta = 2$, shown by Linial [30] for deterministic algorithms, and by Naor [35] for randomized ones. Improved coloring lower bounds have not been forthcoming, and $\Omega(\log^* n)$ rounds is still the best known lower bound complexity for the $(\Delta+1)$-coloring problem in LOCAL and CONGEST.

This lower bound does not hold in the CongestedClique model, and in fact we can color faster. After years of gradual improvements, Parter [36] exploited the LOCAL shattering approach from [10] to give the first sublogarithmic-time randomized $(\Delta+1)$-coloring algorithm for CongestedClique, which runs in $O(\log\log\Delta\cdot\log^*\Delta)$ rounds. This bound was later improved by Parter and Su [37] to $O(\log^*\Delta)$ rounds. Finally, Chang et al. [9] settled the randomized complexity of $(\Delta+1)$-coloring (and also for $(\Delta+1)$-list coloring) and obtained a randomized CongestedClique algorithm that runs in a constant number of rounds. This result has been later simplified and extended into a deterministic constant-round CongestedClique algorithm by Czumaj et al. [15].

(degree+1)-list coloring (DILC). The D1LC problem in distributed setting has been studied both on its own, and also as a tool in designing distributed algorithms for other coloring problems, like $(\Delta+1)$-coloring, $(\Delta+1)$-list coloring, and Δ-coloring. The problem is more general than the $(\Delta+1)$-coloring and the $(\Delta+1)$-list coloring problems, and the difficulty of dealing with vertices having color palettes of significantly different sizes adds an additional challenge. As the result, until very recently the obtained complexity bounds have been significantly weaker than the bounds for the $(\Delta+1)$-coloring problem, see, e.g., [5, 20, 28]. This changed last year, when in a recent breakthrough Halldórsson et al. [24] gave a randomized $O(\log^3\log n)$-round distributed algorithm for D1LC in the LOCAL distributed model. Observe that this bound matches the state-of-the-art complexity for the (easier) $(\Delta+1)$-coloring problem [10]. This work has been later extended to the CONGEST model by Halldórsson et al. [25], who designed a randomized CONGEST algorithm for D1LC that runs in $O(\log^5\log n)$ rounds. Similarly as for the LOCAL model, this bound matches the state-of-the-art complexity for the $(\Delta+1)$-coloring problem in CONGEST [23].

Specifically for the CongestedClique model, the only earlier D1LC result we are aware of is by Bamberger et al. [5], who extended their own CONGEST algorithm for the problem to obtain a deterministic D1LC algorithm requiring $O(\log\Delta\log\log\Delta)$ rounds in CongestedClique. However, the randomized state-of-the-art D1LC bound in the CongestedClique model follows from the aforementioned $O(\log^5\log n)$-round CONGEST algorithm by Halldórsson et al. [25], which works directly in CongestedClique. This should be compared with the state-of-the-art $O(1)$-round CongestedClique algorithms for $(\Delta+1)$-coloring [9, 15].

Recent work in DILC on MPC. Various coloring problems have been also studied in a related model of parallel computation, the so-called *Massively Parallel Computation* (MPC) model. The MPC model, introduced by Karloff et al. [27], is now a standard theoretical model for parallel algorithms. The MPC model with $O(n)$ local space and n machines is essentially equivalent to the CongestedClique model (see, e.g., [7, 26]), and this implies that many MPC algorithms can be easily transferred to the CongestedClique model. (However, this relationship requires that the local space of MPC is $O(n)$ words, not more.)

Both the $(\Delta+1)$-coloring and $(\Delta+1)$-list coloring problems have been studied in MPC extensively (see, e.g., [5, 15] for linear local space MPC and [5, 9, 14] for sublinear local space MPC). We are aware only of a few works for the D1LC problem on MPC, see [5, 11, 24]. The work most relevant to our paper is the result of Halldórsson et al. [24]. They give a constant-round MPC algorithm assuming the local MPC space is *slightly superlinear*, i.e., $\Omega(n\log^4 n)$ [24, Corollary 2]. This result relies on the palette sparsification approach due to Alon and Assadi [1] (see also [3]) to the D1LC problem, which reduces the problem to a sparse instance of size $O(n\log^4 n)$; hence, on an MPC with $\Omega(n\log^4 n)$ local space one can put the entire graph on a single MPC machine and then solve the problem in a single

round. Given the similarity of CongestedClique and the MPC model with *linear local space*, one could hope that the use of "slightly superlinear" MPC local space in [24] can be overcome and the approach can allow the problem to be solved in linear local space, resulting in a CongestedClique algorithm with a similar performance. Unfortunately, we do not think this is the case. The palette sparsification approach of Alon and Assadi does not reduce the number of vertices in an input graph, and can only hope to reduce the maximum degree of the graph down to, at best, $\Theta(\log n)$. It has no effect on graphs that already have $\Delta = O(\log n)$, and these sparse graphs are still hard instances for D1LC, with no better known upper bound than on general graphs.

Further, we have recently seen a similar situation in $(\Delta + 1)$-coloring. The palette sparsification by Assadi et al. [3] trivially implies a constant-round MPC algorithm for $(\Delta + 1)$-coloring with local space $\Omega(n \log^2 n)$, but does not give a constant-round algorithm for $(\Delta + 1)$-coloring in CongestedClique. Only by using a fundamentally different approach were Chang et al. [9] and then (deterministically) Czumaj et al. [13] able to obtain constant-round $(\Delta+1)$-coloring algorithms in CongestedClique. Hence, despite having a constant-round algorithm for D1LC in MPC with local space $\Omega(n \log^4 n)$, possibly a different approach than palette sparsification is needed to achieve a similar performance for D1LC in CongestedClique.

Derandomization tools for distributed coloring algorithms. In our paper we rely on a recently developed general scheme for derandomization in the CongestedClique model (and used also extensively in the MPC model) by combining the methods of bounded independence with efficient computation of conditional expectations. This method was first applied by Censor-Hillel et al. [8], and has since been used in several other works for graph coloring problems, (see, e.g., [15, 36]), and for other problems in CongestedClique and MPC.

The underlying idea begins with the design of a randomized algorithm using random choices with only *limited independence*, e.g., $O(1)$-wise-independence. Then, each round of the randomized algorithm can be simulated by giving all nodes a shared random seed of $O(\log n)$ bits. Next, the nodes deterministically compute a seed which is at least as good as a random seed is in expectation. This is done by using an appropriate estimation of the local quality of a seed, which can be aggregated into a global measure of the quality of the seed. Combining this with the techniques of conditional expectation, pessimistic estimators, and bounded independence, this allows selection of the bits of the seed "batch-by-batch," where each batch consists of $O(\log n)$ bits. Once all bits of the seed are computed, we can used it to simulate the random choices of that round, as it would have been performed by a randomized algorithm. A more detailed explanation of this approach is given in Section 2.2.

1.2 Technical Overview

The core part of our constant-round deterministic CongestedClique D1LC algorithm (given as BucketColor, Algorithm 4) does *not* follow the route of recent D1LC algorithms for LOCAL and CONGEST due to Halldórsson et al. [24, 25]. Instead, it uses fundamentally different techniques, extending the approach developed recently in a simple deterministic $O(1)$-round CongestedClique algorithm for $(\Delta + 1)$-list coloring of Czumaj, Davies, and Parter [15]. Their $(\Delta + 1)$-list coloring algorithm works by partitioning nodes into Δ^{ε} *buckets*, for a small constant ε. (This partitioning is initially at random, but then it is derandomized using the method of conditional expectations). The available colors are also distributed among all buckets, except for one *"leftover" bucket*, which is left without colors and is set aside to be colored later. Then, each node's palette is restricted to only the colors assigned to its bucket (except those in the leftover bucket, whose palettes are not restricted). This ensures

that nodes in different buckets have entirely disjoint palettes, and therefore edges between different buckets can be removed from the graph, since they would never cause a coloring conflict. One important property is that nodes still have sufficient colors when their palettes are restricted in this way. This is achieved in [15] using two main arguments: firstly, the fact that colors are distributed among one fewer buckets than nodes provides enough "slack" to ensure that with reasonably high probability, a node would receive more colors than neighbors in its bucket. Secondly, the few nodes that do *not* satisfy this property induce a small graph (of size $O(n)$), and therefore can be collected onto a single network node and colored separately.

Using the approach from [15] sketched above, a $(\Delta + 1)$-list coloring instance can be reduced into multiple smaller $(\Delta + 1)$-list coloring instances (i.e., on fewer nodes and with a new, lower maximum degree) that are *independent* (since they had disjoint palettes), and so can be solved in parallel without risking coloring conflicts. The final part of the analysis of [15] was to show that, after recursively performing this bucketing process $O(1)$ times, these instances are of $O(n)$ size and therefore they could be collected to individual nodes and solved in a constant number of rounds in CongestedClique.

There are major barriers to extending this approach to (degree+1)-list coloring. Crucially, it required the number of buckets to be dependent on Δ, and all nodes' palette sizes to be at least Δ. Dividing nodes among too few buckets would cause the induced graphs to be too large, and the algorithm would not terminate in $O(1)$ rounds; using too few buckets would fail to provide nodes with sufficient colors in their bucket. In the D1LC problem, we no longer have a uniform bound on palette size, so it is unclear how to perform this bucketing.

Our first major conceptual change is that, rather than simply partitioning among a number (dependent on Δ) of equivalent buckets, we instead use a tree-structured hierarchy of buckets, with the $O(\log \log \Delta)$ levels in the hierarchy corresponding to doubly-exponentially increasing degree ranges (see Figure 1). Nodes with degree $d(v)$ will be mapped to a bucket in a level containing (very approximately) $d(v)^{0.7}$ buckets. Colors will be mapped to a top-level (leaf) bucket, but will also be assigned to every bucket on the leaf-root path in the bucket tree (see Figure 2). We can therefore discard all edges between different buckets that do not have an ancestor-descendant relationship, since these buckets will have disjoint palettes.

This change allows nodes to be bucketed correctly according to their own degree. However, it introduces several new difficulties:

- We no longer get a good bound on the number of lower-degree neighbors of a node v that may share colors with it. We can only hope to prove that v receives enough colors relative to its higher (or same)-degree neighbors.
- The technique of having a leftover bucket which is not assigned colors no longer works to provide slack (nor even makes sense - we would need a leftover bucket at every level, but, for example, level 0 only contains one bucket).

In order to cope with these challenges, firstly, we employ the observation that if we were to greedily color in non-increasing order of degree, we would require nodes to have a palette size of $d^+(v) + 1$ (where $d^+(v)$ is the number of v's neighbors of equal or higher degree), rather that $d(v) + 1$ (since $d^+(v)$ of v's neighbors will have been colored at the point v is considered). Therefore, we argue that we can still show that the graph is colorable even though our bucketing procedure may leave nodes with many more lower-degree neighbors than palette colors. (It is not necessarily clear how to find such a coloring in a parallel fashion, but in our analysis, we will be able to address this issue.)

This observation also helps us with the problem of generating slack without a leftover bucket. We show that, since lower-degree neighbors are now effectively providing slack,

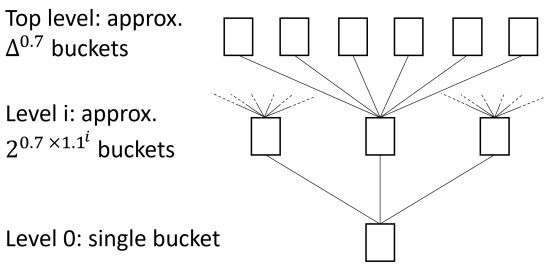


Figure 1 Bucket structure.

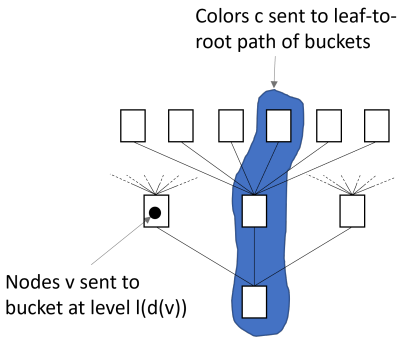


Figure 2 Partitioning nodes and colors.

only nodes with very few lower-degree neighbors may not receive enough colors (relative to higher-degree neighbors) in their bucket. It transpires that we can easily generate slack for these nodes prior to BucketColor via derandomizations of fairly standard procedures (ColorTrial, Algorithm 2, and Subsample, Algorithm 3). The randomized bases for all these procedures would inevitably result in some nodes failing to meet the necessary properties for the next stage. To overcome this, we derandomize all of these procedures, using the method of conditional expectations. As well as making the algorithm deterministic, this has the important property of ensuring that *failed* nodes form an $O(n)$-size induced graph, which can be easily dealt with later.

Having solved the problem of slack for the bucketing process (by showing that nodes have received palettes of size at least $d^+(v) + 1$ within their buckets), it remains to find a parallel analog to greedily coloring in non-increasing order of degree. Our approach here is to repeatedly move all nodes from their current bucket to a child of that bucket in the bucket tree (which further restricts their neighborhood and available palette). We show that, by correct choice of bucket and order of node consideration, we will always be able to find child buckets such that each node still has palette size at least $d^+(v) + 1$ according to the new bucket assignment. We also show that, after $O(1)$ iterations of this process, nodes only have one palette color in their bucket, and zero neighbors earlier in the coloring order. Then, all nodes can safely color themselves this palette color, and the coloring is complete.

The overall structure of the main algorithm Color (Algorithm 1) is more complicated, since Subsample produces a graph G' of leftover nodes that are deferred to be colored later. We recursively run Algorithm 1 on this graph G', and show that it is sufficiently smaller than the original input graph that after $O(1)$ recursive calls, the remaining graph has size $O(n)$ and can be collected and solved on a single node.

If we combine all these tools together then we will be able to obtain a randomized CongestedClique algorithm that finds a (degree+1)-list coloring of any graph in $O(1)$ rounds. Using the *method of conditional expectation* using *bounded-independence hash function* (see Section 2.2 and Appendix A), each randomized step of our algorithm can be derandomized.

2 Preliminaries

The main model considered in this paper is CongestedClique, as introduced by Lotker et al. [31]. It is a variant of CONGEST, in which nodes can send a message of size $O(\log n)$ to each neighboring node in the graph in each communication round: the difference is that CongestedClique allows all-to-all communication, and hence the underlying communication

network is a complete graph on the nodes V. In particular, this allows the communication to be performed between all pairs of nodes rather than being restricted to the edges of the input graph. CongestedClique has been introduced as a theoretical model to study overlay networks: an abstraction that separates the problems emerging from the topology of the communication network from the problems emerging from the structure of the problem at hand. It allows us to study a model in which each pair of nodes can communicate, and we do not consider any details of how this communication is executed by the underlying network.

The **degree+1 list coloring (D1LC) problem** is for a given graph $G = (V, E)$ on n nodes and given color palettes $\Psi(u)$ assigned to each node $u \in V$, such that $|\Psi(u)| \geq d(u)+1$, the objective to find a proper coloring of nodes in G such that each node as assigned to a color from its color palette (and, as in proper coloring, no edge in G is monochromatic). The input to the D1LC problem in CongestedClique is a graph G, where each node v of G has assigned a network node and this network node knows $\Psi(v)$ and all neighbors of v in G.

A useful property of the CongestedClique model is that thanks to the constant-round routing algorithm of Lenzen [29], information can be redistributed essentially arbitrarily in the communication network, so there is no need to associate the computational entities with nodes in the input graph G. (This is in stark contrast to the related LOCAL and CONGEST distributed models in which the link between computation and input graph locality is integral.) In particular, this allows us to collect graphs of size $O(n)$ on a single node in $O(1)$ rounds. Because of this "decoupling" of the computation from the input graph, where appropriate we will distinguish the nodes in their roles as computational entities ("network nodes") from the nodes in the input graph ("graph nodes").

2.1 Notation

For $k \in \mathbb{N}$ we let $[k] := \{1, \ldots, k\}$. We consider a graph $G = (V(G), E(G))$ with $V(G)$ as the node set and $E(G)$ as the edge set. The size of a graph G refers to the number of edges in G and is denoted by $|G|$. The set of neighbors of a node v is denoted by $N_G(v)$ and the degree of a node v is denoted by $d_G(v)$. The maximum degree of any node in G is denoted by Δ_G. For any node v, we partition its neighbors into two sets $N^+_G(v) := \{u \in N_G(v) : d_G(u) \geq d_G(v)\}$ and $N^-_G(v) := \{u \in N_G(v) : d_G(u) < d_G(v)\}$, and let $d^+_G(v) := |N^+_G(v)|$ and $d^-_G(v) := |N^-_G(v)|$. When G is clear from the context, we suppress G from the subscripts of the notation.

For the coloring problem, for a node v, $\Psi_G(v)$ denotes the list of colors in the color palette of v and $p_G(v) := |\Psi_G(v)|$. As we proceed in coloring the nodes of the input graph G the graph will be changing and the color palettes of the nodes may also change. We will ensure that at any moment, if G denotes the current graph then we have $p_G(v) \geq d_G(v) + 1$. We use $\mathcal{C}$ to denote the set of all colors present in the palette of any node (in a given moment).

For binary strings a and a' in $\{0,1\}^*$, $a \sqsubseteq a'$ denotes that a is a prefix of a', and $a \sqsubset a'$ denotes that it is a *strict* prefix of a'. Furthermore, $a' \sqsupseteq a$ iff $a \sqsubseteq a'$, and $a' \sqsupset a$ iff $a \sqsubset a'$.

Due to the space constraint, the missing proofs are deferred to the full version.

2.2 Derandomization in CongestedClique

The *method of conditional expectations* using *bounded-independence hash functions* is a nowadays classical technique for the derandomization of algorithms [17, 33, 34, 39]. Starting with the recent work of Censor-Hillel et al. [8], this approach has been found very powerful also in the setting of distributed and parallel algorithms, see e.g., [5, 12, 13, 14, 15, 16, 18, 21, 36].

This technique requires that we show that our randomized algorithm can be made to work in expectation using only bounded-independence. It is known that small families of bounded-independence hash functions exist, and that hash functions in these families can be specified by a short seed. It is also known that such a family must contain a hash function which beats the expectation due to the probabilistic method. Using these facts, we can perform an efficient search for a hash function which beats the expectation by iteratively setting a larger and larger prefix of the seed of the hash function.

We give a more detailed explanation of bounded-independence hash functions and the method of conditional expectations with its implementation in Appendix A.

3 The DILC Algorithm

The framework of our CongestedClique algorithm is Color(G, x) (Algorithm 1), which colors graph G relying on three main procedures: ColorTrial, Subsample, and BucketColor.

ColorTrial is a derandomized version of a simple and frequently used coloring procedure: all nodes nominate themselves with some constant probability, and nominated nodes then pick a color from their palette. If no neighbors choose this same color, the node is successful and takes this color permanently. For our algorithm, the goal of ColorTrial is to provide permanent slack for nodes whose neighbors mostly have higher degree than their own.

Subsample is a derandomized version of sampling: nodes v *defer* themselves to S (to be colored later) with probability $d(v)^{-0.1}$. The purpose of this is to provide temporary slack to nodes whose neighbors mostly have similar degrees to their own. We will then recursively run the whole algorithm on S, and we will show that after $O(1)$ recursive calls the remaining graph will be of size $O(n)$, which can be trivially colored in CongestedClique in $O(1)$ rounds.

BucketColor is our main coloring procedure, and is designed to color all nodes for which ColorTrial and Subsample have generated sufficient slack, as well as all nodes whose neighbors mostly have lower degree than their own.

All these three algorithms begin with a randomized procedure, and use the method of conditional expectations on a family of $O(1)$-wise independent hash functions to derandomize it. Note that *this derandomization is an essential part of the algorithm* even if one is only concerned with probabilistic success guarantees. This is because in low-degree graphs, we cannot obtain the necessary properties with high probability, and some nodes will fail. The method of conditional expectations ensures that these graphs of failed nodes are of $O(n)$ size (and hence can collected onto a single network node in $O(1)$ rounds to color sequentially).

Using ColorTrial, Subsample, and BucketColor, we can present our main algorithm Color(G, x) (Algorithm 1) to color graph G. The algorithm assumes that $\Delta_G \leq O(\sqrt{n})$ and it uses a parameter x, $0 \leq x \leq 0.9$, that quantifies the size of the remaining graph over recursive calls (the algorithm starts with $x = 0$ and recursively increases by 0.1 until $x = 1$). In Section 6 (Lemma 19), we extend the analysis to arbitrary graphs, allowing arbitrary Δ_G.

Step 1 in Color(G, x) uses the fact that if G is of size $O(n)$, then in CongestedClique, the entire graph can be collected onto a single network node in $O(1)$ rounds and the coloring can be done locally. In the same way, since L_0 consists of vertices of constant degree, we can color them in step 7 in $O(1)$ rounds. Similarly, we will argue that the graph F (of failed nodes in ColorTrial) is of size $O(n)$, and hence it can be colored in step 7 in $O(1)$ rounds. The central part of our analysis will be to show that after a constant number of recursive calls the algorithm terminates with a correct solution to D1LC of G.

To prove the correctness of our algorithm, we show the following properties of Color(G, x):

Algorithm 1 COLOR(G, x): $\Delta_G \leq O(\sqrt{n})$; $0 \leq x \leq 0.9$; C is a sufficiently large constant.

1 If $|G| = O(n)$, then collect G in a single network node and solve the problem locally.
2 Set $L_0 := \{v \in G : p_G(v) < C\}$ and $G_0 := G \setminus L_0$.
3 $G_1, F \leftarrow$ COLORTRIAL(G_0).
4 $G_2, G' \leftarrow$ SUBSAMPLE(G_1, x).
5 BUCKETCOLOR(G_2).
6 COLOR$(G', x + 0.1)$.
7 Collect and solve L_0 and then F at a single node.

1. COLORTRIAL, SUBSAMPLE, and BUCKETCOLOR run deterministically in $O(1)$ rounds.
2. The size of F is $O(n)$.
3. Each node in G_2 has sufficient slack to be colored by BUCKETCOLOR. For each node v of G_2, either $p_{G_2}(v) \geq d_{G_2}(v) + \frac{1}{4} d_{G_2}(v)^{0.9}$, or $|N^-_{G_2}(v)| \geq \frac{1}{3} d_{G_2}(v)$.
4. The size of the (remaining) graph reduces over recursive calls in the following sense:

$$\sum_{v \in G'} d_{G_1}(v)^{x+0.1} \leq Cn + 2 \sum_{v \in G_1} d_{G_1}(v)^x \ . \tag{1}$$

   Observe that when $x = 0.9$, expression (1) bounds the number of edges of G'. In particular, we show that the total size of the remaining graph is $O(n)$ after 10 recursive calls.

In Section 4, we describe the procedures COLORTRIAL and SUBSAMPLE. Also, we prove the desired properties of F, G_2, and G' in Section 4. In Section 5, we describe the procedure BUCKETCOLOR. Finally, we prove our main theorem (Theorem 1) in Section 6.

For simplicity of the presentation, in the pseudocode of our algorithms in the following sections, we will only present the randomized bases of each procedure. In each case, the full deterministic procedure comes from applying the method of conditional expectations to the randomized bases, with some specific cost function we will make clear in the analysis.

4 ColorTrial and Subsample

We describe procedure COLORTRIAL and SUBSAMPLE in Section 4.1 and Section 4.2, respectively, along with some of their crucial useful properties. In particular, we show that $|F| = O(n)$ in Lemma 9 of Section 4.1 and show that graph G' has the desired property in Lemma 11 of Section 4.2. Finally, we give a lemma capturing the desired property of graph G_2 (which is the input to BUCKETCOLOR).

4.1 ColorTrial

We first note that, because nodes with palette size less than C are removed immediately prior to COLORTRIAL(G_0) in COLOR(G, x), we may assume that all nodes v in G_0 have $p_{G_0}(v) \geq C$. The randomized procedure on which COLORTRIAL is based is Algorithm 2. COLORTRIAL has two major steps: nomination step (line 1) and coloring step (line 3). The coloring of a node can be deferred if it is a *failed* node either in nomination step and coloring step. Note that the notions of failed nodes are different in nomination step and coloring step, and we will define both the notions in the following part of this section.

We define some notions that will be useful to define failed nodes in both the nomination step and the coloring step of COLORTRIAL.

Algorithm 2 ColorTrial(G_0) – Randomized Basis.

1 Each node v in G_0 independently self-nominates with probability $\frac{1}{4}$.
2 Each node v decides if it is successful or failed in the nomination step.
3 For each self-nominated node v (that is successful in the nomination step):
- v chooses a random palette color $c(v) \in \Phi_{G_0}(v)$;
- v colors itself with color $c(v)$ if no neighbor u of v choose $c(u) = c(v)$;
- v decides if it is successful or failed in the coloring step.

Return
- G_1, the induced graph of remaining (non-failed) uncolored nodes, with updated palettes,
- F, the induced graph of failed nodes (either in nomination step or in the coloring step), with updated palettes.

▶ **Definition 2.** *$N^*(v) \subseteq N_{G_0}(v)$ is defined as the subset of neighbors u of v that have $d_{G_0}(u) \geq 3d_{G_0}(v)$. $\mathrm{Nom}_v \subseteq N_{G_0}(v)$ is defined as the subset of v's neighbors that self-nominate, and $\mathrm{Nom}_v^* := \mathrm{Nom}_v \cap N^*(v)$.*

Next, we define the notion of failed nodes in the nomination step.

▶ **Definition 3.** *A node v is* successful *during the nomination step of ColorTrial if both of the following hold (if either condition does not hold, node v* fails*):*
- $|\mathrm{Nom}_v| \leq \frac{1}{4}d_{G_0}(v) + p_{G_0}(v)^{0.7}$;
- $|\mathrm{Nom}_v^*| \geq \frac{1}{4}|N^*(v)| - p_{G_0}(v)^{0.7}$.

To derandomize ColorTrial, we replace each of the random choices of lines 1 and 3 (the nomination step and the coloring step respectively) with choices determined by a random hash function from a $O(1)$-wise independent family $[n^{O(1)}] \to [n^{O(1)}]$. We show that, under such a choice of hash function, the subgraph induced by the failed nodes in the nomination step is of size $O(n)$ in expectation (Lemma 4). We are then able to derandomize this selection using the method of conditional expectations to obtain Lemma 5.

▶ **Lemma 4.** *When nomination choices of ColorTrial are determined by a random hash function from a $O(1)$-wise independent hash family $[n^{O(1)}] \to [n^{O(1)}]$, any node v fails in the nomination step of ColorTrial with probability at most $1/p_{G_0}(v)$.*

▶ **Lemma 5.** *We can deterministically choose a hash function in $O(1)$ rounds, from a $O(1)$-wise independent family $[n^{O(1)}] \to [n^{O(1)}]$, to run the nomination step of ColorTrial such that the size of the subgraph induced by the failed nodes (in the nomination step) is $O(n)$.*

Besides the nomination step, a node can also fail in the coloring step of ColorTrial. Now we formally define what it means for a node to fail in the coloring step.

▶ **Definition 6.** *A node v is* successful *during the coloring step of ColorTrial if any of the following hold (if none hold, node v* fails*):*
- $p_{G_0}(v) \geq 1.1d_{G_0}(v)$;
- $|N^*(v)| < \frac{1}{3}d_{G_0}(v)$;
- *at least* $0.03d_{G_0}(v)$ *of v's neighbors failed in the nomination step;*
- *at least* $0.01p_{G_0}(v)$ *of v's neighbors successfully color themselves a color not in v's palette.*

Notice that the first three properties are already determined by the nomination step. Here, we need to handle the fourth property. Similar to our analysis for the nomination step, we are able to show that choosing a hash function uniformly at random from a $O(1)$-wise independent family to make decisions in the coloring step yields a subgraph of failed nodes of size $O(n)$ in expectation (Lemma 7). We can then derandomize this result using the method of conditional expectations, achieving Lemma 8.

▶ **Lemma 7.** *When color choices in the coloring step of* ColorTrial *are determined by a random hash function from a $O(1)$-wise independent hash family $[n^{O(1)}] \to [n^{O(1)}]$, any node v that did not fail in the nomination step fails in the coloring step of* ColorTrial *with probability at most $1/p_{G_0}(v)$.*

▶ **Lemma 8.** *We can deterministically choose a hash function in $O(1)$ rounds, from a $O(1)$-wise independent family $[n^{O(1)}] \to [n^{O(1)}]$, to run the coloring step of* ColorTrial *such that the size of the subgraph induced by the failed nodes (in the coloring step) is at most n.*

Note that Lemma 8 is the only lemma whose prove requires the assumption $\Delta_G = O(\sqrt{n})$.

Recall that F denotes the subgraph of G induced by the nodes that is either failed in the nomination step or in the coloring step of ColorTrial. The following lemma bounds $|F|$, and follows immediately from Lemma 5 and Lemma 8.

▶ **Lemma 9.** *We can deterministically choose hash functions in $O(1)$ rounds, from a $O(1)$-wise independent hash family $[n^{O(1)}] \to [n^{O(1)}]$, to run each step of* ColorTrial *such that the size of the subgraph induced by the failed nodes is at most $O(n)$, i.e., $|F| = O(n)$.*

4.2 Subsample

After executing ColorTrial(G_0), Color(G, x) executes procedure Subsample(G_1, x). The randomized procedure on which Subsample is based is Algorithm 3. To derandomize Subsample, we replace the random choice of line 1 (to generate a set S of vertices) with a choice determined by a hash function from a $O(1)$-wise independent family $[n^{O(1)}] \to [n^{O(1)}]$.

Algorithm 3 Subsample(G_1, x) – Randomized Basis.

1 Each node v in G_1 independently joins S with probability $d_{G_1}(v)^{-0.1}$
2 Each node v decides whether it succeeds or fails. Let F_1 be the set of failed nodes.
3 Let L denote the nodes with $p_{G_1}(v) < C$. Return:
- G_2, consisting of $G_1 \setminus (F_1 \cup S \cup L)$
- $G' = (S \cup F_1 \cup L)$

Note that while x is not used explicitly in Algorithm 3, it increases by 0.1 in each recursive call to Color, and this plays a significant role in the analysis (see Lemma 11). Our aim is to show that after 10 levels of recursion of Color$(G, 0)$, the remaining graph is of size $O(n)$.

We start by defining the notion of failed nodes in Subsample:

▶ **Definition 10.** *Let us define $N^{\approx}(v) \subseteq N_{G_1}(v)$ to be the subset of v's neighbors u with $\frac{1}{2}d_{G_1}(v) \leq d_{G_1}(u) \leq 6d_{G_1}(v)$. A node v is classed as* successful *during* Subsample *if either*
- *$p_{G_1}(v) \geq 1.1d_{G_1}(v)$; or*
- *$|N^{\approx}(v)| \leq \frac{1}{3}d_{G_1}(v)$; or*
- *at least $\frac{1}{4}p_{G_1}(v)^{0.9}$ of v's neighbors join S.*

v is classed as failed *if none of the above three conditions hold.*

In a similar way to the analysis in Section 4.1, we can express the analysis of SubSample in terms of bounded-independence hash functions and derandomize it, obtaining the following:

▶ **Lemma 11.** *Let x be such that $0 \leq x \leq 0.9$. We can deterministically choose a hash function in $O(1)$ rounds, from a $O(1)$-wise indepedenet hash family, to execute line 1 of SubSample to generate set S such that the following holds:*

$$\sum_{v \in G'} d_{G_1}(v)^{x+0.1} \leq Cn + 2 \sum_{v \in G_1} d_{G_1}(v)^x.$$

We end with a lemma which explains what properties the graph G_2 has. Recall that G_2 is the graph of successful nodes that results from running ColorTrial and SubSample on our input graph, and it is the input graph to our main coloring procedure BucketColor in Section 5. Here, we show that each node in G_2 has sufficient slack to be colored in $O(1)$ rounds by BucketColor.

▶ **Lemma 12.** *For any $v \in G_2$, either $p_{G_2}(v) \geq d_{G_2}(v) + \frac{1}{4} d_{G_2}(v)^{0.9}$, or $|N^-_{G_2}(v)| \geq \frac{1}{3} d_{G_2}(v)$.*

5 BucketColor

In this section, we describe our core coloring procedure BucketColor(G_2). Note that, each node in the input graph G_2 to BucketColor has sufficient slack as mentioned in Lemma 12. Throughout this section, the graph under consideration is always G_2, so we omit the subscript G_2 from $N_{G_2}(v), d(v), N^+_{G_2}(v), d^+_{G_2}(v), N^-_{G_2}(v), d^-_{G_2}(v), \Psi(v)$, $p(v)$ and Δ_G.

In Section 5.1, we first formalize the bucket structure of nodes (as discussed in Section 1.2), and then introduce some useful definitions. Then we describe algorithm BucketColor in Section 5.1. In Sections 5.2, we analyze the correctness of BucketColor.

5.1 Assigning nodes to buckets

We use two special functions in the description of our algorithm in this section: $l : V(G_2) \to \mathbb{N}_{\geq 0}$ and $b : \mathbb{N}_{\geq 0} \to \mathbb{N}_{\geq 0}$. l is defined as $l(v) := \max\{\lfloor \log_{1.1} \log_2 d(v) \rfloor, 0\}$ for node v, and b is defined as $b(i) := \lfloor 0.7 \cdot 1.1^i \rfloor$ for $i \in \mathbb{N}_{\geq 0}$. If $d(v)$ is at least a suitable constant, then $b(l(v)) = \Theta(\log d(v))$ and $b(l(v)) \leq 0.7 \log_2 d(v)$.

We consider a partition of the nodes of G_2 into $O(\log \log \Delta)$ levels, with the *level* of a node v equal to $l(v) = \max\{\lfloor \log_{1.1} \log_2 d(v) \rfloor, 0\}$. The nodes of a particular level will be further partitioned into buckets. The *level of a bucket x* is the level of a possible node that can be put into this bucket, and is denoted by $level(x)$. The buckets of *level i* (or *level-i* buckets) are identified by binary strings of length $b(i)$, where $i \in \mathbb{N}_{\geq 0}$, as well as their level. [1] So, there are $2^{b(i)}$ *level-i* buckets. To put a node v to a bucket, (in our algorithm) we generate a random binary string of length $b(l(v))$.

The set of buckets forms a hierarchical tree structure as described below. We say that a bucket a' is a *child* of a (and a is the *parent* of a') if $level(a') = level(a) + 1$ and $a \sqsubseteq a'$. We say that a' is a *descendant* of a (and a is an *ancestor* of a') if $level(a') \geq level(a)$ and

[1] Note that, at low levels, buckets in different levels can be identified by the same string, because the function $b(i) = \lfloor 0.7 \cdot 1.1^i \rfloor$ is not injective for $i \leq 24$. Therefore, for example, $b(0) = b(1) = 0$, and so levels 0 and 1 both in fact contain a single bucket specified by the empty string. We treat these as different buckets in order to conform to a standard rooted tree structure, and therefore must identify buckets by their level as well as their specifying string.

$a \sqsubseteq a'$ (note that by definition a is a descendant and ancestor of itself). The buckets form a rooted tree structure: the root is the single level 0 bucket, specified by the empty string; each bucket in level $i > 0$ has one parent in level $i-1$ and multiple children in level $i+1$.

We also put colors into the buckets. For any color c, we put c into a bucket of level $\lfloor \log_{1.1} \log_2 \Delta \rfloor$ by generating a random binary string of length $b(\lfloor \log_{1.1} \log_2 \Delta \rfloor + 20)$. Consider a bucket a and a color c which is put in a. We say c is assigned to bucket a' (and that bucket a' contains c) iff $a' \sqsubseteq a$. Note that a' is a leaf, since the string generated for c is of maximum length; note also that c is assigned to all buckets on the path from a to the root bucket.

Our algorithm uses a hash function to generate the binary strings (and hence the buckets) for the colors and nodes. Based on the partition of the nodes and colors into buckets, it is sufficient to color a set of reduced instances (one per bucket) of the original D1LC instance. The following definition formalizes the effective palettes and neighborhoods of a node under any function mapping nodes and colors to strings.

▶ **Definition 13.** *Let $h : (\mathcal{C} \cup V(G_2)) \to \{0,1\}^*$ be a function mapping colors and nodes to binary strings. For each node $v \in G_2$, define:*

- *the graph $G^+_{h(v)}$ to contain all edges $\{u, w\} \in G_2$ with $d(u) \leq d(w)$ for which $h(u) = h(v)$ and $h(w) \sqsupseteq h(v)$, and all nodes which are endpoints of such edges;*
- $\Psi_{h(v)}(v) = \{c \in \Psi(v) : h(c) \sqsupseteq h(v)\}$ *{$\Psi_{h(v)}(v)$ is the set of palette colors v has in $h(v)$};*
- $N^+_{h(v)}(v) := \{w \in N^+(v) : h(w) \sqsupseteq h(v)\}$ *{$d^+_{h(v)}(v)$ is the number of neighbors u that v has in descendants of $h(v)$ with $d(v) \leq d(u)$};*
- $p_{h(v)}(v) = \left|\Psi_{h(v)}(v)\right|$ *and* $d^+_{h(v)}(v) = \left|N^+_{h(v)}(v)\right|$.

Observe that there is a reduced instance for each bucket. Notice that each node u is present in only one reduced instance, i.e., in $G^+_{h(u)}$ (the reduced instance corresponding to the bucket where u is present); and each edge $\{u, w\}$ with $d_{G_2}(u) \leq d_{G_2}(w)$ is present in at most one reduced instance, i.e., possibly in $G^+_{h(u)}$ only when $h(u) \sqsubseteq h(w)$ (i.e., w is present in some descendent bucket of u). Consider a node u in the reduced instance G^+_x (i.e., $h(u) = x$). $N^+_x(u)$ and $d^+_x(u)$ denote the set of neighbors and the degree of u in G^+_x, respectively. Moreover, for coloring the reduced instance G^+_x, let $\Psi_x(u)$ and $p_x(u)$ denote the color palette and the size of the the color palette of u, respectively.

Observe that the reduced G^+_x instances are not independent, and they can be of size $\omega(n)$. Also, it may be the case that G^+_x may not be a valid D1LC instance. To handle the issue, we define the notion of *bad nodes* in Definition 14. Intuitively, bad nodes are those who do not behave as expected when mapped to their bucket (e.g. have too many neighbors or too few colors therein), and we will show that the subgraphs of buckets restricted to good nodes are of size $O(n)$ and can be colored in $O(1)$ rounds. If we choose our hash function uniformly at random from a $O(1)$-wise independent family of hash functions, the subgraph G_{bad} (induced by the bad nodes) has size $O(n)$ in expectation. We also show that it is possible to choose a of hash function deterministically in $O(1)$ rounds such that the size of G_{bad} is $O(n)$.

▶ **Definition 14.** *Given a hash function $h : (\mathcal{C} \cup V(G_2)) \to \{0,1\}^*$ mapping colors and nodes to binary strings, define a node v to be* bad *if any of the following occur:*

1. $d^+_{h(v)}(v) \geq d^+(v) 2^{-b(l(v))} + \frac{1}{8} d(v)^{0.9} 2^{-b(l(v))}$;
2. $p_{h(v)}(v) \leq p(v) 2^{-b(l(v))} - \frac{1}{8} d(v)^{0.9} 2^{-b(l(v))}$;
3. *any of v's level $l(v) + 20$ descendant buckets contain more than one of v's palette colors;*
4. *more than $2n2^{-b(l(v))}$ nodes v' have $h(v) = h(v')$.*

(1) and (2) ensure that each reduced instance (after removing the bad nodes) are valid D1LC instances; (3) ensure that the dependencies among the reduced instances are limited; and (4) when combined with (1) ensures that the subgraph induced by bad nodes is $O(n)$.

Now we are ready to discuss our algorithm BucketColor. The randomized procedure on which BucketColor is based is Algorithm 4. Note that only line 1 of Algorithm 4 is a randomized step, and it can be derandomized by replacing its random choices with choices determined by a hash function from a $O(1)$-wise independent family $[n^{O(1)}] \to [n^{O(1)}]$. The subgraph induced by the bad nodes, G_{bad}, is deferred to be colored later. Then in Lines 3 to 11, BucketColor colors the (good) nodes in $G_2 \setminus G_{bad}$ in $O(1)$ rounds deterministically.

Algorithm 4 BucketColor(G_2) – Randomized Basis.

1 Each node v uniformly randomly chooses a $b(l(v))$-bit binary string $h(v)$, and each color is uniformly randomly assigned a $b(\lfloor \log_{1.1} \log_2 \Delta \rfloor + 20)$-bit binary string $h(c)$.
2 Each node v decides whether it is bad or good. Let G_{bad} be the subgraph induced by the bad nodes.
3 **for** $O(1)$ *iterations* **do**
4 Each node $v \in G_2 \setminus G_{bad}$ restricts its palettes to colors c with $h(v) \sqsubseteq h(c)$, i.e., $\Psi_{h(v)}(v) = \{c \in \Psi(v) : h(c) \sqsupset h(v)\}$ is the current palette of v.
5 **for** *each* $i \in [\lfloor \log_{1.1} \log_2 \Delta \rfloor + 20]$ *and each string* $x \in \{0,1\}^{b(i)}$ **do**
6 collect the graph G_x^+ to a dedicated network node $node_x$.
7 **end**
8 **for** *each node* $v \in G_2 \setminus G_{bad}$ *in a bucket* $h(v)$*, in non-increasing order of degree, performed on* $node_{h(v)}$ **do**
9 $h(v) \leftarrow h^*$, where $h^* \sqsupseteq h(v)$ is a child bucket of $h(v)$ with $d^+_{h^*}(v) < p_{h^*}(v)$.
10 **end**
11 **end**
12 Color each node $v \in G_2 \setminus G_{bad}$ with the only palette color in its current bucket.
13 Update the palettes of G_{bad}, collect to a single node, and color sequentially.

Overview of coloring good nodes. To color the (good) nodes in $G_2 \setminus G_{bad}$, we proceed in non-increasing order of node degree and start with the hash function h chosen in Line 1 of BucketColor. Recall that $h(v)$ denotes the bucket in which node v is present. The algorithm goes over iterations and the bucket status of the nodes change over iterations.

In every iteration, for every node v, we restrict the color palettes of v to the colors present in the descendant bucket of (current) $h(v)$. Also, for every binary string x such that the bucket x has at least one node, we gather the graph having set of edges with one endpoint in bucket x and the other endpoint in some descendent bucket of x, i.e., G_x^+ (of size $O(n)$) at a network node in $O(1)$ rounds. Though all G_x^+'s are a valid D1LC instance in any iteration, G_x^+'s are not necessarily independent: there can be an edge between a node v in G_x^+ and a node w outside G_x^+ such that the current palettes of v intersects with the current palette of w. We can show that every node v satisfies $d^+_{h(v)}(v) < p_{h(v)}(v)$ in the first iteration – i.e., each node has enough colors in its bucket to be greedily colored in the non-increasing order of degree. That is, (in first iteration) each graph G_x^+ is a valid D1LC instance.

In each iteration, we move each node down to a child bucket h^* of its current bucket $h(v)$, in such a way that we maintain this colorability property (having more colors in the palette than the degree). This will imply that, when we find graphs G_x^+ in the next iteration, those are also valid D1LC instances. We will show that after $O(1)$ iterations, each node has

only 1 palette color in its bucket (and therefore zero higher-degree neighbors in descendant buckets, since $d^+_{h^*}(v) < p_{h^*}(v)$). At this point, nodes can safely color themselves the single palette color in their bucket. To decide on child buckets for the nodes in any iteration, it is essential that each G^+_x will always fit onto a single network node (which is in fact the case).

5.2 Correctness of BucketColor

To prove the correctness of BucketColor formally, we give Lemma 15 and Lemma 16, which jointly imply Lemma 17.

▶ **Lemma 15.** *All network nodes can simultaneously choose a hash function (in line 1 of BucketColor) such that the size of G_{bad} is $O(n)$.*

▶ **Lemma 16.** *After 20 iterations of the outer for-loop of BucketColor, all nodes in $G_2 \setminus G_{bad}$ can be colored without conflicts.*

▶ **Lemma 17.** *BucketColor successfully colors graph G_2 in $O(1)$ rounds.*

6 Proof of the main theorem

Now, we are ready to complete our analysis of a constant-round CongestedClique and prove Theorem 1. We begin with a theorem summarizing the properties of Color$(G, 0)$.

▶ **Theorem 18.** *Color$(G, 0)$ colors any* D1LC *instance G with $\Delta_G \leq O(\sqrt{n})$ in $O(1)$ rounds.*

Proof. From the description of Color$(G, 0)$ and its subroutines, it is evident that Color$(G, 0)$ colors a graph G successfully when $\Delta_G \leq O(\sqrt{n})$. It remains to analyze the total number of rounds spent by Color$(G, 0)$.

Note that the steps of Color$(G, 0)$, other than the call to subroutines ColorTrial, Subsample, BucketColor and recursive call, can be executed in $O(1)$ rounds. ColorTrial and Subsample can be executed in $O(1)$ rounds by Lemma 9 and Lemma 11, respectively. Also, $O(1)$ rounds are sufficient for BucketColor due to Lemma 17.

To analyze the round complexity of recursive calls in Color$(G, 0)$, let G^i denote the graph on which the i^{th}-level recursive call of Color, i.e., Color$(G^i, 0.1i)$ is made. Color$(G^i, 0.1i)$ does $O(1)$ rounds of operations and makes a recursive call Color$(G^{i+1}, 0.1(i+1))$.

We show by induction that $\sum_{v \in G_i} d_{G_i}(v)^{0.1i} \leq 3^i Cn$ for $i \leq 10$. This is true for $G^0 = G$, since $\sum_{v \in G} d_G(v)^0 = n$. For the inductive step, for $1 \leq i \leq 9$, by Lemma 11 using $x = 0.1i$,

$$\sum_{v \in G^{i+1}} d_{G^{i+1}}(v)^{0.1(i+1)} \leq \sum_{v \in G^{i+1}} d_{G^i}(v)^{0.1(i+1)} \leq Cn + 2\sum_{v \in G^i} d_{G^i}(v)^{0.1i} \leq Cn + 2 \cdot 3^i Cn \leq 3^{i+1} Cn \ .$$

So, $|E(G^{10})| \leq \sum_{v \in G^{10}} d_{G^{10}} \leq 3^{10} Cn = O(n)$. Therefore, after 10 recursive calls, the remaining uncolored graph can simply be collected to a single network node and solved. ◀

While Theorem 18 requires that $\Delta_G \leq O(\sqrt{n})$, we note that we can generalize this result to any maximum degree:

▶ **Lemma 19.** *In $O(1)$ rounds of* CongestedClique*, we can recursively partition an input* D1LC *instance into sub-instances, such that each sub-instance has maximum degree $O(\sqrt{n})$. The sub-instances can be grouped into $O(1)$ groups where each group can be colored in parallel.*

Proof. We use the LowSpacePartition procedure from [14], which reduces a coloring instance to $O(1)$ sequential instances of maximum degree n^ε for any constant $\varepsilon > 0$. The procedure is for $\Delta + 1$-coloring, but it extends immediately to D1LC, as discussed in Section 5 of [11]. Since we can simulate low-space MPC in CongestedClique, we can execute LowSpacePartition, setting ε appropriately to reduce the maximum degree of all instances to $O(\sqrt{n})$. By subsequent arguments in [14], $O(1)$ sequential sets of base cases are created. ◀

Now the proof of Theorem 1 follows immediately from Theorem 18 and Lemma 19. ◀

References

1. Noga Alon and Sepehr Assadi. Palette sparsification beyond $(\Delta + 1)$ vertex coloring. In *Proceedings of the 24th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 6:1–6:22, 2020.
2. Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4):567–583, 1986.
3. Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear algorithms for $(\Delta+1)$ vertex coloring. In *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 767–786, 2019.
4. Étienne Bamas and Louis Esperet. Distributed coloring of graphs with an optimal number of colors. In *Proceedings of the 36th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 10:1–10:15, 2019.
5. Philipp Bamberger, Fabian Kuhn, and Yannic Maus. Efficient deterministic distributed coloring with small bandwidth. In *Proceedings of the 39th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 243–252, 2020.
6. Leonid Barenboim and Michael Elkin. *Distributed Graph Coloring: Fundamentals and Recent Developments*. Morgan & Claypool Publishers, 2013.
7. Soheil Behnezhad, Mahsa Derakhshan, and MohammadTaghi Hajiaghayi. Brief announcement: Semi-MapReduce meets Congested Clique. *Preprint arXiv*, 2018. `arXiv:1802.10297`.
8. Keren Censor-Hillel, Merav Parter, and Gregory Schwartzman. Derandomizing local distributed algorithms under bandwidth restrictions. *Distributed Computing*, 33(3):349–366, 2020.
9. Yi-Jun Chang, Manuela Fischer, Mohsen Ghaffari, Jara Uitto, and Yufan Zheng. The complexity of $(\Delta + 1)$ coloring in Congested Clique, Massively Parallel Computation, and centralized local computation. In *Proceedings of the 38th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 471–480, 2019.
10. Yi-Jun Chang, Wenzheng Li, and Seth Pettie. Distributed $(\Delta + 1)$-coloring via ultrafast graph shattering. *SIAM Journal on Computing*, 49(3):497–539, 2020.
11. Sam Coy, Artur Czumaj, Peter Davies, and Gopinath Mishra. Fast parallel degree+1 list coloring. *Preprint arXiv*, 2023. `arXiv:2302.04378`.
12. Artur Czumaj, Peter Davies, and Merav Parter. Component stability in low-space massively parallel computation. In *Proceedings of the 40th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 481–491, 2021.
13. Artur Czumaj, Peter Davies, and Merav Parter. Graph sparsification for derandomizing massively parallel computation with low space. *ACM Transactions on Algorithms*, 17(2), May 2021.
14. Artur Czumaj, Peter Davies, and Merav Parter. Improved deterministic $(\Delta + 1)$ coloring in low-space MPC. In *Proceedings of the 40th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 469–479, 2021.
15. Artur Czumaj, Peter Davies, and Merav Parter. Simple, deterministic, constant-round coloring in Congested Clique and MPC. *SIAM Journal on Computing*, 50(5):1603–1626, 2021.

16 Janosch Deurer, Fabian Kuhn, and Yannic Maus. Deterministic distributed dominating set approximation in the CONGEST model. In *Proceedings of the 38th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 94–103, 2019.

17 Paul Erdös and John L. Selfridge. On a combinatorial game. *Journal of Combinatorial Theory, Series A*, 14(3):298–301, 1973.

18 Manuela Fischer, Jeff Giliberti, and Christoph Grunau. Improved deterministic connectivity in massively parallel computation. In *Proceedings of the 36th International Symposium on Distributed Computing (DISC)*, pages 22:1–22:17, 2022.

19 Manuela Fischer, Magnús M. Halldórsson, and Yannic Maus. Fast distributed Brooks' theorem. In *Proceedings of the 34th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2567–2588, 2023.

20 Pierre Fraigniaud, Marc Heinrich, and Adrian Kosowski. Local conflict coloring. In *Proceedings of the 57th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 625–634, 2016.

21 Mohsen Ghaffari and Fabian Kuhn. Derandomizing distributed algorithms with small messages: Spanners and dominating set. In *Proceedings of the 32nd International Symposium on Distributed Computing (DISC)*, pages 29:1–29:17, 2018.

22 Mohsen Ghaffari and Fabian Kuhn. Deterministic distributed vertex coloring: Simpler, faster, and without network decomposition. In *Proceedings of the 62nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1009–1020, 2021.

23 Magnús M. Halldórsson, Fabian Kuhn, Yannic Maus, and Tigran Tonoyan. Efficient randomized distributed coloring in CONGEST. In *Proceedings of the 53rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 1180–1193, 2021.

24 Magnús M. Halldórsson, Fabian Kuhn, Alexandre Nolin, and Tigran Tonoyan. Near-optimal distributed degree+1 coloring. In *Proceedings of the 54th Annual ACM Symposium on Theory of Computing (STOC)*, pages 450–463, 2022.

25 Magnús M. Halldórsson, Alexandre Nolin, and Tigran Tonoyan. Overcoming congestion in distributed coloring. In *Proceedings of the 41st ACM Symposium on Principles of Distributed Computing (PODC)*, pages 26–36, 2022.

26 James W. Hegeman and Sriram V. Pemmaraju. Lessons from the Congested Clique applied to MapReduce. *Theoretical Computer Science*, 608:268–281, 2015.

27 Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for MapReduce. In *Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 938–948, 2010.

28 Fabian Kuhn. Faster deterministic distributed coloring through recursive list coloring. In *Proceedings of the 31st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1244–1259, 2020.

29 Christoph Lenzen. Optimal deterministic routing and sorting on the Congested Clique. In *Proceedings of the 32nd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 42–50, 2013.

30 Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, February 1992.

31 Zvi Lotker, Boaz Patt-Shamir, Elan Pavlov, and David Peleg. Minimum-weight spanning tree construction in $O(\log \log n)$ communication rounds. *SIAM Journal on Computing*, 35(1):120–131, 2005.

32 Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15(4):1036–1053, 1986.

33 Michael Luby. Removing randomness in parallel computation without a processor penalty. *Journal of Computer and System Sciences*, 47(2):250–286, 1993.

34 Rajeev Motwani, Joseph Naor, and Moni Naor. The probabilistic method yields deterministic parallel algorithms. *Journal of Computer and System Sciences*, 49(3):478–516, 1994.

35 Moni Naor. A lower bound on probabilistic algorithms for distributive ring coloring. *SIAM Journal on Discrete Mathematics*, 4(3):409–412, 1991.

36 Merav Parter. $(\Delta+1)$ coloring in the Congested Clique model. In *Proceedings of the 45th Annual International Colloquium on Automata, Languages and Programming (ICALP)*, pages 160:1–160:14, 2018.

37 Merav Parter and Hsin-Hao Su. Randomized $(\Delta+1)$-coloring in $O(\log^* \Delta)$ Congested Clique rounds. In *Proceedings of the 32nd International Symposium on Distributed Computing (DISC)*, pages 39:1–39:18, 2018.

38 David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, PA, 2000.

39 Prabhakar Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *Journal of Computer and System Sciences*, 37(2):130–143, 1988.

40 Václav Rozhoň and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *Proceedings of the 52nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 350–363, 2020.

A Derandomization in CongestedClique

In this section we first give some useful lemmas regarding $O(1)$-wise independence and the existence of small families of $O(1)$-wise independent hash functions, and then we give a formal description of the method of conditional expectations and how it is implemented in the CongestedClique model.

A.1 Bounded Independence

Our algorithm will be finding a hash function of sufficient quality from a family of $O(1)$-independent hash functions. In the following, we recall the standard notions of *k-wise independent hash functions* and *k-wise independent random variables*. Then, we recall that we can construct small families of bounded-independence hash functions, and that each hash function in this family can be specified by a short seed.

▶ **Definition 20.** *Let $k \geq 2$ be an integer. A set $\{X_1, \ldots, X_n\}$ of n random variables taking values in S are said to be* k-wise independent *if for any $I \subset [n]$ with $|I| \leq k$ and any $x_i \in S$ for $i \in I$, we have*

$$\mathbf{Pr}\left[\bigwedge_{i\in[k]} X_i = x_i\right] = \prod_{i=1}^{k} \mathbf{Pr}\left[X_i = x_i\right] \quad .$$

▶ **Definition 21.** *A family of hash functions $\mathcal{H} : \{h : X \to Y\}$ is said to be k-wise independent if $\{h(x) : x \in X\}$ are k-wise independent when h is drawn uniformly at random from $\mathcal{H}$.*

We use the property that small families of $O(1)$-wise independent hash functions can be constructed, and each hash function in such a family can be specified with a small number of bits:

▶ Remark 22. For all positive integers c_1, c_2, there is a family of k-wise independent hash functions $\mathcal{H} = \{h : [n^{c_1}] \to [n^{c_2}]\}$ such that each function from $\mathcal{H}$ can be specified using $O(k \log n)$ bits.

A.2 The Method of Conditional Expectations

We now describe in more detail the method of conditional expectations and its implementation in CongestedClique. We briefly recall the setup to the problem: we have a randomized algorithm which "succeeds" if a "bad" outcome occurs for less than some number T of

nodes. This algorithm succeeds in expectation using bounded-independence randomness. We would like to derandomize this algorithm. In order to achieve this, given a family of $O(1)$-independent hash functions $\mathcal{H}$, we need to find a "good" hash function $h^* \in \mathcal{H}$ which solves our problem, when used to make decisions for nodes instead of randomness.

First, we define some cost function $f : \mathcal{H} \times V \to \{0, 1\}$ such that $f(h, v) = 1$ if the node v has a "bad" outcome when h is the selected hash function, and $f(h, v) = 0$ if the outcome is "good". We further define $F(h) = \sum_{v \in V} f(h, v)$ as the total cost of the hash function h: i.e., the number of bad nodes when h is the selected hash function. Finally, we use $\mathbb{E}_{h \in \mathcal{H}} x(h)$ to denote the expected value of some function $x(h)$ when h is drawn uniformly at random from $\mathcal{H}$.

To successfully derandomize our algorithm, we need to find a hash function $h^* \in \mathcal{H}$ such that $F(h^*) \leq T$. We need the following conditions to hold for our derandomization to work:

- $\mathbb{E}_{h \in \mathcal{H}}[F(h)] \leq T$ (i.e., the expected cost of a hash function selected uniformly at random from $\mathcal{H}$ is at most T); and
- Node v can locally (i.e., without communication) evaluate $f(h, v)$ for all $h \in \mathcal{H}$.

We can now use the method of conditional expectations to find a $h^* \in \mathcal{H}$ for which $F(h^*) \leq T$. We first recall that each hash function in our family of $O(1)$-wise independent hash functions $\mathcal{H}$ can be specified using $O(\log n)$ bits, by Remark 22. Next, let $\Pi = \{0, 1\}^{\log n}$ be the set of binary strings of length $\log n$, and for each $\pi \in \Pi$, let $\mathcal{H}_\pi$ denote the hash functions in $\mathcal{H}$ whose seeds begin with the prefix π.

Our goal is to find some seed-prefix $\pi \in \Pi$ for which $\mathbb{E}_{h \in \mathcal{H}_\pi}[F(h)] \leq T$: the existence of such a prefix is guaranteed by the probabilistic method. Since each node v can locally evaluate $f(h, v)$ for all $h \in \mathcal{H}$, nodes can also compute $\mathbb{E}_{h \in \mathcal{H}_\pi}[f(h, v)]$ for all $\pi \in \Pi$. Since $|\Pi| = n$, each node v can be made responsible for a prefix $\pi_v \in \Pi$. Node v can then collect the value of $\mathbb{E}_{h \in \mathcal{H}_{\pi_v}}[f(h, u)]$ for each $u \in V \setminus \{v\}$: since this requires all nodes sending and receiving $O(n)$ messages it can be done in $O(1)$ rounds using Lenzen's routing algorithm [29]. Now, by linearity of expectation:

$$\sum_{v \in V} \left(\mathbb{E}_{h \in \mathcal{H}_{\pi_v}} [f(h, v)] \right) = \mathbb{E}_{h \in \mathcal{H}_{\pi_v}} [F(h)] \quad .$$

Therefore v can compute the expected value of F for the sub-family of hash functions which are prefixed with π_v. Nodes can broadcast this expected value to all other nodes in $O(1)$ rounds, again using Lenzen's routing algorithm [29]. All nodes then know the expected value of F for all $(\log n)$-bit prefixes and can, without communication (breaking ties in a predetermined and arbitrary way), pick the prefix with the lowest expected value of F. Recall that this prefix is guaranteed to have an expected value of at most T by the probabilistic method.

We have now fixed the first $(\log n)$ bits of the prefix and obtained a smaller set $\mathcal{H}_1 \subset \mathcal{H}$ of hash functions. We can then perform the same procedure described above on $\mathcal{H}_1$ to set the next $(\log n)$ bits of the seed, obtaining a smaller set $\mathcal{H}_2 \subset \mathcal{H}_1 \subset \mathcal{H}$ of hash functions. After repeating this procedure $O(1)$ times we will have fixed the entire seed, since we fix $(\log n)$ bits each time and the seeds of hash functions in $\mathcal{H}$ were $O(\log n)$ bits in length.

Incremental Maximization via Continuization

Yann Disser
TU Darmstadt, Germany

Max Klimm
TU Berlin, Germany

Kevin Schewior
University of Southern Denmark, Odense, Denmark

David Weckbecker
TU Darmstadt, Germany

Abstract

We consider the problem of finding an incremental solution to a cardinality-constrained maximization problem that not only captures the solution for a fixed cardinality, but also describes how to gradually grow the solution as the cardinality bound increases. The goal is to find an incremental solution that guarantees a good competitive ratio against the optimum solution for all cardinalities simultaneously. The central challenge is to characterize maximization problems where this is possible, and to determine the best-possible competitive ratio that can be attained. A lower bound of 2.18 and an upper bound of $\varphi + 1 \approx 2.618$ are known on the competitive ratio for monotone and accountable objectives [Bernstein et al., Math. Prog., 2022], which capture a wide range of maximization problems. We introduce a continuization technique and identify an optimal incremental algorithm that provides strong evidence that $\varphi + 1$ is the best-possible competitive ratio. Using this continuization, we obtain an improved lower bound of 2.246 by studying a particular recurrence relation whose characteristic polynomial has complex roots exactly beyond the lower bound. Based on the optimal continuous algorithm combined with a scaling approach, we also provide a 1.772-competitive randomized algorithm. We complement this by a randomized lower bound of 1.447 via Yao's principle.

2012 ACM Subject Classification Theory of computation → Online algorithms; Mathematics of computing → Combinatorial algorithms; Mathematics of computing → Combinatorial optimization

Keywords and phrases incremental optimization, competitive analysis, robust matching, submodular function

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.47

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2305.01310` [5]

Funding *Max Klimm*: Supported by Deutsche Forschungsgemeinschaft under Germany's Excellence Strategy, Berlin Mathematics Research Center (grant EXC-2046/1, Project 390685689).
Kevin Schewior: Supported in part by the Independent Research Fund Denmark, Natural Sciences, grant DFF-0135-00018B.
David Weckbecker: Supported by DFG grant DI 2041/2.

1 Introduction

A classical optimization problem takes as input a single instance and outputs a single solution. While this paradigm can be appropriate in static situations, it fails to capture scenarios that are characterized by perpetual growth, such as growing infrastructure networks, expanding companies, or private households with a steady income. In these cases, a single static solution may be rendered useless unless it can be extended perpetually into larger, more expansive solutions that are adequate for the changed circumstances.

To capture scenarios like this more adequately, we adopt the *incremental optimization* framework formalized as follows. An instance of the IncMax problem is given by a countable set U of elements and a monotone objective function $f\colon 2^U \to \mathbb{R}_{\geq 0}$ that assigns each subset $X \subseteq U$ a value $f(X)$. A solution for an IncMax instance is an order $\sigma = (e_1, e_2, \dots)$ of the elements of U such that each prefix of σ yields a good solution with respect to the objective function f. Formally, for $k \in [n]$, let $\textsc{Opt}(k) = \max\{f(X) : |X| = k, X \subseteq U\}$ denote the optimal value of the problem of maximizing $f(X)$ under the cardinality-constraint $|X| = k$. A deterministic solution $\sigma = (e_1, e_2, \dots)$ is called α-competitive if $\textsc{Opt}(k)/f(\{e_1, \dots, e_k\}) \leq \alpha$ for all $k \in [n]$. A randomized solution is a probability distribution $\Sigma = (E_1, E_2, \dots)$ over deterministic solutions (where $E_1, E_2, \dots$ are random variables). It is called α-competitive if $\textsc{Opt}(k)/\mathbb{E}[f(\{E_1, \dots, E_k\})] \leq \alpha$ for all $k \in [n]$. In both cases, we call the infimum over all $\alpha \geq 1$, such that the solution is α-competitive, the *(randomized) competitive ratio* of the solution. A (randomized) algorithm is called α-competitive for some $\alpha \geq 1$ if, for every instance, it produces an α-competitive solution, and its *(randomized) competitive ratio* is the infimum over all such α. The *(randomized) competitive ratio* of a class of problems (or a problem instance) is the infimum over the competitive ratios of all (randomized) algorithms for it.

Clearly, in this general form, no meaningful results regarding the existence of competitive solutions are possible. For illustration consider the instance $U = \{a, b, c\}$ where for some $M \in \mathbb{N}$ we have

$$f(X) = \begin{cases} M, & \text{if } \{b, c\} \subseteq X, \\ |\{a\} \cap X|, & \text{otherwise} \end{cases} \qquad \text{for all } X \subseteq U.$$

Then, every solution needs to start with element a in order to be competitive for $k = 1$, but any such order cannot be better than M-competitive for $k = 2$. The underlying issue is that the optimal solution for $k = 2$ given by $\{b, c\}$ does not admit a competitive partial solution of cardinality $k = 1$. To circumvent this issue, Bernstein et al. [1] consider *accountable* functions, i.e., functions f, such that, for every $X \subseteq U$, there exists $e \in X$ with $f(X \setminus \{e\}) \geq f(X) - f(X)/|X|$. They further show that many natural incremental optimization problems are monotone and accountable such as the following.

Weighted matching: U is the set of edges of a weighted graph, and $f(X)$ is the maximum weight of a matching contained in X;

Set packing: U is a set of weighted subsets of a ground set, and $f(X)$ is the maximum weight of a set of mutually disjoint subsets of X;

Submodular function maximization: U is arbitrary, and f is monotone and submodular;

(Multi-dimensional) Knapsack: U is a set of items with (multi-dimensional) sizes and values, and $f(X)$ is the maximum value of a subset of items of X that fits into the knapsack.

Bernstein et al. [1] gave an algorithm to compute a $(1 + \varphi)$-competitive incremental solution and showed that the competitive ratio of the IncMax problem is at least 2.18. Throughout this work, we assume that the objective f is accountable.

Our results. As a first step, we reduce the general IncMax problem to the special case of IncMaxSep, where the elements of the instance can be partitioned into a (countable) set of uniform and modular subsets such that the overall objective is the maximum over the modular functions on the subsets. We then define the IncMaxCont problem as a continuization, where there exists one such subset with (fractional) elements of every size $c \in \mathbb{R}_{>0}$. The smooth structure of this problem better lends itself to analysis.

We consider the continuous algorithm GreedyScaling(c_1, ρ) that adds a sequence of these subsets, starting with the subset of size $c_1 > 0$ and proceeding along a sequence of subsets of largest possible sizes under the constraint that ρ-competitiveness is maintained for as long as possible. We first show that there always exists an optimal solution of this form.

▶ **Theorem 1.** *For every instance of* IncMaxCont*, there exists a starting value c_1 such that the algorithm* GreedyScaling(c_1, ρ^*) *achieves the best-possible competitive ratio $\rho^* \geq 1$.*

Our continuous embedding allows us to view every algorithm as an increasing sequence of sizes of subsets that are added one after the other. Using elementary calculus, we can show that, with the golden ratio $\varphi := \frac{1}{2}(1 + \sqrt{5}) \approx 1.618$, GreedyScaling$(c_1, \rho)$ achieves the known upper bound of $\varphi + 1$ for a range of starting values. Here, $d(c)$ refers to the density, i.e., value per size, of the subset of size c (see Sec. 2).

▶ **Theorem 2.** GreedyScaling$(c_1, \varphi + 1)$ *is $(\varphi + 1)$-competitive if and only if $d(c_1) \geq \frac{1}{\varphi+1}$.*

On the other hand, we are able to, for every starting value c_1, construct an instance of IncMaxCont where GreedyScaling(c_1, ρ) is not better than $(\varphi + 1)$-competitive for any $\rho > 1$. We emphasize that the optimum value of $\varphi + 1$ emerges naturally from the geometry of complex roots. Based on this evidence, we conjecture that $\varphi + 1$ is the best-possible competitive ratio.

Of course, proving a general lower bound requires to construct a single instance such that GreedyScaling is not better than $(\varphi + 1)$-competitive for *every* starting value. Careful chaining of our construction for a single starting value yields the following.

▶ **Proposition 3.** *For every countable set $S \subset \mathbb{R}_{>0}$ of starting values, there exists an instance of* IncMaxCont *such that* GreedyScaling(c_1, ρ) *is not ρ-competitive for any $c_1 \in S$ and any $\rho < \varphi + 1$.*

Crucially, while this gives a lower bound if we only allow rational starting values $c_1 \in \mathbb{Q}$, transferring the lower bound back to IncMax requires excluding all reals. Even though we are not able to achieve this, we can extrapolate our analysis in terms of complex calculus to any IncMaxCont algorithm. With this, we beat the currently best known lower bound of 2.18 in [1].

▶ **Theorem 4.** *The* IncMax *problem has a competitive ratio of at least* 2.246*.*

We can also apply our technique, specifically the reduction to separable problem instances and the structure of the GreedyScaling algorithm, to the analysis of randomized algorithms for IncMax. We employ a scaling approach based on the algorithms in [1], combined with a randomized selection of the starting value c_1 inspired by a randomized algorithm for the CowPath problem in [16]. The resulting algorithm has a randomized competitive ratio that beats our deterministic lower bound.

▶ **Theorem 5.** *IncMax admits a* 1.772*-competitive randomized algorithm.*

We complement this result with a lower bound via Yao's principle for separable instances of IncMax.

▶ **Theorem 6.** *Every randomized* IncMax *algorithm has competitive ratio at least* 1.447*.*

Related work. Our work is based on the incremental maximization framework introduced by Bernstein et al. [1]. We provide a new structural understanding that leads to a better lower bound and new randomized bounds.

A similar framework is considered for matchings by Hassin and Rubinstein [13]. Here, the objective f is the total weight of a set of edges and the solution is additionally required to be a matching. Hassin and Rubinstein [13] show that the competitive ratio in this setting is $\sqrt{2}$ and Matuschke, Skutella, and Soto [19] show that the randomized competitive ratio is $\ln(4) \approx 1.38$. The setting was later generalized to the intersection of matroids [7] and to independence systems with bounded exchangeability [15, 21]. Note that, while our results hold for a broader class of objective functions, we require monotonicity of the objective and cannot model the constraint that the solution must be a matching. We can, however, capture the matching problem by letting the objective f be the largest weight of a matching contained as a subset in the solution (i.e., not all parts of the solution need to be used). That being said, it is easy to verify that the lower bound of $\sqrt{2}$ on the competitiveness of any deterministic algorithm in the setting of [13] also applies in our case.

Hassin and Segev [14] studied the problem of finding a small subgraph that contains, for all k, a path (or tree) of cardinality at most k with weight at least α times the optimal solution and show that for this $\alpha|V|/(1-\alpha^2)$ edges suffice. There are further results on problems where the items have sizes and the cardinality-constraint is replaced by a knapsack constraint [4, 6, 17, 20]. Goemans and Unda [9] studied general incremental maximization problems with a sum-objective.

Incremental *minimization* problems further been studied for a variety of minimzation problems such as k-median [3, 22, 18], facility location [18, 23], and k-center [10, 18]. As noted by Lin et al. [18], the results for the minimum latency problem in [2, 8] implicitly yield results for the incremental k-MST problem. There are further results on incremental optimization problems where in each step the set of feasible solution increases [11, 12].

2 Separability of Incremental Maximization

As a first step to bound the competitive ratio of IncMax, we introduce a subclass of instances of a relatively simple structure, and show that it has the same competitive ratio as IncMax. Thus, we can restrict ourselves to this subclass in our search for bounds on the competitive ratio.

▶ **Definition 7.** *An instance of IncMax with objective $f\colon 2^U \to \mathbb{R}_{>0}$ is called* separable *if there exist a partition $U = R_1 \cup R_2 \cup \dots$ of U and values $d_i > 0$ such that*

$$f(X) = \max_{i \in \mathbb{N}} \{|X \cap R_i| \cdot d_i\} \quad \text{for all } X \subseteq U.$$

We refer to d_i as the density *of set R_i and to $v_i := |R_i| \cdot d_i$ as the* value *of set R_i. The restriction of IncMax to separable instances will be denoted by IncMaxSep.*

We start our analysis of IncMaxSep with the following immediate observation.

▶ **Lemma 8.** *Any instance of IncMaxSep can be transformed into one with the same or a worse competitive, that satisfies the following properties.*

1. *There is exactly one set of every cardinality, i.e., $|R_i| = i$.*
2. *Densities are decreasing, i.e., $1 \geq d_1 \geq d_2 \geq \dots$.*
3. *Values are increasing, i.e., $v_1 \leq v_2 \leq \dots$.*

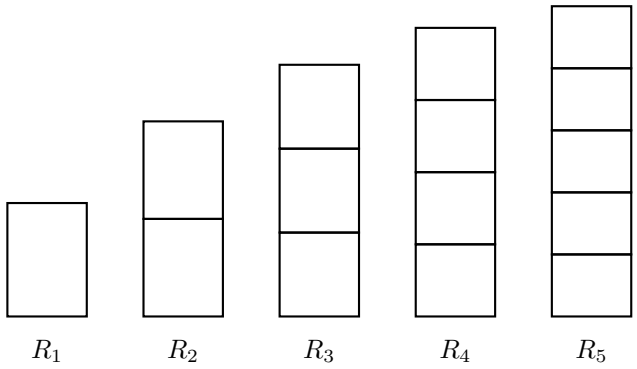


Figure 1 Illustration of an instance of IncMaxSep with $N = 5$ sets. Each set R_i consists of i elements. The height of the elements represents their value. As in Lemma 8, the values of the single elements becomes less the larger i is, while the value of the whole set R_i increases.

Proof. We will show that every instance that does not satisfy the assumptions can be transformed into one that does, without changing the optimum value for any size, and without changing the value of the best incremental solution. Thus the competitive ratio of the two instances coincide.

If there are two sets R_i, R_j with $|R_i| = |R_j|$, it only makes sense to consider the one with higher density, as every solution adding elements from the set of lower density can be improved by adding elements from the other set instead. If there is $i \in \mathbb{N}_{\geq 2}$ such that there is no set with i elements, we can add a new set with i elements to the instance. This new set will have value v_{i-1} . Then, every solution that adds elements from the newly introduced set can be improved by adding elements from set R_{i-1} instead. If there is no set R_1 with 1 element, we can introduce it with density d_2. Then, every solution that adds this one element can instead also add one element from R_2. Thus, the first assumption can be made.

The assumption that $1 \geq d_1$ is without loss of generality by rescaling the objective f. If there was $i \in \mathbb{N}$ with $d_i < d_{i+1}$, every solution to the problem instance that adds elements from the set R_i could be improved by adding elements from the set R_{i+1} instead. Since $|R_{i+1}| \geq |R_i|$, this is possible.

The third assumption can be made because, if there was $i \in \mathbb{N}$ with $v_i > v_{i+1}$, a solution that adds elements from R_{i+1} can be improved by adding elements from R_i instead. ◀

In the following, we assume that every instance satisfies the properties from Lemma 8.

▶ **Definition 9.** *We say that a solution for* IncMaxSep *is* represented *by a sequence of sizes* $(c_1, c_2, \dots)$ *if it first adds all elements from the set* R_{c_1}*, then all elements from the set* R_{c_2}*, and so on.*

A solution of IncMaxSep can only improve if it is altered in a way that it is represented by a sequence of sizes. Indeed, if not all elements of one set are added, the solution does not degrade if a smaller set is added instead because the density of the smaller set is at least as large as the density of the larger set. Moreover, adding all elements of one set consecutively is better because the value of the solution increases faster this way.

▶ **Lemma 10** ([1], Observation 2). *There is an algorithm achieving the best-possible competitive ratio for* IncMaxSep *such that the solution generated by this algorithm can be represented by a sequence* $(c_1, c_2, \dots)$*. We can assume that* $v_{c_i} < v_{c_{i+1}}$ *and thus, since the values* $(v_i)_{i \in \mathbb{N}}$ *are non-decreasing,* $c_i < c_{i+1}$ *for all* $i \in \mathbb{N}$*.*

From now on, we will only consider solutions of this form and denote a solution X by the sequence it is represented by, i.e., $X = (c_1, c_2, \dots)$. For a size $C \in \mathbb{N}$, we denote by $X(C)$ the first C elements added by X, i.e., $|X(C)| = C$ and, with $O_i := \arg\max\{f(S) \mid S \subseteq U, |S| = i\}$, we have $X\left(\sum_{i=1}^{k} c_i\right) = \bigcup_{i=1}^{k} O_{c_i}$.

▶ **Proposition 11.** *The competitive ratios of* IncMax *and* IncMaxSep *coincide.*[1]

Proof Sketch. As IncMaxSep is a subclass of IncMax, the competitive ratio of IncMaxSep is not larger than that of IncMax.

It remains to show that the competitive ratio of IncMax is smaller or equal to that of IncMaxSep. To see this, consider an instance of IncMax. We will construct an instance of IncMaxSep such that every ρ-competitive solution to this problem instance induces a ρ-competitive solution for the initial instance of IncMax.

To define the instance of IncMaxSep, let $R_1, R_2, \dots$ be disjoint sets with $|R_i| = i$ for all $i \in \mathbb{N}$. For $i \in \mathbb{N}$, let $d_i = \textsc{Opt}(i)/i$. By modularity of the value function within one set R_i, the value of the optimal solution of a given size in this instance is the same as that in the instance of IncMax.

For $\rho \geq 1$, let $(c_1, c_2, \dots)$ be a ρ-competitive solution for the separable instance. We consider the solution for the initial problem that starts by adding the optimal solution of size c_1, then adds the optimal solution of size c_2, and so on. Accountability guarantees that it is possible to add the elements within one optimal solution such that the value of the partially added solution grows at least proportionally with the size of the solution. Since the values of the optimal solutions of a given size in the two instances coincide, the value of the solution for the initial instance we defined above is always greater or equal to that of the solution $(c_1, c_2, \dots)$. Thus, the solution for the initial instance is also ρ-competitive, which implies that the competitive ratio of IncMax is smaller or equal to that of IncMaxSep. ◀

3 Continuization Results

In order to find lower bounds on the competitive ratio of IncMaxSep, we transform the problem into a continuous one.

▶ **Definition 12.** *In the* IncMaxCont *problem, we are given a* density function $d\colon \mathbb{R}_{\geq 0} \to (0, 1]$ *and a* value function $v(c) := cd(c)$. *As for the discrete problem, we denote an incremental solution X for* IncMaxCont *by a sequence of sizes $X = (c_1, c_2, \dots)$. For a given size $c \geq 0$, we denote the solution of this size by $X(c)$. With $n \in \mathbb{N}$ such that $\sum_{i=1}^{n-1} c_i < c \leq \sum_{i=1}^{n} c_i$, the value of $X(c)$ is defined as*

$$f(X(c)) := \max\left\{\max_{i \in \{1,\dots,n-1\}} v(c_i), \left(c - \sum_{i=1}^{n-1} c_i\right) d(d_n)\right\}.$$

An incremental solution X is ρ-competitive if $\rho \cdot f(X(c)) \geq v(c)$ for all $c > 0$. The competitive ratio of X is defined as $\inf\{\rho \geq 1 \mid X \text{ is } \rho\text{-competitive}\}$.

The interpretation of the functions d and v is that the instance is partitioned into sets, one for every positive size $c \in \mathbb{R}$, each consisting of c fractional units with a value of $d(c)$ per unit, for a total value of $v(c)$ for the set. The solution can be interpreted in the following way: It starts by adding the set of size c_1, then the set of size c_2, and so on. With $n \in \mathbb{N}$ such that

[1] A full proof of this and all other results can be found in [5].

$\sum_{i=1}^{n-1} c_i < c \leq \sum_{i=1}^{n} c_i$, the solution $X(c)$ has added all of the sets of sizes $c_1, \ldots, c_{n-1}$ and $c - \sum_{i=1}^{n-1} c_i$ units of the set of size c_n. Unlike the IncMaxSep problem, the IncMaxCont problem includes subsets of all real sizes instead of only integer sizes and, furthermore, allows fractional elements to be added to solutions instead of only an integral number of elements.

As for the discrete version of the problem, without loss of generality, we assume that the density function d is non-increasing and the value function v is non-decreasing. These assumptions imply that d is continuous: If this was not the case and d was not continuous for some size c', i.e., $\lim_{c \nearrow c'} d(c) > \lim_{c \searrow c'} d(c)$, then $\lim_{c \nearrow c'} v(c) > \lim_{c \searrow c'} v(c)$ by definition of v, i.e., v would not be increasing in c. So d is continuous, and, by definition of v, also v is continuous. Furthermore, without loss of generality, we assume that $d(0) = 1$.

For a fixed size $c \geq 0$, we define $p(c) = \max\{c' \geq 0 \mid v(c') \leq \rho v(c)\}$. This value gives the size up to which a solution with value $v(c)$ is ρ-competitive. Throughout our analysis, we assume that $p(c)$ is defined for every $c \geq 0$, i.e., that $\lim_{c \to \infty} v(c) = \infty$. Otherwise, any algorithm can terminate when the value of its solution is at least $\frac{1}{\rho} \sup_{c \in \mathbb{R}_{\geq 0}} v(c)$.

▶ **Proposition 13.** *The competitive ratio of IncMaxSep is greater or equal to that of IncMaxCont.*

Proof Sketch. Given a lower bound construction for the competitive ratio of IncMaxCont, one can discretize it with arbitrary resolution such that, with an arbitrarily small loss, it carries over to the IncMaxSep problem. ◀

This proposition implies that instead of devising a lower bound for the IncMaxSep problem, we can construct a lower bound for the IncMaxCont problem.

Note that it is not clear whether the competitive ratio of IncMaxSep and IncMaxCont coincide. This is due to the fact that a solution to the IncMaxCont problem may add fractional elements while a solution to the IncMaxSep problem may only add an integral number of items. There are even discrete instances where every continuization of the instance has a competitive ratio smaller than the initial instance.

▶ **Observation 14.** *There exists an instance of IncMaxSep that has a competitive ratio that is strictly larger than that of every instance of IncMaxCont that monotonically interpolates the IncMaxSep instance.*

Proof Sketch. We show that the instance of IncMaxSep with $N = 16$ sets and

$$\begin{aligned} d_1 &= 1, \\ d_3 = d_4 &= \frac{17}{40}, \\ d_{12} = d_{13} = d_{14} = d_{15} = d_{16} &= \frac{16473}{107200}. \end{aligned}$$

has a competitive ratio of at least 1.446, while every monotone interpolation of it has a competitive ratio of at most 1.425. ◀

Note that, even though this shows that there are instances where the continuous problem is easier than the discrete one, this does not rule out that the competitive ratios of IncMaxSep and IncMaxCont coincide. This is due to the fact that the instance in the proof is not a worst-case instance.

3.1 Optimal Continuous Online Algorithm

In this section, we present an algorithm to solve the IncMaxCont problem, and analyze it. To get an idea what the algorithm does, consider the following lemma. It gives a characterization of a solution $(c_1, c_2, \dots)$ being ρ-competitive, depending on $(c_1, c_2, \dots)$, v and d.

▶ **Lemma 15.** *A solution $(c_1, c_2, \dots)$ for an instance of the IncMaxCont problem is ρ-competitive if and only if $d(c_1) \geq \frac{1}{\rho}$ and, for all $i \in \mathbb{N}$, $d(c_{i+1}) \geq \frac{v(c_i)}{p(c_i) - \sum_{j=1}^{i} c_j}$.*

The intuition behind the fraction

$$\frac{v(c_i)}{p(c_i) - \sum_{j=1}^{i} c_j}$$

is the following: The value of the solution $(c_1, \dots, c_{i-1}, c_i)$ is $v(c_i)$ and this value is ρ-competitive up to size $p(c_i)$. The size required for this solution is $\sum_{j=1}^{i} c_j$. Thus, in order to stay competitive, the size added next, namely c_{i+1}, needs to be chosen such that $\big(p(c_i) - \sum_{j=1}^{i} c_j\big) d(c_{i+1}) \geq v(c_i)$, i.e., the density $d(c_{i+1})$ is large enough such that the value of the solution of size $p(c_i)$ is $\big(p(c_i) - \sum_{j=1}^{i} c_j\big) d(c_{i+1})$.

We use this fraction to define an algorithm for solving the IncMaxCont Problem. For the algorithm, we assume that v is strictly increasing and d is strictly decreasing to make the choice of our algorithm unique. Every instance of IncMaxCont can be transformed to satisfy this with an arbitrarily small loss by simpliy "tilting" constant parts of d and v by a small amount. The algorithm GreedyScaling(c_1, ρ) starts by adding the optimal solution of size $c_1 > 0$ and chooses the size c_{i+1} such that

$$d(c_{i+1}) = \frac{v(c_i)}{p(c_i) - \sum_{j=1}^{i} c_j}, \tag{1}$$

i.e., as large as possible while still satisfying the inequality in Lemma 15. An illustration of the algorithm can be found in Figure 2.

Using the definition of the algorithm in (1) and Lemma 15, we are able to prove the following.

▶ **Proposition 16.** *The algorithm GreedyScaling(c_1, ρ) is ρ-competitive if and only if it produces a solution $(c_1, c_2, \dots)$ with $c_i < c_{i+1}$ for all $i \in \mathbb{N}$ and $d(c_1) \geq \frac{1}{\rho}$.*

Proof Sketch. "⇐": If $c_i < c_{i+1}$ for all $i \in \mathbb{N}$ and $d(c_1) \geq 1/\rho$, we can simply apply Lemma 15 and obtain that the solution is ρ-competitive.

"⇒": If $d(c_1) < 1/\rho$, Lemma 15 yields that the solution is not ρ-competitive. If $c_{k+1} \leq c_k$ for some $k \in \mathbb{N}$, one can iteratively show that $c_{i+1} \leq c_i$ for all $i \in \{k, k+1, \dots\}$. This implies that the value of the solution $(c_1, c_2, \dots)$ is smaller or equal to $v(c_k)$ for all sizes. Yet, for large sizes $C \in \mathbb{N}$, we have $v(C) > \rho v(c_k)$ as $\lim_{c \to \infty} v(c) = \infty$. ◀

The algorithm GreedyScaling(c_1, ρ) only depends on the desired competitive ratio ρ and the starting value c_1. Given that some algorithm can achieve a competitive ratio of ρ, we can show that GreedyScaling(c_1^*, ρ) with the correct starting value $c_1^* > 0$ also gives a ρ-competitive solution.

▶ **Theorem 1.** *For every instance of IncMaxCont, there exists a starting value c_1 such that the algorithm GreedyScaling(c_1, ρ^*) achieves the best-possible competitive ratio $\rho^* \geq 1$.*

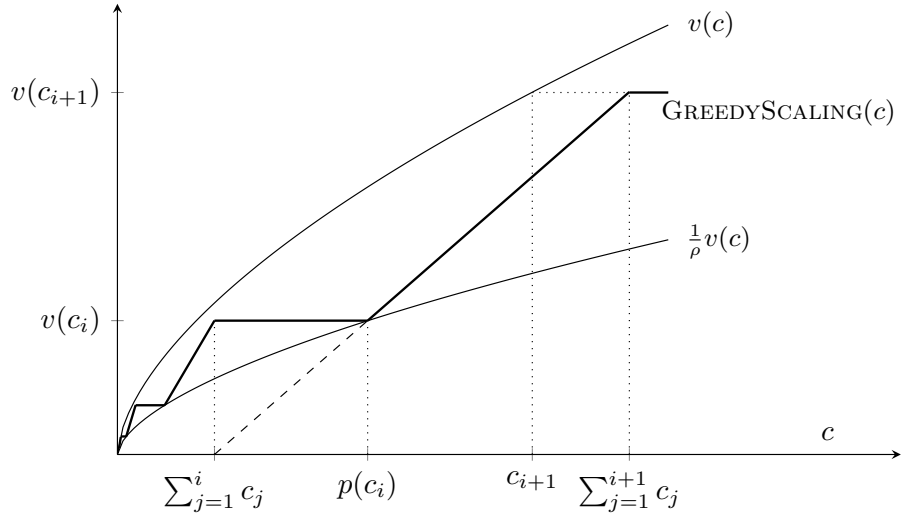


Figure 2 Illustration how GreedyScaling(c_1, ρ) works. Between size $\sum_{j=1}^{i} c_j$ and size $\sum_{j=1}^{i+1} c_j$, the algorithm adds the optimal solution of size c_{i+1}. This size is chosen in a way that the value of the partially added solution has value $v(c_i)$ exactly at size $p(c_i)$, i.e., when the previously added solution of size c_i loses ρ-competitiveness.

Proof sketch. The idea of the proof is to start with a ρ^*-competitive solution $(c_1, c_2, \dots)$ for the instance of IncMaxCont. For every $k \in \mathbb{N}$, we define a new ρ^*-competitive solution $(c_1^k, c_2^k, \dots)$ as follows. For $i \in \mathbb{N}$ with $\sum_{j=1}^{i} c_j \geq k$, we set $c_i^k = c_i$. For $i \in \mathbb{N}$ with $\sum_{j=1}^{i} c_j < k$, we choose $c_i^k \geq 0$ as small as possible without losing ρ^*-competitiveness. This new solution satisfies the inequality

$$d(c_{i+1}^k) \geq \frac{v(c_i^k)}{p(c_i^k) - \sum_{j=1}^{i} c_j^k}$$

from Lemma 15 with equality for $i \in \{1, \dots, k-1\}$. This implies, that we can calculate $c_2^k, \dots, c_k^k$ solely based on c_1^k, d, and v. For every $k \in \mathbb{N}$, we obtain such a solution $(c_1^k, c_2^k, \dots)$. For all $k \in \mathbb{N}$, we have $d(c_1^k) \geq 1/\rho^*$, which implies that all sizes in $\{c_1^1, c_1^2, \dots\}$ are from the finite interval $[0, d^{-1}(1/\rho^*)]$. By the Bolzano-Weierstrass theorem, this implies that the sequence $(c_1^1, c_1^2, \dots)$ contains a converging sub-sequence. If we choose the limit of this sub-sequence to be the starting value c_1^* of the algorithm GreedyScaling(c_1^*, ρ^*), we obtain a ρ^*-competitive algorithm. ◀

For a range of starting values c_1, we are able to show the upper bound on the competitive ratio of GreedyScaling$(c_1, \varphi + 1)$ in Theorem 2, where $\varphi = \frac{1}{2}(1 + \sqrt{5}) \approx 1.618$ is the golden ratio.

▶ **Theorem 2.** *GreedyScaling$(c_1, \varphi+1)$ is $(\varphi+1)$-competitive if and only if $d(c_1) \geq \frac{1}{\varphi+1}$.*

Proof Sketch. By Proposition 16, it suffices to show that, for the solution $(c_1, c_2, \dots)$ produced by GreedyScaling$(c_1, \varphi + 1)$, we have $c_{i+1} > c_i$ for every $c_1 > 0$ with $d(c_1) > \frac{1}{\varphi+1}$. We show iteratively that $c_{i+1} \geq (\varphi + 1)c_i$. In order to do this, we observe that

$$p(c_i) = \frac{(\varphi + 1)v(c_i)}{d(p(c_i))} \geq (\varphi + 1)c_i.$$

By a straighforward induction that uses the fact that $(\varphi+1)^{i-j}c_j \leq c_i$ for all $j \in \{1,\dots,i-1\}$ as well as the definition of GreedyScaling$(c_1, \varphi+1)$ we obtain that $d(c_{i+1}) < d(p(c_i))$. This implies $c_{i+1} > p(c_i) \geq (\varphi+1)c_i$. ◀

Since GreedyScaling(c_1,ρ) with the correct starting value c_1 is the best-possible algorithm for a fixed instance, we can give a lower bound of $\rho > 1$ for the IncMaxCont problem by finding an instance that is a lower bound for GreedyScaling(c_1,ρ) with all starting values $c_1 > 0$ that satisfy $d(c_1) \leq 1/\rho$. In the following, we show that, for every countable set of starting values, there is an instance where GreedyScaling(c_1,ρ) cannot have a competitive ratio of better than $\varphi+1$ for any of these starting values. In order to do this, we need the following lemma.

▶ **Lemma 17.** *For $\alpha,\beta,\rho,\epsilon \in \mathbb{R}_{\geq 0}$ with $\beta > 0$, consider the recursively defined sequence $(t_n)_{n\in\mathbb{N}}$ with*

$$t_0 = \beta, \qquad t_{n+1} = \frac{1}{\frac{\rho}{t_n(1-\epsilon)} - \left(\sum_{j=0}^{n} \frac{(\rho+\epsilon)^{j-n}}{t_j}\right) - \frac{\alpha}{(\rho+\epsilon)^n}} \quad \text{for all } n \in \mathbb{N}\cup\{0\}.$$

If $1 < \rho < \varphi+1$, then there exists $\epsilon' > 0$ such that, for all $\epsilon \in (0,\epsilon']$, there is $\ell \in \mathbb{N}$ with $t_\ell < 0$.

Proof sketch. We define an auxiliary sequence $(a_n)_{n\in\mathbb{N}}$ with $a_n = \frac{1}{t_n}$ for all $n \in \mathbb{N}\cup\{0\}$. This sequence becomes negative if and only if $(t_n)_{n\in\mathbb{N}\cup\{0\}}$ becomes negative. We show that $(a_n)_{n\in\mathbb{N}\cup\{0\}}$ is fully described by the homogeneous recurrence relation

$$a_{n+1} = a_n\left(\frac{1}{\rho+\epsilon} + \frac{\rho}{1-\epsilon} - 1\right) - a_{n-1}\frac{\rho}{(1-\epsilon)(\rho+\epsilon)}$$

for all $n \in \mathbb{N}$, together with the start values $a_0 = 1/\beta$ and

$$a_1 = \frac{1}{t_1} = \frac{\rho}{\beta(1-\epsilon)} - \frac{1}{\beta} - \alpha.$$

Its characteristic polynomial is

$$0 = x^2 - \left(\frac{1}{\rho+\epsilon} + \frac{\rho}{1-\epsilon} - 1\right)x + \frac{\rho}{(1-\epsilon)(\rho+\epsilon)}.$$

We show that the roots x and y of this polynomial are complex if $\rho < \varphi+1$ and $\epsilon > 0$ small enough. Thus, they are also distinct which implies that the sequence $(a_n)_{n\in\mathbb{N}\cup\{0\}}$ has the closed-form expression

$$a_n = \lambda x^n + \mu y^n$$

for all $n \in \mathbb{N}\cup\{0\}$ where $\lambda,\mu \in \mathbb{C}$ are chosen accordingly. The fact that the starting values a_0 and a_1 are real valued imply that λ and μ are complex conjugate. Thus, we obtain

$$a_n = 2\Re(\lambda x^n)$$

for all $n \in \mathbb{N}\cup\{0\}$, where $\Re(\lambda x^n)$ denotes the real part of λx^n. We analyze this equation by visualizing it on the complex plane (cf. Figure 3). Since x is not real valued, multiplying by x corresponds to a rotation by an angle that is not 0 and not π. Thus, for some $n \in \mathbb{N}\cup\{0\}$, $\Re(\lambda x^n)$ must become negative. ◀

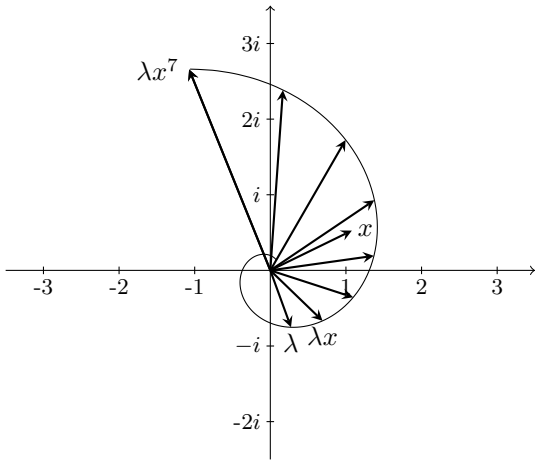


Figure 3 Multiplying λ repeatedly by $x \in (\mathbb{C} \setminus \mathbb{R})$ is equivalent to a rotation around the origin that, at some point, reaches the half-plane corresponding to negative real parts.

▶ **Proposition 3.** *For every countable set $S \subset \mathbb{R}_{>0}$ of starting values, there exists an instance of* IncMaxCont *such that* GreedyScaling(c_1, ρ) *is not ρ-competitive for any $c_1 \in S$ and any $\rho < \varphi + 1$.*

Proof Sketch. We give an overview how to construct an instance where the algorithm GreedyScaling(c_1, ρ) is not ρ-competitive for one fixed starting value $c_1 > 0$ and every $\rho \in [1, \varphi + 1)$. For the sake of simplicity, in this overview, we describe an instance where the density function d and the value function v are locally constant. In the final construction, we avoid this by slightly tilting constant parts of the function.

Let $\epsilon > 0$ be arbitrarily small. The beginning of the instance up to size c_1 can be chosen arbitrarily. We set $d(c) = d(c_1)$ for all $c \in [c_1, (\rho + \epsilon)c_1]$. By doing this, we ensure that the value obtained by adding the optimal solution of the first size c_1 is ρ-competitive for as few sizes as possible, i.e., until $p(c_1) = \rho c_1$. Then, $d(c_2) = \frac{v(c_1)}{\rho c_1 - c_1}$ can be calculated. We set $v(c) = v((\rho+\epsilon)c_1)$ for all $c \in [(\rho + \epsilon)c_1, \frac{v((\rho+\epsilon)c_1)}{d(c_2)}]$. This ensures that c_2 is as small as possible, namely $c_2 = \frac{v((\rho+\epsilon)c_1)}{d(c_2)}$. Now we repeat what we did for c_1, i.e., we define d to be constant so that the value $v(c_2)$ is ρ-competitive for as few sizes as possible. Then, we calculate $d(c_3)$ and define v to be constant so that c_3 is as small as possible. We continue doing this for all larger c_i with $i \geq 3$. It turns out that we have $d(c_i) = t_i$ where the sequence $(t_i)_{i \in \mathbb{N}}$ is defined as in Lemma 17. Thus, at some point, the density GreedyScaling(c_1, ρ) calculates the next capacity to be negative, which is not possible, i.e., the algorithm is not ρ-competitive.

We have seen how to construct an instance that excludes one starting value. This instance is finite and the beginning can be chosen arbitrarily. Thus, we can chain together multiple of these instances by scaling an instance for some set of starting values and modifying the beginning such that it contains an instance for an additional starting value. ◀

3.2 General Lower Bound

Now we want to employ the techniques we used to prove Lemma 17 and Proposition 3 in order to prove a lower bound on the competitive ratio of IncMaxCont. Let ρ^* be the unique real root $\rho \geq 1$ of the polynomial $-4\rho^6 + 24\rho^4 - \rho^3 - 30\rho^2 + 31\rho - 4$. As before, we need to show that a recursively defined sequence becomes negative at some point.

▶ **Lemma 18.** *For $\rho \in \mathbb{R}_{\geq 0}$ and $\epsilon > 0$, consider the recursively defined sequence $(t_n)_{n\in\mathbb{N}}$ with*

$$t_0 = 1, \qquad t_1 = \frac{1-\epsilon}{\rho}, \qquad t_n = \frac{1-\epsilon}{\frac{\rho}{t_{n-1}} - \frac{1}{t_{n-2}} - \frac{1}{\rho}\left(\sum_{j=0}^{n-3} \frac{(\rho+\epsilon)^{j+2-n}}{t_j}\right)} \quad \text{for all } n \in \mathbb{N}_{\geq 2}.$$

If $1 < \rho < \rho^*$, then there exists $\epsilon' > 0$ such that, for all $\epsilon \in [0, \epsilon']$, there is $\ell \in \mathbb{N}$ with $t_\ell < 0$.

The proof of this lemma is along the same lines as the proof of Lemma 17, with additional technical difficulties because the recurrence relation of the sequence is of order 3. With this lemma, we are ready to construct our lower bound on the competitive ratio of IncMaxCont and thus, via Propositions 11 and 13, of IncMax.

▶ **Theorem 4.** *The IncMax problem has a competitive ratio of at least* 2.246.

Proof sketch. We fix a competitive ratio $\rho < \rho^*$ and some small $\epsilon > 0$. Similarly to the construction in the proof of Proposition 3, the lower bound in Theorem 4 is a construction where we have intervals on which, alternatingly, either the density function or the value function is constant (cf. Figure 4). For $i \in \mathbb{N}$, on the $(2i)$-th interval, the value is constant and equals $(\rho+\epsilon)^{i-1}$. On the $(2i-1)$-th interval, the density is constant and equal to t_{i-1}, where $(t_n)_{n\in\mathbb{N}}$ is defined as in Lemma 18. Every solution that contains a size from an interval of constant value can be improved by picking the largest size from the preceding interval of constant density instead. This size has the same value and is smaller. Thus, we assume that algorithms only pick sizes from the intervals with constant density. We denote the solution by $(c_1, c_2, \dots)$. We have $d(c_1) = t_0 = 1$ because $t_1 < 1/\rho$ is too small. In order to be competitive for the first constant value interval of value 1, the solution has to satisfy $c_1 \geq 1/\rho$ to achieve a value of at least $1/\rho$. Then, the following recursive argument is made. Fix $i \in \mathbb{N}$. Whenever, for all $j \in \{1, \dots, i\}$, the solution satisfies $d(c_j) = t_{j-1}$ and $c_j \geq \frac{(\rho+\epsilon)^{i-1}}{\rho}$, then we have $d(c_{i+1}) = t_i$ and $c_{i+1} \geq \frac{(\rho+\epsilon)^i}{\rho}$. The equality $d(c_{i+1}) = t_i$ is due to the definition of the sequence $(t_n)_{n\in\mathbb{N}}$ and Lemma 15. The inequality $c_{i+1} \geq \frac{(\rho+\epsilon)^i}{t_i \rho}$ follows from the fact that, after the size c_{i+1} is added to the solution, the solution has to be competitive on the $(2i+2)$-th interval of value $(\rho+\epsilon)^i$. Since the sequence $(t_n)_{n\in\mathbb{N}}$ becomes negative at some point, the solution is not ρ-competitive. ◀

4 Randomized Incremental Maximization

We turn to analyzing randomized algorithms to solve the (discrete) IncMaxSep problem. In contrast to deterministic algorithms, we do not compare the value obtained by the algorithm to an optimum solution, but rather the expected value obtained by the algorithm. This enables us to find an algorithm with randomized competitive ratio smaller than the lower bound of 2.24 on the competitive ratio of deterministic algorithms in Theorem 4.

4.1 Randomized Algorithm

Scaling algorithms, i.e., algorithms where the size c_i is chosen such that $c_i = \delta c_{i-1}$ with an appropriate scaling factor $\delta > 1$, have been proven to perform well for the deterministic version of the problem. The best known algorithm is, in fact, a scaling algorithm [1]. In the analysis, it turns out that, on average, a scaling algorithm performs better than the actual competitive ratio, which is only tight for few sizes. By randomizing the initial size c_0, we manage to average out the worst-case sizes in the analysis.

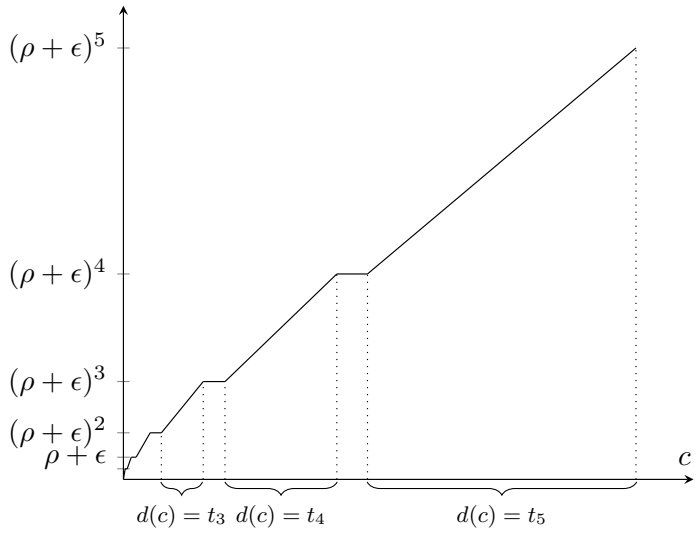


Figure 4 Lower bound construction for $\rho = 2.1$.

We describe the randomized algorithm RandomizedScaling for IncMaxSep. Let $r > 1$ be some scaling parameter to be determined later. The algorithm RandomizedScaling starts by choosing $\epsilon \in (0,1)$ uniformly at random. For all $i \in \mathbb{N}_0$, it calculates $\tilde{c}_i := r^{i+\epsilon}$ and $c_i := \lfloor \tilde{c}_i \rfloor$ and returns the solution $(c_0, c_1, c_2, \dots)$.[2] This approach is similar to a randomized algorithm to solve the CowPath problem in [16], which also calculates such a sequence with a different choice of $r \in \mathbb{R}$ in order to explore a star graph.

We define

$$\tilde{t}_i := \sum_{j=0}^{i} \tilde{c}_j = r^{\epsilon} \frac{r^{i+1}-1}{r-1} \qquad \text{and} \qquad t_i := \sum_{j=0}^{i} c_j.$$

For better readability, we let $\tilde{c}_{-1} = c_{-1} = \tilde{t}_{-1} = t_{-1} = 0$. Note that, for all $i \in \mathbb{N}_0$, we have

$$t_{i-1} \le \tilde{t}_{i-1} = r^{\epsilon} \frac{r^i - 1}{r-1} \overset{r>2}{\le} r^{i+\epsilon} - r^{\epsilon} \le r^{i+\epsilon} - 1 = \tilde{c}_i - 1 \le c_i. \tag{2}$$

For every size $c \in \mathbb{N}_0$, we denote the solution created by the algorithm RandomizedScaling by $X_{\text{ALG}}(c)$. Note that the optimum solution of size $c \in \mathbb{N}_0$ is given by the set R_c because $v_1 \le v_2 \le \dots$ and $d_1 \ge d_2 \ge \dots$. Thus, the value of the optimum solution of size c is v_c.

In order to find an upper bound on the randomized competitive ratio of RandomizedScaling, we need the following lemma. It gives an estimate on the expected value of the solution for a fixed size $C \in \mathbb{N}$ of RandomizedScaling depending on the interval in which C falls.

▶ **Lemma 19.** *Let $C \in \mathbb{N}$.*

1. *For $i \in \mathbb{N} \cup \{0\}$ with $\mathbb{P}[C \in (c_{i-1}, c_i]] > 0$, we have*

$$\mathbb{E}\big[f(X_{\text{ALG}}(C)) \mid C \in (c_{i-1}, c_i]\big] \ge \mathbb{E}\Big[\max\Big\{\frac{c_{i-1}}{C}, \frac{C - t_{i-1}}{\max\{C, c_i\}}\Big\} \;\Big|\; C \in (c_{i-1}, c_i]\Big] \cdot v_C.$$

[2] With this definition, the algorithm does not terminate on finite instances. To avoid this, it suffices to stop calculating the sizes c_i until they are larger than the number of elements in the instance.

2. *For $i \in \mathbb{N}$ with $\mathbb{P}[C \in (\tilde{c}_i, \tilde{t}_i - 1]] > 0$, we have*

$$\mathbb{E}\big[f(X_{ALG}(C)) \mid C \in (\tilde{c}_i, \tilde{t}_i - 1]\big] \geq \mathbb{E}\Big[1 - \frac{\tilde{t}_{i-1}}{C} \;\Big|\; C \in (\tilde{c}_i, \tilde{t}_i - 1]\Big] \cdot v_C.$$

3. *For $i \in \mathbb{N}$ with $\mathbb{P}[C \in (\tilde{t}_{i-1} - 1, \tilde{c}_i]] > 0$, we have*

$$\mathbb{E}\big[f(X_{ALG}(C)) \mid C \in (\tilde{t}_{i-1}{-}1, \tilde{c}_i]\big] \geq \mathbb{E}\Big[\max\Big\{\frac{\tilde{c}_{i-1} - 1}{C}, \frac{C - \tilde{t}_{i-1}}{\tilde{c}_i}\Big\} \;\Big|\; C \in (\tilde{t}_{i-1}{-}1, \tilde{c}_i]\Big] \cdot v_C.$$

By choosing $r \approx 5.1646$ to be the unique maximum of

$$\begin{aligned} g(x) \;=\; & \frac{1 - \sqrt{\big(\frac{x^3-1}{x-1}x^z - 1\big)^2 + 4x^{5+2z}}}{2\log(x)x^{3+z}} - (1-\delta)\frac{1-x^{-3}}{x-1} + z - \frac{1-x^{-3}}{2(x-1)\log(x)} \\ & - \Big(\frac{1-x^{-3}}{x-1} - \frac{1}{x^{3+z}}\Big)\Big(\log_x\Big(\sqrt{\big(\frac{x^3-1}{x-1}x^z - 1\big)^2 + 4x^{5+2z}} - \frac{x^3-1}{x-1}x^z + 1\Big) \\ & - \log_x(2) - 3\Big) - \frac{2x^{2+z}}{\Big(\sqrt{\big(\frac{x^3-1}{x-1}x^z - 1\big)^2 + 4x^{5+2z}} - \frac{x^3-1}{x-1}x^z + 1\Big)\log(x)} \\ & + \frac{2}{\log(x)} - \Big(1 + \frac{1}{x^{3+z}}\Big)\Big(\log_x(x^{3+z}+1) + \log_x(x-1) - \log_x(x^4-1)\Big), \end{aligned}$$

we can show that the following holds.

▶ **Lemma 20.** *Let $k \in \mathbb{N}$ and $\delta \in (0,1]$ such that $r^{k+\delta} \geq \sum_{i=0}^{3} r^i$. Then*

$$\begin{aligned} g(r) \leq I(k,\delta) := & \int_{\min\{1,\mu(k-1)\}}^{1} 1 - \frac{\tilde{t}_{k-2}}{r^{k+\delta}}\,d\epsilon + \int_{\min\{1,\nu(k-1)\}}^{\min\{1,\mu(k-1)\}} \frac{\tilde{c}_{k-1}-1}{r^{k+\delta}}\,d\epsilon \\ & + \int_{\delta}^{\min\{1,\nu(k-1)\}} \frac{r^{k+\delta} - \tilde{t}_{k-1}}{\tilde{c}_k}\,d\epsilon + \int_{\max\{0,\mu(k)\}}^{\delta} 1 - \frac{\tilde{t}_{k-1}}{r^{k+\delta}}\,d\epsilon \\ & + \int_{\max\{0,\nu(k)\}}^{\max\{0,\mu(k)\}} \frac{\tilde{c}_k - 1}{r^{k+\delta}}\,d\epsilon + \int_{0}^{\max\{0,\nu(k)\}} \frac{r^{k+\delta} - \tilde{t}_k}{\tilde{c}_{k+1}}\,d\epsilon \end{aligned}$$

where

$$\begin{aligned} \mu(i) &= \log_r(r^{k+\delta}+1) + \log_r(r-1) - \log_r(r^{i+1}-1), \\ \nu(i) &= \log_r\Big(\sqrt{\Big(r^{k+\delta}\frac{1-r^{-(i+1)}}{r-1} - 1\Big)^2 + 4r^{2k+2\delta-1}} - r^{k+\delta}\frac{1-r^{-(i+1)}}{r-1} + 1\Big) - \log_r(2) - i. \end{aligned}$$

With these lemmas, we are ready to prove an upper bound of $1/g(r) < 1.772$ on the randomized competitive ratio of RandomizedScaling.

▶ **Theorem 5.** *IncMax admits a 1.772-competitive randomized algorithm.*

Proof Sketch. In order to find this estimate, we start by fixing $k \in \mathbb{N}$ such that $C \in [r^k, r^{(k+1)})$. Then, depending on the value of ϵ, C is from one of the intervals

$$I_1 = (\tilde{c}_{k-1}, \tilde{t}_{k-1} - 1], \quad I_2 = (\tilde{t}_{k-1} - 1, \tilde{c}_k], \quad I_3 = (\tilde{c}_k, \tilde{t}_k - 1], \quad I_4 = (\tilde{t}_k - 1, \tilde{c}_{k+1}].$$

Yet, not all of these intervals are relevant to calculate the randomized competitive ratio. Depending on where in the interval $[r^k, r^{(k+1)})$ the value C lies, only 2 or 3 of the intervals I_1 to I_4 have a non-zero probability to contain C. Thus, we distinguish the different cases, where C lies in $[r^k, r^{(k+1)})$ and use Lemma 19 to calculate the randomized competitive ratio to be the integral expression in Lemma 20. Applying this lemma gives the desired bound on the randomized competitive ratio. ◀

4.2 Randomized Lower Bound

We turn to proving the lower bound in Theorem 6 for IncMaxSep.

▶ **Theorem 6.** *Every randomized IncMax algorithm has competitive ratio at least* 1.447.

Proof. We fix N to be the number of sets $R_1, \dots, R_N$, leaving $d_1, \dots, d_N$ as parameters to determine the instance; we denote the resulting instance by $I(d_1, \dots, d_N)$. Note that, given a probability distribution $p_1, \dots, p_N$ over the elements $\{1, \dots, N\}$ in addition, Yao's principle [24] yields

$$\inf_{\text{ALG} \in \mathcal{A}_N} \sum_{i=1}^{N} p_i \cdot \frac{i \cdot d_i}{\text{ALG}(I(d_1, \dots, d_N), i)}$$

as a lower bound on the randomized competitive ratio of the problem. Here, $\text{ALG}(I, i)$ denotes the value of the first i elements in the solution produced by ALG on instance I, and $\mathcal{A}_N$ is the set of all deterministic algorithms on instances with N sets $R_1, \dots, R_N$. As observed earlier, we may assume that

$$\mathcal{A}_N := \left\{ \text{ALG}_{c_1,\dots,c_\ell} \;\middle|\; 1 \le c_1 < \dots < c_\ell \le N, \sum_{i=1}^{\ell} c_i \le N \right\},$$

where $\text{ALG}_{c_1,\dots,c_\ell}$ is the algorithm that first includes all elements of R_{c_1} into the solution, then all elements of R_{c_2}, and so on. Once it has added the c_ℓ elements of R_{c_ℓ}, it adds some arbitrary elements from then onwards.

We can formulate the problem of maximizing the lower bound on the competitive ratio as an optimization problem:

$$\begin{aligned}
\max \quad & \rho \\
\text{s.t.} \quad & \rho \le \sum_{i=1}^{N} p_i \cdot \frac{i \cdot d_i}{\text{ALG}(I(d_1, \dots, d_N), i)} && \forall \text{ALG} \in \mathcal{A}_N, \\
& \sum_{i=1}^{N} p_i = 1, \\
& d_1, \dots, d_N \ge 0, \\
& p_1, \dots, p_N \ge 0.
\end{aligned}$$

Note that the expression $\text{ALG}_{c_1,\dots,c_\ell}(I(d_1, \dots, d_N), i)$ can also be written as a function of $c_1, \dots, c_\ell$, $d_1, \dots, d_N$, and i by taking the maximum over all sets from which $\text{ALG}_{c_1,\dots,c_\ell}$ selects elements:

$$\text{ALG}_{c_1,\dots,c_\ell}(I(d_1, \dots, d_N), i) = \max_{1 \le j \le \ell} \left\{ \max \left\{ i - \sum_{1 \le j' < j} c_{j'}, c_j \right\} \cdot d_{c_j} \right\}.$$

A feasible solution to the above optimization problem with $N = 10$ is given by

$$\begin{aligned}
&(\rho; d_1, \dots, d_{10}; p_1, \dots, p_{10}) \\
&= (1.447; 1, 1/2, 1/2, 1/2, 2/5, 1/3, 1/3, 1/3, 1/3, 1/3; 0.132, 0, 0, 0.395, 0, 0, 0, 0, 0, 0.473),
\end{aligned}$$

with objective value 1.447. ◀

References

1 Aaron Bernstein, Yann Disser, Martin Groß, and Sandra Himburg. General bounds for incremental maximization. *Math. Program.*, 191(2):953–979, 2022. doi:10.1007/s10107-020-01576-0.

2 Avrim Blum, Prasad Chalasani, Don Coppersmith, William R. Pulleyblank, Prabhakar Raghavan, and Madhu Sudan. The minimum latency problem. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing (STOC)*, pages 163–171. ACM, 1994. doi:10.1145/195058.195125.

3 Marek Chrobak, Claire Kenyon, John Noga, and Neal E. Young. Incremental medians via online bidding. *Algorithmica*, 50(4):455–478, 2008. doi:10.1007/s00453-007-9005-x.

4 Yann Disser, Max Klimm, Nicole Megow, and Sebastian Stiller. Packing a knapsack of unknown capacity. *SIAM J. Discret. Math.*, 31(3):1477–1497, 2017. doi:10.1137/16M1070049.

5 Yann Disser, Max Klimm, Kevin Schewior, and David Weckbecker. Incremental maximization via continuization. arXiv:2305.01310v1.

6 Yann Disser, Max Klimm, and David Weckbecker. Fractionally subadditive maximization under an incremental knapsack constraint. In *Proceedings of the 19th International Workshop on Approximation and Online Algorithms (WAOA)*, pages 206–223. Springer, 2021. doi:10.1007/978-3-030-92702-8_13.

7 Ryo Fujita, Yusuke Kobayashi, and Kazuhisa Makino. Robust matchings and matroid intersections. *SIAM J. Discret. Math.*, 27(3):1234–1256, 2013. doi:10.1137/100808800.

8 Michel X. Goemans and Jon M. Kleinberg. An improved approximation ratio for the minimum latency problem. *Math. Program.*, 82:111–124, 1998. doi:10.1007/BF01585867.

9 Michel X. Goemans and Francisco Unda. Approximating incremental combinatorial optimization problems. In *Proceedings of the 20th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 6:1–6:14, 2017. doi:10.4230/LIPIcs.APPROX-RANDOM.2017.6.

10 Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985. doi:10.1016/0304-3975(85)90224-5.

11 Jeff Hartline and Alexa Sharp. An incremental model for combinatorial maximization problems. In *Proceedings of the 5th International Workshop on Experimental Algorithms (WEA)*, pages 36–48, 2006. doi:10.1007/11764298_4.

12 Jeff Hartline and Alexa Sharp. Incremental flow. *Networks*, 50(1):77–85, 2007. doi:10.1002/net.20168.

13 Refael Hassin and Shlomi Rubinstein. Robust matchings. *SIAM J. Discret. Math.*, 15(4):530–537, 2002. doi:10.1137/S0895480198332156.

14 Refael Hassin and Danny Segev. Robust subgraphs for trees and paths. *ACM Trans. Algorithms*, 2(2):263–281, 2006. doi:10.1145/1150334.1150341.

15 Naonori Kakimura and Kazuhisa Makino. Robust independence systems. *SIAM J. Discret. Math.*, 27(3):1257–1273, 2013. doi:10.1137/120899480.

16 Ming-Yang Kao, John H Reif, and Stephen R Tate. Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. *Information and Computation*, 131(1):63–79, 1996.

17 Max Klimm and Martin Knaack. Maximizing a submodular function with bounded curvature under an unknown knapsack constraint. In *Proceedings of the 25th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 49:1–49:19, 2022. doi:10.4230/LIPIcs.APPROX/RANDOM.2022.49.

18 Guolong Lin, Chandrashekhar Nagarajan, Rajmohan Rajaraman, and David P. Williamson. A general approach for incremental approximation and hierarchical clustering. *SIAM J. Comput.*, 39(8):3633–3669, 2010. doi:10.1137/070698257.

19 Jannik Matuschke, Martin Skutella, and José A. Soto. Robust randomized matchings. *Math. Oper. Res.*, 43(2):675–692, 2018. doi:10.1287/moor.2017.0878.

20 Nicole Megow and Julián Mestre. Instance-sensitive robustness guarantees for sequencing with unknown packing and covering constraints. In *Proceedings of the 4th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 495–504, 2013. doi:10.1145/2422436.2422490.

21 Julián Mestre. Greedy in approximation algorithms. In *Proceedings of the 14th Annual European Symposium on Algorithms (ESA)*, pages 528–539, 2006. doi:10.1007/11841036_48.

22 Ramgopal R. Mettu and C. Greg Plaxton. The online median problem. *SIAM J. Comput.*, 32(3):816–832, 2003. doi:10.1137/S0097539701383443.

23 C. Greg Plaxton. Approximation algorithms for hierarchical location problems. *J. Comput. Syst. Sci.*, 72(3):425–443, 2006. doi:10.1016/j.jcss.2005.09.004.

24 Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 222–227, 1977. doi:10.1109/SFCS.1977.24.

Local Computation Algorithms for Hypergraph Coloring – Following Beck's Approach

Andrzej Dorobisz
Theoretical Computer Science Department, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland

Jakub Kozik
Theoretical Computer Science Department, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland

Abstract

We investigate local computation algorithms (LCA) for two-coloring of k-uniform hypergraphs. We focus on hypergraph instances that satisfy strengthened assumption of the Lovász Local Lemma of the form $2^{1-\alpha k}(\Delta+1)\mathrm{e}<1$, where Δ is the bound on the maximum edge degree. The main question which arises here is for how large α there exists an LCA that is able to properly color such hypergraphs in polylogarithmic time per query. We describe briefly how upgrading the classical sequential procedure of Beck from 1991 with Moser and Tardos' Resample yields polylogarithmic LCA that works for α up to $1/4$. Then, we present an improved procedure that solves wider range of instances by allowing α up to $1/3$.

2012 ACM Subject Classification Mathematics of computing → Hypergraphs; Mathematics of computing → Probabilistic algorithms; Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Local Computation Algorithms, Hypergraph Coloring, Property B

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.48

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2305.02831 [8]

Funding This work was partially supported by Polish National Science Center (2016/21/B/ST6/02165).

1 Introduction

The problem of hypergraph coloring often serves as a benchmark for various probabilistic techniques. The task is to answer whether there exist (or to explicitly find) *a proper coloring*, that is, such an assignment of colors to the vertices of a hypergraph that no edge contains vertices all of the same color. In fact, the problem of two-coloring[1] of linear hypergraphs was one of the main motivations for introducing Local Lemma in the seminal paper of Erdős and Lovász [9]. It is well known that determining whether the given hypergraph admits proper two-coloring is NP-complete [15]. This result holds even for hypergraphs with all edges of size 3. In this work, we discuss sublinear algorithms for two-coloring of uniform hypergraphs within the framework of Local Computation Algorithms.

We are going to work with k-uniform hypergraphs[2]. For the rest of the paper, n is used to denote the number of vertices of considered uniform hypergraph, m its number of edges, and k size of the edges. We assume that k is fixed (but sufficiently large to avoid technical

[1] In two-coloring problem we can assign to each vertex one of two available colors.
[2] In *k-uniform hypergraph* each edge contains exactly k vertices.

details) and that n tends to infinity. For a fixed hypergraph, we denote by Δ its maximum edge degree. In the instances with which we are going to work, Δ is bounded by a function of k, so in terms of n it is $\mathcal{O}(1)$. This implies that the number of edges m is at most linear in n. We also assume that the considered hypergraphs do not have isolated vertices. Then, we also have $m = \Theta(n)$.

1.1 Local Computation Algorithms

Rubinfeld, Tamir, Vardi and Xie proposed in [21] a general model of sublinear sequential algorithms called Local Computation Algorithms (LCA). The model is intended to capture the situation where some computation has to be performed on a large instance but, at any specific time, only parts of the answer are required. The interaction with a local computation algorithm is organized in the sequence of queries about fragments of a global solution. The algorithm shall answer each consecutive query in sublinear time (wrt the size of the instance), systematically producing a partial answer that is consistent with some global solution. The model allows for randomness, and algorithm may occasionally fail.

For example, for the hypergraph two-coloring problem, the aim of an LCA procedure is to find a proper coloring of a given hypergraph. The algorithm can be queried about any vertex, and in response, it has to assign to the queried vertex one of the two available colors. For any sequence of queries, with high probability, it should be possible to extend the returned partial coloring to a proper one.

Formally, for a fixed problem, a procedure is a (t, s, δ)-*local computation algorithm*, if for any instance of size n and any sequence of queries, it can consistently answer each of them in time $t(n)$ using up to $s(n)$ space for computation memory. The time $t(n)$ has to be sublinear in n, but a polylogarithmic dependence is desirable. The value $\delta(n)$ shall bound the probability of failure for the whole sequence of queries. It is usually demanded to be small. The computation memory, the input, and the source of random bits are all represented as tapes with random access (the last two are not counted in $s(n)$ limit). The computation memory can be preserved between queries. In particular, it can store some partial answers determined in the previous calls. For the precise general definition of the model consult [21].

A procedure is called *query oblivious* if the returned solution does not depend on the order of the queries (i.e. it depends only on the input and the random bits). It usually indicates that the algorithm uses computation memory only to answer the current query and that there is no need to preserve information between queries. It is a desirable property, since it allows to run queries to algorithm in parallel. In a follow-up paper [3], Alon, Rubinfeld, Vardi, and Xie presented generic methods of removing query order dependence and reducing necessary number of random bits in LCA procedures. In the same paper, these techniques were applied to the example procedures (including hypergraph coloring) from [21] converting them to query oblivious LCAs. The improved procedures work not only in polylogarithmic time but also in polylogarithmic space. Mansour, Rubinstein, Vardi, and Xie in [16] improved analysis of this approach.

1.2 Constructive Local Lemma and LCA

The Lovász Local Lemma (LLL) is one of the most important tools in the field of local algorithms. In its basic form, it allows one to non-constructively prove the existence of combinatorial objects omitting a collection of undesirable properties, so-called bad events. A brief introduction to this topic and a summary of various versions of LLL can be found in the recent survey by Faragó [11].

For a fixed k-uniform hypergraph, let $p = 2^{-k}$ denote the probability that, in a uniformly random coloring, a fixed edge is monochromatic in a specific color. A straighforward application of the symmetric version of Local Lemma (see e.g., [11]) proves that the condition $2p\ (\Delta + 1)\ \mathrm{e} < 1$, is sufficient for a hypergraph with the maximum edge degree Δ, to be two-colorable.

For many years, Local Lemma resisted attempts to make it efficiently algorithmic. The first breakthrough came in 1991, when Beck [5], working on the example of hypergraph two-coloring, showed a method of converting some of LLL existence proofs into polynomial-time algorithmic procedures. However, in order to achieve that, the assumptions of Local Lemma had to be strengthened and took form

$$2\ p^{\alpha}\ (\Delta + 1)\ \mathrm{e} < 1. \tag{1}$$

For $\alpha = 1$ the inequality reduces to the standard assumption. The above inequality constraints Δ, and the constraint becomes more restrictive as α gets smaller. The original proof of Beck worked for $\alpha < 1/48$. From that time, a lot of effort has been put into studying applications to specific problems and pushing α forward, as close as possible to standard LLL criterion [2, 18, 7, 22, 19].

The next breakthrough was made by Moser in 2009. In cooperation with Tardos, Moser's ideas have been recasted in [20] into general constructive formulation of the lemma. They showed that, assuming so called variable setting of LLL, a natural randomized procedure called Resample[3] quickly finds an evaluation of involved random variables for which none of the bad events hold. They also proved that, in typical cases, the expected running time of the procedure is linear in the size of the instance. For the problem of two-coloring of k-uniform hypergraphs, the total expected number of resamplings is bounded by m/Δ (see Theorem 7 in [11]).

Adjusting constructive LLL to LCA model remains one of the most challenging problems in the area. It turns out, however, that previous results on algorithmization of Local Lemma can be adapted in the natural way. In fact, the first LCA algorithm for the hypergraph coloring from [21], is built on the variant of Beck's algorithm that is described in the book by Alon and Spencer [4]. That version works for $\alpha < 1/11$, and runs in polylogarithmic time per query. Later refinements focused on optimizing space and time requirements ([3], [16]), however, for polylogarithmic LCAs the bound on α has not been improved. In a recent work, Achlioptas, Gouleakis, and Iliopoulos [1] showed how to adjust Resample to LCA model. They did not manage, however, to obtain a polylogarithmic time. Their version answers queries in time $t(n) = n^{\beta(\alpha)}$. They establish some trade-off between the bound on α and the time needed to answer a query. In particular, when α approaches $1/2$ then $\beta(\alpha)$ tends to 1, which results in a very weak bound on the running time per query.

1.3 Main result

Our research focuses on the following general question in the area of local constructive versions of the Lovász Local Lemma: up to what value of α there exists a polylogarithmic LCA for the problem of two-coloring of k-uniform hypergraphs satisfying condition $2(\Delta + 1)\mathrm{e} < 2^{\alpha k}$. We prove the following theorem:

[3] As long as some bad events are violated, the procedure picks any such event and resamples all variables on which that event depends.

▶ **Theorem 1** (main result). *For every $\alpha < 1/3$ and all large enough k, there exists a local computation algorithm that, in polylogarithmic time per query, with probability $1 - O(1/n)$ solves the problem of two-coloring for k-uniform hypergraphs with maximum edge degree Δ, that satisfies $2\mathrm{e}(\Delta + 1) < 2^{\alpha k}$.*

Within the notation of [21] we present $(\mathrm{polylog}(n), \mathcal{O}(n), \mathcal{O}(1/n))$-local computation algorithm that properly colors hypergraphs that satisfy the above assumption. Our algorithm is not query oblivious. Moreover, typical methods of eliminating the dependence on the order of queried vertices do not seem to be applicable without sacrificing constant α. Consult the full version of this paper [8] for the complete proof of the theorem.

For comparison, Alon et al. [3] after Rubinfeld et al. [21] present a query oblivious $(\mathrm{polylog}(n), \mathrm{polylog}(n), \mathcal{O}(1/n))$-local computation algorithm working for hypergraphs satisfying

$$\begin{aligned} 16\,\Delta(\Delta-1)^3(\Delta+1) &< 2^{k_1}, \\ 16\,\Delta(\Delta-1)^3(\Delta+1) &< 2^{k_2}, \\ 2e(\Delta+1) &< 2^{k_3}, \end{aligned} \tag{2}$$

where k_1, k_2 and k_3 are positive integers such that $k = k_1 + k_2 + k_3$. These assumptions correspond to $\alpha < 1/11$.

The analysis of the LCA procedure from [3] guarantees only that the running time is of the order $\mathcal{O}\Big(\log^{\Delta}(n)\Big)$. Mansour et al. in [16] focus on improving time and space bounds within polylogarithmic class, removing the dependency on the maximal edge degree from the exponent. They obtain an LCA working in $\mathcal{O}\big(\log^4(n)\big)$ time and space, assuming that $k \geq 16\log(\Delta) + 19$, so it requires even stronger bound on α.

1.4 LOCAL distributed algorithms

The model of Local Computation Algorithms is related to the classical model of local distributed computations by Linial [14] (called LOCAL). For comparison of these two models, see work of Even, Medina, and Ron [10]. Chang and Pettie observed recently in [6] that within LOCAL model, the general problem of solving Local Lemma instances with a dependency graph of bounded degree is in some sense complete for a large class of problems (these are the problems which can be solved in sublogarithmic number of rounds). They also conjectured that for sufficiently strengthened condition of Local Lemma (like taking small enough α in (1)) there exists a distributed LOCAL algorithm that solves the problem in $\mathcal{O}(\log\log n)$ rounds. The straightforward simulation of such an algorithm within LCA framework would yield a procedure that, at least for fixed maximum degree, answers queries in polylogarithmic time.

Recently, progress towards this conjecture has been made by Fischer and Ghaffari [12], who proved that there exists an algorithm for Local Lemma instances that works in $2^{\mathcal{O}\left(\sqrt{\log\log n}\right)}$ rounds. The influence of the degree of underlying dependency graph on running time has been later improved by Ghaffari, Harris and Kuhn in [13]. In particular, for sufficiently constrained problem of hypergraph two-coloring, that result allows one to obtain an LCA procedure that answers queries in sublinear time. The time, however, would be superpolylogarithmic. Moreover, the necessary strengthening of Local Lemma assumptions appears to be much stronger than the one required to apply the result of Rubinfeld et al. [21].

The possibility of simulation of LOCAL algorithms within LCA model implies that if Chang and Pettie conjecture holds, then any problem satisfying sufficiently strengthened LLL conditions can be solved in LCA model in polylogarithmic time per query. We can therefore

formulate a weaker conjecture that for some α every such α-strengthened problem can be solved in LCA in polylogarithmic time per query. For the specific problem of hypergraph coloring, this property is known to hold. We can, however, ask what is the maximum such α for a fixed problem. That is precisely the general problem stated at the beginning of Section 1.3. It is interesting to note that our algorithms make essential use of the sequential nature of LCA. For that reason, they cannot be translated to $\mathcal{O}(\log \log n)$ LOCAL algorithms. This also illustrates an important difference between the models.

2 Main techniques and ideas of the proof

The algorithmic procedure of Beck [5] is divided into two phases. In the first one, which we call *the shattering phase*, it builds a random partial coloring that guarantees that a fraction of all edges are already properly colored. Moreover, the edges which are not yet taken care of have sufficiently many non-colored vertices to make sure that the partial coloring can be completed to a proper one. They also form connected components of logarithmic sizes which can be colored independently. Then, in the second phase, which we call *the final coloring phase*, an exhaustive search is used to complete the coloring of each component. This results in a sequential procedure with polynomial running time. In order to reduce the running time to almost linear, the shattering phase can be applied twice. Then, the final components w.h.p. are of size $\mathcal{O}(\log \log(n))$. The polylogarithmic LCA procedure for hypergraph coloring from [21] followed that approach and simulates locally two shattering phases and an exhaustive search when answering a single query. Division into these three phases is directly reflected in the conditions (2) required by the procedure.

While it is not known whether it is possible to design an LCA algorithm based solely on Resample, combining it with previous local algorithms brings significant improvements. It turns out that, within polylogarithmic time, after only one shattering phase, the coloring can be completed with the use of Resample. This simple modification, with slightly improved analysis, is sufficient to derive Theorem 1 for $\alpha \leq 1/4$. This is our first contribution. That procedure provides a reference point for explaining the intuitions and motivations that underlie the further improvements that we derive. In particular, we define a notion of *component-hypergraph* that allows for a more fine-grained analysis of the components of the residual hypergraph. For that reason, we present our base algorithm in detail in Section 3.

The first modification that we make in order to improve the base algorithm is that within the shattering phase we sample colors for all vertices. Then, for some vertices, the color is final, and for others, it is allowed to change the assigned color in the final coloring phase. Coloring all the vertices during the first phase somehow blurs the border between the shattering and final coloring phases. Its main purpose is to enable a more refined partition of the residual hypergraph into independent fragments. It also allows to determine some components of the residual hypergraph for which no recoloring would be necessary. This corresponds to a situation in which the first sampled colors in Resample happen to define a proper coloring. Altogether, we managed to significantly reduce the pessimistic size of the independent fragments colored in the final coloring phase, which enables further relaxation of the necessary conditions on α to $\alpha < 1/3$. The improved procedure is described in Section 4.

In order to analyze the procedures, we employ a common technique of associating some tree-like *witness structures* with components that require recoloring. Every such structure describes a collection of events associated with some edges of the hypergraph. All these events are determined by the colors assigned in the shattering phase. For the base algorithm, these structures are quite typical. However, in order to achieve the better bound on α, we

developed more sophisticated structures that are capable of tracking different kinds of events, which can also depend on the colors that are allowed to be recolored. Different kinds of events come with different bounds on probability. An important aspect of the analysis concerns amortization of different kinds of events within a single structure. The construction of these structures is our main technical contribution. Its detailed description can be found in the full version of this paper [8].

We finally note that, while our methods are not general enough to work for all instances satisfying the strengthened assumptions of LLL, they can be applied to a number of problems similar to hypergraph coloring, like, e.g. k-SAT.

3 Establishing base result

In this section we show how the Beck's algorithm can be combined with Resample to construct a local computation algorithm that works in polylogarithmic time per query for α up to $1/4$. In other words, we prove Theorem 1 under the stronger assumption that $\alpha \leq 1/4$. To keep the exposition simple, we first present a global randomized algorithm. Then, we comment on how to adapt this procedure to LCA model. The analysis of the procedure can be found in the full version of this paper [8].

Let $H = (V, E)$ be a hypergraph that satisfies the assumptions of Theorem 1 for a fixed $\alpha \leq 1/4$. For technical convenience, we assume that αk is an integer[4]. By assigning a random color, we mean choosing uniformly one of the two available colors. For a set of edges S, by $V(S)$ we mean all vertices covered by the edges from S. For an edge f, $N(f)$ denotes the set of edges intersecting f. We use a naming convention that is similar to other works on the subject – in particular, our view of Beck's algorithm is influenced by its descriptions by Alon and Spencer [4] and Molloy and Reed [17], as well as LCA realization given in [21].

3.1 Global coloring procedure

The algorithm starts with choosing an arbitrary order of vertices. Then, it proceeds in two phases: *the shattering phase* and *the final coloring phase*. The shattering phase colors some vertices of the input hypergraph and then splits the edges of the hypergraph that are not properly colored yet into *final components* – subhypergraphs that can be colored independently. The final coloring phase completes the coloring by considering the final components separately, one by one.

3.1.1 The shattering phase

The procedure processes vertices sequentially according to the fixed ordering. For every vertex, it either assigns a random color to the vertex or leave it non-colored in case it belongs to a *bad* edge. An edge is called *bad* if it contains $(1 - \alpha)k$ colored vertices and is still not colored properly (that is, all these vertices have the same color). Once an edge becomes bad, no more vertices from that edge will be colored – such vertices are called *troubled*. Vertices with assigned colors are called *accepted*.

Upon completion of the shattering phase, there are three types of edges:

- *safe edges* – properly colored by the accepted vertices,
- *bad edges* – containing exactly $(1 - \alpha)k$ accepted vertices, all of the same color,
- *unsafe edges* – containing fewer than $(1 - \alpha)k$ accepted vertices, all of the same color.

[4] In fact, for the given k it is only reasonable to take α in the form of t/k, where t is an integer $2 \leq t \leq k$.

Observe that in the resulting (partial) coloring, every edge that is not colored properly has at least αk troubled vertices, which will be colored in the next phase. Note also that it might happen that some unsafe edge has no colored vertices at all.

The colors of accepted vertices are not going to be changed, so the safe edges are already taken care of. Therefore, we focus on bad and unsafe edges. Let E_{bad} denote the set of all bad edges. Consider hypergraph $(V(E_{bad}), E_{bad})$. It is naturally decomposed into connected components.

▶ **Definition 2.** *Every component of the hypergraph* $(V(E_{bad}), E_{bad})$ *is called a* bad-component.

Note that every troubled vertex belongs to some bad-component. On top of them we build an abstract structure to express dependencies between bad-components through unsafe edges.

▶ **Definition 3.** *A* component-hypergraph *is constructed as follows: its vertices are bad-components of* H *and for every unsafe edge* f *intersecting more than one bad-component, an edge that contains all bad-components intersected by* f *is added to it.*

For each connected component of the component-hypergraph (that is, a maximal set of bad-components that is connected in the component-hypergraph) we construct a *final component* by taking the union of those bad-components (hence a final component is a subhypergraph of H). The shattering phase is *successful* if each final component contains at most $2(\Delta+1)\log(m)$ bad edges. If this is not the case, the procedure declares a failure. It turns out that this is very unlikely to happen.

3.1.2 The final coloring phase

For each final component $\mathcal{C}$ determined during the shattering phase, we add to $\mathcal{C}$ all unsafe edges intersecting it, and then, we restrict $\mathcal{C}$ to troubled vertices[5]. We obtain a hypergraph $\mathcal{C}'$ containing at most $2(\Delta+1)^2\log(m)$ edges, and each of them has at least αk vertices. The maximum edge degree in $\mathcal{C}'$ cannot be larger than Δ, which is the maximum edge degree in H. Since $2e(\Delta+1) < 2^{\alpha k}$ (by the assumptions of Theorem 1), Lovász Local Lemma ensures that $\mathcal{C}'$ is two-colorable. Hence, by the theorem of Moser and Tardos Resample finds a proper coloring of it using on average $|E(\mathcal{C}')|/\Delta$ resamplings (see Theorem 7 in [11]).

When the final coloring phase is over, all final components are properly colored. Since each bad or unsafe edge is dealt within some final component, and each safe edge was properly colored during the shattering phase, it is now guaranteed that the constructed coloring is proper for the whole H.

3.2 LCA realization

We employ quite standard techniques to obtain an LCA realization of the described algorithm. We articulate it below to provide a context for the description of our main algorithm. An important property of the described procedure is that the ordering of vertices does not have to be fixed a priori. In fact it can be even chosen in an on-line manner by an adversary. Following [21], we are going to exploit the freedom of choice of ordering. The LCA version of the algorithm is going to simulate the global version run with a specific ordering. That ordering is constructed dynamically during the evaluation and is driven by the queries. Apart

[5] Restriction of $H=(V,E)$ to $V'\subseteq V$ is defined as $H'=(V',\{e\cap V' |\ e\in E, e\cap V'\neq\emptyset\})$.

from some minor adjustment (resulting from adaptation to LCA model) when the algorithm is queried about vertex v, it performs all the work of the standard algorithm needed to assign a final color to v. The LCA version is presented in Listings 1, 2, 3, and 4. All colors assigned during work of the algorithm are stored in the computation memory (which is preserved between queries). For convenience, we also store there the status of each vertex – *uncolored*, *accepted* or *troubled*. Initially all vertices are uncolored.

Algorithm 1 LCA for uniform hypergraph coloring – main function.

```
1  Procedure QUERY(v - vertex):
2      if v is uncolored then
3          if all edges containing v are not bad then
4              assign a random color to v and mark it as accepted    // shattering
5          else mark v as troubled
6      if v is troubled then
7          C_v ← BUILD_FINAL_COMPONENT(v)                            // shattering
8          COLOR_FINAL_COMPONENT(C_v)                              // final coloring
9      return color assigned to v
```

3.2.1 query

When a vertex v has been already marked as accepted, its color is immediately returned. If it has not been processed before, the algorithm checks whether v belongs to any bad edge (that requires inspecting the current statuses of all the edges that contain v). If not, a random color is assigned to v, the vertex is marked as accepted, and the procedure returns the assigned color. On the other hand, when v belongs to a bad edge, it is marked as troubled. The algorithm then determines the final component containing v in procedure BUILD_FINAL_COMPONENT. These steps can be viewed as the shattering phase. Afterwards, the final coloring phase is performed for the final component in procedure COLOR_FINAL_COMPONENT.

Algorithm 2 Building the final component for v that belongs to some bad edge.

```
1  Procedure BUILD_FINAL_COMPONENT(v - troubled vertex):
2      B ← ∅                  // initialize set of bad edges of the component
3      U ← ∅                     // initialize set of unsafe edges to process
4      e ← any bad edge containing v
5      mark e as explored and run EXPAND_BAD_COMPONENT(e, B, U)
6      // process surrounding unsafe edges
7      while U is not empty do
8          f ← next edge from U (remove it from U)
9          EXPAND_VIA_UNSAFE(f, B, U)
10     // return hypergraph built on set of bad edges
11     return C = (V(B), B)
```

3.2.2 build_final_component

This procedure builds the set B of bad edges of the final component of v, exploring the line graph of H[6]. It uses a temporary flag *explored* to mark visited edges (this flag is not preserved between queries). The construction starts from a bad edge containing troubled vertex v and expands it to a bad-component. Then, as long as possible, set B is extended by edges of neighboring bad-components, which can be reached through unsafe edges adjacent to B. If at some point the number of bad edges in B exceeds the prescribed bound $2(\Delta+1)\log(m)$, then the procedure declares a failure (note that it cannot be restarted since LCA model does not allow to change colors returned for previous queries). Construction of the final component is done when there are no more bad edges to add. Then, the hypergraph $\mathcal{C} = (V(B), B)$ built on the collected bad edges is returned.

The expansion of bad-components is done within subprocedure `EXPAND_BAD_COMPONENT`. It starts from the given bad edge and explores the line graph by inspecting the adjacent edges. For each adjacent edge, its type (safe, unsafe, or bad) is determined using `DETERMINE_EDGE_STATUS`. Determining status of an edge may require processing some uncolored vertices of that edge. For each of them, the procedure check whether it is troubled. If it is not, a random color is assigned to the vertex and the vertex is marked as accepted.

Algorithm 3 Subprocedures for the final component construction.

```
1  Procedure EXPAND_BAD_COMPONENT(e - bad edge, B - bad edges, U - unsafe edges):
2      Q ← {e}                          // initialize set of bad edges to process
3      while Q is not empty do
4          f ← next edge from Q (remove it from Q)
5          add f to B and if |B| > 2(Δ + 1) log(m) then FAIL
6          for g ∈ N(f) which are not explored do
7              mark g as explored and DETERMINE_EDGE_STATUS(g)
8              if g is bad then add g to Q
9              if g is unsafe then add g to U
10
11 Procedure EXPAND_VIA_UNSAFE(f - unsafe edge, B - bad edges, U - unsafe edges):
12     for g ∈ N(f) which are not explored do
13         DETERMINE_EDGE_STATUS(g)
14         if g is bad then
15             mark g as explored and run EXPAND_BAD_COMPONENT(g, B, U)
16
17 Procedure DETERMINE_EDGE_STATUS(g - edge):
18     for each w in g that is uncolored unless g becomes safe do
19         if some edge containing w (including g) is bad then mark w as troubled
20         else assign a random color to w and mark it as accepted
21     count accepted vertices and check their colors to determine status of g
```

During the expansion through unsafe edges we keep a set U of not processed unsafe edges that intersects any edge of B. As long as U is not empty, we pick any unsafe f from U and process it by `EXPAND_VIA_UNSAFE`. Here we determine the statuses of all edges

[6] The line graph $L(H)$ is the graph built on $E(H)$ in which two distinct vertices (representing edges of H) are adjacent if the corresponding edges intersect.

adjacent to f and if we encounter a bad edge which is not in B, then we add it and expand a bad-component containing it. For technical convenience, during bad-component expansion we collect non-explored adjacent unsafe edges and add them to U.

Algorithm 4 Finding coloring inside the final component.

```
1  Procedure COLOR_FINAL_COMPONENT(C - hypergraph):
2      add to C all unsafe edges intersecting C
3      C' ← restriction of C to troubled vertices
4      t_e ← |E(C')|/Δ                    // expected time of one RESAMPLE trial
5      for trial = 1 to 2 log(m) do
6          // RESAMPLE with limited number of steps
7          assign random colors to V(C')
8          for step = 1 to 2t_e do
9              if there is monochromatic f ∈ E(C') then
10                 assign new random colors to all vertices of f
11             else
12                 // C' is properly colored
13                 mark all vertices of C' as accepted and return
14     FAIL
```

3.2.3 color_final_component

Final component $\mathcal{C}$ is extended with unsafe edges that intersect it. Then it is restricted to the set of its troubled vertices. The resulting hypergraph is denoted by $\mathcal{C}'$. The algorithm tries to find a proper coloring of $\mathcal{C}'$ using Resample procedure. To ensure polylogarithmic time, it is run only for the limited number of resampling steps. To decrease the probability of a failure, the procedure may be restarted a few times. When a proper coloring is found, each vertex of $\mathcal{C}'$ is marked as accepted. From now on, all edges of $\mathcal{C}$ are treated as safe. However, if all trials were unsuccessful, the procedure declares a failure.

4 Main result – algorithm

We show how to improve the base procedure described in the previous section to obtain an algorithm that can be used to prove Theorem 1, that is, an algorithm that works in polylogarithmic time per query on input hypergraphs that satisfy strengthened LLL condition (1) for $\alpha < 1/3$. Actually, our procedure can be used to find a proper coloring also for instances that satisfy that condition with any $\alpha \in (0, 1)$, but the running time is not guaranteed for $\alpha \geq 1/3$. We start with introducing the main ideas behind algorithm improvement and describe its global version. Then, we discuss how to adapt it to the model of the local computation algorithms, and finally we present a description of the LCA procedure. The analysis of the algorithm can be found in the full version of this paper [8].

4.1 A general idea

It is a common approach in randomized coloring algorithms to start from an initial random coloring and then make some correction to convert it to a proper one (like in Resample [20] or in Alon's parallel algorithm [2]). This is not the case of Beck's procedure, in which a

proper coloring is constructed incrementally, but coloring of some vertices (those marked as troubled) is postponed to the later phase. Our approach lies somewhere in between. We generally try to follow the latter one, but we sample colors for the troubled vertices already in the shattering phase. Such colors are considered as *proposed*, and we reserve the possibility of changing them in the final coloring phase. We use the information about the proposed colors to shrink the area that will be processed in the final coloring phase. In particular, if we look at the colors proposed for troubled vertices, then only those final components that contain a monochromatic edge require recoloring. Moreover, if we carefully track dependencies between bad-components (see Definition 2), it is also possible to decrease the sizes of the final components. We explain this idea in more detail in the following subsections.

4.1.1 Activation of bad-components

Imagine that all the vertices were colored in the shattering phase and we want to determine the final components. We look at the component-hypergraph (see Definition 3) and have to decide which of the bad-components should be recolored. We start from bad-components that are intersected by monochromatic edges - we mark them as *initially active* and treat them as seeds of final components. The remaining ones are currently *inactive*. Our intention is to recolor only active components in the final coloring phase. Note that it might not be sufficient to alter the coloring in a way that makes initially active components properly colored, because after their recoloring, it is possible that some unsafe edge which get both colors in the shattering phase becomes monochromatic. That is why the activation has to be propagated. We use the following *propagation rule*:

- let A_t be the set of troubled vertices that are covered by active bad-components, and f be an unsafe edge that intersects A_t; if $f \setminus A_t$ is monochromatic, then all inactive bad-components that intersect f become active and all bad-components that intersect f are merged into one (eventually final) component.

The above propagation rule is applied as long as possible. When it stops, it is guaranteed that all monochromatic edges are inside active components and all unsafe and bad edges outside of active components are properly colored by the vertices that are outside of active bad-components. In particular, we can accept all the colors proposed for inactive vertices.

4.1.2 Edge trimming

We employ an additional technique, which can further reduce the area of the final components. Observe that, in order to guarantee two-colorability of the final components, it is enough to ensure that each edge has at least αk vertices to recolor inside one final component. It means that if some active component already contains αk troubled vertices of some edge, then it is not necessary to propagate activation through that edge. Thus, we can improve the propagation rule in the following way. Consider an unsafe edge f for which $f \setminus A_t$ is monochromatic (recall that A_t denotes the set of currently active troubled vertices). If some active component contains at least αk troubled vertices of f, then f is trimmed to that active component. Otherwise, all bad-components intersected by f are activated and merged into one component (as described in the previous section).

We point out that the direct inspiration for this technique came from the work of Czumaj and Scheideler [7] in which the edge trimming is actively used during the construction of the area to be recolored. One of the consequences of using it is that the shapes of the final components depend on the specific order in which activation is propagated.

4.2 Global coloring procedure

Similarly to the base algorithm from Section 3.1, the improved procedure performs the shattering phase and then the final coloring phase. The former is modified according to the ideas described in the previous subsection. In particular, each vertex gets a color but we use the notions of *proposed* and *accepted* colors to distinguish colors that can be changed. The latter phase is almost the same. Pseudocode of the whole procedure can be found in Listing 5 in Appendix A.

4.2.1 The shattering phase

The procedure processes the vertices in a fixed order. For each vertex, it marks it as *accepted* or *troubled*, and then chooses a random color for it. A vertex is accepted if, at the time of processing, it does not belong to any of the bad edges. Otherwise, it is troubled. An edge becomes *bad* when its set of accepted vertices reaches size $(1-\alpha)k$ and is still monochromatic.

After processing all the vertices, *safe* and *unsafe* edges are determined in the same way as in the base algorithm. Additionally, by a *monochromatic* edge, we mean an edge for which all its vertices (accepted and troubled) have the same color. The colors of the accepted vertices are called *accepted colors*. The colors of the troubled vertices are called *proposed colors*. By accepting a color assigned to a vertex, we mean changing its status to accepted.

The next step involves determining the final components. We work with the component-hypergraph. We are going to mark some bad-components and unsafe edges as *active*. By an *active component*, we mean a maximal set of active bad-components which is connected in the component-hypergraph via active unsafe edges. We start with marking as active all monochromatic unsafe edges and all bad-components that are intersected by any (bad or unsafe) monochromatic edge. Let A_t denote the set of troubled vertices that are currently covered by active bad-components. Then, as long as there exists an inactive unsafe edge f satisfying the following conditions:

- f is monochromatic outside the active troubled area (i.e., $f \setminus A_t$ is monochromatic), and
- each active component contains less than αk troubled vertices of f,

we activate f and activate all bad-components intersected by f. When this propagation rule can no longer be applied, we accept the colors of all the troubled vertices from inactive bad-components. At that time, each active component determines a final component as the union of its bad-components. Just like in the base algorithm, the shattering phase is *successful* if each final component contains at most $2(\Delta+1)\log(m)$ bad edges. Otherwise, the procedure declares a failure.

4.2.2 The final coloring phase

We implement one modification at the beginning of the final coloring phase. For each final component $\mathcal{C}$, we add to $\mathcal{C}$ not all unsafe edges intersecting it, but only those that have at least αk troubled vertices in $V(\mathcal{C})$. Then, we proceed exactly as in the base algorithm: we restrict $\mathcal{C}$ to the troubled vertices and apply Resample.

4.3 Ideas behind LCA realization

In the base case, the conversion of the global algorithm to LCA is straightforward. In fact, the LCA version determines the same area to recolor (assuming that both versions process the vertices in the same order). For the improved algorithm described in the previous subsection, conversion to LCA is more complex and alters the behavior of the algorithm. The main

difficulty is that for a bad-component alone that is not initially active, it is not easy to quickly decide whether it is going to be activated or not. There might exist a long chain of activation leading to an activation of the considered bad-component, and we do not know in which direction to search for the sources of this eventual activation. Moreover, even if we find out that it will be activated, it is not obvious what the shape of the final component containing it will be, since it requires performing activation propagation and determining activation statuses of neighboring bad-components as well. To address these problems, when a troubled vertex of some bad-component is queried, we focus on finding an area containing that vertex that can be recolored independently from the remaining part of the input hypergraph. It means that from the beginning of the procedure the component of that vertex is treated as active and we allow trimming unsafe edges to that component. Moreover, we use additional techniques described below to limit the expansion of the processed area in a single query.

4.3.1 Trimming to bad-component

We extend edge trimming to the case when an unsafe edge f has at least αk troubled vertices in some bad-component S, and the set of those vertices together with the accepted vertices of f is not monochromatic. In such a case, f can be trimmed by removing from it the troubled vertices that do not belong to S. Note that we do not check here whether S is active or not. The idea behind this step is that from now on S is responsible for the proper coloring of f. If at some point, the colors of the vertices of S get accepted without any resamplings, then f will be obviously colored properly. Otherwise, if S becomes active, then f will be trimmed anyway, and S has enough troubled vertices of f to not break two-colorability of S.

4.3.2 Activation exclusion

The necessary condition for an inactive bad-component S to be activated is that there is an unsafe edge f whose accepted vertices and troubled vertices in $f \cap V(S)$ are of the same color. When there is no such edge or all such edges were trimmed to other components, then S cannot be activated. Therefore if it is not initially active, it stays inactive. In such a case, we can accept all the proposed colors for the vertices of S. As a result, some unsafe edges become properly colored, and we can treat them as safe. This, in turn, may enable proving that neighboring bad-components will also not be activated. The same reasoning can be applied to a set C of bad-components. If none of the bad-components in C is initially active and there are no unsafe edges intersecting some bad-component outside C that may activate bad-component from C, then we can conclude that all bad-components in C remain inactive.

4.3.3 Conditional expansion

The idea described in the previous subsection can be used for a bad-component to perform some kind of search for a potential reason of activation. If S_1 is not initially active, we inspect unsafe edges that may cause the activation of S_1. We can select any such f, and ask whether other bad-component S_2 intersected by f may become active. We can continue that procedure as long as there is a risk of activating any S_i from the group of bad-components visited so far. In the end, we either find some initially active component or we prove that all the considered bad-components cannot be activated. It turns out that, if we do not follow the edges that can be trimmed with the trimming to bad-component technique, then the processed area during such a search is unlikely to be large.

The possibility of finding an initially active bad-component can be used in expansion of the component to extend it by a neighboring area. For a selected bad-component adjacent to the currently constructed eventually final component, we launch a search and either we find

some monochromatic edge (initially active component) and extend the component with the whole searched area, or convince ourselves that this area cannot be activated. In the latter case we can simply accept the proposed colors in that area. In the former we can perform the expansion because the occurrence of a monochromatic edge, as an unlikely event, in a sense amortizes the expansion of the component. In fact, we can stop the search procedure not only when we find a monochromatic edge but also in a less restrictive case when we find an unsafe edge intersecting at least two disjoint bad edges outside the search area. This possibility follows from the technical details of the analysis.

4.4 LCA procedure

We describe the improved LCA procedure in reference to the base algorithm presented in Section 3.2. As previously, the ordering of the vertices is constructed dynamically and is driven by the queries and the work of the algorithm. For a set of edges S, by $V_t(S)$ we mean all troubled vertices in $V(S)$. For an edge f, we denote by $f|_t$ the set of troubled vertices of f, and by $f|_a$ the set of accepted vertices of f.

4.4.1 query

The main procedure is almost identical to its counterpart in the base algorithm (Listing 1). The only difference is that when processing a vertex v of a bad edge, it is not only marked as troubled, but also a random color is assigned to v.

4.4.2 build_final_component

This procedure is the heart of the algorithm and is substantially more complex than its analogue in the base version. It is presented in Listings 6 and 7 available in Appendix A. It also makes use of subprocedures defined earlier (see Listing 3), with one modification in `DETERMINE_EDGE_STATUS` – once a vertex w is marked as troubled, a random color is also assigned to w. As previously, the procedure works on the line graph of H and grows a set B of bad edges that will be converted to a final component at the end of the procedure. It always starts from the bad-component containing the queried vertex v, and expands it by neighbor bad-components via unsafe edges. The main change is that in the base algorithm each unsafe edge causes expansion of the component, here unsafe edges are processed more carefully. Throughout the procedure we make sure that the size of B does not exceed $2(\Delta+1)\log(m)$ bound on number of edges – if that happens, the procedure stops and declares a failure.

Let U be the set of not processed unsafe edges intersecting $V(B)$. If some edge can be trimmed to $V(B)$, it can be safely removed from U. Thus, we may assume that each f in U has fewer than αk troubled vertices in $V(B)$. Since every unsafe edge has more than αk troubled vertices, each f from U has to intersect at least one bad-component outside $V(B)$. The procedure applies the following *extension rules* as long as possible:

- (r1) if there exists f in U that intersects at least two disjoint bad edges outside B, or
- (r2) if there exists f in U for which all the vertices of f outside of $V_t(B)$ are monochromatic,

then B is extended with all bad edges from the bad-components intersected by f;

- (r3) if there are no edges in U that meet the conditions (r1) or (r2), but there exists f in U that has fewer than αk troubled vertices outside $V(B)$,

then call `EXPAND_OR_ACCEPT` procedure (described in the following subsection) for f, which implements the conditional expansion technique, and extend B with the returned set of bad edges (which may happen to be empty).

Note that, when there are no edges that meet conditions (r1) or (r2), then for any remaining f from U it is guaranteed that f intersects exactly one bad-component outside $V(B)$ and $f \setminus V_t(B)$ is not monochromatic. If such f does not satisfy condition (r3), it has at least αk troubled vertices in that external bad-component, so it can be trimmed to it (according to trimming to bad-component technique). Thus, f can be removed from U.

After each extension rule, the processed edge is removed from U. On the other hand, when B is extended, new unsafe edges may be added to U, but we remove those that can now be trimmed to $V(B)$. Since edges which do not fulfill any of the extension rules are also removed from U, finally U becomes empty and the procedure stops. At this point, B is a set of bad edges which are surrounded only by safe and trimmed unsafe edges.

4.4.3 expand_or_accept

This procedure is an implementation of the conditional expansion technique, through a given unsafe edge e. Similarly to BUILD_FINAL_COMPONENT, it grows a set A of bad edges, which we call a *search area*, and makes sure that its size does not exceed $2(\Delta+1)\log(m)$ bound (if that happens, the whole algorithm stops and declares a failure). Initially, A is empty. Then it becomes expanded by bad-components which may lead to initially active bad-component, starting from the not explored bad-component intersected by e. The expansion naturally stops when there are no more candidate bad-components. The procedure, however, can also stop earlier in case when some monochromatic edge or unsafe edge intersecting two disjoint not explored bad edges is found.

Let Q be the set of unsafe edges to be processed (initially it is empty). Let C be the set of bad edges of the currently expanded bad-component. Let U_C denote the set of unsafe edges intersecting $V(C)$ but not adjacent to the edges of B and A (these are simply those unsafe edges adjacent to the edges in C that were not explored before expansion of C). The procedure extends A with all edges from C, and then looks for the following *amortizing configuration*:

- (e1) if C contains monochromatic edge f

then the procedure stops and returns set A;

- (e2) if U_C contains a monochromatic edge f, or
- (e3) if U_C contains an edge f, which intersects at least two disjoint bad edges outside C,

then first set A is extended with all the bad edges of the bad-components intersected by f, and then the procedure stops and returns A.

When no such configuration is found, all unsafe edges in U_C are not monochromatic and, moreover, each intersects at most one bad-component outside A. We focus on the edges from U_C that can cause an activation of C – these are the edges whose troubled vertices in $V(C)$ together with accepted vertices are monochromatic. Each such an edge f has to intersect exactly one external bad-component and troubled vertices of that component together with $f|_a$ ensure a proper coloring of f. If there are at least αk troubled vertices of f in that external bad-component, f can be trimmed to it (according to the technique of trimming to bad-component). That is why we add to Q only those edges from U_C that may cause activation of C and have fewer than αk troubled vertices outside of $V(C)$.

When processing of C is finished, we pick any edge from Q (the set of unsafe edges to be processed) and repeat the above steps for the external bad-component intersected by the selected edge. It may happen that this component has already been added to A, in a such case the procedure continues picking edges from Q. When the procedure finishes without encountering amortizing configuration, there are no monochromatic edges in A and all unsafe edges intersecting $V(A)$ are either properly colored by the colors of the accepted vertices and

the vertices from $V_t(A)$, or are trimmed to bad-components outside it. Thus, an activation of whole A is excluded. Then we mark all vertices in $V_t(A)$ as accepted and treat edges properly colored by their colors as safe. In that case, the procedure returns the empty set.

Note that during this procedure, we do not apply edge trimming to $V(A)$ when it covers at least αk troubled vertices of some unsafe edge, since it can result in a false activation (in case the edge is monochromatic inside $V(A)$). We also ignore all unsafe edges intersecting $V(B)$ (they were explored before call to `EXPAND_OR_ACCEPT`) since, due to not satisfying (r1) and (r2) they cannot be used in an amortizing configuration or cause an activation (it is guaranteed that they are not monochromatic outside $V_t(B)$).

4.4.4 color_final_component

The last procedure is almost identical to its counterpart in the base algorithm (Listing 4). Recall that the only change is at the beginning of the procedure. Instead of extending $\mathcal{C}$ with all unsafe edges intersecting it, only those unsafe edges that have at least αk troubled vertices in $V(\mathcal{C})$ are added. Then we proceed as in the base algorithm.

References

1 Dimitris Achlioptas, Themis Gouleakis, and Fotis Iliopoulos. Simple local computation algorithms for the general Lovász Local Lemma. In Christian Scheideler and Michael Spear, editors, *SPAA '20: 32nd ACM Symposium on Parallelism in Algorithms and Architectures, Virtual Event, USA, July 15-17, 2020*, pages 1–10. ACM, 2020. doi:10.1145/3350755.3400250.

2 Noga Alon. A parallel algorithmic version of the local lemma. *Random Structures Algorithms*, 2(4):367–378, 1991. doi:10.1002/rsa.3240020403.

3 Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1132–1139. SIAM, 2012. doi:10.1137/1.9781611973099.89.

4 Noga Alon and Joel H. Spencer. *The Probabilistic Method, Second Edition*. John Wiley, 2000. doi:10.1002/0471722154.

5 József Beck. An algorithmic approach to the Lovász local lemma. I. *Random Structures Algorithms*, 2(4):343–365, 1991. doi:10.1002/rsa.3240020402.

6 Yi-Jun Chang and Seth Pettie. A time hierarchy theorem for the LOCAL model. *SIAM J. Comput.*, 48(1):33–69, 2019. doi:10.1137/17M1157957.

7 Artur Czumaj and Christian Scheideler. Coloring non-uniform hypergraphs: a new algorithmic approach to the general Lovász local lemma. In David B. Shmoys, editor, *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA*, pages 30–39. ACM/SIAM, 2000. URL: https://dl.acm.org/doi/10.5555/338219.338229.

8 Andrzej Dorobisz and Jakub Kozik. Local computation algorithms for hypergraph coloring – following Beck's approach (full version), 2023. arXiv:2305.02831.

9 Paul Erdős and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In *Infinite and finite sets (Colloq., Keszthely, 1973; dedicated to P. Erdős on his 60th birthday), Vol. II*, volume 10 of *Colloquia Mathematica Societatis János Bolyai*, pages 609–627. North-Holland, Amsterdam, 1975.

10 Guy Even, Moti Medina, and Dana Ron. Best of two local models: Centralized local and distributed local algorithms. *Inf. Comput.*, 262:69–89, 2018. doi:10.1016/j.ic.2018.07.001.

11 András Faragó. A meeting point of probability, graphs, and algorithms: The Lovász Local Lemma and related results – A survey. *Algorithms*, 14(12):355, 2021. doi:10.3390/a14120355.

12 Manuela Fischer and Mohsen Ghaffari. Sublogarithmic distributed algorithms for Lovász Local Lemma, and the complexity hierarchy. In Andréa W. Richa, editor, *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, volume 91 of *LIPIcs*, pages 18:1–18:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.DISC.2017.18.

13 Mohsen Ghaffari, David G. Harris, and Fabian Kuhn. On derandomizing local distributed algorithms. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 662–673. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00069.

14 Nathan Linial. Distributive graph algorithms global solutions from local data. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 331–335, 1987. doi:10.1109/SFCS.1987.20.

15 László Lovász. Coverings and coloring of hypergraphs. In *Proceedings of the Fourth Southeastern Conference on Combinatorics, Graph Theory, and Computing (Florida Atlantic Univ., Boca Raton, Fla., 1973)*, pages 3–12, 1973.

16 Yishay Mansour, Aviad Rubinstein, Shai Vardi, and Ning Xie. Converting online algorithms to local computation algorithms. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming – 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, volume 7391 of *Lecture Notes in Computer Science*, pages 653–664. Springer, 2012. doi:10.1007/978-3-642-31594-7_55.

17 Michael Molloy and Bruce Reed. *Graph colouring and the probabilistic method*. Springer, 2002. doi:10.1007/978-3-642-04016-0.

18 Michael Molloy and Bruce A. Reed. Further algorithmic aspects of the Local Lemma. In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 524–529. ACM, 1998. doi:10.1145/276698.276866.

19 Robin A. Moser. Derandomizing the Lovasz Local Lemma more effectively. *CoRR*, abs/0807.2120, 2008. arXiv:0807.2120.

20 Robin A. Moser and Gábor Tardos. A constructive proof of the general lovász local lemma. *J. ACM*, 57(2):11:1–11:15, 2010. doi:10.1145/1667053.1667060.

21 Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. In Bernard Chazelle, editor, *Innovations in Computer Science – ICS 2011, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings*, pages 223–238. Tsinghua University Press, 2011.

22 Aravind Srinivasan. Improved algorithmic versions of the Lovász Local Lemma. In Shang-Hua Teng, editor, *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 611–620. SIAM, 2008. URL: https://dl.acm.org/doi/10.5555/1347082.1347150.

A Listings of the improved procedure

A.1 Listing of the global algorithm

Algorithm 5 Improved algorithm for uniform hypergraph coloring.

```
1  Procedure HYPERGRAPH_COLORING(H - hypergraph):
2      // I. SHATTERING PHASE
3      let (v_1, v_2, ...v_n) be an ordering of V(H)
4      for i = 1 to n do
5          if all edges containing v_i are not bad then
6              mark v_i as accepted
7          else
8              mark v_i as troubled
9          assign a random color to v_i
10     determine status of each e ∈ E(H)            // e is bad, safe, or unsafe
11     explore the line graph and build component-hypergraph H_C = (V_C, E_C)
12     // activation of bad-components
13     // - let U_C be the set of unsafe edges corresponding to E_C
14     // - let U(B) denote unsafe edges intersecting component B
15     // - let U_C(B) = U(B) ∩ U_C
16     // - let V_t(C) denote set of troubled vertices in component C
17     A ← ∅                              // initialize set of active components
18     Q ← ∅                                          // unsafe edges to process
19     // - initial activation
20     foreach B ∈ V_C do
21         if some e ∈ E(B) or f ∈ U(B) is monochromatic then
22             mark B as active
23             add B to A and add all edges from U_C(B) to Q
24         else mark B as inactive
25     foreach f ∈ U_C do
26         if f is monochromatic then merge in A all C ∈ A intersected by f
27     // - activation propagation
28     while Q is not empty do
29         f ← next edge from Q (remove it from Q)
30         if ∀_{C∈A} |f ∩ V_t(C)| < αk and f \ V_t(⋃A) is monochromatic then
31             // - activate new bad-components through f
32             foreach B ∈ V_C such that B is inactive and f intersects B do
33                 mark B as active
34                 add B to A and add all edges from U_C(B) to Q
35             // - merge active components through f
36             merge in A all C ∈ A intersected by f
37
38     // II. FINAL COLORING PHASE - color each final component
39     foreach C ∈ A do
40         foreach f ∈ U(C) such that |f ∩ V_t(C)| ≥ αk do add f to C
41         C' ← restriction of C to troubled vertices
42         Resample(C')
```

A.2 Listing of build_final_component (LCA)

Algorithm 6 Improved LCA procedure for the final component construction.

```
1  Procedure BUILD_FINAL_COMPONENT(v - troubled vertex):
2      B ← ∅                    // initialize set of bad edges of the component
3      U ← ∅                    // initialize set of unsafe edges to process
4      U_s ← ∅                  // unprocessed unsafe edges able to launch search
5      e ← any bad edge containing v
6      mark e as explored and run EXPAND_BAD_COMPONENT(e, B, U)
7      // process surrounding unsafe edges according to extension rules
8      while U ≠ ∅ or U_s ≠ ∅ do
9          while U is not empty do
10             f ← next edge from U (remove it from U)
11             if f has < αk troubled vertices in V(B) then
12                 if f satisfies rule (r1) or (r2) then
13                     EXPAND_VIA_UNSAFE(f, B, U)
14                 else if f can satisfy rule (r3) then
15                     add f to U_s        // f \ V(B) has < αk troubled vertices
16         if U_s is not empty then
17             f ← next edge from U_s (remove it from U_s)
18             if f has < αk troubled vertices in V(B) then
19                 // f satisfies rule (r3)
20                 (A, U_A) ← EXPAND_OR_ACCEPT(f, B, U)
21                 B = B ∪ A and if |B| > 2(Δ + 1) log(m) then FAIL
22                 U = U ∪ U_A
23     // return hypergraph built on set of bad edges
24     return C = (V(B), B)
```

A.3 Listing of expand_or_accept (LCA)

Algorithm 7 Conditional expansion via unsafe edge e (exploring a search area).

```
1  Procedure EXPAND_OR_ACCEPT(e - unsafe edge):
2      A ← ∅                    // initialize set of bad edges of the search area
3      U_A ← ∅                  // initialize set of unsafe edges around search area
4      Q ← {e}                  // unprocessed unsafe edges allowing expansion
5      // process selected surrounding unsafe edges
6      while Q is not empty do
7          f ← next edge from Q (remove it from Q)
8          // expand with the external component to which leads f
9          (C, U_C) ← (∅, ∅)
10         EXPAND_VIA_UNSAFE(f, C, U_C)
11         A = A ∪ C and if |A| > 2(Δ + 1) log(m) then FAIL
12         U_A = U_A ∪ U_C
13         // inspect new edges - look for amortizing configuration
14         if (e1) is satisfied (there is a monochromatic edge in C) then
15             return (A, U_A)
16         else if there is an unsafe edge f in U_C satisfying (e2) or (e3) then
17             EXPAND_VIA_UNSAFE(f, A, U_A)
18             return (A, U_A)
19         // select edges that may cause an activation
20         else
21             for g in U_C do
22                 if g|_a ∪ (g|_t ∩ V(C)) is monochromatic then
23                     if g \ V(C) has < αk troubled vertices then add g to Q
24     // activation exclusion
25     mark all troubled vertices in V(A) as accepted
26     return (∅, ∅)
```

An EPTAS for Budgeted Matching and Budgeted Matroid Intersection via Representative Sets

Ilan Doron-Arad
Computer Science Department, Technion, Haifa, Israel

Ariel Kulik
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Hadas Shachnai
Computer Science Department, Technion, Haifa, Israel

Abstract

We study the budgeted versions of the well known matching and matroid intersection problems. While both problems admit a *polynomial-time approximation scheme (PTAS)* [Berger et al. (Math. Programming, 2011), Chekuri, Vondrák and Zenklusen (SODA 2011)], it has been an intriguing open question whether these problems admit a *fully* PTAS (FPTAS), or even an *efficient* PTAS (EPTAS).

In this paper we answer the second part of this question affirmatively, by presenting an EPTAS for budgeted matching and budgeted matroid intersection. A main component of our scheme is a construction of *representative sets* for desired solutions, whose cardinality depends only on ε, the accuracy parameter. Thus, enumerating over solutions within a representative set leads to an EPTAS. This crucially distinguishes our algorithms from previous approaches, which rely on *exhaustive* enumeration over the solution set.

2012 ACM Subject Classification Theory of computation

Keywords and phrases budgeted matching, budgeted matroid intersection, efficient polynomial-time approximation scheme

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.49

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2209.04654` [5]

Funding *Ariel Kulik*: Research supported by the European Reseach Concil (ERC) consolidator grant no. 725978 SYSTEMATICGRAPH.

Acknowledgements We thank an anonymous reviewer for pointing us to the work of Huang and Ward [9], and for other helpful comments and suggestions.

1 Introduction

A wide range of NP-hard combinatorial optimization problems can be formulated as follows. We are given a ground set E and a family $\mathcal{M}$ of subsets of E called the *feasible sets*. The elements in the ground set are associated with a cost function $c : E \to \mathbb{R}_{\geq 0}$ and a profit function $p : E \to \mathbb{R}$, and we are also given a budget $\beta \in \mathbb{R}_{\geq 0}$. A *solution* is a feasible set $S \in \mathcal{M}$ of bounded cost $c(S) \leq \beta$.[1] Generally, the goal is to find a solution S of maximum profit, that is:

$$\max p(S) \text{ s.t. } S \in \mathcal{M}, c(S) \leq \beta. \tag{1}$$

[1] For a function $f : A \to \mathbb{R}$ and a subset of elements $C \subseteq A$, we define $f(C) = \sum_{e \in C} f(e)$.

Notable examples include shortest weight-constrained path [7], constrained minimum spanning trees [16], and knapsack with a conflict graph [15]. In this work, we focus on two prominent problems which can be formulated as (1).

In the *budgeted matching (BM)* problem we are given an undirected graph $G = (V, E)$, profit and cost functions on the edges $p, c : E \to \mathbb{R}_{\geq 0}$, and a budget $\beta \in \mathbb{R}_{\geq 0}$. A *solution* is a *matching* $S \subseteq E$ in G such that $c(S) \leq \beta$. The goal is to find a solution S such that the total profit $p(S)$ is maximized. Observe that BM can be formulated using (1), by letting $\mathcal{M}$ be the set of matchings in G.

In the *budgeted matroid intersection (BI)* problem we are given two matroids $(E, \mathcal{I}_1)$ and $(E, \mathcal{I}_2)$ over a ground set E, profit and cost functions on the elements $p, c : E \to \mathbb{R}_{\geq 0}$, and a budget $\beta \in \mathbb{R}_{\geq 0}$. Each matroid is given by a membership oracle. A *solution* is a *common independent set* $S \in \mathcal{I}_1 \cap \mathcal{I}_2$ such that $c(S) \leq \beta$; the goal is to find a solution S of maximum total profit $p(S)$. The formulation of BI as (1) follows by defining the feasible sets as all common independent sets $\mathcal{M} = \mathcal{I}_1 \cap \mathcal{I}_2$.

Let $\mathrm{OPT}(I)$ be the value of an optimal solution for an instance I of a maximization problem Π. For $\alpha \in (0, 1]$, we say that $\mathcal{A}$ is an α-approximation algorithm for Π if, for any instance I of Π, $\mathcal{A}$ outputs a solution of value at least $\alpha \cdot \mathrm{OPT}(I)$. A *polynomial-time approximation scheme* (PTAS) for Π is a family of algorithms $(A_\varepsilon)_{\varepsilon > 0}$ such that, for any $\varepsilon > 0$, A_ε is a polynomial-time $(1 - \varepsilon)$-approximation algorithm for Π. As ε gets smaller, a running time of the form $n^{\Theta\left(\frac{1}{\varepsilon}\right)}$ for a PTAS may become prohibitively large and thus impractical; therefore, it is natural to seek approximation schemes with better running times. Two families of such schemes have been extensively studied: an *efficient PTAS* (EPTAS) is a PTAS $(A_\varepsilon)_{\varepsilon > 0}$ whose running time is of the form $f\left(\frac{1}{\varepsilon}\right) \cdot n^{O(1)}$, where f is an arbitrary computable function, and n is the bit-length encoding size of the input instance. In a *fully PTAS* (FPTAS) the running time of A_ε is of the form $\left(\frac{n}{\varepsilon}\right)^{O(1)}$. For comprehensive surveys on approximation schemes see, e.g., [18, 19].

The state of the art for BM and BI is a PTAS developed by Berger et al. [1]. Similar results for both problems follow from a later work of Chekuri et al. [3] for the multi-budgeted variants of BM and BI. The running times of the above schemes are dominated by exhaustive enumeration which finds a set of $\Theta\left(\frac{1}{\varepsilon}\right)$ elements of highest profits in the solution. In this paper we optimize the enumeration procedure by substantially reducing the size of the domain over which we seek an efficient solution. Our main results are the following.

▶ **Theorem 1.** *There is an* EPTAS *for the budgeted matching problem.*

▶ **Theorem 2.** *There is an* EPTAS *for the budgeted matroid intersection problem.*

1.1 Related Work

BM and BI are immediate generalizations of the classic 0/1-knapsack problem. While the knapsack problem is known to be NP-hard, it admits an FPTAS. This raises a natural question whether BM and BI admit an FPTAS as well. The papers [1, 3] along with our results can be viewed as first steps towards answering this question.

Berger et al. [1] developed the first PTAS for BM and BI. Their approach includes an elegant combinatorial algorithm for *patching* two solutions for the *Lagrangian relaxation* of the underlying problem (i.e., BM or BI); one solution is feasible but has small profit, while the other solution has high profit but is infeasible. The scheme of [1] enumerates over solutions containing only high profit elements and uses the combinatorial algorithm to add low profit elements. This process may result in losing (twice) the profit of a low profit element, leading to a PTAS.

Chekuri et al. [3] developed a PTAS for multi-budgeted matching and a randomized PTAS for multi-budgeted matroid intersection; these are variants of BM and BI, respectively, in which the costs are d-dimensional, for some constant $d \geq 2$. They incorporate a non-trivial martingale based analysis to derive the results, along with enumeration to facilitate the selection of profitable elements for the solution. The paper [3] generalizes a previous result of Grandoni and Zenklusen [8], who obtained a PTAS for multi-budgeted matching and multi-budgeted matroid intersection in *representable matroids*.[2] For $d \geq 2$, the multi-budgeted variants of BM and BI generalize the two-dimensional knapsack problem, and thus do not admit an EPTAS unless W[1] = FPT [11].

An evidence for the difficulty of attaining an FPTAS for BM comes from the *exact* variant of the problem. In this setting, we are given a graph $G = (V, E)$, a cost function $c : E \to \mathbb{R}_{\geq 0}$, and a *target* $B \in \mathbb{R}_{\geq 0}$; the goal is to find a perfect matching $S \subseteq E$ with exact specified cost $c(S) = B$. There is a randomized pseudo-polynomial time algorithm for exact matching [13]. On the other hand, it is a long standing open question whether exact matching admits a deterministic pseudo-polynomial time algorithm [14]. Interestingly, as noted by Berger et al. [1], a deterministic FPTAS for BM would give an affirmative answer also for the latter question. A deterministic FPTAS for BI would have similar implications for the *exact* matroid intersection problem, which admits a randomized (but not a deterministic) pseudo-polynomial time algorithm for linear matroids [2]. While the above does not rule out the existence of an FPTAS for BM or BI, it indicates that improving our results from EPTAS to FPTAS might be a difficult task.

For the budgeted matroid independent set (i.e., the special case of BI with two identical matroids), the paper [6] gives an EPTAS using *representative sets* to enhance enumeration over elements of high profits. Their scheme exploits integrality properties of matroid polytopes under budget constraints (introduced in [8]) to efficiently combine elements of low profit into the solution.

1.2 Contribution and Techniques

Given an instance I of BM or BI, we say that an element e is *profitable* if $p(e) > \varepsilon \cdot \mathrm{OPT}(I)$; otherwise, e is *non-profitable*. The scheme for BM and BI of Berger et al. [1] distinguishes between profitable and non-profitable elements. In the main loop, the algorithm enumerates over all potential solutions containing only profitable elements.[3] Each solution is extended to include non-profitable elements using a combinatorial algorithm. The algorithm outputs a solution of highest profit. Overall, this process may lose at most twice the profit of a non-profitable element compared to the optimum, effectively preserving the approximation guarantee; however, an exhaustive enumeration over the profitable elements renders the running time $n^{\Omega(\frac{1}{\varepsilon})}$. In stark contrast, in this paper we introduce a new approach which enhances the enumeration over profitable elements, leading to an EPTAS.

We restrict the enumeration to only a small subset of elements called *representative set*; that is, a subset of elements $R \subseteq E$ satisfying the following property: there is a solution S such that the profitable elements in S are a subset of R, and the profit of S is at least $(1-O(\varepsilon))\cdot\mathrm{OPT}(I)$. If one finds efficiently a representative set R of cardinality $|R| \leq f\left(\frac{1}{\varepsilon}\right)$ for some computable function f, obtaining an EPTAS is straightforward based on the approach of [1].

[2] Representable matroids are also known as *linear matroids*.
[3] A similar technique is used also by Chekuri et al. [3].

Our scheme generalizes the *representative set* framework in [6], developed originally for budgeted matroid independent set. In [6], a representative set is a basis of minimum cost of a matroid, which can be found using a greedy algorithm. Alas, a greedy analogue for the setting of matching and matroid intersection fails; we give an example in Figure 1.[4] Hence, we take a different approach. Our main technical contribution is in the construction of representative sets for each of our problems.

For BM we design a surprisingly simple algorithm which finds a representative set using a union of multiple matchings. To this end, we partition the edges in G into *profit classes* such that each profit class contains edges of *similar* profits. We then use the greedy approach to repeatedly find in each profit class a union of disjoint matchings, where each matching has a bounded cardinality and is greedily selected to minimize cost. Intuitively, to show that the above yields a representative set, consider a profitable edge e in some optimal solution. Suppose that e is not chosen in our union of matchings, then we consider two cases. If each matching selected in the profit class of e contains an edge that is adjacent to (i.e., shares a vertex with) e, we show that at least one of these edges can be exchanged with e; otherwise, there exists a matching with no edge adjacent to e. In this case, we show that our greedy selection guarantees the existence of an edge in this matching which can be exchanged with e, implying the above is a representative set (see the details in Section 4).

For BI, we design a recursive algorithm that relies on an *asymmetric interpretation* of the two given matroids. We have learnt recently that a similar and more powerful construction was already proposed in [9]; we include the full details for completeness. In each recursive call of the algorithm, we are given an independent set $S \in \mathcal{I}_1$. The algorithm adds to the constructed representative set a minimum cost basis B_S of the second matroid $(E, \mathcal{I}_2)$, with the crucial restriction that any element $e \in B_S$ must satisfy $S \cup \{e\} \in \mathcal{I}_1$. Succeeding recursive calls will then use the set $S \cup \{e\}$, for every $e \in B_S$. Thus, we limit the search space to $\mathcal{I}_1$, while bases are constructed w.r.t. $\mathcal{I}_2$. To show that the algorithm yields a representative set, consider a profitable element f in an optimal solution. We construct a sequence of elements which are independent w.r.t. $\mathcal{I}_1$ and can be exchanged with f w.r.t. $\mathcal{I}_2$. Using matroid properties we show that one of these elements can be exchanged with f w.r.t. both matroids (see the details in Section 5).

Interestingly, our framework for solving BM and BI (presented in Section 3) can be extended to solve other problems formulated as (1) which possess similar *exchange properties*. We elaborate on that in Section 6.

Organization of the paper. In Section 2 we give some definitions and notation. Section 3 presents our framework that yields an EPTAS for each of the problems. In Sections 4 and 5 we describe the algorithms for constructing representative sets for BM and BI, respectively. We conclude in Section 6 with a summary and some directions for future work. Due to space constraints, some of the proofs are given in the full version of the paper [5].

2 Preliminaries

For simplicity of the notation, for any set A and an element e, we use $A + e$ and $A - e$ to denote $A \cup \{e\}$ and $A \setminus \{e\}$, respectively. Also, for any $k \in \mathbb{R}$ let $[k] = \{1, 2, \ldots, \lfloor k \rfloor\}$. For a function $f : A \to \mathbb{R}_{\geq 0}$ and a subset of elements $C \subseteq A$, let $f|_C : C \to \mathbb{R}_{\geq 0}$ be the *restriction* of f to C, such that $\forall e \in C : f|_C(e) = f(e)$.

[4] The example becomes clear once the reader is familiar with the definitions given in Section 3.

2.1 Matching and Matroids

Given an undirected graph $G = (V, E)$, a *matching* of G is a subset of edges $M \subseteq E$ such that each vertex appears as an endpoint in at most one edge in M, i.e., for all $v \in V$ it holds that $|\{\{u, v\} \in M \mid u \in V\}| \leq 1$. We denote by $V(M) = \{v \in V \mid \exists u \in V \text{ s.t. } \{u, v\} \in M\}$ the set of endpoints of a matching M of G.

Let E be a finite ground set and $\mathcal{I} \subseteq 2^E$ a non-empty set containing subsets of E called the *independent sets* of E. Then $\mathcal{M} = (E, \mathcal{I})$ is a *matroid* if it satisfies the following.

1. (Hereditary Property) For all $A \in \mathcal{I}$ and $B \subseteq A$, it holds that $B \in \mathcal{I}$.
2. (Exchange Property) For any $A, B \in \mathcal{I}$ where $|A| > |B|$, there is $e \in A \setminus B$ such that $B + e \in \mathcal{I}$.

A *basis* of a matroid $\mathcal{G} = (E, \mathcal{I})$ is an independent set $B \in \mathcal{I}$ such that for all $e \in E \setminus B$ it holds that $B + e \notin \mathcal{I}$. Given a cost function $c : E \to \mathbb{R}_{\geq 0}$, we say that a basis B of $\mathcal{G}$ is a *minimum* basis of $\mathcal{G}$ w.r.t. c if, for any basis A of $\mathcal{G}$ it holds that $c(B) \leq c(A)$. A minimum basis of $\mathcal{G}$ w.r.t. c can be easily constructed in polynomial-time using a greedy approach (see, e.g., [4]). In the following we define several matroid operations. Note that the structures resulting from the operations outlined in Definition 3 are matroids (see, e.g., [17]).

▶ **Definition 3.** *Let $\mathcal{G} = (E, \mathcal{I})$ be a matroid.*

1. *(restriction) For any $F \subseteq E$ define $\mathcal{I}_{\cap F} = \{A \in \mathcal{I} \mid A \subseteq F\}$ and $\mathcal{G} \cap F = (F, \mathcal{I}_{\cap F})$.*
2. *(thinning) For any $F \in \mathcal{I}$ define $\mathcal{I}/F = \{A \subseteq E \setminus F \mid A \cup F \in \mathcal{I}\}$ and $\mathcal{G}/F = (E \setminus F, \mathcal{I}/F)$.*[5]
3. *(truncation) For any $q \in \mathbb{N}$ define $\mathcal{I}_{\leq q} = \{A \in \mathcal{I} \mid |A| \leq q\}$ and $[\mathcal{G}]_{\leq q} = (E, \mathcal{I}_{\leq q})$.*

2.2 Instance Definition

We give a unified definition for instances of budgeted matching and budgeted matroid intersection. Given a ground set E of elements, we say that $\mathcal{C}$ is a *constraint* of E if one of the following holds.

- $\mathcal{C} = (V, E)$ is a *matching constraint*, where $\mathcal{C}$ is an undirected graph. Let $\mathcal{M}(\mathcal{C}) = \{M \subseteq E \mid M \text{ is a matching in } \mathcal{C}\}$ be the *feasible sets* of $\mathcal{C}$. Given a subset of edges $F \subseteq E$, let $E/F = \{\{u, v\} \in E \mid u, v \notin V(F)\}$ be the *thinning* of F on E, and let $\mathcal{C}/F = (V, E/F)$ be the *thinning* of F on $\mathcal{C}$.
- $\mathcal{C} = (\mathcal{I}_1, \mathcal{I}_2)$ is a *matroid intersection constraint*, where $(E, \mathcal{I}_1)$ and $(E, \mathcal{I}_2)$ are matroids. Throughout this paper, we assume that each of the matroids is given by an independence oracle. That is, determining whether $F \subseteq E$ belongs to $\mathcal{I}_1$ or to $\mathcal{I}_2$ requires a single call to the corresponding oracle of $\mathcal{I}_1$ or $\mathcal{I}_2$, respectively. Let $\mathcal{M}(\mathcal{C}) = \mathcal{I}_1 \cap \mathcal{I}_2$ be the collection of *feasible sets* of $\mathcal{C}$. In addition, given some $F \subseteq E$, let $\mathcal{C}/F = (\mathcal{I}_1/F, \mathcal{I}_2/F)$ be the *thinning* of F on $\mathcal{C}$. We say that $\mathcal{C}$ is a *single matroid constraint* if $\mathcal{I}_1 = \mathcal{I}_2$

When understood from the context, we simply use $\mathcal{M} = \mathcal{M}(\mathcal{C})$. Define an instance of the *budgeted constrained (BC)* problem as a tuple $I = (E, \mathcal{C}, c, p, \beta)$, where E is a ground set of elements, $\mathcal{C}$ is a constraint of E, $c : E \to \mathbb{R}_{\geq 0}$ is a cost function, $p : E \to \mathbb{R}_{\geq 0}$ is a profit function, and $\beta \in \mathbb{R}_{\geq 0}$ is a budget. If $\mathcal{C}$ is a matching constraint then I is a BM instance; otherwise, I is a BI instance. A *solution* of I is a feasible set $S \in \mathcal{M}(\mathcal{C})$ such that $c(S) \leq \beta$. The objective is to find a solution S of I such that $p(S)$ is maximized. Let $|I|$ denote the encoding size of a BC instance I, and $\text{poly}(|I|)$ be a polynomial size in $|I|$.

[5] Thinning is generally known as contraction; we use the term thinning to avoid confusion with edge contraction in graphs.

3 The Algorithm

In this section we present an EPTAS for the BC problem. Our first step is to determine the set of *profitable* elements in the constructed solution.[6] To this end, we generalize the *representative set* notion of [6] to the setting of BC. Our scheme relies on initially finding a set of profitable elements of small cardinality, from which the most profitable elements are selected for the solution using enumeration. Then, *non-profitable* elements are added to the solution using techniques of [1].

For the remainder of this section, fix a BC instance $I = (E, \mathcal{C}, c, p, \beta)$ and an error parameter $0 < \varepsilon < \frac{1}{2}$. Let $H(I, \varepsilon) = \{e \in E \mid p(e) > \varepsilon \cdot \mathrm{OPT}(I)\}$ be the set of *profitable* elements in I, and $E \setminus H(I, \varepsilon)$ the set of *non-profitable* elements; when understood from the context, we use $H = H(I, \varepsilon)$. Now, a representative set is a subset of elements which contains the profitable elements of an *almost* optimal solution. Formally,

▶ **Definition 4.** *Let $I = (E, \mathcal{C}, c, p, \beta)$ be a* BC *instance, $0 < \varepsilon < \frac{1}{2}$ and $R \subseteq E$. We say that R is a* representative set *of I and ε if there is a solution S of I such that the following holds.*

1. $S \cap H \subseteq R$.
2. $p(S) \geq (1 - 4\varepsilon) \cdot \mathrm{OPT}(I)$.

The work of [6] laid the foundations for the following notions of *replacements* and *strict representative sets (SRS)*, for the special case of BC where $\mathcal{C}$ is a single matroid constraint. Below we generalize the definitions of replacements and SRS.

Intuitively, a replacement of a solution S for I of bounded cardinality is another solution for I which preserves the attributes of the profitable elements in S (i.e., $S \cap H$). In particular, the profit of the replacement is close to $p(S \cap H)$, whereas the cost and the number of profitable elements can only be smaller. An SRS is a subset of elements containing a replacement for any solution for I of bounded cardinality.

The formal definitions of replacement and SRS for general BC instances are given in Definitions 5 and 6, respectively. Let $q(\varepsilon) = \left\lceil \varepsilon^{-\varepsilon^{-1}} \right\rceil$, and $\mathcal{M}_{\leq q(\varepsilon)} = \{A \in \mathcal{M} \mid |A| \leq q(\varepsilon)\}$ be all *bounded feasible sets* of $\mathcal{C}$ and ε. Recall that we use $\mathcal{M} = \mathcal{M}(\mathcal{C})$ for the feasible sets of $\mathcal{C}$; similar simplification in notation is used also for bounded feasible sets.

▶ **Definition 5.** *Given a* BC *instance $I = (E, \mathcal{C}, c, p, \beta), 0 < \varepsilon < \frac{1}{2}$, $S \in \mathcal{M}_{\leq q(\varepsilon)}$, and $Z_S \subseteq E$, we say that Z_S is a* replacement *of S for I and ε if the following holds:*

1. $(S \setminus H) \cup Z_S \in \mathcal{M}_{\leq q(\varepsilon)}$.
2. $c(Z_S) \leq c(S \cap H)$.
3. $p((S \setminus H) \cup Z_S) \geq (1 - \varepsilon) \cdot p(S)$.
4. $|Z_S| \leq |S \cap H|$.

▶ **Definition 6.** *Given a* BC *instance $I = (E, \mathcal{C}, c, p, \beta), 0 < \varepsilon < \frac{1}{2}$, and $R \subseteq E$, we say that R is a* strict representative set (SRS) *of I and ε if, for any $S \in \mathcal{M}_{\leq q(\varepsilon)}$, there is a replacement $Z_S \subseteq R$ of S for I and ε.*

Observe that given any solution S of I such that $|S| \leq q(\varepsilon)$, it holds that $S \cap H$ is a replacement of S; also, E is an SRS. In the next result, we demonstrate the power of SRS in solving BC. Specifically, we show that any SRS $R \subseteq E$ is also a representative set. Hence, using enumeration on subsets of R we can find a subset of elements that can be extended by only non-profitable elements to an *almost* optimal solution (see Algorithm 2).

[6] A similar approach is used, e.g., in [8, 1, 6].

▶ **Lemma 7.** *Let $I = (E, \mathcal{C}, c, p, \beta)$ be a* BC *instance, let $0 < \varepsilon < \frac{1}{2}$, and let R be an* SRS *of I and ε. Then R is a representative set of I and ε.*

The proof of Lemma 7 is given in [5]. We proceed to construct an SRS whose cardinality depends only on ε. First, we partition the profitable elements (and possibly some more elements) into a small number of *profit classes*, where elements from the same profit class have *similar* profits. The profit classes are derived from a 2-approximation α for $\mathrm{OPT}(I)$, which can be easily computed in polynomial time. Specifically, for all $r \in [\log_{1-\varepsilon}\left(\frac{\varepsilon}{2}\right) + 1]$ define the *r-profit class* as

$$\mathcal{K}_r(\alpha) = \left\{ e \in E \;\middle|\; \frac{p(e)}{2 \cdot \alpha} \in \left((1-\varepsilon)^r, (1-\varepsilon)^{r-1}\right] \right\}. \tag{2}$$

In the following we give a definition of an *exchange set* for each profit class. This facilitates the construction of an SRS. In words, a subset of elements X is an exchange set for some profit class $\mathcal{K}_r(\alpha)$ if any feasible set Δ and element $a \in (\Delta \cap \mathcal{K}_r(\alpha)) \setminus X$ can be replaced (while maintaining feasibility) by some element $b \in (X \cap \mathcal{K}_r(\alpha)) \setminus \Delta$, such that the cost of b is no larger than the cost of a. Formally,

▶ **Definition 8.** *Let $I = (E, \mathcal{C}, c, p, \beta)$ be a* BC *instance, $0 < \varepsilon < \frac{1}{2}$, $\frac{\mathrm{OPT}(I)}{2} \leq \alpha \leq \mathrm{OPT}(I)$, $r \in [\log_{1-\varepsilon}\left(\frac{\varepsilon}{2}\right) + 1]$, and $X \subseteq \mathcal{K}_r(\alpha)$. We say that X is an* exchange set *for I, ε, α, and r if:*

- *For all $\Delta \in \mathcal{M}_{\leq q(\varepsilon)}$ and $a \in (\Delta \cap \mathcal{K}_r(\alpha)) \setminus X$ there is $b \in (\mathcal{K}_r(\alpha) \cap X) \setminus \Delta$ satisfying*
  - *$c(b) \leq c(a)$.*
  - *$\Delta - a + b \in \mathcal{M}_{\leq q(\varepsilon)}$.*

The similarity between SRS and exchange sets is not coincidental. We show that if a set $R \subseteq E$ satisfies that $R \cap \mathcal{K}_r(\alpha)$ is an exchange set for any $r \in [\log_{1-\varepsilon}\left(\frac{\varepsilon}{2}\right) + 1]$, then R is an SRS, and thus also a representative set by Lemma 7. This allows us to construct an SRS using a union of disjoint exchange sets, one for each profit class.

▶ **Lemma 9.** *Let $I = (E, \mathcal{C}, c, p, \beta)$ be a* BC *instance, $0 < \varepsilon < \frac{1}{2}$, $\frac{\mathrm{OPT}(I)}{2} \leq \alpha \leq \mathrm{OPT}(I)$ and $R \subseteq E$. If for all $r \in [\log_{1-\varepsilon}\left(\frac{\varepsilon}{2}\right) + 1]$ it holds that $R \cap \mathcal{K}_r(\alpha)$ is an exchange set for I, ε, α, and r, then R is a representative set of I and ε.*

We give the formal proof in [5]. We now present a unified algorithm for finding a representative set for both types of constraints, namely, matching or matroid intersection constraints. This is achieved by taking the union of exchange sets of all profit classes. Nevertheless, for the construction of exchange sets we distinguish between the two types of constraints. This results also in different sizes for the obtained representative sets. Our algorithms for finding the exchange sets are the core technical contribution of this paper.

For matching constraints, we design an algorithm which constructs an exchange set for any profit class by finding multiple matchings of $\mathcal{C}$ from the given profit class. Each matching has a bounded cardinality, and the edges are chosen using a greedy approach to minimize the cost. We give the full details and a formal proof of Lemma 10 in Section 4.

▶ **Lemma 10.** *There is an algorithm* ExSet-Matching *that given a* BM *instance I, $0 < \varepsilon < \frac{1}{2}$, $\frac{\mathrm{OPT}(I)}{2} \leq \alpha \leq \mathrm{OPT}(I)$, and $r \in [\log_{1-\varepsilon}\left(\frac{\varepsilon}{2}\right) + 1]$, returns in time $q(\varepsilon) \cdot \mathrm{poly}(|I|)$ an exchange set X for I, ε, α, and r, such that $|X| \leq 18 \cdot q(\varepsilon)^2$.*

Our algorithm for matroid intersection constraints is more involved and generates an exchange set by an *asymmetric interpretation* of the two given matroids. As the technique was introduced by Huang and Ward [9], the proof of the next lemma follows immediately from Theorem 3.6 in [9]. For completeness, we give the full details in Section 5.

▶ **Lemma 11.** *There is an algorithm* ExSet-MatroidIntersection *that given a* BI *instance* I, $0 < \varepsilon < \frac{1}{2}$, $\frac{\mathrm{OPT}(I)}{2} \leq \alpha \leq \mathrm{OPT}(I)$, *and* $r \in [\log_{1-\varepsilon}\left(\frac{\varepsilon}{2}\right) + 1]$, *returns in time* $q(\varepsilon)^{O(q(\varepsilon))} \cdot \mathrm{poly}(|I|)$ *an exchange set* X *for* I, ε, α, *and* r, *such that* $|X| \leq q(\varepsilon)^{O(q(\varepsilon))}$.

Using the above, we design an algorithm that returns a representative set for both types of constraints. This is done by computing a 2-approximation α for $\mathrm{OPT}(I)$, and then finding exchange sets for all profit classes, for the corresponding type of constraint. Finally, we return the union of the above exchange sets. The pseudocode of our algorithm, RepSet, is given in Algorithm 1.

Algorithm 1 RepSet$(I = (E, \mathcal{C}, c, p, \beta), \varepsilon)$.

input : A BC instance I and error parameter $0 < \varepsilon < \frac{1}{2}$.
output : A representative set R of I and ε.

```
Compute a 2-approximation S* for I using a PTAS for BC with parameter ε' = 1/2.
Set α ← p(S*).
Initialize R ← ∅.
for r ∈ [log_{1−ε}(ε/2) + 1] do
    if I is a BM instance then
        R ← R ∪ ExSet-Matching(I, ε, α, r).
    else
        R ← R ∪ ExSet-MatroidIntersection(I, ε, α, r).
Return R.
```

▶ **Lemma 12.** *Given a* BC *instance* $I = (E, \mathcal{C}, c, p, \beta)$ *and* $0 < \varepsilon < \frac{1}{2}$, *Algorithm 1 returns a representative set* R *of* I *and* ε, *such that one of the following holds.*

- *If* $\mathcal{C}$ *is a matching constraint the running time is* $q(\varepsilon)^2 \cdot \mathrm{poly}(|I|)$, *and* $|R| \leq 54 \cdot q(\varepsilon)^3$.
- *If* $\mathcal{C}$ *is a matroid intersection constraint the running time is* $q(\varepsilon)^{O(q(\varepsilon))} \cdot \mathrm{poly}(|I|)$, *and* $|R| \leq q(\varepsilon)^{O(q(\varepsilon))}$.

The proof of the lemma is given in [5]. Next, we use a result of [1] for adding elements of smaller profits to the solution. The techniques of [1] are based on a non-trivial patching of two solutions of the Lagrangian relaxation of BC (for both matching and matroid intersection constraints). This approach yields a feasible set of almost optimal profit; in the worst case, the difference from the optimum is twice the maximal profit of an element in the instance. Since we use the latter approach only for non-profitable elements, this effectively does not harm our approximation guarantee. The following is a compact statement of the above result of [1].

▶ **Lemma 13.** *There is a polynomial-time algorithm* NonProfitableSolver *that given a* BC *instance* $I = (E, \mathcal{C}, c, p, \beta)$ *computes a solution* S *for* I *of profit* $p(S) \geq \mathrm{OPT}(I) - 2 \cdot \max_{e \in E} p(e)$.

Using the algorithm above and our algorithm for computing a representative set, we obtain an EPTAS for BC. Let R be the representative set returned by RepSet(I, ε). Our scheme enumerates over subsets of R to select profitable elements for the solution. Using algorithm NonProfitableSolver of [1], the solution is extended to include also non-profitable elements. Specifically, let $\frac{\mathrm{OPT}(I)}{2} \leq \alpha \leq \mathrm{OPT}(I)$ be a 2-approximation for the optimal profit for I. In addition, let $E(\alpha) = \{e \in E \mid p(e) \leq 2\varepsilon \cdot \alpha\}$ be the set including the non-profitable elements, and possibly also profitable elements $e \in E$ such that $p(e) \leq 2\varepsilon \cdot \mathrm{OPT}(I)$. Given a feasible set $F \in \mathcal{M}$, we define a residual BC instance containing elements which can *extend* F by adding elements from $E(\alpha)$. More formally,

▶ **Definition 14.** *Given a* BC *instance* $I = (E, \mathcal{C}, c, p, \beta)$, $\frac{\mathrm{OPT}(I)}{2} \leq \alpha \leq \mathrm{OPT}(I)$, *and* $F \in \mathcal{M}(\mathcal{C})$, *the* residual instance of F and α *for* I *is the* BC *instance* $I_F(\alpha) = (E_F, \mathcal{C}_F, c_F, p_F, \beta_F)$ *defined as follows.*

- $E_F = E(\alpha) \setminus F$.
- $\mathcal{C}_F = \mathcal{C}/F$.
- $p_F = p|_F$ *(i.e., the restriction of* p *to* F*).*
- $c_F = c|_F$.
- $\beta_F = \beta - c(F)$.

▶ **Observation 15.** *Let* $I = (E, \mathcal{C}, c, p, \beta)$ *be a* BC *instance,* $\frac{\mathrm{OPT}(I)}{2} \leq \alpha \leq \mathrm{OPT}(I)$, $F \in \mathcal{M}(\mathcal{C})$, *and let* T *be a solution for* $I_F(\alpha)$. *Then,* $T \cup F$ *is a solution for* I.

For all solutions $F \subseteq R$ for I with $|F| \leq \varepsilon^{-1}$, we find a solution T_F for the residual instance $I_F(\alpha)$ using Algorithm NonProfitableSolver and define $K_F = T_F \cup F$ as the *extended solution* of F. Our scheme iterates over the extended solutions K_F, for all such solutions F, and chooses an extended solution K_{F^*} of maximal total profit. The pseudocode of the scheme is given in Algorithm 2.

Algorithm 2 EPTAS$(I = (E, \mathcal{C}, c, p, \beta), \varepsilon)$.

input : A BC instance I and an error parameter $0 < \varepsilon < \frac{1}{2}$.
output : A solution for I.

```
Construct the representative set R ← RepSet(I, ε).
Compute a 2-approximation S* for I using a PTAS for BC with parameter ε' = 1/2.
Set α ← p(S*).
Initialize an empty solution A ← ∅.
for F ⊆ R s.t. |F| ≤ ε^{-1} and F is a solution of I do
    Find a solution for I_F(α) by T_F ← NonProfitableSolver(I_F(α)).
    Let K_F ← T_F ∪ F.
    if p(K_F) > p(A) then
        Update A ← K_F
Return A.
```

The running time of Algorithm 2 crucially depends on the cardinality of the representative set. Roughly speaking, the running time is the number of subsets of the representative set containing at most ε^{-1} elements, multiplied by a computation time that is polynomial in the encoding size of the instance. Moreover, since $R = \mathsf{RepSet}(I, \varepsilon)$ is a representative set (by Lemma 12), there is an almost optimal solution S of I such that the profitable elements in S are a subset of R. Thus, there is an iteration of the **for** loop in Algorithm 2 such that $F = S \cap H$. In the proof of Lemma 16 we focus on this iteration and show that it yields a solution K_F of I with an almost optimal profit.

▶ **Lemma 16.** *Given a* BC *instance* $I = (E, \mathcal{C}, c, p, \beta)$ *and* $0 < \varepsilon < \frac{1}{2}$, *Algorithm 2 returns a solution for* I *of profit at least* $(1 - 8\varepsilon) \cdot \mathrm{OPT}(I)$ *such that one of the following holds.*

- *If* I *is a* BM *instance the running time is* $2^{O\left(\varepsilon^{-2} \log \frac{1}{\varepsilon}\right)} \cdot \mathrm{poly}(|I|)$.
- *If* I *is a* BI *instance the running time is* $q(\varepsilon)^{O(\varepsilon^{-1} \cdot q(\varepsilon))} \cdot \mathrm{poly}(|I|)$, *where* $q(\varepsilon) = \left\lceil \varepsilon^{-\varepsilon^{-1}} \right\rceil$.

The proof of Lemma 16 is given in [5]. We are ready to prove our main results.

Proofs of Theorem 1 and Theorem 2. Given a BC instance I and $0 < \varepsilon < \frac{1}{2}$, using Algorithm 2 for I with parameter $\frac{\varepsilon}{8}$ we have by Lemma 16 the desired approximation guarantee. Furthermore, the running time is $2^{O\left(\varepsilon^{-2} \log \frac{1}{\varepsilon}\right)} \cdot \mathrm{poly}(|I|)$ or $q(\varepsilon)^{O(\varepsilon^{-1} \cdot q(\varepsilon))} \cdot \mathrm{poly}(|I|)$, depending on whether I is a BM instance or a BI instance, respectively. ◀

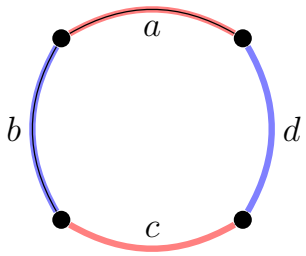


Figure 1 An example showing that bipartite matching may not yield an exchange set. Consider the two matchings $\Delta_1 = \{a, c\}, \Delta_2 = \{b, d\}$ marked in red and blue, and suppose that $\mathcal{K}_r(\alpha) = \{a, b\}$ is a profit class. The only exchange set for $\mathcal{K}_r(\alpha)$ is $\{a, b\}$, which is not a matching. Note that a bipartite matching can be cast as matroid intersection. For a bipartite graph $G = (L \cup R, E)$, define the matroids $\mathcal{M}_1 = (E, \mathcal{I}_1)$ and $\mathcal{M}_2 = (E, \mathcal{I}_2)$, where $\mathcal{I}_1 = \{F \subseteq E |\ \forall v \in L : |F \cap N(v)| \leq 1\}$, and $\mathcal{I}_2 = \{F \subseteq E |\ \forall v \in R : |F \cap N(v)| \leq 1\}$, where $N(v)$ is the set of neighbors of v. Thus, bipartite matching is a special case of both matching and matroid intersection.

4 Exchange Set for Matching Constraints

In this section we design an algorithm for finding an exchange set for a BM instance and a profit class, leading to the proof of Lemma 10. For the remainder of this section, fix a BM instance $I = (E, \mathcal{C}, c, p, \beta)$, an error parameter $0 < \varepsilon < \frac{1}{2}$, a 2-approximation for $\mathrm{OPT}(I)$, $\frac{\mathrm{OPT}(I)}{2} \leq \alpha \leq \mathrm{OPT}(I)$, and an index $r \in [\log_{1-\varepsilon}\left(\frac{\varepsilon}{2}\right) + 1]$ of the profit class $\mathcal{K}_r(\alpha)$.

We note that for a single matroid constraint an exchange set can be constructed by finding a minimum cost basis in the matroid [6]. More specifically, given a matroid $\mathcal{G} = (E, \mathcal{I})$, it is shown in [6] that a minimum cost basis in the matroid $[\mathcal{G} \cap \mathcal{K}_r(\alpha)]_{\leq q(\varepsilon)}$ is an exchange set for $\mathcal{K}_r(\alpha)$. Such exchange set can be easily computed using a greedy approach. An analogue for the setting of matching constraints is to find a matching of cardinality $\Omega(q(\varepsilon))$ and minimum total cost in $\mathcal{K}_r(\alpha)$. However, as shown in Figure 1, this idea fails. Thus, we turn to use a completely different approach.

A key observation is that even if a greedy matching algorithm may not suffice for the construction of an exchange set, applying such an algorithm multiple times can be the solution. Thus, as a subroutine our algorithm finds a matching using a greedy approach. The algorithm iteratively selects an edge of minimal cost while ensuring that the selected set of edges is a matching. This is done until the algorithm reaches a given cardinality bound, or no more edges can be added. The pseudocode of GreedyMatching is given in Algorithm 3.[7]

Algorithm 3 GreedyMatching$(G = (V, E), N, c)$.

input : A graph G, an integer $N \in \mathbb{N} \setminus \{0\}$, and a cost function $c : E \to \mathbb{R}_{\geq 0}$.
output : A matching M of G.

```
Initialize M ← ∅.
while |M| < N and E/M ≠ ∅ do
    Find e ∈ E/M of minimal cost w.r.t. c.
    Update M ← M + e.
Return M.
```

[7] Given a graph $G = (V, E)$ and a matching M of G, the definition of thinning E/M is given in Section 2.

Given a graph $G = (V, E)$ and two edges $a, b \in E$, we say that a, b are *adjacent* if there are $x, y, z \in V$ such that $a = \{x, y\}$ and $b = \{y, z\}$; for all $e \in E$, let $\mathsf{Adj}_G(e)$ be the set of edges adjacent to e in G. In the next result we show that if an edge a is not selected for the solution by GreedyMatching, then either the algorithm selects an adjacent edge of cost at most $c(a)$, or all of the selected edges have costs at most $c(a)$.

▶ **Lemma 17.** *Given a graph $G = (V, E)$, $N \in \mathbb{N} \setminus \{0\}$, and $c : E \to \mathbb{R}_{\geq 0}$, Algorithm 3 returns in polynomial time a matching M of G such that for all $a \in E \setminus M$ one of the following holds.*

1. *$|M| \leq N$ and there is $b \in \mathsf{Adj}_G(a) \cap M$ such that $c(b) \leq c(a)$.*
2. *$|M| = N$, for all $b \in M$ it holds that $c(b) \leq c(a)$, and $M + a$ is a matching of G.*

Proof. Clearly, Algorithm 3 returns in polynomial time a matching M of G. Observe that $|M| \leq N$ by Step 2. To prove that either 1. or 2. hold, we distinguish between two cases.

- $a \notin E/M$. Then $\mathsf{Adj}_G(a) \cap M \neq \emptyset$. Let e be the first edge in $\mathsf{Adj}_G(a) \cap M$ that is added to M in Step 4; also, let L be the set of edges added to M before e. Then $a \in E/L$, since L does not contain edges adjacent to a. By Step 3, it holds that $c(e) = \min_{e' \in E/L} c(e') \leq c(a)$.
- $a \in E/M$. Thus, $|M| = N$; otherwise, a would be added to M. Also, $M + a$ is a matching of G. Now, let $b \in M$, and let K be the set of edges added to M before b. Since $M + a$ is a matching of G, by the hereditary property of $(E, \mathcal{M}(G))$ it holds that $K + a$ is a matching of G; thus, $a \in E/K$ and by Step 3 it follows that $c(b) = \min_{e' \in E/K} c(e') \leq c(a)$. ◀

By Lemma 17, we argue that an exchange set can be found by using Algorithm GreedyMatching iteratively. Specifically, let $k(\varepsilon) = 6 \cdot q(\varepsilon)$ and $N(\varepsilon) = 3 \cdot q(\varepsilon)$. We run Algorithm GreedyMatching for $k(\varepsilon)$ iterations, each iteration with a bound $N(\varepsilon)$ on the cardinality of the matching. In iteration i, we choose a matching M_i from the edges of the profit class $\mathcal{K}_r(\alpha)$ and remove the chosen edges from the graph. Therefore, in the following iterations, edges adjacent to previously chosen edges can be chosen as well. A small-scale illustration of the algorithm is presented in Figure 2. The pseudocode of Algorithm ExSet-Matching, which computes an exchange set for the given profit class, is presented in Algorithm 4.

Algorithm 4 ExSet-Matching$(I = (E, \mathcal{C}, c, p, \beta), \varepsilon, \alpha, r)$.

input : a matching-BC instance I , $0 < \varepsilon < \frac{1}{2}$, $\frac{\mathrm{OPT}(I)}{2} \leq \alpha \leq \mathrm{OPT}(I)$, $r \in [\log_{1-\varepsilon}\left(\frac{\varepsilon}{2}\right) + 1]$.
output : An exchange set for I, ε, α, and r.

```
Initialize X ← ∅ and E_0 ← K_r(α).
for i ∈ {1, ..., k(ε)} do
    Define G_i = (V, E_{i−1}) where V is the vertex set of C.
    Compute M_i ← GreedyMatching(G_i, N(ε), c|_{E_{i−1}}).
    Update X ← X ∪ M_i and define E_i ← E_{i−1} \ M_i.
Return X.
```

Algorithm ExSet-Matching outputs a union X of disjoint matchings $M_1, \ldots, M_{k(\varepsilon)}$ taken from the edges of the profit class $\mathcal{K}_r(\alpha)$. For some $\Delta \in \mathcal{M}(\mathcal{C})$ and $a \in (\Delta \cap \mathcal{K}_r(\alpha)) \setminus X$, by Lemma 17, there are two options summarizing the main idea in the proof of Lemma 10.

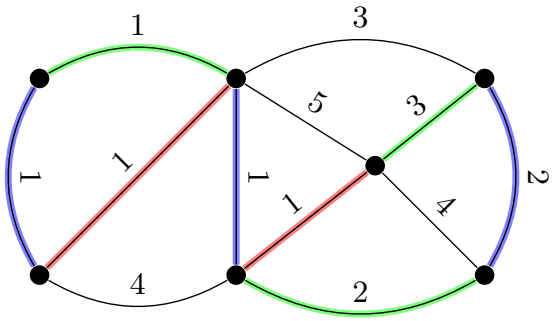


Figure 2 An execution of Algorithm ExSet-Matching with the (illegally small) parameters $N(\varepsilon) = k(\varepsilon) = 3$. The numbers by the edges are the costs. The edges chosen in iterations $i = 1, 2, 3$ are marked in blue, red, and green, respectively.

- all matchings M_i contain some b_i adjacent to a such that $c(b_i) \leq c(a)$. Then, as $k(\varepsilon)$ is sufficiently large, one such b_i is not adjacent to any edge in $\Delta - a$. Hence, $\Delta - a + b_i$ is a matching.
- One such M_i contains only edges of costs at most $c(a)$; as $N(\varepsilon)$ is sufficiently large, there is $b \in M_i$ such that $\Delta - a + b$ is a matching.

Proof of Lemma 10. For all $i \in \{1, \ldots, k(\varepsilon)\}$, let G_i and M_i be the outputs of Steps 3 and 4 in iteration i of the **for** loop in ExSet-Matching$(I, \varepsilon, \alpha, r)$, respectively. Also, let X be the output of the algorithm; observe that $X = \bigcup_{i \in [k(\varepsilon)]} M_i$. We show that X is an exchange set for I, ε, α and r (see Definition 8). Let $\Delta \in \mathcal{M}_{\leq q(\varepsilon)}$ and $a \in (\Delta \cap \mathcal{K}_r(\alpha)) \setminus X$. We use the next inequality in the claim below.

$$\frac{k(\varepsilon)}{2} = N(\varepsilon) = 3 \cdot q(\varepsilon) > 2 \cdot |\Delta| = |V(\Delta)|. \tag{3}$$

The inequality holds since $\Delta \in \mathcal{M}_{\leq q(\varepsilon)}$. The last equality holds since each vertex appears as an endpoint in a matching at most once.

▷ Claim 18. There is $b \in (X \cap \mathcal{K}_r(\alpha)) \setminus \Delta$ such that $\Delta - a + b \in \mathcal{M}_{\leq q(\varepsilon)}$, and $c(b) \leq c(a)$.

Proof. Let $a = \{x, y\}$, $I = (E, \mathcal{C}, c, p, \beta)$, and $\mathcal{C} = (V, E)$. Since $a \notin X$, for all $i \in \{1, \ldots, k(\varepsilon)\}$ it holds that $a \notin M_i$; thus, $a \in \mathcal{E}_i = \mathcal{E}_{i-1} \setminus M_i$. Hence, by Lemma 17, one of the following holds.

1. For all $i \in [k(\varepsilon)]$ there is $b_i \in \mathsf{Adj}_{G_i}(a) \cap M_i$ such that $c(b_i) \leq c(a)$. For $z \in \{x, y\}$ let

$$J_z = \{i \in [k(\varepsilon)] \mid \exists u \in V : \ b_i = \{z, u\}\}$$

be the set of indices of edges b_i neighboring to z. Since $b_i \in \mathsf{Adj}_{G_i}(a)$ it holds that $J_x \cup J_y = [k(\varepsilon)]$. Thus, there is $z \in \{x, y\}$ such that $|J_z| \geq \frac{k(\varepsilon)}{2} > |V(\Delta)|$, where the last inequality follows from (3). For any $i \in J_z$ let $v_i \in V$ be the vertex connected to z in b_i, that is $b_i = \{z, v_i\}$. Since the matchings $M_1, \ldots M_{k(\varepsilon)}$ are disjoint and $b_i \in M_i$ it follows that the vertices v_i for $i \in J_z$ are all distinct. As $|J_z| > |V(\Delta)|$ there is $i^* \in J_z$ such that $v_{i^*} \notin V(\Delta)$. Therefore, $\Delta - a + b_{i^*} \in \mathcal{M}_{\leq q(\varepsilon)}$ and $c(b_{i^*}) \leq c(a)$.
2. There is $i \in \{1, \ldots, k(\varepsilon)\}$ such that $|M_i| = N(\varepsilon)$, and for all $b \in M_i$ it holds that $c(b) \leq c(a)$. Then,

$$|M_i| = N(\varepsilon) > |V(\Delta)|. \tag{4}$$

The equality follows by the definition of M_i in Case 2. The inequality follows from (3). Since each vertex appears as an endpoint in a matching at most once, by (4) there is $b \in M_i$ such that both endpoints of b are not in $V(\Delta)$. Thus, $\Delta + b \in \mathcal{M}$; by the hereditary property and since $a \in \Delta$, it holds that $\Delta - a + b \in \mathcal{M}_{\leq q(\varepsilon)}$. ◁

By Claim 18 and Definition 8, we have that X is an exchange set for I, ε, α, and r as required. To complete the proof of the lemma we show (in [5]) the following.

▷ Claim 19. $|X| \leq 18 \cdot q(\varepsilon)^2$, and the running time of Algorithm 4 is $q(\varepsilon) \cdot \text{poly}(|I|)$. ◀

5 Exchange Set for Matroid Intersection Constraints

In this section we design an algorithm for finding an exchange set for a profit class in a BI instance. For the remainder of this section, fix a BI instance $I = (E, \mathcal{C}, c, p, \beta)$, an error parameter $0 < \varepsilon < \frac{1}{2}$, a 2-approximation for $\text{OPT}(I)$, $\frac{\text{OPT}(I)}{2} \leq \alpha \leq \text{OPT}(I)$, and an index $r \in [\log_{1-\varepsilon}(\frac{\varepsilon}{2}) + 1]$ of the profit class $\mathcal{K}_r(\alpha)$. Also, let $\mathcal{C} = (\mathcal{I}_1, \mathcal{I}_2)$ be the matroid intersection constraint $\mathcal{C}$ of I. For simplicity, when understood from the context, some of the lemmas in this section consider the given parameters (e.g., I) without explicit mention. The proofs of the lemmas in this section are given in the full version of the paper [5].

As shown in Figure 1, a simple greedy approach which finds a feasible set of minimum cost (within $\mathcal{K}_r(\alpha)$) in the intersection of the matroids may not output an exchange set for $\mathcal{K}_r(\alpha)$. Instead, our approach builds on some interesting properties of matroid intersection. The next definition presents a *shifting property* for a feasible set $\Delta \in \mathcal{M}_{\leq q(\varepsilon)}$ and an element $a \in \Delta \cap \mathcal{K}_r(\alpha)$ w.r.t. the two matroids. We use this property to show that our algorithm constructs an exchange set.

▶ **Definition 20.** *Let $\Delta \in \mathcal{M}_{\leq q(\varepsilon)}$, $a \in \Delta \cap \mathcal{K}_r(\alpha)$ and $b \in \mathcal{K}_r(\alpha) \setminus \Delta$. We say that b is a* shift *to a for Δ if $c(b) \leq c(a)$ and $\Delta - a + b \in \mathcal{M}_{\leq q(\varepsilon)}$. Furthermore, b is a* semi-shift *to a for Δ if $c(b) \leq c(a)$ and $\Delta - a + b \in \mathcal{I}_2$ but $\Delta - a + b \notin \mathcal{I}_1$.*

As a starting point for our exchange set algorithm, we show how to obtain small cardinality sets which contain either a shift or a semi-shift for every pair $\Delta \in \mathcal{M}_{\leq q(\varepsilon)}$ and $a \in \Delta \cap \mathcal{K}_r(\alpha)$.

▶ **Lemma 21.** *Let $U \subseteq \mathcal{K}_r(\alpha)$, $\Delta \in \mathcal{M}_{\leq q(\varepsilon)}$, and B be a minimum basis of $[(E, \mathcal{I}_2) \cap U]_{\leq q(\varepsilon)}$ w.r.t. c. Also, let $a \in (U \cap \Delta) \setminus B$. Then, there is $b \in B \setminus \Delta$ such that b is a semi-shift to a for Δ, or b is a shift to a for Δ.*

Observe that to obtain an exchange set, our goal is to find a subset of $\mathcal{K}_r(\alpha)$ which contains a shift for every pair $\Delta \in \mathcal{M}_{\leq q(\varepsilon)}$ and $a \in \Delta \cap \mathcal{K}_r(\alpha)$. Thus, using Lemma 21 we design the following recursive algorithm ExtendChain, which finds a union of minimum bases of matroids w.r.t $\mathcal{I}_2$, of increasingly restricted ground sets w.r.t. $\mathcal{I}_1$. The pseudocode of Algorithm ExtendChain is given in Algorithm 5.

We can view the execution of ExtendChain as a tree, where each node (called below a *branch*) corresponds to the subset $S \subseteq \mathcal{K}_r(\alpha)$ in a specific recursive call. We now describe the role of S in Algorithm ExtendChain. If $|S| \geq q(\varepsilon) + 1$, we simply return $\emptyset$; such a branch is called a *leaf*, and does not contribute elements to the constructed exchange set. Otherwise, define the *universe* of the branch S as $U_S = \{e \in \mathcal{K}_r(\alpha) \setminus S \mid S + e \in \mathcal{I}_1\}$; that is, elements in the universe of S which can be added to S to form an independent set w.r.t. $\mathcal{I}_1$. Next, we construct a minimum basis B_S w.r.t. c of the matroid $[(E, \mathcal{I}_2) \cap U_S]_{\leq q(\varepsilon)}$. Observe that B_S contains up to $q(\varepsilon)$ elements, selected from the universe of S, and that B_S is independent w.r.t. $\mathcal{I}_2$. Note that the definition of the universe relates to $\mathcal{I}_1$ while the construction of the bases to $\mathcal{I}_2$; thus, the two matroids play completely different roles in the algorithm.

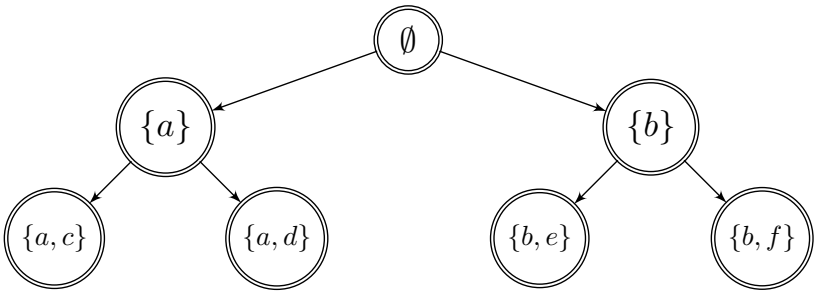


Figure 3 An illustration of the branches in Algorithm 5 for $S = \emptyset$. Note that $B_\emptyset = \{a, b\}$, $B_{\{a\}} = \{c, d\}$ and $B_{\{b\}} = \{e, f\}$. Also, $\{a, c\}$ and $\{a, d\}$ are the child branches of $\{a\}$.

For every element $e \in B_S$ we apply Algorithm ExtendChain recursively with $S' = S + e$ to find the corresponding basis B_{S+e}. The algorithm returns (using recursion) the union of the constructed bases over all branches. Finally, algorithm ExSet-MatroidIntersection constructs an exchange set for I, ε, α, and r by calling Algorithm ExtendChain with the initial branch (i.e., *root*) $S = \emptyset$:

$$\mathsf{ExSet\text{-}MatroidIntersection}(I, \varepsilon, \alpha, r) = \mathsf{ExtendChain}(I, \varepsilon, \alpha, r, \emptyset). \tag{5}$$

For an illustration of the algorithm, see Figure 3.

Algorithm 5 ExtendChain$(I = (E, \mathcal{C}, c, p, \beta), \varepsilon, \alpha, r, S)$.

input : a matroid-BC instance I , where $\mathcal{C} = (\mathcal{I}_1, \mathcal{I}_2)$, $0 < \varepsilon < \frac{1}{2}$, $\frac{\mathrm{OPT}(I)}{2} \leq \alpha \leq \mathrm{OPT}(I)$, $r \in [\log_{1-\varepsilon}\left(\frac{\varepsilon}{2}\right) + 1]$, and $S \subseteq E$.
output : (for $S = \emptyset$) An exchange set X for I, ε, α, and r.

```
if |S| ≥ q(ε) + 1 then
    Return ∅
Define U_S = {e ∈ K_r(α) \ S | S + e ∈ I_1}.
Compute a minimum basis B_S w.r.t. c of the matroid [(E, I_2) ∩ U_S]_{≤q(ε)}.
Return B_S ∪ (⋃_{e∈B_S} ExtendChain(I, ε, α, r, S + e)).
```

In the analysis of the algorithm, we consider branches with useful attributes, called *chains*; these are essentially sequences of semi-shifts to some $\Delta \in \mathcal{M}_{\leq q(\varepsilon)}$ and $a \in \Delta \cap \mathcal{K}_r(\alpha)$. Let $X = \mathsf{ExSet\text{-}MatroidIntersection}(I, \varepsilon, \alpha, r)$, and let $\mathcal{S}$ be the set of all branches $S \subseteq \mathcal{K}_r(\alpha)$ such that $\mathsf{ExtendChain}(I, \varepsilon, \alpha, r, S)$ is computed during the construction of X.

▶ **Definition 22.** *Let $S \in \mathcal{S}$, $\Delta \in \mathcal{M}_{\leq q(\varepsilon)}$, and $a \in (\mathcal{K}_r(\alpha) \cap \Delta) \setminus X$. We say that S is a* chain *of a and Δ if $a \in U_S$, and for all $e \in S$ it holds that e is a semi-shift to a for Δ.*

Note that there must be a chain for a and Δ since the empty set satisfies the conditions of Definition 22. Moreover, we can bound the cardinality of a chain by $q(\varepsilon)$ using the exchange property of the matroid $(E, \mathcal{I}_1)$. The above arguments are formalized in the next lemmas.

▶ **Lemma 23.** *For all $\Delta \in \mathcal{M}_{\leq q(\varepsilon)}$ and $a \in (\mathcal{K}_r(\alpha) \cap \Delta) \setminus X$ there is $S \subseteq X$ such that S is a chain of a and Δ.*

▶ **Lemma 24.** *For all $\Delta \in \mathcal{M}_{\leq q(\varepsilon)}$, $a \in (\mathcal{K}_r(\alpha) \cap \Delta) \setminus X$, and a chain S of a and Δ, it holds that $|S| \leq q(\varepsilon)$.*

For a chain S of a and Δ, let B_S be the result of the first computation of Step 4 (i.e., not within a recursive call) in $\mathsf{ExtendChain}(I, \varepsilon, \alpha, r, S)$. The key argument in the proof of Lemma 11 is that for a chain S^* of maximal cardinality, B_{S^*} contains a shift to a and Δ, using the maximality of S^* and Lemma 21.

▶ **Lemma 25.** *For all $\Delta \in \mathcal{M}_{\leq q(\varepsilon)}$, $a \in (\mathcal{K}_r(\alpha) \cap \Delta) \setminus X$, and a chain S^* of a and Δ of maximum cardinality, there is a shift $b^* \in B_{S^*}$ to a for Δ.*

In the proof of Lemma 11, for every $\Delta \in \mathcal{M}_{\leq q(\varepsilon)}$ and $a \in (\mathcal{K}_r(\alpha) \cap \Delta) \setminus X$, we take a chain S^* of a and Δ of maximum cardinality (which exists by Lemma 23 and Lemma 24). Then, by Lemma 25, there is a shift b^* to a for Δ, and it follows that X is an exchange set for I, ε, α, and r. The formal proof is given in [5].

6 Discussion

In this paper we present the first EPTAS for budgeted matching and budgeted matroid intersection, thus improving upon the existing PTAS for both problems. We derive our results via a generalization of the representative set framework in [6]; this ameliorates the exhaustive enumeration applied in similar settings [1, 3].

We note that the framework based on representative sets may be useful for solving other problems formulated as (1). Indeed, the proofs of Lemma 7 and Lemma 9, which establish the representative set framework, are oblivious to the exact type of constraints and only require having a *k-exchange system* for some constant k.[8]

Furthermore, our exchange sets algorithms can be applied with slight modifications to other variants of our problems and are thus of independent interest. In particular, we can use a generalization of Algorithm 4 to construct an exchange set for the *budgeted b-matching* problem. Also, using the techniques of [9], Algorithm 5 can be generalized to construct exchange sets for budgeted *multi-matroid intersection* for any constant number of matroids; this includes the *budgeted multi-dimensional matching* problem. While this problem does not admit a PTAS unless P=NP [10], our initial study shows that by constructing a representative set we may obtain an FPT-*approximation scheme* by parameterizing on the number of elements in the solution.[9]

Finally, to resolve the complexity status of BM and BI, the gripping question of whether the problems admit an FPTAS needs to be answered. Unfortunately, this may be a very difficult task. Even for special cases of a single matroid, such as graphic matroid, the existence of an FPTAS is still open. Moreover, a deterministic FPTAS for budgeted matching would solve deterministically the exact matching problem, which has been open for over four decades [14].

References

1 André Berger, Vincenzo Bonifaci, Fabrizio Grandoni, and Guido Schäfer. Budgeted matching and budgeted matroid intersection via the gasoline puzzle. *Mathematical Programming*, 128(1):355–372, 2011.

2 Paolo M. Camerini, Giulia Galbiati, and Francesco Maffioli. Random pseudo-polynomial algorithms for exact matroid problems. *Journal of Algorithms*, 13(2):258–273, 1992.

[8] A set system $(E, \mathcal{I})$ satisfies the k-exchange property if for all $A \in \mathcal{I}$ and $e \in E$ there is $B \subseteq A, |B| \leq k$, such that $(A \setminus B) \cup \{e\} \in \mathcal{I}$.

[9] We refer the reader, e.g., to [12] for the definition of parameterized approximation algorithms running in fixed-parameter tractable (FPT)-time.

3 Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Multi-budgeted matchings and matroid intersection via dependent rounding. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1080–1097. SIAM, 2011.

4 Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

5 Ilan Doron-Arad, Ariel Kulik, and Hadas Shachnai. An EPTAS for budgeted matching and budgeted matroid intersection. *arXiv preprint*, 2023. `arXiv:2302.05681`.

6 Ilan Doron-Arad, Ariel Kulik, and Hadas Shachnai. An EPTAS for budgeted matroid independent set. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 69–83, 2023.

7 Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

8 Fabrizio Grandoni and Rico Zenklusen. Approximation schemes for multi-budgeted independence systems. In *European Symposium on Algorithms*, pages 536–548. Springer, 2010.

9 Chien-Chung Huang and Justin Ward. FPT-algorithms for the ℓ-matchoid problem with a coverage objective. *arXiv preprint*, 2020. `arXiv:2011.06268`.

10 Viggo Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters*, 37(1):27–35, 1991.

11 Ariel Kulik and Hadas Shachnai. There is no EPTAS for two-dimensional knapsack. *Information Processing Letters*, 110(16):707–710, 2010.

12 Dániel Marx. Parameterized complexity and approximation algorithms. *The Computer Journal*, 51(1):60–78, 2008.

13 Ketan Mulmuley, Umesh V Vazirani, and Vijay V Vazirani. Matching is as easy as matrix inversion. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 345–354, 1987.

14 Christos H Papadimitriou and Mihalis Yannakakis. The complexity of restricted spanning tree problems. *Journal of the ACM (JACM)*, 29(2):285–309, 1982.

15 Ulrich Pferschy and Joachim Schauer. The knapsack problem with conflict graphs. *J. Graph Algorithms Appl.*, 13(2):233–249, 2009.

16 Ram Ravi and Michel X Goemans. The constrained minimum spanning tree problem. In *Scandinavian Workshop on Algorithm Theory*, pages 66–75. Springer, 1996.

17 Alexander Schrijver et al. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

18 Petra Schuurman and Gerhard J Woeginger. Approximation schemes – A tutorial. *Lectures on scheduling*, 2001.

19 Hadas Shachnai and Tami Tamir. Polynomial time approximation schemes. In *Handbook of Approximation Algorithms and Metaheuristics, Second Edition, Volume 1: Methologies and Traditional Applications*, pages 125–156. CRC Press, 2018.

Connected k-Center and k-Diameter Clustering

Lukas Drexler
Heinrich-Heine Universität Düsseldorf, Germany

Jan Eube
Universität Bonn, Germany

Kelin Luo
Universität Bonn, Germany

Heiko Röglin
Universität Bonn, Germany

Melanie Schmidt
Heinrich-Heine Universität Düsseldorf, Germany

Julian Wargalla
Heinrich-Heine Universität Düsseldorf, Germany

Abstract

Motivated by an application from geodesy, we study the *connected k-center problem* and the *connected k-diameter problem*. These problems arise from the classical k-center and k-diameter problems by adding a side constraint. For the side constraint, we are given an undirected *connectivity graph* G on the input points, and a clustering is now only feasible if every cluster induces a connected subgraph in G. Usually in clustering problems one assumes that the clusters are pairwise disjoint. We study this case but additionally also the case that clusters are allowed to be non-disjoint. This can help to satisfy the connectivity constraints.

Our main result is an $O(1)$-approximation algorithm for the disjoint connected k-center and k-diameter problem for Euclidean spaces of low dimension (constant d) and for metrics with constant doubling dimension. For general metrics, we get an $O(\log^2 k)$-approximation. Our algorithms work by computing a non-disjoint connected clustering first and transforming it into a disjoint connected clustering.

We complement these upper bounds by several upper and lower bounds for variations and special cases of the model.

2012 ACM Subject Classification Theory of computation → Facility location and clustering

Keywords and phrases Approximation algorithms, Clustering, Connectivity constraints

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.50

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2211.02176`

Funding This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 390685813; 459420781.

Acknowledgements The authors thank anonymous reviewers of a previous draft for helpful comments and pointing out relevant related work. We thank Jürgen Kusche and Christian Sohler for raising the problem and for fruitful discussion on the modeling. We also thank Xiangyu Guo for the discussion on the algorithm design and analysis.

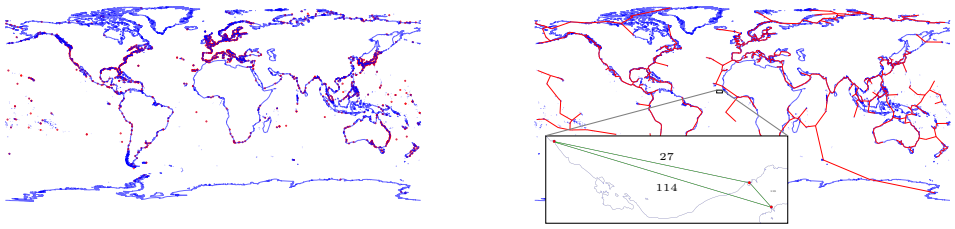


Figure 1 Gauge stations around the globe, with station location data from PSMSL (`http://www.psmsl.org/data/obtaining/`), plotted onto the map from the Natural Earth data set (`https://www.naturalearthdata.com/downloads/10m-physical-vectors/10m-coastline/`). Highlighted are three stations in Central America, and the numbers are Fréchet distances computed on the curves defined by sea levels between 1953 and 1968.

1 Introduction

Clustering problems occur in a wide range of application domains. Because of the general importance and interesting combinatorial properties, well-known k-clustering problems like k-center, k-median, and k-means have also been vastly studied in theory. These problems are NP-hard and APX-hard, but many constant-factor approximation algorithms for them are known. All k-clustering problems ask to partition a set of points (usually in a general metric space or in Euclidean space) into k clusters, often by picking k centers and assigning every point to its closest center. The clusters are then evaluated based on the distances between the points and their corresponding centers. For example in the case of k-center, the objective is to minimize the maximum distance between any point and its closest center.

In applications, clustering problems are often subject to side constraints. Consequently, clustering with side constraints has also become a thriving topic for designing approximation algorithms. Probably the most known example is clustering with capacities where the number of points in a cluster is limited. Notice how this constraint prevents us from assigning points to their closest center because there might not be enough space. So, for example, uniform capacitated (center-based) clustering consists of finding k centers and an assignment of points to those centers such that every center gets at most U points (and then evaluating the desired objective). Finding a constant factor approximation for uniform capacitated k-median clustering is a long standing open problem. Other constraints that have been studied are for example lower bounds (here, a cluster has to have a certain minimum number of points, so it may be beneficial to open less than k clusters) and clustering with outliers (here we are allowed $k + z$ clusters, but z of them have to be singletons, i.e. outliers). There are also results on constraints that restrict the choice of centers, for example by demanding that the centers satisfy a given matroid constraint. Among the newer clustering problems with constraints are those that evolve around aspects of fairness. These constraints are typically more complex and can either be point-based or center-based. Each constrained clustering problem, old or new, comes with a unique combinatorial structure, giving rise to a plethora of insights on designing approximation algorithms.

In this paper, we study a constraint that stems from the area of sea level geodesy but which is also of interest for other domains (discussed briefly below). For the application that motivated our work, consider the left picture in Figure 1. We see the location of tide gauge stations around the globe from the PSMSL data set [13, 9]. At every station, sea level heights have been collected over the years, constituting monthly time series. These records

can be used to reconstruct regional or global mean sea levels. However, the tide gauges have usually been constructed for practical purposes and not for sea level science. As a result, they are unevenly distributed over the globe. One way out of this is to replace clusters of tide gauges by representative records to thin out the data set. Our general goal is therefore to cluster the tide gauges into a given number k of clusters. However, the objective is not based on the gauge stations' geographic distance but on the time series. We wish to combine gauge stations with similar time series into one, i.e., when we cluster, we want to find clusters where the center's time series is similar to the records collected at the tide gauges represented by that center. We can model the distance between time series by a metric distance measure for time series or curves (like the Fréchet distance). As the objective we pick k-center, so we want to minimize the maximum distance between the center and the points that are replaced by it. Now we get to the complication: The gauge stations are *also* points on the map. We do not want to have points in the same cluster that are geographically very far away.

It is not immediately clear how to best model this scenario. We could resort to bicriteria approximation and look for solutions where both the time series of points in a cluster are similar and the radius of clusters is small, by either looking at the Pareto front or weighting the two objectives. Alternatively, we could fix a threshold and limit the geographic distance between centers and points, i.e., demand that a point x can only be assigned to center c if its geographic distance is at most some T. Both modelings have the drawback that they really only capture the distance on the map, while in reality, we would like to have somewhat coherent clusters that correspond to non-overlapping areas on the map. Indeed, we might be fine with having points of large geographic distance in the same cluster if all points 'between' them are also in the same cluster (i.e., that larger area of the sea behaves very similar with respect to the gauge station measurements).

The modeling that we study incorporates this via a preprocessing step. We assume that the points have been preprocessed such that we get a connectivity graph like shown on the right in Figure 1. The graph on the map was computed by finding a minimum spanning tree of the points, but it could be computed in other ways, too. The important part is that it captures a neighborhood structure. To model coherence, we now demand that clusters are *connected* in this graph. Figure 2 gives an example.

▶ **Problem 1.** *In a connected k-clustering problem, we are given points V, a metric d on V, a number k, and an unweighted and undirected connectivity graph $G = (V, E)$. A feasible solution is a partitioning of V into k clusters $C_1, \dots, C_k$ which satisfies that for every $i \in \{1, \dots, k\}$ the subgraph of G induced by C_i is* connected.

For the connected k-center problem, a solution also contains centers $c_1, \dots, c_k$ corresponding to the clusters $C_1, \dots, C_k$ and the objective is to minimize the maximum radius $\max_{i\in[k], x\in C_i} d(x, c_i)$. For the connected k-diameter problem the objective is to minimize the maximum diameter $\max_{i\in[k]} \max_{x,y\in C_i} d(x, y)$. It is easy to see that the connected k-clustering problem generalizes the classic k-center and k-diameter problem whose connectivity graph G is a complete graph.

Interestingly, the connected k-center problem was independently defined in an earlier paper by Ge et al. [4] (previously unknown to us. We thank the anonymous reviewer who pointed us to this reference). In that paper, connected clustering is motivated in the context of applications where both attribute and relationship data is present. It is applied to scenarios of community detection and gene clustering, showing the wide applicability of the modeling. We discuss their work further in the related work section.

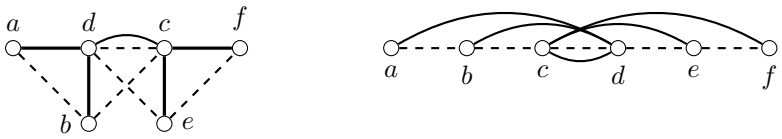


Figure 2 An example. The solid edges form the metric: Vertices connected by a solid edge have distance 1 and all other distances are 2. The dashed edges form the connectivity graph. Both pictures show the same graph. The optimal k-center solution with centers $\{c, d\}$ and clusters $\{a, b, d\}$ and $\{c, e, f\}$ is not connected. Any optimal (disjoint) connected k-center solution has radius 2.

Disjoint vs non-disjoint clusters, restricted graph classes

Notice that we demand that the C_i are *disjoint*. For some clustering problems with constraints the objective value can be decreased when we are allowed to assign points to more than one cluster: For example, lower bounds are easier to satisfy when points can be reused. The same is true for connected clustering: It is easier to satisfy connectivity when we can put important points into multiple clusters. For our application, we want to have disjoint clusters, but we still study the variation for completeness and also since it allows for better approximation algorithms that can be at least tested for their usefulness in the application (e.g., leaving it to the user to resolve overlaps). Notice that in Figure 2, allowing non-disjoint clusters enables the solution $\{c, d\}$ with clusters $\{a, b, c, d\}$, $\{c, d, e, f\}$ which has cost 1.

▶ **Definition 2.** *We distinguish between connected k-clustering* with disjoint clusters *and* with non-disjoint *clusters, referring to whether the clusters C_i have to be pairwise disjoint or not.*

Finally, we observe that in our application the connectivity graph is not necessarily arbitrary. Depending on the way that we build the graph, it could be a tree (the minimum spanning tree) or even a line (if we follow the coast line). Thus, we are interested in the problem on restricted graph classes as well.

Results and techniques

Our main result is an approximation algorithm that works for both the disjoint connected k-center problem and the disjoint connected k-diameter problem for general connectivity graphs G. For general metrics, the algorithm computes an $O(\log^2 k)$-approximation. If the metric has bounded doubling dimension D, the approximation ratio improves to $O(2^{3 \cdot D})$, and for Euclidean spaces, to $O(d \cdot 2^d)$. To obtain these results we first compute a non-disjoint clustering. Then we develop a method using a concept of a layered partitioning (see Definition 10) to make the clusters disjoint. We show how to obtain such a partitioning for different metrics. Both steps are novel and form the main contribution of this paper. In addition, in the full version of this paper we study how to compute well-separated partitions if the number of clusters is small, particularly when $k = 2$.

We also study restricted connectivity graphs (lines, stars and trees) and also the easier case of non-disjoint connected clustering. In this context we discuss greedy algorithms and obtain hardness results via reductions. The rest hardness proof in the full version of this paper is technically more involved. An overview of our results is given in Table 1, the more details are given in Section 2.

Table 1 An overview of the bounds shown in this paper and the literature for connected k-clustering. The notation $[\ell, u]$ stands for a lower bound ℓ and an upper bound u on the best possible approximation factor (achievable in polynomial time and assuming $P \neq NP$). Results marked by "*" are proven in the full version of this paper.

Restriction \ Objective	k-Center		k-Diameter	
	disjoint	non-disjoint	disjoint	non-disjoint
G is a line	1 Ge et al. [4]	1 Cor. 4	1 Cor. 4	
G is a star / tree		$[2, 2]$ Cor. 7, Lem. 9	$[2, 2]$ Lem. 5, Thm. 6	$[2, 2]$ Cor. 7, Lem. 9
Doubling dimension D	$O(2^{3D})$ Thm. 23		$O(2^{3D})$ Thm. 23	
L_p metric in dimension d	$O(d \cdot 2^d)$ Thm. 20		$O(d \cdot 2^d)$ Thm. 20	
No Restrictions	$[3^*, O(\log^2 k)]$ Thm. 18		$[2, O(\log^2 k)]$ Lem. 5, Thm. 18	

Related work

The k-center problem and the k-diameter problem are both NP-hard to approximate better than by a factor of 2 (see [10, 7] for k-center, k-diameter follows along the same lines). There are two popular 2-approximation algorithms for k-center which both also work for k-diameter with the same approximation guarantee [5, 8]. There are various results on side constraints for k-center and related k-clustering problems, including [1, 2, 3, 11] and many others. A more extensive list of results is contained in the full version of this paper, and we only review closely related work in the following. The connected k-center problem with disjoint clusters has been introduced and studied by Ge et al. [4][1]. Besides other results, Ge et al. present a greedy algorithm for the problem and claim that it computes a 6-approximation. In the full version of this paper we present an example showing that this greedy algorithm actually only obtains an $\Omega(k)$-approximation. The greedy algorithm is based on the approach of transforming a non-disjoint clustering into a disjoint one. In this transformation, it does not change the centers, i.e., it uses the given centers of the non-disjoint clustering also as centers for the disjoint clustering. In addition, we prove in the full version of this paper a lower bound showing that no algorithm based on transforming a non-disjoint clustering into a disjoint one with the same centers can compute an $O(1)$-approximation. Hence, without fundamental changes of the algorithm, no $O(1)$-approximation can be obtained. We even show that in general the optimal non-disjoint clustering can be better than the optimal disjoint clustering by a factor of $\Omega(\log \log k)$. Hence, if one uses only the radius of an optimal non-disjoint clustering as a lower bound for the radius of an optimal disjoint clustering, one cannot show a better approximation factor than $\Omega(\log \log k)$. To the best of our knowledge, no other approximation algorithms with provable guarantees for the connected k-center or k-diameter problem are known.

Ge et al. introduce the connected k-center problem to model clustering problems where both attribute and relationship data is present. They perform experiments in the context of gene clustering and community detection and demonstrate that for both these applications modelling them as connected clustering problems leads to superior results compared to standard clustering formulations without connectivity constraint. For community detection for example, they construct datasets from DBLP[2] where researchers are supposed to be

[1] We thank an anonymous reviewer for pointing us to this reference
[2] https://dblp.org/

clustered according to their main research area. Based on keyword frequencies they defined a distance measure for the researchers. At the same time, the coauthor network can be used as a connectivity graph. The advantage of connected clustering compared to traditional models is that it naturally takes into account both the distance measure and the coauthor network. For their experiments, Ge et al. develop a heuristic called NetScan for the connected k-center problem with disjoint clusters, which is reminiscent of the k-means method, and efficient on large datasets. In their experiments, the outcomes of this heuristic were significantly better than the outcomes of state-of-the-art clustering algorithms that take into account either only the distance measure or only the coauthor network. The work of Ge et al. has attracted some attention and it is cited in many other articles on community detection and related subjects.

Furthermore, Ge et al. show that already for $k = 2$, the connected k-center problem with disjoint clusters is NP-hard. They also argue that it is even NP-hard to obtain a $(2 - \epsilon)$-approximation for any $\epsilon > 0$. Additionally they give an algorithm based on dynamic programming with running time $O(n^2 \log n)$ that solves the connected k-center problem with disjoint clusters optimally when the connectivity graph is a tree.

Gupta et al. [6] study the connected k-median and k-means problem and prove upper and lower bounds on their approximability. Related to our motivation, Liao and Peng [12] consider the connected k-means problem to model clustering of spatial data with a geographic constraint. They develop a local-search based heuristic and conduct an experimental evaluation.

Outline

In Section 2 we discuss the general setting and results for restricted graph classes. Section 3 covers the case of non-disjoint connected clustering. Then in Section 4, we show the results on the connected clustering problems for general connectivity graphs and disjoint clusterings.

2 Setup and review of results on restricted graph classes

For all approximation algorithms in this paper, we use the following well-known framework for k-center approximation due to Hochbaum and Shmoys [8]. It is built upon the following fact: For the k-diameter or k-center problem (connected or not), the value of the cost function is always equal to one of the at most n^2 different distances between two points in V where $n = |V|$. This is true because it is either the distance between two points in the same cluster (k-diameter) or it is the distance between a point and its center (k-center). Thus, a standard scheme to follow is to sort these distances in time $O(n^2 \log n)$ and then search for the optimum value by binary search. The problem then reduces to finding a subroutine for the following task.

▶ **Problem 3.** *If there is a solution which costs r for a given r, find a solution that costs at most $\alpha \cdot r$. Otherwise, report that r is too small.*

An algorithm that solves this task can easily be turned into an α-approximation by searching for the smallest r for which the algorithm returns a solution. The running time of the resulting algorithm is $O(n^2 \log n)$ for the preprocessing plus $O(\log n)$ times the running time of the subroutine.

Lines, stars and trees

Connected k-clustering demands that the clusters are connected in a given connectivity graph G. How tricky is this condition? Maybe it can actually *help* to solve the problem? This is true if G is very simple, i.e., a line. We include the following proof as a warm-up.

▶ **Corollary 4.** *When the connectivity graph G is a line graph, then the connected k-center problem and the connected k-diameter problem can be solved optimally in time $O(n^2 \log n)$ both with disjoint and non-disjoint clusters. This is true even if the distances are not a metric.*

Proof. We only show how to solve the connected k-center problem with non-disjoint clusters. The full proof can be found in the full version of this paper. The line graph G is defined by vertices $V = \{v_1, v_2, ..., v_n\}$ and edges $E = \{\{v_i, v_{i+1}\} \mid i \in \{1, \dots, n-1\}\}$. Assume that r is given.

Notice that any connected cluster is a subpath of G. We start by precomputing for every v_i how far a cluster with center at v_i can stretch to the left and right: Let a_i be the smallest ℓ such that $d(v_j, v_{j'}) \leq r$ for all $j, j' \in \{\ell, \dots, i\}$ and let b_i be the largest ℓ such that $d(v_j, v_{j'}) \leq r$ for all $j, j' \in \{i, \dots, \ell\}$. We can compute all a_i and all b_i in time $O(n^2)$. Now we cut the line into clusters. We start by finding an index i with $a_i = 1$ for which b_i is as large as possible because we have to cover the first vertex and want to cover as many other vertices as possible. We place a center at v_i and know that all vertices until v_{b_i} are covered by the cluster. Now we know that the next cluster has to contain v_{b_i+1}, so we search for an i' which satisfies $b_i + 1 \in \{a_{i'}, \dots, b_{i'}\}$, if there are multiple, we take the one with maximum $b_{i'}$. This finds the center which covers v_{b_i+1} and the largest number of additional vertices. We place a center at $v_{i'}$. It may be that $i' < i$ as in Figure 2) and thus the clusters have to overlap (recall that we are in the non-disjoint case). The process is iterated until v_n is covered. If the number of clusters is more than k, we report that r was too small, otherwise, we report the clustering. This way we solve Problem 3 for $\alpha = 1$ in time $O(n^2)$. ◀

For trees, k-center and k-diameter differ. Surprisingly, the connected k-diameter problem is already NP-hard if G is a star. We prove the following lemma by a reduction from the uniform minimum multicut problem on stars in the full version of this paper.

▶ **Lemma 5.** *Let $\epsilon > 0$. Assuming $P \neq NP$, there is no $(2 - \epsilon)$-approximation algorithm for the connected k-diameter problem with disjoint clusters even if G is a star.*

Notice how the connected k-diameter problem with G being a star is thus very different from the k-diameter problem where the *metric* is given by a graph metric that is a star. The latter problem can be solved optimally by sorting the edges by weight and then deleting the $k - 1$ most expensive edges to form k connected components which form an optimal clustering. Say we have distances $d(e_1) \geq d(e_2) \geq \dots \geq d(e_n)$, then this optimal clustering has cost $d(e_k) + d(e_{k+1})$. However, any clustering that keeps an edge from $\{e_1, \dots, e_{k-1}\}$ costs at least $d(e_{k+1}) + d(e_{k-1}) \geq d(e_k) + d(e_{k+1})$ since it deletes at most $k - 1$ edges.

Ge et al. [4] show that the connected k-center problem is still solvable optimally for trees by dynamic programming.

▶ **Theorem 6** (Ge et al. [4])**.** *When the connectivity graph G is a tree, then the connected k-center problem with disjoint clusters can be solved optimally in time $O(n^2 \log n)$. This is true even if the distances are not a metric.*

It follows immediately that the connected k-diameter problem with disjoint clusters on trees can be 2-approximated by the same algorithm because the diameter of the produced solution is always at most twice the radius. This is interesting because our reduction in Lemma 5 shows that this is tight, i.e., using the dynamic programming algorithm for k-diameter achieves the best possible approximation ratio (assuming $P \neq NP$).

3 General G, non-disjoint clusters

The connected k-center and k-diameter problems with non-disjoint clusters behave similarly to the unconstrained versions. On the positive side, there is a 2-approximation; on the negative side, it is NP-hard to approximate these problems better than 2. In contrast to the case of disjoint clusters, APX-hardness starts with stars for *both* k-center and k-diameter. We show this via reductions from clique cover and set cover in the full version of this paper.

▶ **Corollary 7.** *Let $\epsilon > 0$. Assuming $P \neq NP$, there is no $(2-\epsilon)$-approximation algorithm for the connected k-diameter problem with non-disjoint clusters, even if G is a star. The same is true for the connected k-center problem with non-disjoint clusters.*

For the positive result, the classical result by Hochbaum and Shmoys [8] can be used. We discuss it in detail because we need it as a basis for our algorithms. For the unconstrained k-center problem, Problem 3 for $\alpha = 2$ can be solved as follows: Given input V, k, and a radius r, one picks an arbitrary point $x \in V$ and puts all nodes within distance $2r$ of x into one cluster. When r is at least the radius of the optimal k-clustering, this cluster will contain all nodes that are in the same optimal cluster as x. The cluster is then removed from V and the process is repeated until all nodes are covered. If the number of clusters is at most k, the solution is returned, otherwise, it is reported that r was too small.

This algorithm can easily be adapted to the connected k-center problem with non-disjoint clusters by the following observation: Let x and y be two nodes from the same optimal cluster with center c and radius r. Then x and y are connected in the connectivity graph by a path that contains only nodes within distance $2r$ from x and y. So the algorithm is: When a node x is selected, put all nodes into a cluster that have distance at most $2r$ from x and are reachable from x in the connectivity graph via a path on which all nodes have a distance of at most $2r$ from x. This set can be determined by the BFS-type algorithm `ComputeCluster` (see Algorithm 1 with $R = 2r$). Say the resulting cluster is T. Do not remove T from G

Algorithm 1 ComputeCluster(G, M, R, c).

Input: points V, graph $G = (V, E)$, metric $M = (V, d)$, radius R, node $c \in V$
$T \leftarrow \{c\}$;
$N \leftarrow \{u \in V \setminus T \mid \exists v \in T, (v, u) \in E : d(u, c) \leq R\}$;
while $N \neq \emptyset$ **do**
  $T \leftarrow T \cup N$;
  $N \leftarrow \{u \in V \setminus T \mid \exists v \in T, (v, u) \in E : d(u, c) \leq R\}$;
Output: cluster T

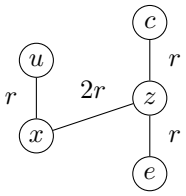
but only mark all nodes in T as covered. As long as there are uncovered nodes, pick an arbitrary such node and form a cluster of radius $2r$ around it (in general this cluster will also contain nodes that are already covered). This will result in at most k connected clusters with radius $2r$ if r is at least the radius of an optimal connected k-clustering. We call this algorithm `GreedyClustering` and we give its pseudocode as Algorithm 2. In general, the sets T_c computed by this algorithm are not disjoint but the centers are pairwise distinct.

▶ **Lemma 8.** *Let r^* denote the radius of an optimal connected k-center clustering with non-disjoint clusters. For $r \geq 2r^*$, Algorithm 2 computes a center set C with $|C| \leq k$.*

Proof. Consider a node $c \in V$ that is chosen as a center by the algorithm and the optimal cluster O node c is contained in. This cluster is centered around some node c' and has a radius of at most r^*. Hence, by the triangle inequality all nodes in O have a distance of at

Algorithm 2 GreedyClustering(G, M, r).

Input: graph $G = (V, E)$, metric $M = (V, d)$, radius r
$C \leftarrow \emptyset$; `// center nodes`
$V' \leftarrow V$; `// uncovered nodes`
while $V' \neq \emptyset$ **do**
  select a node $c \in V'$ and add it to C;
  $T_c \leftarrow$ ComputeCluster(G, M, r, c);
  $V' \leftarrow V' \setminus T_c$;
Output: centers C, sets T_c for all $c \in C$



The optimal connected 2-clustering has centers x and z with clusters $\{x, u\}$ and $\{z, c, e\}$ and a radius of r. The greedy algorithm started with x forms $\{x, u, z\}$ as the first cluster. After that, only c and e remain. Without z, they are not connected anymore and have to go into different clusters.

Figure 3 An example where greedy disconnects an optimum cluster.

most $2r^*$ from c. Also since O is connected, all nodes in O are reachable from c. In particular, all nodes in O are reachable from c on paths that contain only nodes within distance $2r^*$ of c. This implies that for $r \geq 2r^*$, the set T_c is a superset of the optimal cluster O. Since the centers in Algorithm 2 are chosen among the uncovered nodes, all chosen centers must be from distinct optimal clusters. This implies that there can be at most k centers in C. ◀

The same algorithm works for the connected k-diameter problem when `ComputeCluster` is evoked with $R = r$ (not $2r$) if r is at least the optimal diameter. By adding all points in distance r to the cluster of the chosen center x, it is ensured that the optimum cluster is added if r is at least the optimum value (since the distance between two points is then at most r). Furthermore, the resulting cluster has diameter at most $2r$ by the triangle inequality.

▶ **Lemma 9.** *There exists a 2-approximation algorithm for the connected k-center problem with non-disjoint clusters and also for the connected k-diameter problem with non-disjoint clusters.*

4 General G, disjoint clusters

The disjoint case for general connectivity graphs is more challenging. To keep the presentation simple, we focus in the following on the connected k-center problem: Given an unweighted graph $G = (V, E)$ and a metric space $M = (V, d)$ with $d : V \times V \to \mathbb{R}$, find k node-disjoint connected subgraphs of G (clusters) that cover all vertices and minimize the maximum radius of these subgraphs. An adaptation to the connected k-diameter problem can be found in the full version of this paper.

We start with the algorithm `GreedyClustering` from the previous section on the non-disjoint case. Notice that in general, the output of this algorithm is not node-disjoint. We could opt to delete the nodes in T computed by Algorithm 1 to enforce disjointness, however, the problem is this: The first cluster that the algorithm forms around a vertex x is guaranteed to be a superset of the optimal cluster that x is contained in. It might be a strict superset and contain a node that belongs to a different optimal cluster. This node will get removed from G together with all other nodes in the cluster around x. However, its removal might

make the optimal cluster it is contained in unconnected. This is problematic because then k connected clusters might not suffice anymore to cover all points from G even if we guessed the optimal radius r correctly. See Figure 3 for an example where this happens.

4.1 Making the clusters disjoint

In this section we describe how to transform the set of non-disjoint clusters computed by `GreedyClustering` into a set of pairwise disjoint clusters that cover all points at the cost of increasing the radius or diameter. This transformation has to be performed very carefully in order to not increase the radius or diameter by too much.

Let C with $|C| \leq k$ denote the set of centers around which the non-disjoint clusters have been formed by the algorithm and let r denote their radius. The following two observations are helpful: (1) When two centers are more than $2r$ apart then their corresponding clusters are disjoint. (2) If a set of centers have pairwise distance at most L then merging the corresponding clusters results in a single cluster with radius at most $r + L$ and diameter at most $2r + L$.

If it is possible to partition the centers into groups such that all centers within the same group have a distance of at most L and all centers from different groups have a distance of more than $2r$, we could make the clusters disjoint as follows: as long as there are two non-disjoint clusters whose centers are in the same group of the partition, merge them into a single cluster. In the end, the algorithm will return no more than $|C| \leq k$ clusters. By isolating some singletons as new clusters, we obtain a solution with exactly k clusters as required without worsening the solution. After this, all clusters whose centers are in the same group are disjoint (if not they would have been merged) and clusters whose centers are in different groups are disjoint because their centers are far enough from each other. Hence, such a partition results in a solution with disjoint clusters with radius $r + L$ and diameter $2r + L$. A key idea in our algorithm for the general case is to find such a partition of the centers in C with small L. However, observe that this is not possible in general. A simple counterexample would be that all centers are equally spaced on a line with distance r between two consecutive centers. Then all centers have to be in the same group and L would be $(k-1)r$, resulting in an approximation factor of $\Omega(k)$.

To circumvent this problem, we do not partition all centers from C at once but we start with a partition of a subset of C that satisfies the properties above (i.e., centers in the same group have distance at most L, while centers in different groups have a distance of more than $2r$). We call this the first layer of the partition. Then we remove all centers contained in the first layer from C and proceed with the remaining centers analogously: Let C' denote the set of centers not contained in the first layer. We find a partition of a subset of C' that satisfies the properties above and call this the second layer of the partition. We repeat this process until all points from C are in some layer. We call such a partition a *well-separated partition*. Figure 4 shows possible partitions for the example above.

▶ **Definition 10.** *Let $M = (C, d)$ be a metric and $r > 0$. An* r-well-separated partition *with $\ell \in \mathbb{N}$ layers and with parameters $(h_1, \ldots, h_\ell)$ is a partition of C into groups $\{C_{1,1}, \ldots, C_{1,\ell_1}\}, \{C_{2,1}, \ldots, C_{2,\ell_2}\}, \ldots, \{C_{\ell,1}, \ldots, C_{\ell,\ell_\ell}\}$ with the following properties.*

- **(i)** *The groups cover all points from C, i.e., $\bigcup_{i\in[\ell], j\in[\ell_i]} C_{i,j} = C$.*
- **(ii)** *The groups are pairwise disjoint, i.e., $\forall i, i', j, j'$ with $i \neq i'$ or $j \neq j'$, $C_{i,j} \cap C_{i',j'} = \emptyset$.*
- **(iii)** *For $i \in [\ell]$, we call the sets $C_{i,1}, \ldots, C_{i,\ell_i}$ the* sets on layer i. *Two different sets from the same layer are more than $2r$ away, i.e., $\forall i \in [\ell], v \in C_{i,j}, v' \in C_{i,j'}$ with $j \neq j'$, $d(v, v') > 2r$.*
- **(iv)** *For $i \in [\ell]$, the maximum diameter of a group on layer i is at most h_i, i.e., $\max_j \max_{v,v'\in C_{i,j}} d(v, v') \leq h_i$.*

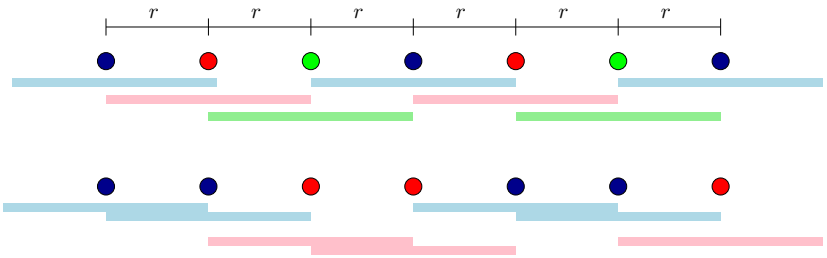


Figure 4 We consider an instance with 7 centers on a line where consecutive centers have a distance of r. The top figure shows a well-separated partition of this instance with $L = 0$ and $\ell = 3$ layers. The colors depict the different layers and the colored rectangles depict the clusters of radius R around these centers. On the blue layer there are, e.g., three groups where each group consists of a single blue center. The bottom figure shows a well-separated partition of the same instance with $L = r$ and $\ell = 2$. The blue layer contains two groups of two centers each, while the red layer contains two groups, one with two centers and one with only one center.

It is not clear at first glance why a well-separated partition is helpful for obtaining a solution with disjoint clusters. For every layer of the partition, we can use the reasoning above. That is, we merge all non-disjoint clusters whose centers are in the same group to obtain disjoint clusters with radius $r + L$ and diameter $2r + L$. However, a cluster is then only disjoint from all clusters on the same layer but in general not from clusters on other layers (see Figure 4). A main ingredient of our algorithm is a non-trivial way to merge clusters on different layers. For this, we add the layers one after another. Consider the case of two layers. The clusters from the first layer are disjoint from each other. We add the clusters of the second layer one after another. For each cluster from the second layer, we first check with which clusters from the first layer it overlaps. If there is more than one, we split the cluster from the second layer into multiple parts and merge the parts with different clusters from the first layer with which they overlap. This is done in such a way that the final result is a set of disjoint connected clusters. We prove with an inductive argument that the radius and diameter of these clusters is $O(\ell \cdot L)$, where ℓ denotes the number of layers of the well-separated partition.

The following lemma describes an algorithm that adjusts the clusters layer by layer to make them pairwise disjoint.

▶ **Lemma 11.** *Consider an instance $(G = (V, E), M = (V, d), k)$ of the connected k-center problem and assume that Algorithm 2 computes a center set $C \subseteq V$ with $|C| \leq k$ for some radius r. Furthermore, let an r-well-separated partition of C with ℓ layers and parameters $(h_1, \ldots, h_\ell)$ be given. Then we can efficiently find a feasible solution for the connected k-center problem with disjoint clusters with radius at most $(2\ell - 1)r + \sum_{i=1}^{\ell} h_i$.*

Proof. According to Definition 10 and Algorithm 2, we have the following properties:

(i) $\bigcup_{i\in[\ell],j\in[\ell_i]} C_{i,j} = C$
(ii) $\forall i, i', j, j'$ with $i \neq i'$ or $j \neq j'$: $C_{i,j} \cap C_{i',j'} = \emptyset$
(iii) $\forall i \in [\ell], c \in C_{i,j}, c' \in C_{i,j'}$ with $j \neq j'$: $d(c, c') > 2r$ and $T_c \cap T_{c'} = \emptyset$
(iv) $\forall i \in [\ell]$, $j \in [\ell_i]$, $c, c' \in C_{i,j}$: $d(c, c') \leq h_i$
(v) $\bigcup_{i\in[\ell],j\in[\ell_i]} \bigcup_{c\in C_{i,j}} T_c = V$

In the first step, we adjust the clusters by merging all non-disjoint clusters whose centers belong to the same group. To be precise, for each group $C_{i,j}$ we do the following: As long as there are two different centers $c \in C_{i,j}$ and $c' \in C_{i,j}$ with $T_c \cap T_{c'} \neq \emptyset$, we remove c' from

$C_{i,j}$ and replace T_c by $T_c \cup T_{c'}$. That is, we merge the two clusters T_c and $T_{c'}$ and define c as its center. Since centers in the same group on layer i have a distance of at most h_i, after this step the clusters in each group $C_{i,j}$ are pairwise disjoint and have a radius of at most $r + h_i$ and a diameter of at most $2r + h_i$. They are still connected because we only merge connected clusters that have at least one node in common.

Since clusters in different groups of the same layer are pairwise disjoint anyway, all clusters on the same layer are pairwise disjoint after this step. Hence, in the next step we only need to describe how clusters from different layers can be made disjoint. For this, it will be helpful to view the clusters as trees. To make this more precise, consider a cluster T_c with center c. We know that the subgraph of G induced by T_c is connected. For any cluster T_c we choose an arbitrary spanning tree in this induced subgraph and consider c to be the root of this tree. Let $\mathcal{T}_i$ denote the set of all such trees in the i-th layer for $i \in [\ell]$. In the following we will use the terms clusters and trees synonymously. By abuse of notation we will use T_c to denote both the cluster with center c and the spanning tree with root c, depending on the context.

For every $i \in [\ell]$, all trees in $\mathcal{T}_i$ are node-disjoint. We will now describe how to ensure that trees on different layers are also node-disjoint. For this, we will go through the layers $i = 1, 2, \ldots, \ell$ in this order and replace $\mathcal{T}_i$ by an adjusted set of trees $\mathcal{T}'_i$. We will construct these trees so that at each step $i \in [\ell]$ all trees from $\cup_{j\in[i]}\mathcal{T}'_j$ are pairwise disjoint. Furthermore, at step i the radius of any tree from $\cup_{j\in[i]}\mathcal{T}'_j$ will be bounded from above by $(2i-1)r + \sum_{j\in[i]} h_j$. Finally, our construction ensures that in the end, the trees in $\cup_{i\in[\ell]}\mathcal{T}'_i$ cover all nodes in V. Hence, these trees form a feasible solution to the connected k-center problem with disjoint clusters with the desired radius.

We set $\mathcal{T}'_1 = \mathcal{T}_1$. Then for $i = 1$, the desired properties are satisfied because the trees on layer 1 are pairwise disjoint and have a radius of at most $r + h_1$. Now assume that the properties are true for some i and let us discuss how to ensure them also for $i + 1$. We start with $\mathcal{T}'_{i+1} = \emptyset$ and add trees to it one after another. Consider an arbitrary tree $T \in \mathcal{T}_{i+1} = (V', E')$ with center c and let $V^* \subseteq V'$ denote the nodes that also occur in some tree $T' \in \mathcal{T}'_j$ for some $j \in [i]$. Observe that any node from V^* can be contained in at most one such tree T' because by the induction hypothesis all trees in $\cup_{j\in[i]}\mathcal{T}'_j$ are pairwise disjoint. If V^* is empty then the tree T is disjoint from all trees in $\cup_{j\in[i+1]}\mathcal{T}'_j$ and does not need to be adjusted. In this case we simply add it to $\mathcal{T}'_{i+1}$.

If V^* contains only a single node v then we merge the tree T with the unique tree T' from $\mathcal{T}'_j$ for some $j \leq i$ that also contains node v, i.e., we replace T' by $T \cup T'$ in $\mathcal{T}'_j$. Tree T' has a radius of at most $(2i-1)r + \sum_{j\in[i]} h_j$. Since the diameter of T is at most $2r + h_{i+1}$, the radius of the union of T and T' with respect to the center of T' is at most (see Figure 5)

$$(2r + h_{i+1}) + (2i-1)r + \sum_{j\in[i]} h_j = (2(i+1)-1)r + \sum_{j\in[i+1]} h_j. \tag{1}$$

Now consider the case that V^* contains more than one node. In this case we cannot simply merge T with some tree from $\cup_{j\in[i]}\mathcal{T}'_j$ because the resulting tree would not be disjoint from the other trees. We also cannot merge all trees that contain nodes from V' into a single cluster because the radius of the resulting cluster could be too large. Instead we split the tree T into multiple components and we merge these components separately with different trees from $\cup_{j\in[i]}\mathcal{T}'_j$. For each node $v \in V^*$ that is not the root c of T we consider the path from c to v and let e denote the last edge on this path (i.e., the edge leading to v). We remove edge e from the tree T and thereby split the tree T into two components. Since we do this for every node from $V^* \setminus \{c\}$, the tree T will be split into $|V^* \setminus \{c\}| + 1$ pairwise disjoint connected components. Each of these components that does not contain the root c contains

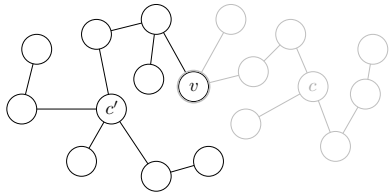


Figure 5 This figure shows the tree T' with center c' in black and the tree T with center c in gray. These trees have node v in common. When T and T' are merged into a single tree, the radius of this new tree with respect to c' is larger than the radius of T' by at most the diameter of T.

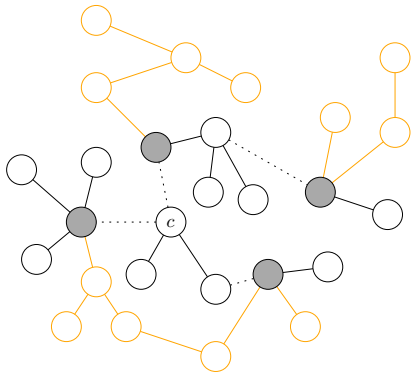


Figure 6 This figure shows the tree T in black. The nodes in V^* are marked gray and the edges that are removed from T are shown dotted. The orange trees depict the trees on lower layers that contain the nodes from V^* and with which the corresponding components are merged.

exactly one node from V^*. Hence, for each of these components there is a unique tree from $\cup_{j\in[i]}\mathcal{T}'_j$ from which it is non-disjoint. We merge every component with the tree from which it is non-disjoint (see Figure 6). In the component that contains the root, only the root might belong to V^*. If this is the case, we merge it with the unique tree from $\cup_{j\in[i]}\mathcal{T}'_j$ from which it is non-disjoint. Otherwise, we add this component to $\mathcal{T}'_{i+1}$. Since T has a diameter of at most $2r + h_{i+1}$, the same is true for each of the components. By the induction hypothesis, each tree from $\cup_{j\in[i]}\mathcal{T}'_j$ has a radius of at most $(2i-1)r + \sum_{j\in[i]} h_j$. Hence, as in (1), the radius of the merged clusters is bounded from above by $(2(i+1)-1)r + \sum_{j\in[i+1]} h_j$. ◀

▶ **Corollary 12.** *If there exists a polynomial-time algorithm that computes for any metric (C, d) and any r an r-well-separated partition with ℓ layers and parameters $(h_1, \ldots, h_\ell)$ then there exists an approximation algorithm for the connected k-center problem with disjoint clusters that achieves an approximation factor of $4\ell - 2 + 2\sum_{i=1}^{\ell} h_i/r$.*

Proof. To obtain the desired approximation factor, we first determine the smallest r for which Algorithm 2 returns a center set C with $|C| \leq k$. Due to Lemma 8, this radius r will be at most $2r^*$, where r^* denotes the radius of an optimal connected k-clustering with non-disjoint clusters. Let r^*_D denote the radius of an optimal connected k-clustering with disjoint clusters. Then $r^*_D \geq r^* \geq r/2$. According to Lemma 11, the polynomial-time algorithm for computing an r-well-separated partition can then be used to compute a connected k-clustering with disjoint clusters and radius at most $(2\ell-1)r + \sum_{i\in[\ell]} h_i$. The approximation factor of this k-clustering is

$$\frac{(2\ell-1)r + \sum_{i\in[\ell]} h_i}{r^*_D} \leq \frac{(2\ell-1)r + \sum_{i\in[\ell]} h_i}{r/2} = 4\ell - 2 + 2\sum_{i\in[\ell]} \frac{h_i}{r}.$$ ◀

The same algorithm that we developed in this sections for the connected k-center problem can also be used for the connected k-diameter problem without any modifications. Only the analysis of the approximation factor needs to be adapted slightly.

Lemma 8 is changed as follows.

▶ **Lemma 13.** *Let r^* denote the diameter of an optimal connected k-diameter clustering with non-disjoint clusters. For $r \geq r^*$, Algorithm 2 computes a center set C with $|C| \leq k$.*

Observe that the diameter of the clusters T_c that are computed by Algorithm 2 for some r can be at most $2r$.

A straightforward adaption of Lemma 11 yields the following result.

▶ **Lemma 14.** *Consider an instance $(G = (V, E), M = (V, d), k)$ of the connected k-diameter problem and assume that Algorithm 2 computes a center set $C \subseteq V$ with $|C| \leq k$ for some radius r. Furthermore, let an r-well-separated partition of C with ℓ layers and parameters $(h_1, \ldots, h_\ell)$ be given. Then we can efficiently find a feasible solution for the connected k-diameter problem with disjoint clusters with diameter at most $(4\ell - 2)r + h_1 + 2\sum_{i=2}^{\ell} h_i$.*

Overall we obtain the following corollary.

▶ **Corollary 15.** *If there exists a polynomial-time algorithm that computes for any metric (C, d) and any r an r-well-separated partition with ℓ layers and parameters $(h_1, \ldots, h_\ell)$ then there exists an approximation algorithm for the connected k-diameter problem with disjoint clusters that achieves an approximation factor of $4\ell - 2 + h_1/r + 2\sum_{i=2}^{\ell} h_i/r$.*

4.2 Finding well-separated partitions

With the discussion above, finding a good approximation algorithm is reduced to finding an efficient algorithm for computing an r-well-separated partition with small L and few layers. For general metrics, we present an efficient algorithm that computes a well-separated partition with $L = O(r \cdot \log k)$ and $\ell = O(\log k)$. This yields a clustering of radius and diameter $O(r \cdot \log^2 k)$. Details can be found in the proof of Theorem 18 in the next section. We give better results for computing well-separated partitions for L_p-metrics and metric spaces with bounded doubling dimension in Theorem 20 and Theorem 23. Overall, we get the following results.

▶ **Theorem 16.** *There exists an $O(\log^2 k)$-approximation algorithm for the connected k-center problem with disjoint clusters and for the connected k-diameter problem with disjoint clusters. The approximation ratio improves*

- *to $O(2^{3 \cdot \dim(M)})$ if the metric space has bounded doubling dimension $\dim(M)$, and*
- *to $O(d \cdot 2^d)$ if the distance is an L_p-metric in $\mathbb{R}^d$.*

It is an intriguing question if better well-separated partitions exist for general metrics and for the special metrics that we have considered. By our framework, better partitions would immediately give rise to better approximation factors.

We prove in the full version a lower bound of $\Omega(\log \log k)$ for our algorithmic framework. To be precise, we construct an instance in a general metric space together with a set of k centers C that could be produced by the algorithm `GreedyClustering` such that even the optimal disjoint solution with centers C is worse than the optimal disjoint solution for arbitrary centers by a factor of $\Omega(\log \log k)$. Hence, to prove a constant-factor approximation in general metric spaces, one cannot rely on the centers chosen by `GreedyClustering`.

4.2.1 Well-separated partitions in general metrics

According to Corollaries 12 and 15, we only need to find an efficient algorithm for computing an r-well-separated partition to obtain an approximation algorithm for the connected k-center and k-diameter problem.

Algorithm 3 computes an r-well separated partition layer by layer. For each layer it creates the groups in a greedy fashion: At the beginning of a layer i, the set U' of all nodes that are not assigned to previous layers is considered. The goal is to assign as many of these nodes to the current layer i as possible. For this, we start with an arbitrary node $u \in U'$ that is not assigned to any previous layer and we create a group around u. First the group consists only of u itself. Then we iteratively augment the group by adding all nodes to the group that have a distance of at most $2r$ from some node that already belongs to the group. We repeat this augmentation step multiple times one after another. We stop when the number of new nodes that join the group is smaller than twice the number of nodes that have already been added to the group for the first time. Then the group around u is finished and added to layer i. All nodes in the group are removed from U'. Furthermore, we also remove all nodes that have a distance of at most $2r$ from this group from U'. These nodes have to be assigned to other layers that are created later to ensure property (iii) in Definition 10. As long as U' is not empty, we repeat the process to create another group on layer i. The pseudocode is shown as Algorithm 3.

Algorithm 3 PartitionGeneralMetric$((C, d), r)$.

Input: metric (C, d), radius r
$U \leftarrow C$; // nodes that still have to be assigned
$i \leftarrow 0$;
while $U \neq \emptyset$ **do**
  $i \leftarrow i + 1$; // start a new layer
  $j \leftarrow 0$;
  $U' \leftarrow U$; // nodes that could still be assigned on i-th layer
  while $U' \neq \emptyset$ **do**
    $j = j + 1$; // create a new group in i-th layer
    select a node $u \in U'$, $C_{i,j} \leftarrow \{u\}$ and $N_0(u) \leftarrow \{u\}$;
    $U' \leftarrow U' \setminus \{u\}$;
    $U \leftarrow U \setminus \{u\}$;
    $s = 1$;
    while $s \neq 0$ *and* $U' \neq \emptyset$ **do**
      $N_s(u) \leftarrow \{x \in U' \mid \exists v \in N_{s-1}(u) : d(v, x) \leq 2r\}$;
      // nearby nodes of nodes $C_{i,j}$ in U'
      if $|N_s(u)| \geq 2 \cdot |C_{i,j}|$ **then**
        $C_{i,j} \leftarrow C_{i,j} \cup N_s(u)$; // add nearby nodes to $C_{i,j}$
        $U' \leftarrow U' \setminus N_s(u)$;
        $U \leftarrow U \setminus N_s(u)$;
        $s = s + 1$;
      else
        $U' \leftarrow U' \setminus N_s(u)$; // nearby nodes cannot be on i-th layer
        $s = 0$; // end group of node u
Output: $\{C_{1,1}, C_{1,2}, \ldots\}, \{C_{2,1}, C_{2,2}, \ldots\}, \ldots$

▶ **Lemma 17.** *Let (C, d) be an arbitrary metric with $k := |C|$ and $r > 0$. Let $\ell = 1 + \lfloor \log_{\frac{3}{2}}(k) \rfloor$ and $h = 4r\lfloor \log_3 k \rfloor$. The output of Algorithm 3 is an r-well-separated partition with at most ℓ layers and parameters $(h, \dots, h)$.*

Proof. Let $\{C_{1,1}, \dots, C_{1,\ell_1}\}, \{C_{2,1}, \dots, C_{2,\ell_2}\}, \dots, \{C_{\ell,1}, \dots, C_{\ell,\ell_\ell}\}$ denote the output of Algorithm 3. The algorithm ensures that every point from C is contained in exactly one group $C_{i,j}$ because when nodes are deleted from U in Line 18 they have been added to $C_{i,j}$ in Line 16. Furthermore U' is always a subset of U and so no node can be assigned to multiple clusters. Furthermore, Lines 14 and 21 ensure that nodes in different groups of the same layer are more than $2r$ apart. This shows that the properties (i), (ii), and (iii) in Definition 10 are satisfied.

Next we show property (iv) that the maximum diameter of every group is h. As long as the number of nearby nodes in $N_s(u)$ is at least twice the number of the previously grouped nodes in $\cup_{t=1}^{s-1} N_t(u)$, we add these nearby nodes to the current group. As long as this is true we have

$$|N_s(u)| \geq 2 \cdot \sum_{t=0}^{s-1} |N_t(u)|.$$

Together with $|N_0(u)| = 1$, this implies $|\cup_{t=1}^{s} N_t(u)| \geq 3^s$ for every s by a simple inductive argument. Since this set cannot contain more than $k = |C|$ nodes, we have $C_{i,j} = \bigcup_{s=1}^{h} N_s(u)$ for some $h \leq \lfloor \log_3 k \rfloor$. For any $s \geq 1$, any node in $N_s(u)$ has a distance of at most $2r$ from some node in $N_{s-1}(u)$. Since u is the only node in $N_0(u)$, this implies that any node has a distance of at most $2rh$ from u. Hence, the diameter of every group is at most $4rh \leq 4r\lfloor \log_3 k \rfloor$. This shows property (iv) in Definition 10.

Now it only remains to bound the number of layers of the partition. When a new layer is started, U' is set to U, the set of yet unassigned nodes in Line 6. When a group is formed then its current neighbors $N_s(u)$ get removed from U' in Line 21. These are exactly the nodes that do not get assigned to the current layer and have to be assigned to other layers afterwards. Since line 21 is only reached if $|N_s(u)|$ is smaller than twice $|C_{i,j}|$, at least one third of the initially unassigned nodes get assigned to groups on the current layer and at most two thirds are postponed to other layers afterwards. This implies that after ℓ layers, there are no more than $(\frac{2}{3})^\ell \cdot k$ nodes left to be assigned. Hence, the number of layers cannot be more than $1 + \lfloor \log_{\frac{3}{2}}(k) \rfloor$. ◀

Based on Corollaries 12 and 15, it is now easy to prove the following theorem.

▶ **Theorem 18.** *There exists an $O(\log^2 k)$-approximation algorithm for the connected k-center problem and for the connected k-diameter problem with disjoint clusters.*

Proof. According to Lemma 17, one can efficiently compute for any metric an r-well-separated partition with at most ℓ layers and parameters $(h, \dots, h)$ for $\ell = 1 + \lfloor \log_{\frac{3}{2}}(k) \rfloor = O(\log k)$ and $h = 4r\lfloor \log_3 k \rfloor = O(r \cdot \log k)$.

By Corollary 12 this implies that we can efficiently find a solution for the connected k-center problem with disjoint clusters with approximation factor

$$4\ell - 2 + 2\sum_{i=1}^{\ell} h/r = O(\ell + \ell h/r) = O(\log k + \log^2 k) = O(\log^2 k).$$

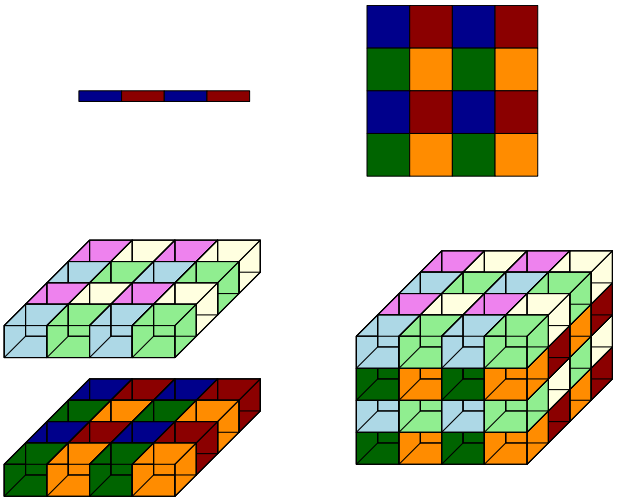


Figure 7 In the upper row, colorings for $d = 1$ and $d = 2$ are shown. In the lower row on the right, a coloring for $d = 3$ is shown. It is composed of alternatingly using the 2-dimensional colorings shown on the left.

By Corollary 15, it also implies that we can efficiently find a solution for the connected k-diameter problem with disjoint clusters with approximation factor

$$4\ell - 2 + h_1/r + 2\sum_{i=2}^{\ell} h_i/r = O(\ell + \ell h/r) = O(\log k + \log^2 k) = O(\log^2 k). \quad \blacktriangleleft$$

4.2.2 Well-separated partitions in Euclidean metrics

In this section, we study how to compute an r-well-separated partition if the metric is an L_p-metric in the d-dimensional space $\mathbb{R}^d$ for some $p \in \{1, 2, \ldots, \infty\}$.

▶ **Lemma 19.** *For any L_p-metric in $\mathbb{R}^d$, an r-well-separated partition with 2^d layers and parameters $(h, \ldots, h)$ with $h = 3d^{1/p}r$ can be computed in polynomial time.*

Proof. First we partition the space $\mathbb{R}^d$ into d-dimensional hypercubes with side length $3r$. These hypercubes are chosen such that they are pairwise disjoint and that they cover the entire space. Then we color these hypercubes such that no two neighboring hypercubes get the same color where also diagonal neighbors are taken into account (see Figure 7). Based on this coloring we create the following r-well-separated partition: each color corresponds to one layer of the partition and within a layer all nodes that belong to the same hypercube form a group. Since the distance of two hypercubes of the same color is at least $3r$, property (iii) of Definition 10 is satisfied. Properties (i) and (ii) are satisfied because the hypercubes partition the space $\mathbb{R}^d$. Finally, the diameter of any of the hypercubes is bounded from above by $(\sum_{i=1}^{d}(3r)^p)^{1/p} = 3d^{1/p}r$, which also proves property (iv).

It remains to bound the number of layers, i.e., the number of different colors necessary to color the hypercubes. One can prove by induction that 2^d colors are sufficient. For $d = 1$, one simply colors the hypercubes alternatingly with two different colors. For $d \geq 2$, we first pick two different colorings of $\mathbb{R}^{d-1}$ with 2^{d-1} colors each such that the two colorings do not have a color in common. Then we color the hypercubes in $\mathbb{R}^d$ by alternatingly using one of the two $(d-1)$-dimensional colorings. This way, we obtain a coloring of the hypercubes in $\mathbb{R}^d$ with 2^d colors (see Figure 7). ◀

Based on Corollaries 12 and 15, it is now easy to prove the following theorem.

▶ **Theorem 20.** *For any L_p-metric in $\mathbb{R}^d$, there exists an $O(d \cdot 2^d)$-approximation algorithm for the connected k-center problem and for the connected k-diameter problem with disjoint clusters.*

Proof. According to Lemma 19, we can efficiently compute an r-well-separated partition with 2^d layers and parameters $(h, \dots, h)$ for $h = 3d^{1/p}r$.

By Corollary 12 this implies that we can efficiently find a solution for the connected k-center problem with disjoint clusters with approximation factor

$$4\ell - 2 + 2\sum_{i=1}^{\ell} h/r = O(d \cdot 2^d).$$

By Corollary 15, it also implies that we can efficiently find a solution for the connected k-diameter problem with disjoint clusters with approximation factor

$$(4\ell - 2) + h_1/r + 2\sum_{i=2}^{\ell} h_i/r = O(d \cdot 2^d).$$ ◀

4.2.3 Well-separated partitions in metrics with small doubling dimension

In this section, we study how to compute an r-well-separated partition if the metric has constant doubling dimension. This generalizes Lemma 19 for Euclidean spaces.

▶ **Definition 21** (doubling dimension)**.** *The* doubling constant *of a metric space $M = (X, d)$ is the smallest number k such that for all $x \in X$ and $r > 0$, the ball $B_r(x) := \{y \in X \mid d(x, y) \leq r\}$ can be covered by at most k balls of radius $r/2$, i.e.,*

$$\forall x \in X : \forall r > 0 : \exists Y \subseteq X, |Y| \leq k : B_r(x) \subseteq \bigcup_{y \in Y} B_{r/2}(y).$$

The doubling dimension *of M is defined as* $\dim(M) = \lceil \log_2 k \rceil$.

▶ **Lemma 22.** *For any metric $M = (X, d)$ with doubling dimension $\dim(M)$, an r-well-separated partition with $2^{3 \cdot \dim(M)}$ layers and parameters $(h, \dots, h)$ with $h = 2r$ can be computed in polynomial time.*

Proof. First we partition X greedily into balls of radius r: As long as not all points of X are covered, we choose arbitrarily an uncovered point x from X and put x into one group together with all uncovered points that have a distance of at most r from x. This way, we get a partition of X into groups with radius at most r.

Next, we try to reduce the number of groups by local improvements. We say that two groups are neighboring if the distance of their centers is at most $4r$. As long as there is a group that has at least $2^{3 \cdot \dim(M)}$ neighbors, we replace this group and its neighbors by $2^{3 \cdot \dim(M)}$ groups as follows: Let x be a center of a group that has at least $2^{3 \cdot \dim(M)}$ neighbors, and let the centers of the neighbors be $Y \subseteq X$. Since x has a distance of at most $4r$ from all centers in Y, we have

$$B_r(x) \cup_{y \in Y} B_r(y) \subseteq B_{5r}(x).$$

By definition of the doubling dimension, the ball $B_{5r}(x)$ can be covered by $2^{\dim(M)}$ balls of radius $5r/2$, each of these can be covered by $2^{\dim(M)}$ balls of radius $5r/4 < 2r$, and each of these can be covered by $2^{\dim(M)}$ balls of radius $5r/8 < r$. Hence, the points in

$B_r(x) \cup_{y \in Y} B_r(y)$ can be covered by $2^{3 \cdot \dim(M)}$ balls of radius r. In our partition, we replace the groups around x and around $y \in Y$ by the groups induced by these balls. Since this reduces the number of groups by at least one, after a linear number of these local improvements, no local improvement is possible anymore, i.e., every group has less than $2^{3 \cdot \dim(M)}$ neighbors.

We have obtained a partition of X into groups, where each group has a radius of at most r. Furthermore, each group has a center and two groups are neighbors if their centers have a distance of at most $4r$. Furthermore, every group has less than $2^{3 \cdot \dim(M)}$ neighbors. The groups will form the groups in the r-well-separated partition. Since points from groups that are not neighbored have a distance of more than $2r$, two groups that are not neighbored can be on the same layer of the partition without contradicting property (iii) from Definition 10. The diameter of each group is at most $h = 2r$. It remains to distribute the groups to the different layers of the partition. For this we find a coloring of the groups such that neighboring groups get different colors. The neighborhood defines implicitly a graph with the groups as vertices with degree at most $2^{3 \cdot \dim(M)} - 1$. Any such graph can be colored with $2^{3 \cdot \dim(M)}$ colors by a greedy algorithm. Now we assign the groups according to the colors to different layers, resulting in an r-well-separated partition with at most $2^{3 \cdot \dim(M)}$ layers. ◀

Based on Corollaries 12 and 15, it is now easy to prove the following theorem.

▶ **Theorem 23.** *For any metric $M = (X, d)$ with doubling dimension $\dim(M)$, there exists an $O(2^{3 \cdot \dim(M)})$-approximation algorithm for the connected k-center problem and for the connected k-diameter problem with disjoint clusters.*

Proof. According to Lemma 19, we can efficiently compute an r-well-separated partition with $2^{3 \cdot \dim(M)}$ layers and parameters $(h, \dots, h)$ for $h = 2r$.

By Corollary 12 this implies that we can efficiently find a solution for the connected k-center problem with disjoint clusters with approximation factor

$$4\ell - 2 + 2 \sum_{i=1}^{\ell} h/r = O(2^{3 \cdot \dim(M)}).$$

By Corollary 15, it also implies that we can efficiently find a solution for the connected k-diameter problem with disjoint clusters with approximation factor

$$4\ell - 2 + h_1/r + 2 \sum_{i=2}^{\ell} h_i/r = O(2^{3 \cdot \dim(M)}).$$ ◀

5 Conclusions

We studied the connected k-center and k-diameter problem and proved several new results on the approximability of different variants of these problems. In particular, we developed a general framework to obtain approximation algorithms for the disjoint versions of these problems that relies on the existence of well-separated partitions. While we obtain constant-factor approximations for L_p-metrics in constant dimension and metrics with constant doubling dimension, our general upper bound is $O(\log^2 k)$. Since all our lower bounds are constant, an obvious open question is to close the gaps between the upper and lower bounds. One possibility to approach this would be to derive better well-separated partitions. However, we also show that with our approach no bound better than $O(\log \log k)$ can be shown.

References

1. Moses Charikar, Samir Khuller, David M. Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In S. Rao Kosaraju, editor, *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms (SODA)*, pages 642–651. ACM/SIAM, 2001.
2. Marek Cygan, MohammadTaghi Hajiaghayi, and Samir Khuller. LP rounding for k-centers with non-uniform hard capacities. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 273–282. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.63.
3. Hu Ding and Jinhui Xu. A unified framework for clustering constrained data without locality property. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1471–1490. SIAM, 2015.
4. Rong Ge, Martin Ester, Byron J. Gao, Zengjian Hu, Binay K. Bhattacharya, and Boaz Ben-Moshe. Joint cluster analysis of attribute data and relationship data: The connected k-center problem, algorithms and applications. *ACM Trans. Knowl. Discov. Data*, 2(2):7:1–7:35, 2008. doi:10.1145/1376815.1376816.
5. Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.
6. Neelima Gupta, Aditya Pancholi, and Yogish Sabharwal. Clustering with internal connectedness. In *Proc. of 5th Intl. Workshop on Algorithms and Computation (WALCOM)*, volume 6552 of *Lecture Notes in Computer Science*, pages 158–169. Springer, 2011. doi:10.1007/978-3-642-19094-0_17.
7. Dorit S. Hochbaum. When are np-hard location problems easy? *Ann. Oper. Res.*, 1(3):201–214, 1984. doi:10.1007/BF01874389.
8. Dorit S. Hochbaum and David B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM*, 33(3):533–550, 1986.
9. Simon J. Holgate, Andrew Matthews, Philip L. Woodworth, Lesley J. Rickards, Mark E. Tamisiea, Elizabeth Bradshaw, Peter R. Foden, Kathleen M. Gordon, Svetlana Jevrejeva, and Jeff Pugh. New data systems and products at the permanent service for mean sea level. *Journal of Coastal Research*, 29:493–504, 2013.
10. Wen-Lian Hsu and George L. Nemhauser. Easy and hard bottleneck location problems. *Discrete Applied Mathematics*, 1(3):209–215, 1979.
11. Ravishankar Krishnaswamy, Shi Li, and Sai Sandeep. Constant approximation for k-median and k-means with outliers via iterative rounding. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 646–659, 2018.
12. Zhung-Xun Liao and Wen-Chih Peng. Clustering spatial data with a geographic constraint: exploring local search. *Knowl. Inf. Syst.*, 31(1):153–170, 2012. doi:10.1007/s10115-011-0402-8.
13. Permanent Service for Mean Sea Level (PSMSL). Tide gauge data, retrieved on 03 February 2022 from http://www.psmsl.org/data/obtaining/.

On Sparsification of Stochastic Packing Problems

Shaddin Dughmi
University of Southern California, Los Angeles, CA, USA

Yusuf Hakan Kalayci
University of Southern California, Los Angeles, CA, USA

Neel Patel
University of Southern California, Los Angeles, CA, USA

Abstract

Motivated by recent progress on stochastic matching with few queries, we embark on a systematic study of the sparsification of stochastic packing problems more generally. Specifically, we consider packing problems where elements are independently active with a given probability p, and ask whether one can (non-adaptively) compute a "sparse" set of elements guaranteed to contain an approximately optimal solution to the realized (active) subproblem. We seek structural and algorithmic results of broad applicability to such problems. Our focus is on computing sparse sets containing on the order of d feasible solutions to the packing problem, where d is linear or at most polynomial in $\frac{1}{p}$. Crucially, we require d to be independent of the number of elements, or any parameter related to the "size" of the packing problem. We refer to d as the "degree" of the sparsifier, as is consistent with graph theoretic degree in the special case of matching.

First, we exhibit a generic sparsifier of degree $\frac{1}{p}$ based on contention resolution. This sparsifier's approximation ratio matches the best contention resolution scheme (CRS) for any packing problem for additive objectives, and approximately matches the best monotone CRS for submodular objectives. Second, we embark on outperforming this generic sparsifier for additive optimization over matroids and their intersections, as well as weighted matching. These improved sparsifiers feature different algorithmic and analytic approaches, and have degree linear in $\frac{1}{p}$. In the case of a single matroid, our sparsifier tends to the optimal solution. In the case of weighted matching, we combine our contention-resolution-based sparsifier with technical approaches of prior work to improve the state of the art ratio from 0.501 to 0.536. Third, we examine packing problems with submodular objectives. We show that even the simplest such problems do not admit sparsifiers approaching optimality. We then outperform our generic sparsifier for some special cases with submodular objectives.

2012 ACM Subject Classification Theory of computation → Packing and covering problems

Keywords and phrases Stochastic packing, sparsification, matroid

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.51

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/pdf/2211.07829.pdf`

Funding Supported by NSF grant CCF-2009060.

Acknowledgements We are grateful to the anonymous reviewers for their thoughtful feedback on the earlier version of this paper.

1 Introduction

Our starting point for this paper is the beautiful line of recent work on variants of the stochastic matching problem, seeking approximate solutions with limited query access to the (stochastic) data [9, 3, 2, 4, 8, 7, 6, 5]. Notably, many of the algorithms in these works are non-adaptive, and can therefore be interpreted as "sparsifiers" for the stochastic problem.

These works feature powerful new algorithmic and analytic sparsification techniques of possibly more general interest, suggesting that effective sparsifiers might exist well beyond matching and closely related problems.

Our goal in this paper is to coalesce a broader agenda on the sparsification of combinatorial stochastic optimization problems more generally, beginning with the natural and broad class of packing problems. We ask, and make progress on, the fundamental questions: For which stochastic packing problems is effective sparsification possible? What are the algorithmic techniques and blueprints which are broadly applicable? What are the barriers to progress?

Concretely, we examine stochastic packing problems (SPPs) of the following (fairly general) form. We are given a set system $(E, \mathcal{I})$, where E is a finite set of *elements* and $\mathcal{I} \subseteq 2^E$ is a downwards-closed family of *feasible sets* (often also referred to as *independent sets*, in particular for matroids). Also given is an objective function $f : 2^E \to \mathbb{R}_+$, which we assume to be either additive (a.k.a. modular) or submodular. The stochastic uncertainty is described by a given probability $p \in [0, 1]$: We assume that each element of E is *active*, i.e., viable for being selected, independently with probability p. The goal of the SPP is to select a feasible set of active elements maximizing the objective function.

When the set R of active elements is given, or can be queried without restriction, this reduces to non-stochastic optimization for the induced subproblem on R. We refer to the output of such an omniscient [approximation] algorithm as an [approximate] *stochastic optimum solution*. We are instead concerned with algorithms that approximate the stochastic optimum by querying the activity status of only a small, a.k.a. "sparse", set of elements $Q \subseteq E$. In particular, as in much of the prior work we require the queried set Q to be chosen non-adaptively. Such algorithms can equivalently be thought of as factoring into two steps: First, a *sparsification algorithm* (or *sparsifier* for short) computes a (possibly random) set of elements Q. Second, we learn $R \cap Q$, and an [approximate] optimization algorithm is applied to the (now fully-specified) subproblem induced by $R \cap Q$. Since the second (optimization) step is familiar and well-studied, our focus is on the first step, namely sparsification.

We evaluate a sparsifier by two quantities. The first quantity is a familiar one, namely its *approximation ratio*. Specifically, a sparsifier is α-approximate if it guarantees an α-approximation to the stochastic optimum solution when combined with a suitable algorithm in the second (optimization) step. The second quantity is a measure of the "sparsity" of the set Q selected by the sparsifier. We say our sparsifier is of *degree* d if it guarantees $\mathbb{E}[|Q|] \leq d \cdot r$, where $r = \max\{|S| : S \in \mathcal{I}\}$ is the *rank* of the set system $(E, \mathcal{I})$. Intuitively, the sparsification degree refers to the level of "contingency" or "redundancy" in the sparsified instance, relative to the size of maximal feasible solutions. Loosely speaking, the degree of a sparsifier roughly measures "how many" feasible solutions are maintained to account for uncertainty in the problem. Somewhat fortuitously, our definition of degree specializes to the (average) graph-theoretic degree in the special case of matching, lending consistency with prior work on stochastic matching with few queries.

We study sparsifiers whose degree admits an upperbound that is independent of the size of the system; The degree bound can not depend on the number of elements or the rank of the set system, for example. We focus especially on the "polynomial regime", where the degree is restricted to be at most polynomial in $\frac{1}{p}$. We pursue sparsifiers which are constant-approximate, or in the best case $(1 - \epsilon)$-approximate for arbitrarily small $\epsilon > 0$.

Results and Techniques

We begin with the observation that a degree of at least $\frac{1}{p}$ is necessary for constant-approximate sparsification, even for the simplest of packing problems: a rank one matroid and the unweighted additive objective. We then establish a "baseline" of possibility for all stochastic

packing problems, through a generic sparsifier with this same degree $\frac{1}{p}$. This sparsifier is simple: it computes (or estimates) the marginals $\{q_e\}_{e \in E}$ of the stochastic optimum solution, and outputs a set Q which includes each element e independently with probability $\frac{q_e}{p}$. For SPPs with an additive objective, we show that this sparsifier's approximation ratio matches the balance ratio of the best contention resolution scheme (CRS)[1]. for the set system. When the objective is submodular, we approximately match the balance ratio of the best *monotone* CRS up to a factor of $1 - \frac{1}{e}$. We note that contention resolution is only used as a proof tool to certify our sparsifier's approximation guarantee, and is not invoked algorithmically. In settings where the marginals $\{q_e\}_{e \in E}$ are intractable to compute, this sparsifier can be made computationally efficient by resorting to approximation, in which case its approximation ratio degrades in the expected manner. This generic result yields constant-approximate sparsifiers of degree $\frac{1}{p}$ for a large variety of set systems for which contention resolution has been studied, including matroids and their intersections.

Next, we embark on "beating" this contention resolution baseline for natural SPPs. We succeed at doing so for additive (weighted) optimization over matroids, matroid intersections, and matchings. For a single matroid, we derive a simple greedy sparsifier which is $(1 - \epsilon)$-approximate and has degree $\frac{1}{p} \cdot \log(1/\epsilon)$. This sparsifier repeatedly adds a maximum weight independent set of the matroid to the sparse set Q, and removes it from the matroid, until the desired degree is reached. Though our sparsifier is simple, its analysis is (we believe necessarily) less so.

For matroid intersections, we first argue that adaptations of our single-matroid sparsifier cannot succeed, due to feasible sets not "combining well" as they do in the case of a single matroid. Instead, our sparsifier for matroid intersections repeatedly samples the stochastic optimum solution and adds it to the sparse set Q, for a degree of $O(\frac{1}{p\epsilon} \cdot \log(1/\epsilon))$. The approximation ratio of our sparsifier for the intersection of k matroids is $\frac{1-\epsilon}{k+1/(k+1)}$, which beats the best known bound on the correlation gap of $1/(k+1)$ [1]. The analysis of this sparsifier is again nontrivial, and utilizes basis exchange maps.

For matroids and matroid intersections, we note that analysis techniques employed by prior work on matching do not appear to suffice. In particular, prior work on matching often employs concentration arguments on the active degree of matroid "flats" containing an element; this is sufficient in the case of matching, since each element is in at most two binding flats (one for each partition matroid). For general matroids, such concentration arguments fail to bound the degree in a manner independent of the number of elements, necessitating alternative proof approaches like ours.

For general (non-bipartite) matching, we augment our contention-resolution-baseline sparsifier with samples from the stochastic optimum solution, for a total degree of $O(1/p)$. We show that the samples from the stochastic optimum combine well with our baseline sparsifier. We obtain an approximation ratio which is a function of the (as yet not fully known) correlation gap of the matching polytope. This function exceeds the identity function everywhere, implying that our sparsifier strictly improves on the contention resolution baseline. Plugging in the best known lowerbound of 0.474 on the correlation gap from [17], we guarantee that our sparsifier is 0.536 approximate. This improves on the state of the art in the polynomial regime, 0.501-approximate sparsifier of degree poly $(1/p)$ due to [8]. In addition, assuming the conjecture from [20] which states the existence of 0.544 balanced CRS for general matching polytope implies that our sparsifier is 0.598 approximate.

[1] This is equal to the set system's *correlation gap*, as shown by [12].

Table 1 Summary of information theoretic sparsifiers for additive objectives. Here, n is the number of elements and W is the maximum element weight.

	Previous Results		This Work	
Constraint	**Approx. Ratio**	**Sparsification Degree**	**Approx. Ratio**	**Sparsification Degree**
Matroid	$1-\epsilon$ [14]	$O\left(\frac{1}{p}\log\frac{\mathbf{Rank}}{\epsilon}\right)$	$1-\epsilon$	$\frac{1}{p}\cdot\log\left(\frac{1}{\epsilon}\right)$
k-Matroid Intersection	$\frac{1-\epsilon}{2k}$ [18]	$O\left(\frac{W}{p}\log n\log\left(\frac{n}{\epsilon}\right)\frac{k}{\epsilon^3}\right)$	$\frac{1-\epsilon}{k+\frac{1}{k+1}}$	$\frac{1}{\epsilon\cdot p}\cdot\log\left(\frac{1}{\epsilon}\right)$
General Matching	$1-\epsilon$ [6]	$O\left(\exp(\exp(\exp(1/p)))\right)$	0.536	$O\left(\frac{1}{p}\right)$
General Matching	0.501 [8]	$O\left(\frac{1}{p}\right)$	0.536	$O\left(\frac{1}{p}\right)$

Finally, we further examine stochastic packing problems with submodular objectives. Our $(1-\epsilon)$-approximate sparsifier for weighted matroid optimization might tempt one to conjecture a similar result for submodular optimization over simple enough set systems. However, we show by way of an information-theoretic impossibility result that no sparsifier with degree bound independent of the number of elements can beat $(1-1/e)$, even for optimizing a coverage function subject to a uniform matroid constraint. We complement this impossibility result with algorithmic sparsification results for optimizing coverage functions over matroids, improving over the guarantees provided by our baseline generic sparsifier. Due to limited space, the results for submodular SPPs are detailed in the full version of this paper [13](Section 8).

Additional Discussion of Related Work

The exploration of sparsifying SPPs was initiated by [9], who focus on the unweighted stochastic matching problem. This problem has since been studied extensively in a series of works [4, 8, 7, 6] which attempt to beat the benchmark set by [9]. In the "polynomial-degree regime", the state-of-the-art sparsifier for unweighted stochastic matching is a 0.66-approximation due to [2]. Recent work by [5] improves this approximation to $\frac{e}{e+1}$ for unweighted bipartite matching. For weighted stochastic matching in the polynomial-degree regime, the current best known sparsifier is a 0.501-approximation due to [8]. Going beyond polynomial degree, [7, 6] constructed a $(1-\epsilon)$-approximate sparsifier with degree $\exp(\exp(\exp(1/p)))$the weighted general matching problem. The sparsifiers designed for the stochastic matching problems rely heavily on structural properties particular to matching. Our techniques, on the other hand, are targeted at more general packing problems.

To the best of our knowledge, the work of [18, 19] stands alone in directly studying the sparsification of SPPs beyond matching. In [18], they proposed a general framework for solving stochastic packing integer programs. As a corollary of their techniques, they obtain non-adaptive sparsifiers for several additive SPPs. However, the degree of their sparsification algorithms intrinsically depends on the number of elements in settings where a single element may be in an exponential number of binding constraints (as is the case for matroids). Our work, in contrast, proposes several algorithmic techniques that yield approximate sparsifiers with degree independent of the number of elements.

Also related is the work of [14], which studies the covering analogue of our question for matroids. They show how to construct a set of size $O(\frac{\mathbf{Rank}}{p}\log\frac{\mathbf{Rank}}{\epsilon})$ which is guaranteed to contain a minimum-weight base of the matroid with high probability. This implicitly

Table 2 Summary of information theoretic sparsifiers for monotone submodular objectives. All mentioned results are shown in this paper.

Constraint	Approximation Ratio	Sparsification Degree	Note
r-Uniform Matroid	$\left(1-\frac{1}{e}\right)\cdot\left(1-\frac{1}{\sqrt{r+3}}\right)$	$\frac{1}{p}$	$\left(1-\frac{1}{e}\right)$ upperbound, Optimal when $r\to\infty$
Matroid	$\left(1-\frac{1}{e}\right)^2$	$\frac{1}{p}$	$1-\frac{1}{e}$ upperbound
k-Matroid Intersection	$\left(1-\frac{1}{e}\right)\cdot\frac{1}{k+1}$	$\frac{1}{p}$	

implies an $O(\frac{1}{p}\log\frac{\mathbf{Rank}}{\epsilon})$-degree sparsifier for weighted stochastic packing on matroids. Their analysis is tight for the covering setting, and it appears nontrivial to adapt their techniques for the packing setting in order to remove the degree's dependence on the rank. We compare our results for additive SPPs with prior work in Table 1.

The manuscript [19] proposes sparsifiers for SPPs with a monotone submodular objectives. However, their sparsification algorithms are intrinsically adaptive in nature. To the best of our knowledge, ours is the first work that analyzes SPPs with submodular objectives in the non-adaptive setting. We summarize our results for submodular SPPs in Table 2.

2 Problem Definition

We consider packing problems of the form $\langle E,\mathcal{I},f\rangle$ where E is a ground set of *elements* with cardinality n, $f:2^E\to\mathbb{R}_{\geq 0}$ is an objective function, and $\mathcal{I}\subseteq 2^E$ is a downwards-closed family of *independent sets* (a.k.a. *feasible sets*). We use $r=\text{argmax}\{|I|:I\in\mathcal{I}\}$ to denote the *rank* of the set system $\mathcal{I}$. The aim of the packing problem is to select an independent set $O\in\mathcal{I}$ that maximizes $f(O)$.

In this paper, we study packing problems in a particular setting with uncertainty parametrized by $p\in[0,1]$. In a *stochastic packing problem (SPP)* $\langle E,\mathcal{I},f,p\rangle$, nature selects a random set $R\subseteq E$ of *active elements* such that $\mathbf{Pr}[e\in R]=p$ independently for all $e\in E$. We are then tasked with solving the induced (random) packing problem on the active elements, namely $\langle R,\mathcal{I}|R,f|R\rangle$ where $\mathcal{I}|R$ and $f|R$ denote the restriction of $\mathcal{I}$ and f to subsets of R, respectively. We refer to an [approximately] optimum solution to $\langle R,\mathcal{I}|R,f|R\rangle$ as an [approximate] *stochastic optimum solution*. We use **OPT** to denote the expected value of a stochastic optimum solution, i.e.,

$$\mathbf{OPT}(E,\mathcal{I},f,p)=\mathbb{E}_R\left[\max_{\substack{T\in\mathcal{I}\\T\subseteq R}}f(T)\right],$$

where $R\subseteq E$ is the random set which each element of E independently with probability p.

We assume that the set R of active elements is a-priori unknown, and that we can *query* elements in E to check their membership in R. Motivated by settings in which queries are costly, we seek algorithms which query a small (we say "sparse") subset of the elements, and moreover choose those queries non-adaptively. Such non-adaptive algorithms can be thought of as factoring into two steps: A *sparsification* step which selects the small set $Q\subseteq E$ of queries, and an *optimization* step which solves the packing problem $\langle R\cap Q,\mathcal{I}|R\cap Q,f|R\cap Q\rangle$ induced by the queried active elements. For the optimization step, we assume access to a traditional [approximation] algorithm. Our focus is on algorithms for the sparsification step, which we define formally next.

A *sparsification algorithm* (or *sparsifier* for short) $\mathcal{A}$ takes as input an SPP $J = \langle E, \mathcal{I}, f, p \rangle$ from some family of SPPs, and outputs a (possibly random) set of elements $Q \subseteq E$. The twin goals here are for Q to be "sparse" in a quantified formal sense we describe shortly, while guaranteeing that optimally solving the "sparsified" SPP $J|Q = \langle Q, \mathcal{I}|Q, f|Q, p \rangle$ yields an approximate solution to the original SPP J. We say that the sparsification algorithm $\mathcal{A}$ is α-*approximate* if it guarantees $\mathbf{OPT}(J|Q) \geq \alpha\, \mathbf{OPT}(J)$ – i.e., an optimal solution to the sparsified SPP is an α-approximate solution to the original SPP. We sometimes identify the sparsified SPP $J|Q$ with Q when J is clear from the context.

To quantify sparsity, we say that $\mathcal{A}$ has *sparsification degree* d if it guarantees that $\frac{\mathbb{E}[|Q|]}{r} \leq d$, where r is the rank of the set system $\mathcal{I}$, and expectation is over the internal random coins of $\mathcal{A}$. Intuitively, the degree of sparsification refers to the level of "contingency" or "redundancy" in the sparsified instance, relative to the size of maximal feasible solutions. Loosely speaking, the degree of a sparsifier roughly measures "how many" feasible solutions it maintains to account for uncertainty in the problem.

In the absence of a bound on degree, an approximation factor of $\alpha = 1$ is trivially achievable. We aim to construct approximate sparsifiers of low degree for natural classes of SPPs. We begin by observing that a degree of $\Omega(1/p)$ is necessary for constant approximation, even for the simplest of constraints.

▶ **Example 1.** Consider the SPP with n elements, the unweighted additive objective function $f(S) = |S|$, a rank-one matroid constraint, and activation probability $p = 1/n$. There is at least one active element with probability $1 - (1-p)^n \geq 1 - 1/e$, therefore $\mathbf{OPT} \geq 1 - 1/e$. On the other hand, a set of elements Q will contain no active elements with probability $(1-p)^{|Q|} \geq 1 - |Q| \cdot p = 1 - \frac{|Q|}{n}$. When $|Q| = o(1/p) = o(n)$, there are no active elements in Q with probability $1 - o(1)$. Therefore, any constant-approximate sparsifier must have degree $\Omega(1/p)$.

We also show in in the full version of this paper [13] that, unsurprisingly, there exist stochastic packing problems which do not admit constant approximate sparsifiers with degree $\text{poly}(1/p)$. Given these simple impossibility results, we ask a natural question:

▶ **Question 2.** *Which stochastic packing problems admit constant approximate sparsifiers of degree* $O\left(\frac{1}{p}\right)$*, or more loosely* $\text{poly}\left(\frac{1}{p}\right)$*?*

In this paper, we focus on designing sparsification algorithms for stochastic packing problems with additive or nonnegative monotone submodular objectives.

A Note on Input Representation

Many of our results are information theoretic, and therefore make no assumptions on how a stochastic packing problem is represented. Most of our algorithmic results, on the other hand, only require solving realized (non-stochastic) instances of the packing problem, possibly approximately. Specifically, for a stochastic packing problem $\langle E, \mathcal{I}, f, p \rangle$ we often assume access to a [β-approximate] *stochastic optimal oracle*. Such an oracle samples a [β-approximate] solution to the (random) packing problem $\langle R, \mathcal{I}|R, f|R \rangle$, where R includes each element of E independently with probability p, and $\mathcal{I}|R$ and $f|R$ denote the restriction of $\mathcal{I}$ and f to R respectively. For our algorithmic results on matroids, we additionally assume access to an independence oracle, as is standard.

3 Sparsification from Contention Resolution

In this section, we show how to generically derive a sparsifier for a stochastic packing problem from bounds on contention resolution for the associated set system. First, we recall the relevant definition of contention resolution.

▶ **Definition 3** ([12]). *Let $(E, \mathcal{I})$ be a set system, and let $P_{\mathcal{I}} = \text{convexhull}\{\mathbb{1}_I : I \in \mathcal{I}\}$ denote the associated polytope. A* Contention Resolution Scheme (CRS) *π for $P_{\mathcal{I}}$ is a (randomized) algorithm which takes as input a point $x \in P_{\mathcal{I}}$ and a set of* active *elements $R(x) \subseteq E$, including each element $i \in E$ independently with probability x_i, and outputs a feasible subset $\pi_x(R(x)) \subseteq R(x)$, $\pi_x(R(x)) \in \mathcal{I}$. For $b, c \in [0, 1]$, we say a CRS is* (b, c)-balanced *if for all $i \in E$ and $x \in b \cdot P_{\mathcal{I}}$, $\mathbf{Pr}[i \in \pi_x(R(x)) \mid i \in R(x)] \geq c$. A CRS π is* monotone *if for every $S \subseteq T \subseteq E$ we have that $\mathbf{Pr}[i \in \pi(S) \mid i \in S] \geq \mathbf{Pr}[i \in \pi(T) \mid i \in T]$.*

Our generic sparsifier is randomized, has degree $\frac{1}{p}$, and is shown in Algorithm 1. Our sparsifier computes estimated marginals $\mathbf{q}$ for the stochastic optimum solution. For an information-theoretic result, we can assume these to be exact. Then it samples each element $e \in E$ in a sparse set Q with probability $\frac{q_e}{p}$.

When the objective function f is additive, our sparsifier has an approximation factor that matches the balance ratio of the best CRS for $P_{\mathcal{I}}$.[2] For nonnegative monotone submodular functions, the approximation factor matches the balance ratio of the best monotone CRS for $P_{\mathcal{I}}$. This is due to the observation that each element $e \in E$ is included in the active subset of the sparse set Q with probability q_e and the fact that $\mathbf{q} \in P_{\mathcal{I}}$. The detailed proof for Theorem 4 can be found in the full version [13].

Algorithm 1 Generic Sparsifier for a Stochastic Packing Problem $\langle E, \mathcal{I}, f, p \rangle$.

Input: Stochastic packing problem $\langle E, \mathcal{I}, f, p \rangle$

Compute the marginals $\mathbf{q}$ of the stochastic optimum solution, or an approximation thereof.
$Q \leftarrow \emptyset$;
for all $e \in E$ **do**
  Add e to Q with probability $\frac{q_e}{p}$ (independently)
end for
Output: Sparse set Q.

▶ **Theorem 4.** *Consider Algorithm 1, implemented with exact (possibly non-polynomial-time) computation of the marginals $\mathbf{q}$. When f is additive, and $P_{\mathcal{I}}$ admits a c-balanced CRS, the algorithm is a c-approximate sparsifier of degree $\frac{1}{p}$. When f is a nonnegative monotone submodular, and $P_{\mathcal{I}}$ admits a c-balanced monotone CRS, the algorithm is a $c\left(1 - \frac{1}{e}\right)$-approximate sparsifier of degree $\frac{1}{p}$.*

To make our sparsifier algorithmically efficient, $\mathbf{q}$ may be estimated by sampling from a (possibly approximate) stochastic optimum oracle, in which case our guarantees degrade in the expected manner due to sampling errors and/or the approximation inherent to the oracle. We present the detailed analysis with approximate stochastic optimal oracles in Appendix B in the full version [13]. Theorem 4 and Theorem 4.3 (In full version [13]) together with contention resolution schemes from prior work [1, 12, 17] and approximate stochastic optimal oracles that employ approximation algorithms from [11, 16], imply constant approximate sparsifier for a broad class of packing constrains summarized in Table 3.

[2] This balanced ratio is equal to the *correlation gap* of the set system $\mathcal{I}$, as per [12].

Table 3 Approximation Ratio of Generic Sparsifier of degree $\frac{1}{p}$ for various packing constraint families with additive and non-negative monotone submodular function.

	Additive Objective		**Submodular Objective**	
Constraint	Information Theoretic	Poly-Time	Information Theoretic	Poly-Time
Matroid	$\left(1-\frac{1}{e}\right)$	$(1-\epsilon)\cdot\left(1-\frac{1}{e}\right)$	$\left(1-\frac{1}{e}\right)^2$	$(1-\epsilon)\cdot\left(1-\frac{1}{e}\right)^3$
k-matroid intersection	$\frac{1}{k+1}$	$(1-\epsilon)\cdot\frac{1}{k^2-1}\cdot$	$\frac{1-1/e}{k+1}$	$(1-\epsilon)\cdot\frac{1-1/e}{k^2-1}\cdot$
Matching	0.474	$(1-\epsilon)\cdot 0.474$	$(1-\frac{1}{e})\cdot 0.43$	$(1-\frac{1}{e})^2\cdot 0.43$

The following proposition (whose proof is delegated to the full version [13]) shows that Algorithm 1 is optimal for matroids and additive objectives among sparsifiers of degree $1/p$. This strongly suggests that sparsification is intimately tied to contention resolution when the degree is restricted to $1/p$. In particular, exceeding degree $1/p$ appears necessary for outperforming the correlation gap of a set system in general.

▶ **Proposition 5.** *Consider the family of stochastic packing problems with matroid constraints and additive objectives. There is no degree $\frac{1}{p}$ sparsifier for this family that achieves an approximation ratio $1-1/e+\Omega(1)$.*

We note that $1-1/e$ is the best possible balance ratio for contention resolution on the rank one matroid, as shown in [12] through the correlation gap. Given the above discussion, it is natural to ask whether we can design sparsifiers of degree $O(1/p)$, or even poly$(1/p)$, whose approximation ratio α exceeds the best CRS balance ratio c, i.e., can we have $\alpha > c$ with degree linear or polynomial in $1/p$? Recent progress on this question for bipartite matching constraints came in a pair of recent works. Behnezhad et.al. [5] designed a $\frac{e}{e+1}\approx 0.731$-approximate sparsifier with degree poly$(1/p)$ for unweighted bipartite matching. Their approximation factor is strictly better than a known upper bound of 0.544 on the correlation gap (and hence the best balance ratio) of bipartite matching, due to [15]. To our knowledge, this is the only sparsifier in the literature with degree polynomial in $\frac{1}{p}$ and approximation ratio provably exceeding the correlation gap of the set system. Another recent result due to Behnezhad et al [7] achieves a 0.501-approximate sparsification with degree polynomial in $1/p$ for *weighted* matching. This outperforms the best *known* contention resolution scheme for matching[10], though not clearly the best possible. Prior to our work, there was no known sparsifier for any *weighted* stochastic packing problem which provably outperforms the correlation gap using degree poly$(1/p)$.

In the following sections, we will construct degree $O(1/p)$ sparsifiers for matroids, matroid intersections, and matching which improve on the contention-resolution-based guarantees provided in this section. For matroids and matchings, our sparsifiers provably outperform contention resolution. For matroid intersections, we outperform the best *known* CRS.

4 Additive Optimization over a Matroid

In this section, we design an improved sparsifier for the stochastic packing problem $\langle E,\mathcal{I},f,p\rangle$ when $\mathcal{M}=(E,\mathcal{I})$ is a matroid and f is additive. For an arbitrary $\epsilon>0$, our sparsifier is $(1-\epsilon)$-approximate and has degree $\frac{1}{p}\log\frac{1}{\epsilon}$. Throughout, we use $\{w_e\}_{e\in E}$ to denote the weights associated with the additive function f, and use $R\subseteq E$ to denote the (random) set of active elements which includes each $e\in E$ independently with probability p. We also sometimes use r as shorthand for $\mathbf{Rank}(\mathcal{M})$. We present basic preliminaries of matroid theory in the full version [13]

▶ **Theorem 6.** *Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid, f be an additive function and $p \in [0,1]$. Algorithm 2 is a $(1-\epsilon)$-approximate polynomial time sparsifier for the stochastic packing problem $\langle E, \mathcal{I}, f, p\rangle$ with sparsification degree $\frac{1}{p} \cdot \log\left(\frac{1}{\epsilon}\right)$.*

Previously, the best known sparsifer for matroid was $(1-\epsilon)$-approximate with degree $O\left(1/p \log\left(\mathbf{Rank}/\epsilon\right)\right)$ implicit in [14]. In contrast, the sparsification degree of our algorithm is independent of the "size" of the matroid. As we argued in introduction, such a size-independent guarantee appears to be beyond the techniques used in earlier works [14, 18].

Algorithm 2 Sparsifier for $(\mathcal{M}, f, p)$, when $\mathcal{M}$ is a matroid and f is additive.

Set $\mathcal{M}_0 = \mathcal{M}$ and $Q = \emptyset$.
for t in $\{1, \ldots, \tau\}$ where $\tau = \frac{1}{p} \cdot \log(\frac{1}{\epsilon})$
  Let $I_t \leftarrow \text{argmax}_{I \in \mathcal{I}_{t-1}} f(I)$, where $\mathcal{I}_{t-1}$ is the collection of independent sets in $\mathcal{M}_{t-1}$.
  Update $\mathcal{M}_t \leftarrow \mathcal{M}_{t-1} \setminus I_t$.
Output: $Q = \bigcup_{t=1}^{\tau} I_t$.

It is clear that the sparsifier in Algorithm 2 has degree $\tau = \frac{1}{p} \cdot \log(\frac{1}{\epsilon})$, and can be implemented in polynomial time given an independence oracle for the matroid $\mathcal{M}$. The remainder of this section is devoted to proving that it is $(1-\epsilon)$-approximate, as needed to complete the proof of Theorem 6. Our proof will consist of two parts. First, we will analyze Algorithm 2 in the special case of unit weights (a.k.a. unweighted). Second, we reduce the analysis of the weighted problem to that of the unweighted problem.

4.1 Special Case: Unweighted Optimization

In this subsection, we assume that elements of the matroid $\mathcal{M}$ all have unit weight. In this case, observe that Algorithm 2 repeatedly removes an arbitrary basis of the matroid and adds it to the sparse set Q. More precisely, in iteration t the set I_t is a basis of the remaining matroid $\mathcal{M}_{t-1} := \mathcal{M} \setminus \bigcup_{j=1}^{t-1} I_j$.

In this unweighted case, the stochastic optimal value is the expected rank of the active elements R, and our claimed approximation guarantee can be expressed as $\mathbb{E}[\mathbf{Rank}(Q \cap R)] \geq (1-\epsilon)\,\mathbb{E}[\mathbf{Rank}(R)]$. To establish this, consider the following informal (but ultimately flawed) argument, starting with the observation that $I_t \cap R$ spans a p fraction of the rank of the remaining matroid $\mathcal{M}_{t-1}$ in expectation. This observation suggests that the rank of elements not spanned by $Q \cap R$ should shrink by a factor of $(1-p)$ with each iteration. Induction would then guarantee that after $\frac{1}{p} \cdot \log\left(\frac{1}{\epsilon}\right)$ iterations we have covered a $(1-\epsilon)$ fraction of the rank of the matroid.

The above rough argument is a good starting point. Indeed, it succeeds when all (or many) of the bases $I_1, \ldots, I_\tau$ are full-rank or close to it. These are precisely the scenarios in which $\mathbb{E}[\mathbf{Rank}(R)] \approx \mathbf{Rank}(\mathcal{M})$. However, in general $\mathbf{OPT} = \mathbb{E}[\mathbf{Rank}(R)]$ can be significantly smaller than $\mathbf{Rank}(\mathcal{M})$ – in the worst case up to a factor of p smaller – in which case the the rank of I_t may drop precipitously with t and the above inductive analysis falls apart. Such scenarios are not simply outliers that we can assume away: they are unavoidable products of the weighted-to-unweighted reduction we present in the next subsection, and can account for a large fraction of the weighted stochastic optimal. This seems to necessitate a more nuanced proof approach in which we compare $\mathbb{E}[\mathbf{Rank}(Q \cap R)]$ with $\mathbb{E}[\mathbf{Rank}(R)]$. We present such a proof next, built upon the following definitions and structural properties.

▶ **Definition 7.** *A* nested system of spanning sets (NSS) *for a matroid $\mathcal{M}$ is a sequence $I_1, I_2, \dots I_\tau$ of sets such that for any $j \in [\tau]$, I_j is a full rank set of elements in $\mathcal{M} \setminus I_{1:j-1}$, where $I_{1:j-1} = \bigcup_{\ell=1}^{j-1} I_\ell$.*

▶ **Observation 8.** *The sets $I_1, \dots, I_\tau$ from Algorithm 2 are an NSS of $\mathcal{M}$.*

The following lemma states that the property of being an NSS is preserved under contraction.

▶ **Lemma 9.** *Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid and let $I_1, \dots I_\tau$ be an NSS of $\mathcal{M}$. For an arbitrary independent set S of $\mathcal{M}$, let $I'_j = I_j \setminus S$ for all j. Then, the sequence $I'_1, \dots I'_\tau$ is an NSS of $\mathcal{M}/S$.*

Proof. Fix an arbitrary $j \in \{1, \dots, \tau\}$. It is clear that I'_j is a subset of the elements of $\mathcal{M}/S \setminus I'_{1:j-1}$. It remains to show that I'_j is full rank in $\mathcal{M}/S \setminus I'_{1:j-1}$, as follows.

$$\begin{aligned}
\mathbf{Rank}^{\mathcal{M}/S}(I'_j) &= \mathbf{Rank}^{\mathcal{M}}(I_j \cup S) - |S| && \text{(By (2) and definition of } I'_j\text{)} \\
&= \mathbf{Rank}^{\mathcal{M}}((E \setminus I_{1:j-1}) \cup S) - |S| && (I_j \text{ is full rank in } \mathcal{M} \setminus I_{1:j-1}) \\
&= \mathbf{Rank}^{\mathcal{M}}((E \setminus S \setminus I'_{1:j-1}) \cup S) - |S| && \text{(By definition of } I'_j\text{)} \\
&= \mathbf{Rank}^{\mathcal{M}/S}(E \setminus S \setminus I'_{1:j-1}) && \text{(By (2))} \\
&= \mathbf{Rank}(\mathcal{M}/S \setminus I'_{1:j-1}) && \blacktriangleleft
\end{aligned}$$

▶ **Observation 10.** *If $I_1, \dots, I_\tau$ is an NSS of $\mathcal{M}$, then $I_2, \dots I_\tau$ is an NSS of $\mathcal{M} \setminus I_1$.*

Now, we will prove the desired result for unweighted matroids.

▶ **Lemma 11.** *Let $\mathcal{M}$ be a matroid, and let $I_1, \dots I_\tau$ be an NSS of $\mathcal{M}$. Then,*

$$\mathbb{E}[\boldsymbol{Rank}(I_{1:\tau} \cap R)] \geq (1 - (1-p)^\tau) \cdot \mathbb{E}[\boldsymbol{Rank}(R)]$$

Proof. Let E denote the elements of $\mathcal{M}$. We will apply induction on τ to prove this result. The base case of $\tau = 0$ is trivial.

Consider $\tau \geq 1$. Let S be an arbitrary maximal independent subset of $R \cap I_1$, and let $\mathbf{Rank}'$ denote the rank function of the (random) matroid $\mathcal{M}' = \mathcal{M}/S \setminus I_1$ with elements $E \setminus I_1$. Using (2) we can write

$$\mathbf{Rank}(R \cap I_{1:\tau}) = \mathbf{Rank}(R \cap I_1) + \mathbf{Rank}'(R \cap I_{2:\tau}) \tag{1}$$

The expectation of the first term is $\mathbb{E}[\mathbf{Rank}(R \cap I_1)] = r \cdot p$. To bound the expectation of the second term, we first condition on $R \cap I_1$, which also fixes S and $\mathcal{M}'$. It follows from Lemma 9 and Observation 10, as well as the fact that $S \subseteq I_1$ is disjoint from $I_{2:\tau}$, that $I_2, \dots I_\tau$ is an NSS of $\mathcal{M}'$. This allows us to invoke the inductive hypothesis to obtain

$$\mathbb{E}[\mathbf{Rank}'(R \cap I_{2:\tau})] \geq (1 - (1-p)^{\tau-1}) \cdot \mathbb{E}[\mathbf{Rank}'(R \setminus I_1)].$$

We use a well-known fact about the rank function of the contracted matroid given by

$$\mathbf{Rank}^{\mathcal{M}/S}(T) = \mathbf{Rank}^{\mathcal{M}}(T \cup S) - \mathbf{Rank}^{\mathcal{M}}(S) = \mathbf{Rank}^{\mathcal{M}}(T \cup S) - |S|. \tag{2}$$

Equation (2) and the definition of S implies that $\mathbf{Rank}'(R \setminus I_1) = \mathbf{Rank}((R \setminus I_1) \cup S) - \mathbf{Rank}(S) = \mathbf{Rank}(R) - \mathbf{Rank}(R \cap I_1)$. Also using the fact $\mathbb{E}[\mathbf{Rank}(R \cap I_1)] = r \cdot p$, we obtain

$$\begin{aligned}\mathbb{E}[\mathbf{Rank}'(R \cap I_{2:\tau})] &\geq (1-(1-p)^{\tau-1}) \cdot \mathbb{E}[\mathbf{Rank}'(R \setminus I_1)] \\ &= (1-(1-p)^{\tau-1})(\mathbb{E}[\mathbf{Rank}(R)] - \mathbb{E}[\mathbf{Rank}(R \cap I_1)]) \\ &= (1-(1-p)^{\tau-1})\,\mathbb{E}[\mathbf{Rank}(R)] - (1-(1-p)^{\tau-1}) \cdot r \cdot p \end{aligned} \tag{3}$$

Finally, we combine (1), and (3) to conclude

$$\begin{aligned}\mathbb{E}[\mathbf{Rank}(R \cap I_{1:\tau})] &\geq (1-(1-p)^{\tau-1})\,\mathbb{E}[\mathbf{Rank}(R)] + (1-p)^{\tau-1} \cdot r \cdot p \\ &\geq (1-(1-p)^{\tau-1} + p(1-p)^{\tau-1})\,\mathbb{E}[\mathbf{Rank}(R)] \\ &= (1-(1-p)^{\tau})\,\mathbb{E}[\mathbf{Rank}(R)] \end{aligned}$$

◀

By observation 8, we get the following corollary of Lemma 11.

▶ **Corollary 12.** *Consider a stochastic matroid optimization problem $\langle E, \mathcal{I}, f, p\rangle$ for $p \in [0,1]$ and $f(S) = |S|$ for all $S \subseteq E$. Algorithm 2 is a $(1-\epsilon)$-approximate sparsifier with degree $\frac{1}{p} \cdot \log\left(\frac{1}{\epsilon}\right)$.*

4.2 Proof of Theorem 6

In this section, we will complete the proof of Theorem 6 by reducing the analysis for a general (weighted) additive function to that of the unweighted case. We order the elements $e_1, \ldots e_n$ in decreasing order of their weights $w_1 \geq \ldots \geq w_n$. Without loss of generality we assume $w_n > 0$, and for notational convenience we define $w_{n+1} = 0$. The following lemma says that if a sparsifier is α-approximate for the unweighted problem on elements above any given weight threshold, then it is also α-approximate for the weighted problem.

▶ **Lemma 13.** *For all $j \in [n]$ with $w_j > w_{j+1}$, if a set $Q \subseteq E$ satisfies*

$$\mathbb{E}[\boldsymbol{Rank}(Q \cap R \cap \{e_1, \ldots e_j\})] \geq (1-\epsilon)\,\mathbb{E}[\boldsymbol{Rank}(R \cap \{e_1, \ldots e_j\})], \tag{4}$$

then $\mathbb{E}[f(\mathrm{opt}(Q \cap R))] \geq (1-\epsilon)\,\mathbb{E}[f(\mathrm{opt}(R))]$. Here, we denote $\mathrm{opt}(S) \in \operatorname{argmax}_{\substack{I \subseteq S \\ I \in \mathcal{I}}} f(I)$, with ties broken arbitrarily.

The above lemma follows from the optimality of the greedy algorithm for weighted optimization over matroids. We relegate the (fairly standard) proof to the full version [13].

To conclude the proof of Theorem 6, we show in the following lemma that the output of Algorithm 2 satisfies condition (4).

▶ **Lemma 14.** *For all $j \in [n]$ with $w_j > w_{j+1}$, the output set Q of Algorithm 2 satisfies*

$$\mathbb{E}[\boldsymbol{Rank}(Q \cap R \cap \{e_1, \ldots, e_j\})] \geq (1-(1-p)^{\tau})\,\mathbb{E}[\boldsymbol{Rank}(R \cap \{e_1, \ldots, e_j\})]$$

To provide more intuition, let $I_1, \ldots, I_\tau$ be the sets defined in Algorithm 2, and $\overline{E} = \{e_1, \ldots e_j\}$ be the top weight j elements. It is sufficient to show that the sets $I_t \cap \overline{E}$ form a sequence of nested spanning sets for the restricted matroid $\overline{\mathcal{M}}$ on elements $\overline{E}$. The optimal choice of of I_t in Algorithm 2, together with the matroid structure, implies that $I_t \cap \overline{E}$ has full rank in $\overline{\mathcal{M}} \setminus I_{1:t-1}$. We complete the proof in the in the full version [13]. Combining Lemmas 13 and 14 completes the proof of Theorem 6.

5 Improved Sparsifier for Stochastic Weighted Matching

In the instance of stochastic weighted matching $\langle E, \mathcal{I}, f, p\rangle$, the elements E are the edges of a known weighted graph $G := (V, E, w)$, $\mathcal{I}$ is the set of all matchings in the graph G, and f is an additive function with element weights $\{w_e\}_{e\in E}$. For simplicity, we sometimes denote the stochastic matching instance $\langle E, \mathcal{I}, f, p\rangle$ by $\langle G, p\rangle$ when it is clear from the context.

The aim of a sparsifier for this problem is to query a poly $(1/p)$-degree subgraph H of G such that the expected weight of the maximum matching on active edges of H approximates the optimum value of $\langle G, p\rangle$. The current state-of-the-art poly $(1/p)$-degree sparsifier for the stochastic weighted matching problem achieves a 0.501 approximation ratio due to [8][3]. In this section, we present a new poly$(1/p)$-degree sparsifier for the stochastic weighted matching that improves the approximation ratio to 0.536.

Our sparsifier for the stochastic weighted matching problem consists of two phases. In the first phase, it samples a set of edges $Q_{\texttt{CRS}}$ using the generic sparsifier described in Algorithm 1. In the second phase, we independently select T samples $Q_1, \ldots, Q_T$ from the stochastic optimum oracle $\mathcal{D}_{\text{opt}}$, which is similar to the method used in [8]. This second phase alone already provides a 0.501 approximation, but by incorporating the edges sampled in the first phase, we are able to improve the approximation ratio to 0.536. The main result of this section is presented in the following theorem.

Algorithm 3 Sparsifier for Weighted Stochastic Matching Problem $\langle G, p\rangle$.

1: Compute the marginals $\mathbf{q}$ of the stochastic optimum solution.
2: Add each edge $e \in E$ to the set $Q_{\texttt{CRS}}$ independently with probability $\frac{q_e}{p}$.
3: Sample $Q_1, \ldots, Q_T \sim D_{\text{opt}}$ independently and add them to $Q_{\texttt{Greedy}}$ for $T = 1/\epsilon^8 p$.
4: **Output:** $Q = Q_{\texttt{CRS}} \cup Q_{\texttt{Greedy}}$.

▶ **Theorem 15.** *Let $G = (V, E, w)$ be a weighted graph and $p \in (0, 1)$. If the matching polytope of G admits an α-balanced contention resolution scheme, then Algorithm 3 is the $(1 - O(\epsilon)) \cdot \max\left\{\frac{1}{2}, \left(\frac{1+\alpha e^2}{1+e^2}\right)\right\}$-approximate polynomial time sparsifier for the stochastic weighted matching problem $\langle G, p\rangle$ with sparsification degree $O(1/\epsilon^8 p)$.*

Our theorem combined with 0.474-balanced CRS for machining polytope from [17] implies 0.536-approximate sparsifier for stochastic weighted matching. Assuming the conjecture from [20] which states the existence of 0.544 balanced CRS for general matching polytope implies that Algorithm 3 is ~ 0.6 approximate.

The proof of Theorem 15 relies on p being small. So, before we prove the theorem, in Lemma 16, we show that for any $\epsilon > 0$ (constant), without loss of generality we can assume $p \le \epsilon^4$. The proof of this part is rather technical and, we defer it to the full version [13] due to space constraints.

▶ **Lemma 16** (Reduction Lemma). *If there exists an α-approximate sparsifier with degree d/p for the class of stochastic weighted matching with $p \le \epsilon^4$ then there exists an α-approximate sparsifier for the same problem class and arbitrary $p \in (0, 1)$ with sparsification degree $\frac{d}{p\cdot\epsilon^4}$.*

For the rest of the section, we assume that $p \le \epsilon^4$. We first define the set of crucial edges and non-crucial edges formally in the following definition.

[3] Recent work by [6] constructs $(1-\epsilon)$-approximate sparsifier with degree $\exp(\exp(\exp(1/\epsilon, 1/p)))$, however, in this work, we focus on sparsifiers with degree poly$(1/p)$

▶ **Definition 17.** *Given $\langle G, p\rangle$, let q_e be the probability of an edge e being in the stochastic optimum solution. We define* crucial edges *as* $\mathtt{C} := \{e \in E \ : \ q_e \geq \tau(\epsilon)\}$ *and* non-crucial edges *as* $\mathtt{NC} := \{e \in E \ : \ q_e < \tau(\epsilon)\}$ *where* $\tau(\epsilon) := \frac{\epsilon^3 p}{20 \cdot \log \frac{1}{\epsilon}}$ *is the threshold.*

Given $\langle G, p\rangle$ and set of crucial and non-crucial edges $\mathtt{C}$ and $\mathtt{NC}$, we let $\mathbf{OPT}_{\mathtt{C}}$ and $\mathbf{OPT}_{\mathtt{NC}}$ be the contributions of crucial and non-crucial edges in the stochastic optimum, i.e. $\sum_{e \in \mathtt{C}} w_e \cdot q_e$ and $\sum_{e \in \mathtt{NC}} w_e \cdot q_e$. Note that

$$\mathbf{OPT} = \mathbf{OPT}_{\mathtt{C}} + \mathbf{OPT}_{\mathtt{NC}}\,.$$

In order to prove Theorem 15, we provide a procedure to construct a matching $M \subseteq Q \cap R$ such that $\mathbb{E}\left[\sum_{i \in M} w_e\right] \geq (1 - O(\epsilon)) \cdot \max\left\{\frac{1}{2}, \left(\frac{1+\alpha e^2}{1+e^2}\right)\right\} \cdot \mathbf{OPT}$. Our procedure constructs three matchings $M_{\mathtt{C}}, M_{\mathtt{NC}}, M_{\mathtt{AUG}} \subseteq R \cap Q$ and then picks the matching with the maximum weight. We construct matchings $M_{\mathtt{C}}, M_{\mathtt{NC}}$ on the queried active crucial and non-crucial edges in $Q_{\mathtt{Greedy}}$ similar to the [8] which satisfies the desired properties described in Lemma 18 and Lemma 19. First, we state that each crucial edge $e \in \mathtt{C}$ appears in the $Q_{\mathtt{Greedy}}$ with probability $1 - \epsilon$ which shows the existence of matching $M_{\mathtt{C}} \subseteq Q \cap R \cap \mathtt{C}$ with expected weight at least $(1 - \epsilon) \cdot \mathbf{OPT}_{\mathtt{C}}$.

▶ **Lemma 18** (Crucial Edge Lemma [8])**.** *Given a stochastic weighted matching instance $\langle G, p\rangle$ and $Q_{\mathtt{Greedy}}$ is the set defined in Algorithm 3, let $M_{\mathtt{C}}$ be the maximum weight matching in the graph $Q_{\mathtt{Greedy}} \cap \mathtt{C} \cap R$, then $\mathbb{E}\left[\sum_{e \in M_{\mathtt{C}}} w_e\right] \geq (1 - \epsilon) \cdot \boldsymbol{OPT}_{\mathtt{C}}$.*

Now, following the [8, Lemma 4.7], in Lemma 19, we construct a matching $M_{\mathtt{NC}} \subseteq R \cap Q_{\mathtt{Greedy}} \cap \mathtt{NC}$ on active queried non-crucial edges, such that each $e \in \mathtt{NC}$ is present in $M_{\mathtt{NC}}$ with probability at least $(1 - O(\epsilon)) \cdot q_e$. We further prove an important property of $M_{\mathtt{NC}}$ that states that for any non-crucial edge $e \in \mathtt{NC}$, the probability of $e \in M_{\mathtt{NC}}$ can not decrease when we condition on the events that some of the neighbors of e are inactive[4].

▶ **Lemma 19** (Non-Crucial Edges)**.** *Given a stochastic weighted matching instance $\langle G, p\rangle$, let $Q_{\mathtt{Greedy}}$ be the set defined in Algorithm 3. There exists a matching $M_{\mathtt{NC}} \subseteq Q_{\mathtt{Greedy}} \cap \mathtt{NC} \cap R$ such that for any non-crucial edge $e \in \mathtt{NC}$, $\mathbf{Pr}[e \in M_{\mathtt{NC}}] \geq (1 - 12\epsilon) \cdot q_e$. This implies that, $\mathbb{E}\left[\sum_{e \in M_{\mathtt{NC}}} w_e\right] \geq (1 - 12\epsilon) \cdot \boldsymbol{OPT}_{\mathtt{NC}}$. Moreover, for any subset $S \subseteq N(e)$ where $N(e)$ is the set of edges incident to e in graph G, we have*

$$\mathbf{Pr}[e \in M_{\mathtt{NC}} \mid S \cap R = \emptyset] \geq (1 - 12 \cdot \epsilon) \cdot q_e. \tag{5}$$

The proof of the lemma is technically involved and therefore it is delegated to the full version [13]. Lemma 18 and Lemma 19 together imply that our sparsifier is at least 1/2 approximate.

We note that $Q_{\mathtt{CRS}}$ is the output of generic sparsifier discussed in Algorithm 1 (Section 3). Let $M_{\mathtt{CRS}} := \pi(Q_{\mathtt{CRS}} \cap R)$ be the matching constructed by an α-balanced CRS π which ensures $\mathbf{Pr}[e \in M_{\mathtt{CRS}}] \geq \alpha \cdot q_e$ for all $e \in E$. We refer $M_{\mathtt{CRS}}$ as `CRS-BaseMatching`. Crucially, $M_{\mathtt{CRS}}$ is independent of the edges sampled in $Q_{\mathtt{Greedy}}$ as well as $M_{\mathtt{NC}}$ and $M_{\mathtt{C}}$. Using Independence between $M_{\mathtt{CRS}}$ and $M_{\mathtt{NC}}$, we construct the third matching $M_{\mathtt{AUG}}$ on the set of edges $Q_{\mathtt{CRS}} \cup (Q_{\mathtt{Greedy}} \cap \mathtt{NC}) \cup R$. Our augmentation simply adds a non-crucial edge $e \in M_{\mathtt{NC}}$ to `CRS-BaseMatching` if both endpoints of the edge e are unmatched in `CRS-BaseMatching`. Algorithm 4 describes our augmentation procedure in detail.

[4] We noticed a bug in the proof of a similar lemma presented in [8], further used in [7, 6]. In order to prove the lemma and the monotonicity property (5), we require slightly different proof techniques.

Our key observation is that any non-crucial edge $e \in \texttt{NC}$ has a small probability of being sampled in the set $Q_{\texttt{CRS}}$. However, with some non-trivial probability, both endpoints of the edge e will be unmatched in `CRS-BaseMatching`. More formally, first we show that for any non-crucial edge $e := (u, v) \in \texttt{NC}$, both endpoints of e are unmatched in the `CRS-BaseMatching` with probability at least $1/e^2$. The intuition here is that as $p \leq \epsilon^4$, the number of incident edges on the endpoints of the edge e in the set $Q_{\texttt{CRS}}$ are concentrated around $2/p$ with high probability. Such a property ensures that if all these incident edges are inactive, then both endpoints of e are unmatched in `CRS-BaseMatching`.

Later, we use the property (5) of $M_{\texttt{NC}}$ from Lemma 19 to guarantee that when a non-crucial edge $e \notin Q_{\texttt{CRS}}$ and both endpoints of e are unmatched in `CRS-BaseMatching`, we can guarantee that $e \in M_{\texttt{NC}}$ with probability approximately q_e. Therefore, we can add such a non-crucial edge e to `CRS-BaseMatching` with probability approximately $\frac{q_e}{e^2}$. Combining this intuition, we prove the following key lemma whose proof is delegated to the full version [13].

▶ **Lemma 20.** *Let M_{AUG} be the output of the procedure described in Algorithm 4 then,*

$$\mathbf{Pr}[e \in M_{AUG}] \geq q_e \cdot \alpha \ \forall e \in \mathcal{C} \qquad \text{and} \qquad \mathbf{Pr}[e \in M_{AUG}] \geq q_e \cdot \left(\alpha + \frac{1 - O(\epsilon)}{e^2}\right) \ \forall e \in \mathit{NC}.$$

Algorithm 4 Construction of the matching $M_{\texttt{AUG}}$ on $Q \cap R$.

1: $M_{\texttt{NC}}$ be the matching on $Q_{\texttt{Greedy}} \cap R \cap \texttt{NC}$ satisfying property of stated Lemma 19.
2: $M_{\texttt{CRS}} \leftarrow \pi(Q_{\texttt{CRS}} \cap R)$ be the matching produced by α-balanced truncated CRS.
3: $M_{\texttt{AUG}} \leftarrow M_{\texttt{CRS}}$.
4: $\forall e \in M_{\texttt{NC}}$, add e to the matching $M_{\texttt{AUG}}$ if both endpoints of e are unmatched in $M_{\texttt{AUG}}$.

Combining Lemma 18, Lemma 19 and Lemma 20, we show that the expected weight of the best matching among $M_{\texttt{CRS}}, M_{\texttt{NC}}$, and $M_{\texttt{AUG}}$ exhibits the desired approximation ratio. We complete the proof of Theorem 15 in the full version [13].

6 Additive Optimization over the Intersection of k Matroids

Given our $(1 - \epsilon)$-approximate sparsifier for additive optimization over a single matroid constraint, a natural question is whether the natural generalization of this algorithm to the intersection of matroids is $(1 - \epsilon)$-approximate. This turns out to not be the case even for bipartite matching (the intersection of two partition matroids) due to [9]. The main challenge here is that, unlike for a single matroid, multiple solutions for matroid intersection do not always "combine" well. In this section, we prove a slightly weaker sparsification result for additive optimization over the intersection of k matroid constraints, which nevertheless beats the best known bound of $1/(k + 1)$ on the correlation gap of k-matroid intersection (see [1]), and therefore outperforms our generic sparsifier for this problem. The following theorem is the main result of this section.

▶ **Theorem 21.** *For each $\epsilon > 0$, there is a $\frac{(1-\epsilon)}{k+\frac{1}{k+1}}$-approximate sparsifier of degree $O\left(\frac{1}{\epsilon \cdot p} \log \frac{1}{\epsilon}\right)$ for stochastic packing problem $\langle E, \mathcal{I}, f, p\rangle$ when $(E, \mathcal{I})$ is the intersection of k matroids and f is additive.*

Our sparsifier samples $Q_1, \ldots, Q_\tau$ independently from stochastic optimum oracle $\mathcal{D}_{\text{opt}}$ as a sparsifier. Similar algorithms with degree poly$(1/p)$ have been considered for the stochastic matching [8, 7], and were shown to be 0.6568-approximate for the unweighted and 0.501-approximate for a weighted matching with degree poly$(1/p)$.

Algorithm 5 Sparsifier for additive optimization over the intersection of k matroid constraints.

Input: $\langle E, \mathcal{I}, f, p\rangle$ with the intersection of k matroids constraints and additive f; $D_{\mathbf{OPT}}$
Sample $Q_1, \ldots, Q_\tau \sim D_{\mathbf{OPT}}$ independently for $\tau \leftarrow \frac{2}{\epsilon p} \log \frac{2}{\epsilon}$;
Output: $Q = \cup_{i=1}^{\tau} Q_i$.

In order to prove Theorem 21, we provide a procedure for constructing a feasible solution $I \subseteq Q \cap R$ such that $\mathbb{E}[\sum_{i\in I} w_e] \geq \frac{(1-\epsilon)}{k+1/(k+1)}\,\mathbf{OPT}$. The backbone of our analysis lies in Lemma 22. As a first step, let S_1 and S_2 be two independent sets of the same matroid and $R \subseteq E$ be the (random) set of active elements with parameter p. We propose a procedure (details in Algorithm 6 in the full version [13]) that swaps active elements from S_1, i.e. $S_1 \cap R$, with elements of S_2 such that each element of S_2 is "protected" independently with probability $1-p$. Hence, the expected value of updated set S_2 is $\geq \mathbb{E}[f(S_1 \cap R)] + (1-p)\cdot f(S_2)$.

The key intuition here is that the exchange property of matroids allows us to swap any element $e \in S_1$ with a different element $f \in S_2$ without violating the feasibility of S_2. Therefore, if e is inactive then e can not swap out f from S_2 and hence we "protect" f in S_2 with probability $1-p$. However, the main challenge here is after a single swap between e and f, sets S_1 and S_2 get updated and f can potentially be swapped with some $f' \in S_2$. Our procedure overcomes this challenge by carefully choosing swaps of elements between S_1 and S_2 while maintaining feasibility.

We extend this idea to when S_1 and S_2 are two independent sets in the intersection of k matroids. We run the procedure described in Algorithm 6 in the full version [13] for each matroid and obtain sets T feasible in the intersection of all matroids such that each element of S_2 is added to T independently with probability $(1-p)^k$. The details of procedure and proof of Lemma 22 is relegated to the full version of the paper [13]

▶ **Lemma 22.** *Let $\mathcal{M}_1, \ldots \mathcal{M}_k$ be matroids with $\mathcal{M}_\ell = (E, \mathcal{I}_\ell)$, and let $\mathcal{I} = \bigcap_{\ell=1}^{k} \mathcal{I}_\ell$ be their common independent sets. Let S_1 and S_2 be in $\mathcal{I}$. Let $R \subseteq E$ include each element of E independently with probability p. Let $T(\ell) \in \mathcal{I}_\ell$ be the output of Algorithm 6 in full version [13] for matroid $\mathcal{M}_\ell$, for each $\ell \in [k]$. The set $T := \bigcap_{\ell=1}^{k} T(\ell)$ satisfies:*

1. *$S_1 \cap S_2 \subseteq T$ with probability 1.*
2. *$T \in \mathcal{I}$ with probability 1.*
3. *$(S_1 \setminus S_2) \cap R \subseteq T$, i.e. $\mathbf{Pr}[e \in T] = p$ for all $e \in S_1 \setminus S_2$.*
4. *$\mathbf{Pr}[f \in T] \geq (1-p)^k$ for all $f \in S_2 \setminus S_1$*

We utilize the above lemma and propose a procedure to construct a feasible set $I \subseteq Q \cap R$. At a high level, our procedure iteratively observes active elements in the set Q_i and swaps elements in $Q_{i+1}, \ldots, Q_n$ by $Q_i \cap R$ using Lemma 22. To this end, Lemma 22 ensures that each element in Q_j for $j > i$ is not swapped ("protected") with probability at least $(1-p)$. Using this argument inductively, we prove the following lemma that lower bounds the probability of selecting each element $e \in Q$ whose proof is in the full version [13]. We then use the lemma and carefully analyze the probability of each element $e \in E$ in the constructed set $I \subseteq R \cap Q$ to conclude the proof of Theorem 21.

▶ **Lemma 23.** *Let $I^* = I(\tau)$ be the output of Algorithm 7 in full version [13]. For any $e \in E$, we have*

$$\mathbf{Pr}[e \in I^* \mid e \in Q_i \setminus \cup_{\ell=1}^{i-1} Q_i] \geq p \cdot (1-p)^{k(i-1)}$$

7 Open Questions

- We believe that our results portend a deeper connection between the sparsification and contention resolution. The results of Section 3 show that contention resolution serves to lower-bound the sparsification ratio. We ask whether the connection goes both ways. In particular, does the existence of a c-sparsifier of degree $1/p$ imply a contention resolution scheme with balance c? This is intimated by Proposition 5. Does the existence of a c-sparsifier of degree $\text{poly}(1/p)$ imply a contention resolution scheme with balance $\Omega(c)$ (or some other expression involving c and the degree)? Formalizing a tighter connection between sparsification and contention resolution (equivalently, the correlation gap) might lead to new structural and computational insights for the latter.
- In Section 4, we show that a greedy sparsifier $1-\epsilon$ approximate with degree $O(1/p)$ for additive optimization subject to a matroid constraint. We conjecture that a similar greedy sparsifier exists for the intersection of k matroids, obtaining a $\frac{1-\epsilon}{k}$-approximation with degree $O(1/p)$. A similar greedy sparsifier, albeit with degree $O(1/p^{1/\epsilon})$, was shown to be 1/2-approximate for the special case of unweighted bipartite matching in [9].
- Our results in Section 5 improve the state of the art sparsifier for weighted (non-bipartite) matching in the polynomial degree regime. Moreover, since our approximation guarantee is a function of the correlation gap, progress on the correlation gap of the matching polytope will lead to further improved sparsifiers. Finding the best possible sparsification ratio in the polynomial degree regime remains open, however, with $1-\epsilon$ still on the table. Beyond polynomial degree, a $1-\epsilon$ approximate sparsifier with degree $\exp(\exp(\exp(1/p)))$ was already shown by [6].

References

1. Marek Adamczyk and Michał Włodarczyk. Random order contention resolution schemes. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 790–801. IEEE, 2018.
2. Sepehr Assadi and Aaron Bernstein. Towards a unified theory of sparsification for matching problems. *arXiv preprint*, 2018. `arXiv:1811.02009`.
3. Sepehr Assadi, Sanjeev Khanna, and Yang Li. The stochastic matching problem: Beating half with a non-adaptive algorithm. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 99–116, 2017.
4. Sepehr Assadi, Sanjeev Khanna, and Yang Li. The stochastic matching problem with (very) few queries. *ACM Transactions on Economics and Computation (TEAC)*, 7(3):1–19, 2019.
5. Soheil Behnezhad, Avrim Blum, and Mahsa Derakhshan. Stochastic vertex cover with few queries. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1808–1846. SIAM, 2022.
6. Soheil Behnezhad and Mahsa Derakhshan. Stochastic weighted matching: $(1-\epsilon)$ approximation. *arXiv preprint*, 2020. `arXiv:2004.08703`.
7. Soheil Behnezhad, Mahsa Derakhshan, and MohammadTaghi Hajiaghayi. Stochastic matching with few queries:(1-ε) approximation. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1111–1124, 2020.
8. Soheil Behnezhad, Alireza Farhadi, MohammadTaghi Hajiaghayi, and Nima Reyhani. Stochastic matching with few queries: New algorithms and tools. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2855–2874. SIAM, 2019.
9. Avrim Blum, John P Dickerson, Nika Haghtalab, Ariel D Procaccia, Tuomas Sandholm, and Ankit Sharma. Ignorance is almost bliss: Near-optimal stochastic matching with few queries. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 325–342, 2015.

10 Simon Bruggmann and Rico Zenklusen. An optimal monotone contention resolution scheme for bipartite matchings via a polyhedral viewpoint. *Mathematical Programming*, pages 1–51, 2020.

11 Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

12 Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *SIAM Journal on Computing*, 43(6):1831–1879, 2014.

13 Shaddin Dughmi, Yusuf Hakan Kalayci, and Neel Patel. On sparsification of stochastic packing problems. *arXiv preprint*, 2022. `arXiv:2211.07829`.

14 Michel X. Goemans and Jan Vondrák. Covering minimum spanning trees of random subgraphs. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 934–941, USA, 2004. Society for Industrial and Applied Mathematics.

15 Richard M Karp and Michael Sipser. Maximum matching in sparse random graphs. In *22nd Annual Symposium on Foundations of Computer Science (sfcs 1981)*, pages 364–375. IEEE, 1981.

16 Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Mathematics of Operations Research*, 35(4):795–806, 2010.

17 Calum MacRury, Will Ma, and Nathaniel Grammel. On (random-order) online contention resolution schemes for the matching polytope of (bipartite) graphs, 2022. `doi:10.48550/arXiv.2209.07520`.

18 Takanori Maehara and Yutaro Yamaguchi. Stochastic packing integer programs with few queries. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 293–310, USA, 2018. Society for Industrial and Applied Mathematics.

19 Takanori Maehara and Yutaro Yamaguchi. Stochastic monotone submodular maximization with queries. *arXiv preprint*, 2019. `arXiv:1907.04083`.

20 Pranav Nuti and Jan Vondrák. Towards an optimal contention resolution scheme for matchings. *arXiv preprint*, 2022. `arXiv:2211.03599`.

Triangle Counting with Local Edge Differential Privacy

Talya Eden
Bar Ilan University, Ramat Gan, IL

Quanquan C. Liu
Northwestern University, Evanston, IL, US

Sofya Raskhodnikova
Boston University, MA, US

Adam Smith
Boston University, MA, US

Abstract

Many deployments of differential privacy in industry are in the local model, where each party releases its private information via a differentially private randomizer. We study triangle counting in the noninteractive and interactive local model with edge differential privacy (that, intuitively, requires that the outputs of the algorithm on graphs that differ in one edge be indistinguishable). In this model, each party's local view consists of the adjacency list of one vertex.

In the noninteractive model, we prove that additive $\Omega(n^2)$ error is necessary, where n is the number of nodes. This lower bound is our main technical contribution. It uses a reconstruction attack with a new class of linear queries and a novel mix-and-match strategy of running the local randomizers with different completions of their adjacency lists. It matches the additive error of the algorithm based on Randomized Response, proposed by Imola, Murakami and Chaudhuri (USENIX2021) and analyzed by Imola, Murakami and Chaudhuri (CCS2022) for constant ε. We use a different postprocessing of Randomized Response and provide tight bounds on the variance of the resulting algorithm.

In the interactive setting, we prove a lower bound of $\Omega(n^{3/2})$ on the additive error. Previously, no hardness results were known for interactive, edge-private algorithms in the local model, except for those that follow trivially from the results for the central model. Our work significantly improves on the state of the art in differentially private graph analysis in the local model.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases local differential privacy, reconstruction attacks, lower bounds, triangle counting

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.52

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2305.02263`

Funding T.E. was supported by the NSF TRIPODS program, award DMS-2022448, and Boston University. This work was partially done while affiliated with Boston University and MIT. A.S. was supported in part by NSF awards CCF-1763786 and CNS-2120667 as well as Faculty Awards from Google and Apple.

Acknowledgements We thank Iden Kalemaj and Satchit Sivakumar for helpful comments on the initial version of our results.

1 Introduction

Triangle counting is a fundamental primitive in graph analysis, used in numerous applications and widely studied in different computational models [3, 12, 26, 37, 7, 44, 45, 48, 50]. Statistics based on triangle counts reveal important structural information about networks (as discussed, e.g., in [30, 46, 51]). They are used to perform many computational tasks on social networks, including community detection [49], link prediction [25], and spam filtering [5]. See [1] for a survey on algorithms for and applications of triangle counting.

In applications where a graph (e.g., a social network) holds sensitive information, the algorithm that computes on the graph has to protect personal information, such as friendships between specific individuals. Differential privacy [22] has emerged as the standard of rigorous privacy guarantees. See [53] for a survey of differentially private graph analysis. The most investigated setting of differential privacy is called the *central model*. It implicitly assumes a curator that collects all the data, computes on it, and provides data releases. In some situations, however, it might be undesirable to collect all information in one place, for instance, because of trust or liability issues. To address this, the *local model* of differential privacy was proposed [29, 22, 40] and is now used in many industry deployments [28, 8, 14, 2, 17].

In this model, each party releases its private information via a differentially private randomizer. Then the algorithm processes the information and, in the case of the local *noninteractive* model, outputs the answer. In the case of the local *interactive* model, the algorithm may have multiple rounds where it asks all parties to run different randomizers on their private data. These randomizers can have arbitrary dependencies on previous messages. Differential privacy in the local model is defined with respect to the whole transcript of interactions between the parties and the algorithm. In the local model applied to graph data, each vertex represents a party. It receives the list of its neighbors as input and applies local randomizers to it. In contrast to the typical datasets, where information belongs to individual parties, in the graph setting, each pair of parties (vertices) share the information of whether there is an edge between them.

Differential privacy, intuitively, guarantees that, for any two neighboring datasets, the output distributions of the algorithm are roughly the same. There are two natural notions of neighboring graphs: *edge-neighboring* and *node-neighboring*. Two graphs are edge-neighboring if they differ in one edge; they are node-neighboring if they differ in one node and its adjacent edges. Edge differential privacy is, in general, easier to attain, but node differential privacy provides stronger guarantees. Edge differential privacy was introduced and first applied to triangle counting in [47]. The edge-differentially private algorithm from [47] was generalized and implemented in [38]. The first node-differentially private algorithms appeared in [9, 41, 13], and all three of these articles considered the problem of triangle counting. Edge differential privacy in the local model has been studied in [52, 31, 54, 56, 33, 34, 16] with most of the listed articles focusing on triangle counting.

In this work, we investigate edge differentially private algorithms for estimating the number of triangles in a graph in the local model. Our goal is to understand the additive error achievable by such algorithms both in the noninteractive and in the interactive model. For the noninteractive model, we provide upper and lower bounds on additive error. Our bounds are tight in terms of n, the number of nodes in the input graph. For the interactive model, we provide the first lower bound specific to local, edge differentially private (LEDP) algorithms. There are trivial lower bounds for the central model (based on global sensitivity) which apply to the local model, but no lower bounds specific to the local model were previously known for any graph problem, even for 2-round algorithms. Together, our results improve our understanding of both noninteractive and interactive LEDP algorithms.

1.1 Results

Our results and comparison to previous work are summarized in Table 1.

Table 1 Summary of lower and upper bounds on the additive error for triangle counting in the noninteractive and interactive models. Note that the largest value of $C_4(G)$ is $\binom{n}{4} = \Theta(n^4)$. For ease of comparison, the results of [33] and [34] are stated for graphs with $d_{max} = \Theta(n)$.

Model		Previous Results		Our Results	
Non-interactive	Lower Bound	$\Omega(n^{3/2})$	[33]	$\Omega(n^2)$	Thm. 1.1
	Upper Bound	$O(n^2)$ (constant ε)	[35]	$O\left(\frac{\sqrt{C_4(G)}}{\varepsilon} + \frac{n^{3/2}}{\varepsilon^3}\right)$	Thm. 1.2
Interactive	Lower Bound	$\Omega(n)$	easy	$\Omega(\frac{n^{3/2}}{\varepsilon})$	Thm. 1.3
	Upper Bound	$O\left(\frac{\sqrt{C_4(G)}}{\varepsilon} + \frac{n^{3/2}}{\varepsilon^2}\right)$	[34]		

1.1.1 Lower Bound for the Noninteractive Local Model

Our main technical contribution is a lower bound in the noninteractive setting. It uses a reconstruction attack (for the central model) with a new class of linear queries and a novel mix-and-match strategy of running local randomizers with different completions of their adjacency lists. While reconstruction attacks are a powerful tool in proving lower bounds in the central model of differential privacy, they have not been used to obtain bounds in the local model. Previous lower bounds in the local model are based on quite different techniques – typically, information-theoretic arguments (see, for example, [40, 6, 19] and many subsequent works).

▶ **Theorem 1.1** (Noninteractive Lower Bound, informal version). *Let $\varepsilon \in (0, 1/20)$ and $\delta \geq 0$ be a sufficiently small constant. There exists a family of graphs such that every noninteractive (ε, δ)-local edge differentially private algorithm that gets an n-node graph from the family as input and approximates the number of triangles in the graph within additive error at most α (with sufficiently high constant probability) must have $\alpha = \Omega(n^2)$.*

Our lower bound holds for all small $\delta \geq 0$ (the case referred to as "approximate" differential privacy). Observe that such lower bounds are stronger than those for $\delta = 0$ (the case referred to as "pure" differential privacy), because they include $\delta = 0$ as a special case. The only previously known lower bound, due to Imola et al. [33], showed that noninteractive algorithms must have error $\Omega(\sqrt{n} \cdot d_{max})$.

To prove the lower bound in Theorem 1.1, we develop a novel mix-and-match technique for noninteractive local model. For a technical overview of the proof of Theorem 1.1, see Section 1.2.

Our lower bound matches the upper bound of $O(n^2)$ proved by [35, Theorem G.3] (for constant ε) for an algorithm based on randomized response. In this work, we give a simpler variant of the algorithm and a more refined analysis, which works for all ε.

1.1.2 Tight Analysis of Randomized Response

The most natural algorithm for the noninteractive model is Randomized Response, which dates back to Warner [55]. In this algorithm, each bit is flipped with probability $\frac{1}{e^{\varepsilon}+1}$, where ε is the privacy parameter. In the case of graphs, each bit represents a presence or absence of an edge. An algorithm based on Randomized Response for triangle counting was first analyzed by [33] for the special case of Erdős-Rényi graphs, and then [35] proved that this algorithm has $O(n^2)$ additive error for constant ε for general graphs. These works first compute the number of triangles and other induced subgraphs with three vertices as though the noisy edges are real edges and then appropriately adjust the estimate using these counts to make it unbiased.

We use a different postprocessing of Randomized Response. We rescale the noisy edges right away, so we need not compute counts for graphs other than triangles, which makes the analysis much simpler. We obtain tight upper and lower bounds on the variance of the resulting algorithm that hold for all ε. Our bounds are more refined, as they are stated in terms of $C_4(G)$, the number of four cycles in the graph.

▶ **Theorem 1.2** (Analysis of Randomized Response). *For all $\varepsilon > 0$, there exists a noninteractive ε-LEDP algorithm based on Randomized Response that gets an n-node graph as input and returns an unbiased estimate $\hat{T}$ of the number of triangles in a graph that has variance $\Theta\Big(\frac{C_4(G)}{\varepsilon^2} + \frac{n^3}{\varepsilon^6}\Big)$.*

In particular, with high constant probability, $\hat{T}$ has additive error $\alpha = O\Big(\frac{\sqrt{C_4(G)}}{\varepsilon} + \frac{n^{3/2}}{\varepsilon^3}\Big)$.

Note that for constant ε, Theorem 1.2 implies an upper bound of $O(n^2)$ on the additive error of the algorithm's estimate. Thus, Randomized Response is optimal for graphs that have $C_4 = \Theta(n^4)$ by our lower bound in Theorem 1.1. Also, observe that Randomized Response achieves pure differential privacy (with $\delta = 0$), whereas the lower bound in Theorem 1.1 holds even for approximate differential privacy. Even though allowing $\delta > 0$ results in better accuracy for many problems, it does not give any additional utility for noninteractive triangle counting. The proof of Theorem 1.2 is deferred to the full version.

1.1.3 Lower Bound for the Interactive Local Model

Next, we investigate triangle counting in the interactive setting. Imola et al. [33] present an ε-LEDP for triangle counting in the interactive model with additive error of $O\big(\sqrt{C_4(G)}/\varepsilon + \sqrt{n} \cdot d_{max}/\varepsilon^2\big)$, where d_{max} is an upper bound on the maximum degree.

We give a lower bound on the additive error of LEDP algorithms for triangle counting in the interactive model. Note that $\Omega(n)$ additive error is unavoidable for triangle counting even in the central model, because the (edge) global sensitivity of the number of triangles is $n-2$ (and this lower bound is tight in the central model). There were no previously known lower bounds for this problem (or any other problem on graphs) specific to the interactive LEDP model that applied to even 2-round algorithms. Our lower bound applies to interactive algorithms with *any* number of rounds.

▶ **Theorem 1.3** (Interactive Lower Bound). *There exist a family of graphs and a constant $c > 0$ such that for every $\varepsilon \in (0,1)$, $n \in \mathbb{N}, \alpha \in (0, n^2]$ and $\delta \in \Big[0, \frac{1}{10^5} \cdot \frac{\varepsilon^3 \alpha_0^2}{n^5 \ln(n^3/\varepsilon\alpha_0)}\Big]$, every (potentially interactive) (ε, δ)-local edge differentially private algorithm that gets an n-node graph from the family as input and approximates the number of triangles in the graph with additive error at most α (with probability at least 2/3) must have $\alpha \geq c \cdot \frac{n^{3/2}}{\varepsilon}$.*

Our lower bound is obtained via a reduction from the problem of computing the summation of n randomly sampled bits in $\{0,1\}$ in the LDP model, studied in a series of works [6, 10, 20, 36]. Our lower bound matches the upper bound of [33] for constant ε and for graphs where $d_{max} = \Theta(n)$ and $C_4(G) = O(n^3)$. It is open whether additive error of $o(n^2)$ can be achieved for general graphs.

1.2 Technical Overview of the Noninteractive Lower Bound

Typical techniques for proving lower bounds in the local model heavily rely on two facts that hold for simpler datasets: first, each party's information is not seen by other parties; second, arbitrary changes to the information of one party have to be protected. Both of these conditions fail for graphs in the LEDP model: each edge is shared between two parties, and only changes to one edge are protected in the strong sense of neighboring datasets, imposed by differential privacy.

To overcome these difficulties, we develop a new lower bound method, based on reconstruction attacks in the central model. Such attacks use accurate answers to many queries to reconstruct nearly all the entries of a secret data set [18, 23, 24, 42, 43, 15]. They are usually applied to algorithms that release many different values. However, a triangle-estimation algorithm returns a single number. Consider a naïve attempt to mount an attack using the algorithm as a black box, that is, by simulating every query using a separate invocation of the triangle counting algorithm. This would require us to run the local randomizers many times, degrading their privacy parameters and making a privacy breach vacuous.

To overcome this difficulty, in our attack, we use the noninteractive triangle-estimation algorithm as a *gray box*. Since the algorithm is noninteractive, it is specified by local randomizers for all vertices and a postprocessing algorithm that runs on the outputs of the randomizers. We use a secret dataset X to create a secret subgraph, run the randomizers for the vertices in the secret subgraph only twice, and publish the results. By properties of the randomizers and by composition, the resulting procedure is differentially private. In the next phase, we postprocess the published information to complete the secret subgraph to different graphs corresponding to the queries needed for our attack. Then we feed these graphs to the triangle approximation algorithm, except that for the vertices in the secret subgraph, we rely only on the published outputs. If the triangle counting algorithm is accurate, we get accurate answers to our queries. Even though the randomness used to answer different queries is correlated, we show that a good approximation algorithm for triangle counts allows us to get most of the queries answered correctly. Finally, we use a novel anti-concentration bound (Lemma 1.4, below) to demonstrate that our attack succeeds in reconstructing most of the secret dataset with high probability. This shows that the overall algorithm we run in this process is not differentially private, leading to the conclusion that a very accurate triangle counting algorithm cannot exist in the noninteractive LEDP setting.

We call the queries used in our attack *outer-product queries*. The queries are linear, but their entries are dependent. To define this class of queries, we represent the secret dataset X with n^2 bits as an $n \times n$ matrix. An outer-product query to X specifies two vectors A and B of length n with entries in $\{-1,1\}$ and returns $A^T X B$, that is, $\sum_{i,j\in[n]} A_i X_{ij} B_j$.

To analyze our reconstruction attack, we prove the following anti-concentration bound for random outer-product queries, which might be of independent interest.

▶ **Lemma 1.4** (Anti-concentration for random outer-product queries). *Let M be an $n \times n$ matrix with entries $M_{ij} \in \{-1,0,1\}$ for all $i,j \in [n]$ and m be the number of nonzero entries in M. Let A and B be drawn uniformly and independently from $\{-1,1\}^n$. If $m \geq \gamma n^2$ for some constant γ, then*

$$\Pr\left[|A^T M B| > \frac{\sqrt{m}}{2}\right] \geq \frac{\gamma^2}{16}.$$

The literature on reconstruction attacks describes other classes of dependent queries [42]; the outer-product queries arising here required a new and qualitatively different analysis.

1.3 Additional Related Work

One of the difficulties with proving lower bounds in the local model is that Randomized Response, despite providing strong privacy guarantees, supplies enough information to compute fairly sophisticated statistics. For example, Gupta, Roth and Ullman [32] show how the output of Randomized Response can be used to estimate the density of all cuts in a graph. Karwa et al. [39] show how to fit exponential random graph models based on randomized response output. For certain model families, this would entail estimation of the number of triangles; however, they provide no theoretical error analysis, only experimental evidence for convergence. Randomized Response has also been studied in the statistics literature with a focus on small probabilities of flipping an edge. Balachandran et al. [4] analyze the distribution of the naive estimator that counts the number of triangles in the randomized responses (when flip probabilities are very low). Chang et al. [11] give estimation strategies for settings where the flip rate is unknown but multiple replicates with independent noise are available. To the best of our understanding, these works do not shed light on the regime most relevant to privacy, where edge-flip probabilities are close to 1/2.

A number of works have looked at triangle counting and other graph problems in the empirical setting [54, 52, 31, 56] in "decentralized" privacy models. In all but [54], the local view consists of the adjacency list. The local views in Sun et al. [54] consist of two-hop neighborhoods. Such a model results in less error since nodes can see all of their adjacent triangles and can report their adjacent triangles using the geometric mechanism.

1.4 Organization

Various models of differential privacy, including LEDP, are defined in Section 2. Our proof of the lower bound for the noninteractive model, Theorem 1.1, appears in Section 3. The anti-concentration lemma for out-product queries (Lemma 1.4) is proved in Section 3.2. Our analysis of Randomized Response and the proof of Theorem 1.2 appears in the full version [27]. The proof of Theorem 1.3 for the interactive LEDP model appears in Section 4.

2 Background on Differential Privacy

We begin with the definition of differential privacy that applies to datasets represented as vectors as well as to graph datasets.

▶ **Definition 2.1** (Differential Privacy [22, 21]). *Let $\varepsilon > 0$ and $\delta \in [0, 1)$. A randomized algorithm $\mathcal{A}$ is* (ε, δ)-differentially private (DP) *(with respect to the neighbor relation on the universe of the datasets) if for all events S in the output space of $\mathcal{A}$ and all neighboring datasets X and X',*

$$\Pr[\mathcal{A}(x) \in S] \leq \exp(\varepsilon) \cdot \Pr[\mathcal{A}(X') \in S] + \delta.$$

When $\delta = 0$, the algorithm is ε-differentially private (sometimes also called "purely differentially private").

Differential privacy can be defined with respect to any notion of neighboring datasets. When datasets are represented as vectors, datasets X and Y are consider neighbors if they differ in one entry. In the context of graphs, there are two natural notions of neighboring graphs that can be used in the definition: edge-neighboring and node-neighboring. We use predominantly the former, but define both to make discussion of previous work clear.

▶ **Definition 2.2.** *Two graphs $G = (V, E)$ and $G' = (V', E')$ are **edge-neighboring** if G and G' differ in exactly one edge, that is, if $V = V'$ and E and E' differ in exactly one element. Two graphs are **node-neighboring** if one can be obtained from the other by removing a node and its adjacent edges.*

If the datasets are graphs with edge (respectively, node) neighbor relationship, we call a differentially private algorithm simply *edge-private* (respectively, *node-private*).

2.1 The local model

The definition of differential privacy implicitly assumes a trusted curator that has access to the data, runs a private algorithm on it, and releases the result. This setup is called the *central model* of differential privacy. In contrast, in the *local model* of differential privacy, each party participating in the computation holds its own data. The interaction between the parties is coordinated by an algorithm $\mathcal{A}$ that accesses data via *local randomizers*. A local randomizer is a differentially private algorithm that runs on the data of one party. In the context of graph datasets, the input graph is distributed among the parties as follows: each party corresponds to a node of the graph and its data is the corresponding row in the adjacency matrix of the graph. In each round of interaction, the algorithm $\mathcal{A}$ assigns each party a local randomizer (or randomizers) that can depend on the information obtained in previous rounds.

We adapt the definition of local differential privacy from [36, 40] to the graph setting. Consider an undirected graph $G = ([n], E)$ represented by an $n \times n$ adjacency matrix A. Each party $i \in [n]$ holds the i-th row of A, denoted $\mathbf{a}_{i*}$. We sometimes refer to $\mathbf{a}_{i*}$ as the *adjacency vector* of party i. Entries of A are denoted a_{ij} for $i, j \in [n]$.

▶ **Definition 2.3** (Local Randomizer)**.** *Let $\varepsilon > 0$ and $\delta \in [0, 1)$. An (ε, δ)-**local randomizer** $R : \{0,1\}^n \to \mathcal{Y}$ is an (ε, δ)-edge DP algorithm that takes as input the set of neighbors of one node, represented by an adjacency vector $\mathbf{a} \in \{0,1\}^n$. In other words, $\mathsf{Pr}[R(\mathbf{a}) \in Y] \leq e^\varepsilon \cdot \mathsf{Pr}[R(\mathbf{a}') \in Y] + \delta$ for all $\mathbf{a}$ and $\mathbf{a}'$ that differ in one bit and all sets of outputs $Y \subseteq \mathcal{Y}$. The probability is taken over the random coins of R (but* not *over the choice of the input). When $\delta = 0$, we say that R is an ε-local randomizer.*

A randomized algorithm $\mathcal{A}$ on a distributed graph is (ε, δ)-LEDP if it satisfies Definition 2.4.

▶ **Definition 2.4** (Local Edge Differential Privacy)**.** *A **transcript** π is a vector consisting of 5-tuples $(S_U^t, S_R^t, S_\varepsilon^t, S_\delta^t, S_Y^t)$ – encoding the set of parties chosen, set of randomizers assigned, set of randomizer privacy parameters, and set of randomized outputs produced – for each round t. Let S_π be the collection of all transcripts and S_R be the collection of all randomizers. Let $\perp$ denote a special character indicating that the computation halts. An **algorithm** in this model is a function $\mathcal{A} : S_\pi \to (2^{[n]} \times 2^{S_R} \times 2^{\mathbb{R}^{\geq 0}} \times 2^{\mathbb{R}^{\geq 0}}) \cup \{\perp\}$ mapping transcripts to sets of parties, randomizers, and randomizer privacy parameters. The length of the transcript, as indexed by t, is its round complexity.*

*Given $\varepsilon \geq 0$ and $\delta \in [0,1)$, a randomized algorithm $\mathcal{A}$ on a (distributed) graph G is (ε,δ)-**locally edge differentially private (LEDP)** if the algorithm that outputs the entire transcript generated by $\mathcal{A}$ is (ε,δ)-edge differentially private on graph G. When $\delta = 0$, we say that $\mathcal{A}$ is an ε-LEDP.*

*If $t = 1$, that is, if there is only one round, then $\mathcal{A}$ is called **noninteractive**. Otherwise, $\mathcal{A}$ is called **interactive**.*

Observe that a noninteractive LEDP algorithm is specified by a local randomizer for each node and a postprocessing algorithm $\mathcal{P}$ that takes the outputs of the local randomizers as input.

We use a local algorithm known as *randomized response*, initially due to [55], but since adapted to differential privacy [40].

▶ **Definition 2.5** (Randomized Response). *Given a privacy parameter $\varepsilon > 0$ and a k-bit vector $\mathbf{a}$, the algorithm $\mathsf{RandomizedResponse}_\varepsilon(\mathbf{a})$ outputs a k-bit vector, where for each $i \in [k]$, bit i is a_i with probability $\frac{e^\varepsilon}{e^\varepsilon+1}$ and $1-a_i$ otherwise.*

▶ **Theorem 2.6** (Randomized Response is ε-LR). *Randomized response is an ε-local randomizer.*

Additional privacy tools are described in the full version of this paper.

3 The Noninteractive Lower Bound

In this section, we prove Theorem 1.1, which we restate formally here.

▶ **Theorem 3.1.** *There exists a family of graphs, such that every noninteractive (ε,δ)-LEDP algorithm with $\varepsilon \in (0, \frac{1}{20})$ and $\delta \in [0, \frac{1}{100})$ that gets an n-node graph from the family as an input and approximates the number of triangles in the graph within additive error α with probability at least $1 - \frac{1}{3^6 \cdot 2^7}$, must have $\alpha = \Omega(n^2)$.*

At a high level, the lower bound is proved by showing that a noninteractive local algorithm for counting triangles can be used to mount a reconstruction attack on a secret dataset X in the central model of differential privacy. A groundbreaking result of Dinur and Nissim [18] – generalized in subsequent works [23, 24, 42, 43, 15] – shows that if an algorithm answers too many random linear queries on a sensitive dataset of N bits too accurately then a large constant fraction of the dataset can be reconstructed. This is referred to as a "reconstruction attack". Specifically, Dinur and Nissim show that N random linear queries answered to within $\pm O(\sqrt{N})$ are sufficient for reconstruction. It is well known that if the output of an algorithm on a secret dataset can be used for reconstruction, then this algorithm is not differentially private. This line of reasoning leads to a lower bound of $\Omega(\sqrt{N})$ on the additive error of any differentially private algorithm answering N random linear queries.

Suppose we could show that an LEDP triangle counting algorithm with $O(n^2)$ additive error can be used to construct a DP algorithm for answering n linear queries with $O(\sqrt{n})$ additive error on some data set of size n – then by the above, we reach a contradiction to the privacy of the algorithm. While indeed a triangle counting algorithm can be used to answer a *single* linear query, the main challenge is that the Dinur-Nissim reconstruction attack requires answering not one, but rather n, linear queries on the same dataset. Let $\mathcal{A}$ be an (ε,δ)-LEDP triangle counting algorithm. If we naively try to answer each linear query to X using a new invocation of the triangle counting algorithm in a black-box manner, this would result in n invocations of $\mathcal{A}$. This in turn would cause the privacy parameters to grow linearly with n, making the privacy breach vacuous. That is, the result would be of

the following sort. An (ε, δ)-LEDP algorithm for triangle counting with low additive error implies an $(O(\varepsilon n), O(n\delta))$-DP algorithm for answering linear queries with low additive error. Since the latter statement is too weak to be used with the results of Dinur and Nissim, we take a different approach.

In order to avoid making n invocations of a triangle counting algorithm, we develop a new type of reconstruction attack on a secret dataset X, where the set of allowed linear queries has a special combinatorial structure. We call the new type of queries *outer-product queries*. We show that, given access to an (ε, δ)-LEDP algorithm $\mathcal{A}$ that approximates the number of triangles up to $O(n^2)$ additive error, we can design a $(2\varepsilon, 2\delta)$-DP algorithm $\mathcal{B}$ for answering $\Theta(n^2)$ outer-product queries on dataset X of size $N = n^2$, so that a constant fraction of them is answered with $O(n)$ additive error. (The dataset size is n^2, so asymptotically the number of random queries and the required accuracy are the same as in the Dinur-Nissim attack.) The main insight is that instead of using $\mathcal{A}$ as a black-box, we use it in a "gray-box" manner. This allows us to answer all $\Theta(n^2)$ queries without degrading the privacy parameters of $\mathcal{B}$. This in turn allows us to reconstruct X, which is a contradiction to the privacy of algorithm $\mathcal{B}$, and thus also to the privacy of algorithm $\mathcal{A}$. Hence, we conclude that any LEDP triangle-counting algorithm must have $\Omega(n^2)$ additive error.

The rest of Section 3.1 is organized as follows. In Section 3.1, we define outer-product queries, and show that an (ε, δ)-DP algorithm $\mathcal{A}$ for triangle-counting with low additive error can be used to construct a $(2\varepsilon, 2\delta)$-DP algorithm $\mathcal{B}$ for answering outer-product queries with low additive error. In Section 3.2, we prove an anti-concentration result for random outer-product queries. In Section 3.3, we use the anti-concentration result to show that an algorithm $\mathcal{B}$ that accurately answers $\Theta(n^2)$ outer-product queries on a sensitive data set $X \in \{0,1\}^{n \times n}$ can be used to reconstruct most of X and complete the proof of Theorem 3.1.

3.1 Reduction from Outer-product Queries to Triangle Counting

In this section, we prove Lemma 3.3, which is at the heart of our reduction. It shows that, given access to an (ε, δ)-LEDP algorithm $\mathcal{A}$ for approximating the number of triangles with low additive error, we can construct an $(2\varepsilon, 2\delta)$-DP algorithm $\mathcal{B}$ (in the central model) that accurately answers $\Theta(n^2)$ outer-product queries on a sensitive data set X. We start by formally defining this new class of queries.

▶ **Definition 3.2** (Outer-product queries). *Let $X \in \{0,1\}^{n \times n}$. An* outer-product query *to X specifies two vectors A and B of length n with entries in $\{-1, 1\}$ and returns $A^T X B$, that is, $\sum_{i,j \in [n]} A_i X_{ij} B_j$.*

Let γ be the desired reconstruction parameter that indicates that the attack has been successful if we reconstruct at least $(1-\gamma)n^2$ bits of X correctly. (Later, in Section 3.3, γ will be set to $\frac{1}{9}$ and the number of queries, k, will be set to $\Theta(n^2)$.)

▶ **Lemma 3.3** (Answering Outer-product Queries via Triangle Counting). *Let $\varepsilon, \delta > 0$ and $\gamma \in (0, 1/2)$. Assume that there is a noninteractive (ε, δ)-LEDP algorithm $\mathcal{A}$ that, for every $3n$-node graph, approximates the number of triangles with probability at least $1 - \frac{\gamma^2}{9 \cdot 128}$ and has additive error at most $\frac{\sqrt{\gamma} n^2}{20}$. Then there is an $(2\varepsilon, 2\delta)$-DP algorithm $\mathcal{B}$ in the central model that, for every secret dataset $X \in \{0,1\}^{n \times n}$ and every set of k outer-product queries $(A^{(1)}, B^{(1)}), \ldots, (A^{(k)}, B^{(k)})$, gives answers $a_1, \ldots, a_k$ satisfying*

$$\Pr\left[\left|\left\{\ell \in [k] : \left|(A^{(\ell)})^T X B^{(\ell)} - a_\ell\right| > \frac{\sqrt{\gamma} n}{4}\right\}\right| > \frac{\gamma^2 k}{64}\right] \leq \frac{1}{6}.$$

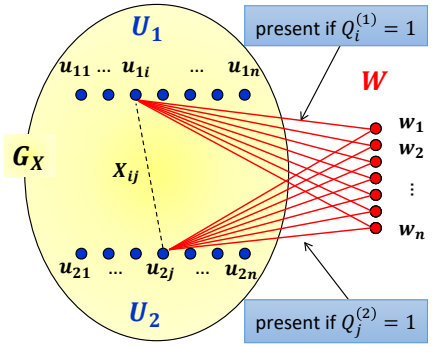


Figure 1 The construction of the query graph $G_{X,Q}$. Each of the parts U_1, U_2, W consists of n nodes. The dashed line is an edge iff $X_{ij} = 1$. Only the subgraph G_X (induced by $U_1 \cup U_2$) holds secret information.

That is, with probability at most $5/6$*, for every dataset* X *and a set of* k *outer-product queries, Algorithm* $\mathcal{B}$ *answers inaccurately at most* $\frac{\gamma^2 k}{64}$ *of the* k *queries, where by inaccurately we mean with additive error more than* $\frac{\sqrt{\gamma}n}{4}$*.*

Proof. Consider an algorithm $\mathcal{A}$ described in the premise of the lemma. Since $\mathcal{A}$ is local noninteractive, it is specified by a local randomizer $R_v(\mathbf{a})$ for each vertex v, as well as a postprocessing algorithm $\mathcal{P}$. Each randomizer takes an adjacency vector $\mathbf{a} \in \{0,1\}^n$ as input and passes its output to $\mathcal{P}$. Next, we define algorithm $\mathcal{B}$ that, given a sensitive dataset X and a set of k outer-product queries, uses the randomizers and the postprocessing algorithm as subroutines to obtain accurate answers to the outer-product queries.

Fix a dataset $X \in \{0,1\}^{n \times n}$. For each outer-product query (A, B), algorithm $\mathcal{B}$ constructs several corresponding *query* graphs. All query graphs are on the same vertex set V of size $3n$, partitioned into three sets U_1, U_2, and W of size n. The vertices in U_t for $t \in \{1,2\}$ are denoted $u_{t1}, \ldots, u_{tn}$. The vertices of W are denoted $w_1, \ldots, w_n$. See Figure 1 for an illustration.

Algorithm $\mathcal{B}$ first forms a bipartite graph G_X with parts U_1 and U_2 with X as the adjacency matrix; that is, it adds an edge (u_{1i}, u_{2j}) for each $i, j \in [n]$ with $X_{ij} = 1$. We call G_X the *secret subgraph*, because it will be included as a subgraph in every query graph and it will be the only part of that graph that contains any information about the original sensitive dataset X. Note that G_X does not depend on the outer-product query. The remaining edges of each query subgraph are between $U_1 \cup U_2$ and W and are specific to each query graph, so that overall the resulting graph is tripartite. For each $v \in U_1 \cup U_2$, let $\Gamma_X(v)$ denote the neighbors of v in the secret subgraph G_X. A key idea in the construction is that every node in the secret subgraph G_X will have one of only two possible neighborhoods in each query graph. This allows algorithm $\mathcal{B}$ to simulate triangle-counting computations on all query graphs by invoking a local randomizer on each vertex in $U_1 \cup U_2$ only twice. For each vertex $v \in U_1 \cup U_2$, algorithm $\mathcal{B}$ runs its local randomizer $R_v(\cdot)$ twice: once with the adjacency list specified by $\Gamma_X(v)$ and once with the adjacency list specified by $\Gamma_X(v) \cup W$. Algorithm $\mathcal{B}$ then records the output of the former invocation as as $r_0(v)$, and the latter as $r_1(v)$.

By the composition property of differential privacy, the algorithm that simply outputs the vector of all $4n$ responses of the local randomizers is $(2\varepsilon, 2\delta)$-DP by composition, because each bit of X is encoded as a potential edge and used in two executions of the randomizers for its endpoints, where each execution (of all randomizers) is (ε, δ)-LEDP. In the remaining steps, algorithm $\mathcal{B}$ only postprocesses the vector of responses, and thus it is $(2\varepsilon, 2\delta)$-DP.

Next, we describe how to postprocess the vector of responses to obtain an answer to an outer-product query (A, B). To answer each outer-product query, algorithm $\mathcal{B}$ will first obtain answers to three linear queries that we call *submatrix queries*. Submatrix queries are defined the same way as outer-product queries, except that vectors A and B have entries in $\{0, 1\}$ instead of $\{-1, 1\}$. Next, we explain how to answer submatrix queries on X, deferring to Claim 3.5 the description of the simulation of each outer-product query with submatrix queries.

To answer a submatrix query $Q = (Q^{(1)}, Q^{(2)})$ on dataset X, algorithm $\mathcal{B}$ completes the secret subgraph G_X to a query graph $G_{X,Q}$ as follows. For each vertex $u_{ti} \in U_1 \cup U_2$, where $t \in \{1, 2\}$ and $i \in [n]$, it adds edges determined by $Q^{(t)}$: specifically, if $Q_i^{(t)} = 1$, it adds edges from u_{ti} to all vertices in W. Next claim states the relationship between the number of triangles in $G_{X,Q}$ and the answer to the submatrix query Q.

▷ Claim 3.4. The number of triangles in graph $G_{X,Q}$ is equal to $n \cdot (Q^{(1)})^T X Q^{(2)}$.

Proof. Observe that $G_{X,Q}$ is tripartite with parts (U_1, U_2, W), so all triangles must have one vertex in each part. The answer to the submatrix query $Q = (Q^{(1)}, Q^{(2)})$ is

$$(Q^{(1)})^T X Q^{(2)} = \sum_{i,j \in [n]} Q_i^{(1)} Q_j^{(2)} X_{ij}.$$

For each term in the sum, both u_{1i} and u_{2j} are adjacent to all nodes in W iff $Q_i^{(1)} = Q_j^{(2)} = 1$. If the edge (u_{1i}, u_{2j}) is present in the graph, then this results in n triangles. Thus, each term where $Q_i^{(1)} = Q_j^{(2)} = X_{ij} = 1$ corresponds to n triangles of the form (u_{1i}, u_{2j}, w_ℓ), where $\ell \in [n]$. All other terms create no triangles, since either $X_{ij} = 0$, in which case the edge (u_{1i}, u_{2j}) is not present in the graph, or either $Q_i^{(1)} = 0$ or $Q_j^{(2)} = 0$, in which case u_{1i} and u_{2j} do not have common neighbors. ◁

To answer a submatrix query Q, algorithm $\mathcal{B}$ simulates a call to the triangle-counting algorithm $\mathcal{A}$ on the corresponding query graph $G_{X,Q}$. First, $\mathcal{B}$ runs the local randomizers for the vertices in W with their adjacency vectors specified by the graph $G_{X,Q}$. Note that these vertices do not have access to any private information, so this operation does not affect privacy. For each vertex $u_{ti} \in U_1 \cup U_2$, where $t \in \{1, 2\}$ and $i \in [n]$, algorithm $\mathcal{B}$ uses the result $r_b(u_{ti})$ from the previously run randomizer, where $b = Q_i^{(t)}$. E.g., if $Q_i^{(1)} = 0$, then $\mathcal{B}$ uses the result $r_0(u_{1i})$, and if $Q_i^{(1)} = 1$, it uses the result $r_1(u_{1i})$. Now algorithm $\mathcal{B}$ has results from all vertex randomizers on the graph $G_{X,Q}$ and it simply runs the postprocessing algorithm $\mathcal{P}$ on these results. To obtain the answer to the submatrix query, $\mathcal{B}$ divides the output of $\mathcal{P}$ by n.

Finally, algorithm $\mathcal{B}$ answers each outer-product query as specified in the following claim, by getting answers to three submatrix queries.

▷ Claim 3.5. An outer-product query to X can be simulated with three submatrix queries to X. Moreover, if all three submatrix queries are answered with additive error at most α, then the outer product query can be answered with additive error at most 5α.

Proof. Consider an outer-product query to an $n \times n$ matrix X specified by $A, B \in \{-1, 1\}^n$. Define n-bit vectors $A' = \frac{1}{2}(A + \vec{1})$ and $A'' = \frac{1}{2}(-A + \vec{1})$, where $\vec{1}$ denotes a vector of 1s of length n. Define B' and B'' analogously. Then, as illustrated in Figure 2,

$$A^T X B = 2((A')^T X B' + (A'')^T X B'') - \vec{1}^T X \vec{1}.$$

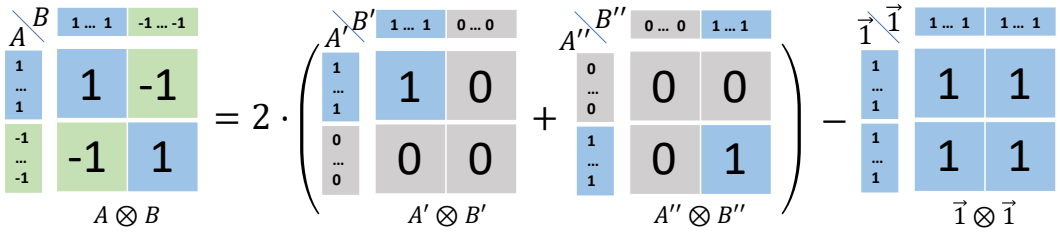


Figure 2 Every outer-product query can be simulated using three submatrix queries. For the illustration, the entries of all vectors are rearranged to group the same values together. The outer product is denoted $\otimes$.

That is, the answer to the outer-product query (A, B) can be computed from the answers to the submatrix queries $(A', B'), (A'', B'')$, and $(\vec{1}, \vec{1})$, and the additive error increases from α to 5α, as stated. ◁

It remains to prove the following claim.

▷ Claim 3.6. Let $\mathcal{A}$ be as in the premise of Lemma 3.3. For every secret dataset $X \in \{0,1\}^{n \times n}$ and every set of k outer-product queries $\{(A^{(\ell)}, B^{(\ell)})\}_{\ell \in [k]}$, algorithm $\mathcal{B}$ gives answers $a_1, \ldots, a_k$ satisfying

$$\Pr\left[\left|\left\{\ell \in [k] : \left|(A^{(\ell)})^T X B^{(\ell)} - a_\ell\right| > \frac{\sqrt{\gamma}n}{4}\right\}\right| > \frac{\gamma^2 k}{64}\right] \leq \frac{1}{6}.$$

That is, the number of "incorrectly" answered outer-product queries exceeds $\frac{\gamma^2 k}{64}$ with probability at most $1/6$.

Proof. By the assumption on $\mathcal{A}$, for every graph G, algorithm $\mathcal{A}$ returns the number of triangles in G within an additive error at most $\frac{\sqrt{\gamma}n^2}{20}$ with probability at least $1 - \frac{\gamma^2}{9 \cdot 128}$.

Given a secret dataset X and k outer-product queries, algorithm $\mathcal{B}$ first creates k triples of submatrix queries corresponding to the outer-product queries. Then $\mathcal{B}$ uses $\mathcal{A}$ as a gray box, to answer all $3k$ submatrix queries simultaneously. Recall that this is achieved by invoking the local randomizers on vertices holding private information (that is, vertices in parts U_1, U_2) twice, once for each potential value of the bit that corresponds to this vertex in a specific query. Then for each individual submatrix query, one local randomizer is invoked on each of the n vertices in W with the adjacency list that corresponds to that specific query graph. Then, to answer each specific submatrix query, algorithm $\mathcal{B}$ combines the new outputs of the vertices from W with the stored outputs from running randomizers on $U_1 \cup U_2$ that correspond to that specific query, and invokes the postprocessing algorithm $\mathcal{P}$ on this vector of $3n$ outputs. Finally, $\mathcal{B}$ divides $\mathcal{P}$'s answer by n to obtain the answer to the submatrix query.

Each invocation of $\mathcal{P}$ by $\mathcal{B}$ simulates one triangle-counting computation. Overall, we have $3k$ (*dependent*) simulated triangle-counting computations. By the assumption on $\mathcal{A}$, stated in the premise of Lemma 3.3, the postprocessing algorithm $\mathcal{P}$ answers each simulated triangle-counting computation inaccurately (i.e., with additive error exceeding $\frac{\sqrt{\gamma}n^2}{20}$) with probability at most $\frac{\gamma^2}{9 \cdot 128}$ (where this probability is taken over the random coins of the individual $3n$ local randomizers, as well as the random coins of $\mathcal{P}$). Overall, there are $3k$ (dependent) simulations, and so the expected number of simulated triangle-counting computations for which $\mathcal{A}$ returns additive error greater than $\frac{\sqrt{\gamma}n^2}{20}$ is at most $\frac{\gamma^2 \cdot (3k)}{9 \cdot 128} = \frac{\gamma^2 \cdot k}{6 \cdot 64}$. Hence, by Markov's inequality, the probability that the number of inaccurate simulated triangles queries exceeds $\frac{\gamma^2 \cdot k}{64}$ is at most $\frac{1}{6}$.

Condition on the event that at most $\frac{\gamma^2 \cdot k}{64}$ of the triangle-counting computations are answered inaccurately, so that the remaining computations are answered with error at most $\alpha = \frac{\sqrt{\gamma} n^2}{20}$, and denote this even by E. Recall that each triangle-counting computation is used to answer a single submatrix query, and that by Claim 3.4, if a triangle-counting computation is answered with additive error α, then the corresponding submatrix query is answered with additive error α/n. Hence, by the above conditioning, at most $\frac{\gamma^2 \cdot k}{64}$ of the submatrix queries are answered with additive error greater than α/n. Each inaccurate answer to a triangle-counting computation can spoil the answer to at most one outer-product query. Furthermore, by Claim 3.5, if all three submatrix queries used to compute a single outer-product query are answered to within additive error α/n, then the outer-product query is answered to within additive error $5\alpha/n$. Hence, by the above conditioning, at most $\frac{\gamma^2 \cdot k}{64}$ of the outer-product queries are answered with additive error greater than $5\alpha/n = \frac{\sqrt{\gamma} n}{4}$. Since event E occurs with probability at least $5/6$, we get that with probability at least $5/6$, the fraction of outer-product queries that is answered with additive error greater than $\frac{\sqrt{\gamma} n}{4}$ is at most $\frac{\gamma^2 \cdot k}{64}$, so that Claim 3.6 holds. ◁

This completes the proof of Lemma 3.3. ◀

3.2 Anti-Concentration for Random Outer-Product Queries

In this section, we prove Lemma 1.4. To analyze our reconstruction attack, we will consider the differences between the true dataset X and a potential reconstructed dataset Y. Let M denote $X - Y$. Then, for an outer-product query (A, B), the difference between the answers to this query on dataset X and on dataset Y is $A^T X B - A^T Y B = A^T M B$. The main result of this section shows that if X and Y differ on many entries (that is, M has lots of nonzero entries) then a random outer-product query is likely to produce significantly different answers on X and Y.

Proof of Lemma 1.4. Let $Z_{ij} = A_i B_j$ for all $i, j \in [n]$, and $U = A^T M B$. We prove the lemma by computing the expectation and the second and the fourth moments of U, and then apply the Paley-Zigmund inequality to U^2.

By independence of A_i and B_j for all $i, j \in [n]$, we have $\mathbb{E}[Z_{ij}] = \mathbb{E}[A_i] \cdot \mathbb{E}[B_j] = 0$ and $\mathrm{Var}[Z_{ij}] = \mathbb{E}[Z_{ij}^2] = \mathbb{E}[A_i^2 B_j^2] = 1$. By definition of U and the linearity of expectation,

$$\mathbb{E}[U] = \mathbb{E}[A^T M B] = \mathbb{E}\Big[\sum_{i,j \in [n]} M_{i,j} Z_{i,j}\Big] = \sum_{i,j \in [n]} M_{i,j} \mathbb{E}[Z_{i,j}] = 0.$$

Note that random variables Z_{ij} are pairwise independent. This is an important feature of random outer-product queries and the main reason to use them instead of the submatrix queries. This feature greatly simplifies the analysis. Since U is unbiased, $\mathbb{E}[U^2] = \mathrm{Var}[U]$. By pairwise independence of Z_{ij},

$$\mathrm{Var}[U] = \mathrm{Var}\Big[\sum_{i,j \in [n]} M_{ij} Z_{ij}\Big] = \sum_{i,j \in [n]} M_{ij}^2 \, \mathrm{Var}[Z_{ij}] = \sum_{i,j \in [n]} M_{ij}^2 = m.$$

Next, we give an upper bound on the 4th moment of U.

▷ Claim 3.7. $\mathbb{E}[U^4] \leq 9n^4$.

Proof. We use the definition of U, write it out as a sum, and multiply out the terms of the product:

$$\begin{aligned}\mathbb{E}[U^4] = \mathbb{E}[(A^T MB)^4] &= \mathbb{E}\Big[\Big(\sum_{i,j\in[n]} M_{ij}Z_{ij}\Big)^4\Big] \\ &= \sum_{(i_1,j_1),\ldots,(i_4,j_4)\in[d]\times[d]} M_{i_1j_1}M_{i_2j_2}M_{i_3j_3}M_{i_4j_4}\ \mathbb{E}[Z_{i_1j_1}Z_{i_2j_2}Z_{i_3j_3}Z_{i_4j_4}], \qquad (1)\end{aligned}$$

where Equation (1) is obtained by using the linearity of expectation. Next, we evaluate the expectation of the product in Equation (1):

$$\begin{aligned}\mathbb{E}[Z_{i_1j_1}Z_{i_2j_2}Z_{i_3j_3}Z_{i_4j_4}] &= \mathbb{E}[A_{i_1}B_{j_1}A_{i_2}B_{j_2}A_{i_3}B_{j_3}A_{i_4}B_{j_4}] \\ &= \mathbb{E}[A_{i_1}A_{i_2}A_{i_3}A_{i_4}]\ \mathbb{E}[B_{j_1}B_{j_2}B_{j_3}B_{j_4}],\end{aligned}$$

where the last equality follows by independence of A_i and B_j for all $i, j \in [n]$. The expression $\mathbb{E}[A_{i_1}A_{i_2}A_{i_3}A_{i_4}]$ is 0 if at least one of the indices appears only once in the tuple (i_1, i_2, i_3, i_4), since, in this case, we can use the independence of the corresponding factor A_i from the remaining factors to represent this expression as $\mathbb{E}[A_i]$ multiplied by the expectation of the product of the remaining factors. Since $\mathbb{E}[A_i] = 0$ for all $i \in [n]$, the overall expression evaluates to 0.

Note that if one of the factors appears exactly three times, then another factor appears exactly once. Therefore, the remaining case is when each factor appears an even number of times. If there are two factors, say A_i and A_j that appear twice, then the expression evaluates to $\mathbb{E}[A_i^2 A_j^2] = 1$. It also evaluates to 1 when $i = j$.

Thus, each term in Equation (1) is either 0 or 1. By symmetry, it can potentially be 1 only if each index in the tuple (i_1, i_2, i_3, i_4) and each index in the tuple (j_1, j_2, j_3, j_4) appears an even number of times. It remains to give an upper bound on the number of such terms. There are $\binom{n}{2}$ ways to choose two distinct i-indices and $\binom{4}{2} = 6$ possible positions for them in the 4-tuple. In addition, there are n ways to choose an index that appears 4 times in the 4-tuple. So, the number of possibilities for nonzero $\mathbb{E}[A_{i_1}A_{i_2}A_{i_3}A_{i_4}]$ is at most $3n^2$. The same bounds holds for $\mathbb{E}[B_{j_1}B_{j_2}B_{j_3}B_{j_4}]$. Consequently, the number of terms equal to 1 in Equation (1) is at most $9n^4$. Thus, the sum evaluates to at most $9n^4$. This completes the proof of Claim 3.7. ◁

Since U^2 is a nonnegative random variable with finite variance, the Paley-Zygmund inequality gives that, for all $\theta \in [0, 1]$,

$$\Pr\left[U^2 > \theta\ \mathbb{E}[U^2]\right] \geq (1-\theta)^2 \frac{(\mathbb{E}[U^2])^2}{\mathbb{E}[U^4]} \geq (1-\theta)^2 \frac{m^2}{9n^4} \geq (1-\theta)^2 \frac{(\gamma n^2)^2}{9n^4} = (1-\theta)^2 \frac{\gamma^2}{9},$$

where the last inequality uses the bound $m \geq \gamma n^2$ stated in the lemma. Finally, we set $\theta = 1/4$ and get:

$$\Pr\left[|A^T MB| > \frac{\sqrt{m}}{2}\right] = \Pr\left[|U| > \frac{\sqrt{m}}{2}\right] = \Pr\left[U^2 > \frac{m}{4}\right] \geq \frac{3^2}{4^2}\frac{\gamma^2}{9} = \frac{\gamma^2}{16},$$

completing the proof of Lemma 1.4. ◀

3.3 Reconstruction Attack Using Outer-Product Queries

To simplify notation in this section, we represent our datasets and outer-product queries as vectors. Formally, X here denotes the vectorization of the original sensitive dataset, i.e., a vector in $\{0,1\}^{n^2}$. For an outer-product query (A, B), we let $Q \in \{0,1\}^{n^2}$ represent the vectorization of $A \otimes B$, the outer product of A and B. (In other words, Q is the Kronecker product of A and B.) Then the answer to the query is the dot product $Q \cdot X$.

In this section, we define and analyze the attacker's algorithm $\mathcal{C}$ and complete the proof of Theorem 3.1. The attacker $\mathcal{C}$ runs algorithm $\mathcal{B}$ from Section 3.1 on the sensitive dataset X and a set of k random outer-product queries $Q_1, \ldots, Q_k$ to obtain answers $a_1, \ldots, a_k$. For all $\ell \in [k]$, we call the answer a_ℓ *accurate for a dataset* Y if $|Q_\ell \cdot Y - a_\ell| \leq \frac{\sqrt{\gamma}n}{4}$; otherwise, we call a_ℓ *inaccurate for* Y. The attacker $\mathcal{C}$ outputs any dataset $Y^* \in \{0,1\}^{n^2}$ for which at most $\frac{\gamma^2 k}{64}$ answers among $a_1, \ldots, a_k$ are inaccurate for Y^*. By Lemma 3.3, the probability that X satisfies this requirement is at least $\frac{5}{6}$. If this event occurs, algorithm $\mathcal{C}$ will be able to output some Y^*. (Otherwise, the attack fails.)

Next, we analyze the attack. Let $\|X - Y\|_1$ denote the Hamming distance between datasets X and Y. Call a dataset Y *bad* if $\|X - Y\|_1 > \gamma n^2$, i.e., if it differs from X on more than γn^2 entries. We will show that $\mathcal{C}$ is unlikely to choose a bad data set as Y^*. Fix a bad dataset Y. Let $M = X - Y$, and observe that M has $m > \gamma n^2$ nonzero entries. We say that a set of queries $\{Q_1, \ldots, Q_k\}$ *catches* the dataset Y if more than $\frac{\gamma^2 k}{32}$ entries in $(|Q_1 \cdot M|, \ldots, |Q_k \cdot M|)$ exceed $\frac{\sqrt{\gamma}n}{2}$.

▶ **Lemma 3.8.** *Suppose the attacker $\mathcal{C}$ makes $k = \frac{128n^2}{\gamma^2}$ uniformly random outer-product queries. Then the probability that there exists a bad dataset not caught by the attacker's set of queries is at most $\frac{1}{6}$.*

Proof. Consider a set of k uniformly random outer-product queries $\{Q_\ell\}_{\ell \in [k]}$. Fix a bad dataset Y. Then $\|X - Y\|_1 > \gamma n^2$. Let $M = X - Y$.

For every $\ell \in [k]$, let $\chi_\ell = 1$ if $|Q_\ell \cdot M| > \frac{\sqrt{\gamma}n}{2}$, and otherwise let $\chi_\ell = 0$. Also, let $\chi = \sum_{\ell=1}^{k} \chi_\ell$. By definition, the difference vector $M = X - Y$ has more than γn^2 nonzero entries. By the anti-concentration bound in Lemma 1.4, $\Pr\left[|Q_\ell \cdot M| > \frac{\sqrt{\gamma}n}{2}\right] \geq \frac{\gamma^2}{16}$. Therefore, $\mathbb{E}[\chi_\ell] \geq \frac{\gamma^2}{16}$. By the Chernoff bound, we have that for $k = \frac{128n^2}{\gamma^2}$ and for $n \geq 3$,

$$\Pr\left[\chi \leq \frac{\gamma^2 \cdot k}{32}\right] \leq \exp\left(-\frac{\gamma^2 k}{128}\right) = \exp\left(-n^2\right) < \frac{1}{6 \cdot 2^{n^2}}.$$

Hence, the set $\{Q_\ell\}_{\ell \in [k]}$ fails to catch each specific bad dataset with probability at most $\frac{1}{6 \cdot 2^{n^2}}$. By a union bound over at most 2^{n^2} bad datasets, the probability that there exists a bad dataset not caught by the attacker's queries is at most $1/6$. ◀

▶ **Lemma 3.9** (Reconstruction Lemma)**.** *If algorithm $\mathcal{B}$ has additive error at most $\frac{\sqrt{\gamma}n}{4}$ on all but at most $\frac{\gamma^2 k}{64}$ answers, and the set of queries it uses catches all bad datasets Y, then the reconstruction attack is successful, that is, the attacker $\mathcal{C}$ outputs Y^* that differs from X on at most γn^2 entries, i.e., $\|X - Y^*\|_1 \leq \gamma n^2$.*

Proof. By the first premise of the lemma, the dataset X "disagrees" with at most $\frac{\gamma^2 k}{64}$ of the answers a_ℓ. Hence, necessarily, the attacker $\mathcal{C}$ outputs some dataset Y^*. Assume towards a contradiction that Y^* is a bad dataset. Let $\{Q_\ell\}_{\ell \in [k]}$ be the set of queries chosen by $\mathcal{B}$. Let $M^* = X - Y^*$ be the difference vector. By the triangle inequality, $|Q_\ell M^*| = |Q_\ell X - Q_\ell Y^*| \leq |Q_\ell X - a_\ell| + |Q_\ell Y^* - a_\ell|$. From the first assumption in the

lemma, $|Q_\ell X - a_\ell| \le \frac{\sqrt{\gamma}n}{4}$ for all but at most $\frac{\gamma^2 k}{64}$ of the queries. By the description of the attack $\mathcal{C}$, the output Y^* is such that for all but at most $\frac{\gamma^2 k}{64}$ of the queries, $|Q_\ell Y^* - a_\ell| \le \frac{\sqrt{\gamma}n}{4}$. Therefore, for all but at most $\frac{\gamma^2 k}{32}$ of the queries, $|Q_\ell M^*| \le |Q_\ell X - a_\ell| + |Q_\ell Y^* - a_\ell| \le \frac{\sqrt{\gamma}n}{2}$. Since $\{Q_\ell\}_{\ell\in[k]}$ catches all bad datasets, it in particular catches Y^*, because Y^* is bad. By definition of catching, $|Q_\ell M^*| > \frac{\sqrt{\gamma}n}{2}$ for more than $\frac{\gamma^2 k}{32}$ of the values $Q_\ell M^*$. Hence, we have reached a contradiction, implying that Y^* is a good dataset. ◀

The final ingredient for proving Theorem 3.1 is the following lemma, which is based on an argument of [15]. Any algorithm that outputs a large fraction of its secret dataset is definitely not private, for any reasonable notion of privacy. Lemma 3.10 states that such an algorithm is not differentially private.

▶ **Lemma 3.10.** *Let $\mathcal{C}$ be an algorithm that takes as input a secret data set X in $\{0,1\}^N$ and outputs a vector in the same set, $\{0,1\}^N$. If $\mathcal{C}$ is (ε,δ)-differentially private and X is uniformly distributed in $\{0,1\}^N$, then*

$$\mathbb{E}\left[\|\mathcal{C}(X) - X\|_1\right] \ge e^{-\varepsilon}\left(\tfrac{1}{2} - \delta\right)N\,.$$

Lemma 3.10 above only bounds the expectation of $\|\mathcal{C}(X) - X\|_1$. The more sophisticated argument in [15] yields much tighter concentration results. We use the simpler version here since it allows for a self-contained presentation.

Proof. Fix an index $i \in [N]$ and a bit $r \in \{0,1\}$. Let $X_{i\to r}$ denote the vector obtained by replacing the i-th entry of X with the bit r.

Consider the pair of random variables $(X, \mathcal{C}(X))$. Because $\mathcal{C}$ is (ε,δ)-differentially private, this is distributed similarly to the pair $(X_{i\to R}, \mathcal{C}(X))$, where R is a uniformly random bit independent of the other values. Specifically, for any event $E \subseteq \{0,1\}^N \times \{0,1\}^N$,

$$\Pr[(X_{i\to R}, \mathcal{C}(X)) \in E] \le e^{\varepsilon} \Pr[(X, \mathcal{C}(X)) \in E] + \delta.$$

Applying this inequality to the event $E_i = \{(x,y) : x_i \ne y_i\}$ shows that

$$\tfrac{1}{2} = \Pr[\mathcal{C}(X)_i \ne R] \le e^{\varepsilon} \Pr[\mathcal{C}(X)_i \ne X_i] + \delta \quad \text{and thus} \quad \Pr[\mathcal{C}(X)_i \ne X_i] \ge e^{-\varepsilon}(\tfrac{1}{2} - \delta)\,.$$

The Hamming distance $\|\mathcal{C}(X) - X\|_1$ is the sum of the indicator random variables for the events $\mathcal{C}(X)_i \ne X_i$. By linearity of expectation, the expected Hamming distance is at least $e^{-\varepsilon}\left(\frac{1}{2} - \delta\right)N$. ◀

Finally, we use Lemmas 3.3 and 3.8–3.10 to compete the proof of the main theorem.

Proof of Theorem 3.1. We set $\gamma = \frac{1}{9}$. Assume towards a contradiction that for some ε and δ as in the statement of the theorem, there exists an (ε,δ)-LEDP algorithm $\mathcal{A}$ that for every $3n$-node graph approximates the number of triangles in the graph up to additive error $\alpha = \frac{\sqrt{\gamma}n^2}{20}$ with probability at least $1 - \frac{\gamma^2}{9\cdot 128} = 1 - \frac{1}{3^6\cdot 2^7}$. Then by Lemma 3.3, there exists a $(2\varepsilon, 2\delta)$-DP algorithm $\mathcal{B}$ that, for every secret dataset X and every set of k outer-product queries, answers inaccurately (i.e., with additive error more than $\frac{\sqrt{\gamma}n}{4}$) on at most $\frac{\gamma^2 k}{64}$ of the k queries with probability at least $\frac{5}{6}$. By Lemma 3.8, the probability that a set of $k = \frac{128n^2}{\gamma^2}$ random outer-product queries chosen by the attacker $\mathcal{C}$ does not catch all bad datasets is at most $\frac{1}{6}$. By a union bound, with probability at least $\frac{2}{3}$, the attacker $\mathcal{C}$ satisfies the premise of Lemma 3.9 and the set of chosen queries catches all bad data sets. Hence, with probability at least $\frac{2}{3}$, the attacker $\mathcal{C}$ outputs a dataset Y^* which coincides with X on at least $(1-\gamma)n^2$ entries. The expected Hamming distance $\mathbb{E}[\|X - Y^*\|_1]$ is therefore at most $\frac{2}{3}\cdot\gamma n^2 + \frac{1}{3}n^2 = \frac{1+2\gamma}{3}n^2$. When $\gamma = \frac{1}{9}$, the expected distance is less than $0.41n^2$.

Recall that the attacker $\mathcal{C}$ runs $(2\varepsilon, 2\delta)$-DP algorithm $\mathcal{B}$ on a secret dataset X and then post processes the output of $\mathcal{B}$. Thus, $\mathcal{C}$ is $(2\varepsilon, 2\delta)$-DP, and we can apply Lemma 3.10 to conclude that the expected Hamming distance $\mathbb{E}\|\mathcal{C}(X) - X\|_1$ is at most $e^{-2\varepsilon}(\frac{1}{2} - 2\delta)n^2$. Since, by assumption, $\varepsilon \leq 1/20$ and $\delta \leq 1/100$, we have $\mathbb{E}\|\mathcal{C}(X) - X\|_1 \geq 0.43n^2$. This contradicts the upper bound of $0.41n^2$ above. ◀

4 The Interactive Lower Bound

In this section, we present an $\Omega\left(\frac{n^{3/2}}{\varepsilon}\right)$ lower bound on the additive error of every ε-LEDP algorithm for estimating the number of triangles in a graph, stated formally in Theorem 1.3. We reduce from the problem of computing the summation in the LDP model.

▶ **Definition 4.1** (Summation function). *Let SUM_n be the following function. For all $x_1, \ldots, x_n \in \{0,1\}$, $SUM_n(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i$.*

This problem was shown to have an additive error lower bound of $\Omega(\sqrt{n}/\varepsilon)$ [36, Theorem 5.3 of arxiv v2]. We substitute $\alpha = \alpha_0/n$ and $\beta = \varepsilon\alpha_0/n$ to obtain the following lemma.

▶ **Lemma 4.2** ([10, 6, 36]). *There exists a constant $c > 0$ such that for every $\varepsilon \in (0,1)$, $n \in \mathbb{N}, \alpha_0 \in (0,n]$ and $\delta \in \left[0, \frac{1}{10^5} \cdot \frac{\varepsilon^3\alpha_0^2}{n^3 \ln(n^2/\varepsilon\alpha_0)}\right]$, if $\mathcal{B}$ is an (ε,δ)-LDP algorithm where each party i receives input $x_i \in \{0,1\}$ and $\mathcal{B}$ estimates SUM_n up to additive error α_0 with probability at least $2/3$, then $\alpha_0 \geq c \cdot \sqrt{n}/\varepsilon$.*

Proof of Theorem 1.3. We reduce from SUM_n in the local model. By Lemma 4.2, every (potentially interactive) algorithm that approximates SUM_n with additive error at most α_0 (with sufficiently high constant probability) must have $\alpha_0 = \Omega(\frac{\sqrt{n}}{\varepsilon})$. In our reduction, we will set the additive error of the triangle-counting algorithm, $\alpha = \alpha_0 n$.

Our reduction is black-box. Given an instance of SUM_n, where each local party holds one bit X_i of the vector $(X_1, \ldots, X_n)$, the parties implicitly create the following graph G. The vertex set consists of two sets of nodes, V_1 and V_2, where V_1 has size n and V_2 has size $2n$. The nodes in V_1 will not have any secret information and can be simulated by any local party. The nodes in V_2 are $[2n]$, and each party $i \in [n]$ is responsible for simulating nodes $2i-1$ and $2i$ in V_2. To create the edges of G, we add edges of the complete bipartite graph between V_1 and V_2. In addition, each pair of nodes $(2i-1, 2i)$ in V_2 has an edge between them if and only if $x_i = 1$. See Figure 3 for an illustration.

Let $S = x_1 + \ldots x_n$. Observe that any triangle in G must have two vertices in V_2 and an edge between a pair of matched nodes. Any such edge contributes exactly n triangles. So, the total number of triangles in G is $T = Sn$.

For the sake of contradiction, suppose there is an (ε,δ)-LEDP algorithm $\mathcal{A}$ that estimates the number of edges with error $o(\frac{n^{3/2}}{\varepsilon})$. We run it on G. By construction, party i can simulate the two nodes assigned to it, and anybody can simulate nodes in V_1. When the algorithm gets an estimate $\widehat{T}$ for the number of triangles, it outputs $\hat{S} = \widehat{T}/n$. If $\widehat{T} = T \pm o(\frac{n^{3/2}}{\varepsilon})$, then $\hat{S} = \widehat{T}/n = T/n \pm o(\frac{\sqrt{n}}{\varepsilon}) = S \pm o(\frac{\sqrt{n}}{\varepsilon})$, which is a contradiction to Lemma 4.2.

Moreover, if $\mathcal{A}$ is (ε,δ)-LEDP, then the reduction algorithm is (ε,δ)-LDP with respect to the secret dataset X. However, the latter contradicts Lemma 4.2. Thus, $\mathcal{A}$ cannot exist. ◀

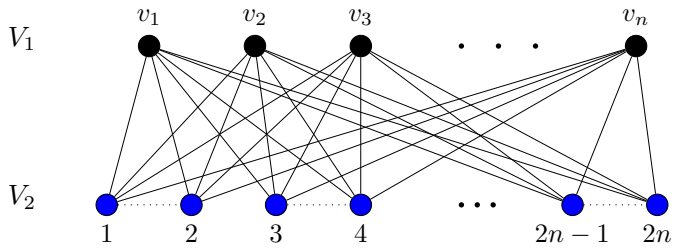


Figure 3 An instance of the interactive $\Omega(n^{3/2})$ lower bound consists of a complete bipartite graph with parts V_1, V_2 of sizes n and $2n$, respectively; in addition, there is an edge between each pair $\{2i-1, 2i\}$ iff the secret input bit $X_i = 1$.

References

1 Mohammad Al Hasan and Vachik S Dave. Triangle counting in large networks: a review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(2):e1226, 2018.

2 Differential Privacy Team Apple. Learning with privacy at scale differential, 2017.

3 Sepehr Assadi, Michael Kapralov, and Sanjeev Khanna. A simple sublinear-time algorithm for counting arbitrary subgraphs via edge sampling. In *ITCS*, volume 124 of *LIPIcs*, pages 6:1–6:20. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2019.

4 Prakash Balachandran, Eric D. Kolaczyk, and Weston D. Viles. On the propagation of low-rate measurement error to subgraph counts in large networks. *JMLR*, 18(61):1–33, 2017.

5 Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *Proceedings of the 14th ACM SIGKDD*, pages 16–24, 2008.

6 Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In *CRYPTO*, pages 451–468. Springer, 2008.

7 Amartya Shankha Biswas, Talya Eden, Quanquan C. Liu, Ronitt Rubinfeld, and Slobodan Mitrovic. Massively parallel algorithms for small subgraph counting. In Amit Chakrabarti and Chaitanya Swamy, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022, September 19-21, 2022, University of Illinois, Urbana-Champaign, USA (Virtual Conference)*, volume 245 of *LIPIcs*, pages 39:1–39:28. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.APPROX/RANDOM.2022.39.

8 Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th SOSP*, SOSP '17, pages 441–459, New York, NY, USA, 2017.

9 Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In Robert D. Kleinberg, editor, *ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pages 87–96. ACM, 2013. doi:10.1145/2422436.2422449.

10 TH Hubert Chan, Elaine Shi, and Dawn Song. Optimal lower bound for differentially private multi-party aggregation. In *ESA*, pages 277–288. Springer, 2012.

11 Jinyuan Chang, Eric D. Kolaczyk, and Qiwei Yao. Estimation of subgraph densities in noisy networks. *Journal of the American Statistical Association*, 117(537):361–374, 2022.

12 Justin Y Chen, Talya Eden, Piotr Indyk, Honghao Lin, Shyam Narayanan, Ronitt Rubinfeld, Sandeep Silwal, Tal Wagner, David Woodruff, and Michael Zhang. Triangle and four cycle counting with predictions in graph streams. In *ICLR*, 2021.

13 Shixi Chen and Shuigeng Zhou. Recursive mechanism: towards node differential privacy and unrestricted joins. In Kenneth A. Ross, Divesh Srivastava, and Dimitris Papadias, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pages 653–664. ACM, 2013. doi:10.1145/2463676.2465304.

14 Graham Cormode, Somesh Jha, Tejas Kulkarni, Ninghui Li, Divesh Srivastava, and Tianhao Wang. Privacy at scale: Local differential privacy in practice. In *Proceedings of the 2018 SIGMOD*, SIGMOD '18, pages 1655–1658, New York, NY, USA, 2018.

15 Anindya De. Lower bounds in differential privacy. In *TCC*, pages 321–338. Springer, 2012.

16 Laxman Dhulipala, Quanquan C. Liu, Sofya Raskhodnikova, Jessica Shi, Julian Shun, and Shangdi Yu. Differential privacy from locally adjustable graph algorithms: k-core decomposition, low out-degree ordering, and densest subgraphs. In *FOCS 2022, Denver, CO, USA, October 31 – November 3, 2022*, pages 754–765. IEEE, 2022. doi:10.1109/FOCS54457.2022.00077.

17 Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 3574–3583, Red Hook, NY, USA, 2017. Curran Associates Inc.

18 Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In Frank Neven, Catriel Beeri, and Tova Milo, editors, *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART PODS, June 9-12, 2003, San Diego, CA, USA*, pages 202–210. ACM, 2003. doi:10.1145/773153.773173.

19 John Duchi, Michael Jordan, and Martin Wainwright. Local privacy and statistical minimax rates. In *IEEE Symposium on Foundations of Computer Science*, FOCS '13, pages 429–438, Berkeley, CA, USA, 2013. arXiv:1302.3203.

20 John C Duchi, Michael I Jordan, Martin J Wainwright, et al. Minimax optimal procedures for locally private estimation. *Journal of the American Statistical Association*, 113(521):182–201, 2018.

21 Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In Serge Vaudenay, editor, *EUROCRYPT 2006, St. Petersburg, Russia, May 28 – June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 486–503. Springer, 2006. doi:10.1007/11761679_29.

22 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, pages 265–284, Berlin, Heidelberg, 2006. Springer-Verlag.

23 Cynthia Dwork, Frank McSherry, and Kunal Talwar. The price of privacy and the limits of LP decoding. In *Proceedings of the Thirty-Ninth ACM STOC*, pages 85–94. ACM, 2007.

24 Cynthia Dwork and Sergey Yekhanin. New efficient attacks on statistical disclosure control mechanisms. In *CRYPTO*, pages 469–480. Springer, 2008.

25 Jean-Pierre Eckmann and Elisha Moses. Curvature of co-links uncovers hidden thematic layers in the world wide web. *Proceedings of the national academy of sciences*, 99(9):5825–5829, 2002.

26 Talya Eden, Amit Levi, Dana Ron, and C Seshadhri. Approximately counting triangles in sublinear time. *SIAM Journal on Computing*, 46(5):1603–1646, 2017.

27 Talya Eden, Quanquan C. Liu, Sofya Raskhodnikova, and Adam D. Smith. Triangle counting with local edge differential privacy. *CoRR*, abs/2305.02263, 2023. doi:10.48550/arXiv.2305.02263.

28 Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC CCS*, CCS '14, pages 1054–1067, New York, NY, USA, 2014.

29 Alexandre V. Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In Frank Neven, Catriel Beeri, and Tova Milo, editors, *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART PODS, June 9-12, 2003, San Diego, CA, USA*, pages 211–222. ACM, 2003. doi:10.1145/773153.773174.

30 Illés J Farkas, Imre Derényi, Albert-László Barabási, and Tamas Vicsek. Spectra of "real-world" graphs: Beyond the semicircle law. *Physical Review E*, 64(2):026704, 2001.

31 Tianchong Gao, Feng Li, Yu Chen, and XuKai Zou. Local differential privately anonymizing online social networks under hrg-based model. *IEEE Transactions on Computational Social Systems*, 5(4):1009–1020, 2018.

32 Anupam Gupta, Aaron Roth, and Jonathan R. Ullman. Iterative constructions and private data release. In Ronald Cramer, editor, *Theory of Cryptography – 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, volume 7194 of *Lecture Notes in Computer Science*, pages 339–356. Springer, 2012. doi:10.1007/978-3-642-28914-9_19.

33 Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. Locally differentially private analysis of graph statistics. In *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 983–1000, 2021. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/imola.

34 Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. Communication-efficient triangle counting under local differential privacy. In *31st USENIX Security Symposium, USENIX Security 2022, August 10-12, 2022*, 2022. arXiv:2110.06485.

35 Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. Differentially private subgraph counting in the shuffle model. *CoRR*, abs/2205.01429, 2022. arXiv:2205.01429, doi:10.48550/arXiv.2205.01429.

36 Matthew Joseph, Jieming Mao, Seth Neel, and Aaron Roth. The role of interactivity in local differential privacy. In David Zuckerman, editor, *60th IEEE Annual FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 94–105. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00015.

37 John Kallaugher and Eric Price. A hybrid sampling scheme for triangle counting. In *Proceedings of the Annual ACM-SIAM SODA*, pages 1778–1797, 2017.

38 Vishesh Karwa, Sofya Raskhodnikova, Adam D. Smith, and Grigory Yaroslavtsev. Private analysis of graph structure. *ACM Trans. Database Syst.*, 39(3):22:1–22:33, 2014. doi:10.1145/2611523.

39 Vishesh Karwa, Aleksandra B. Slavković, and Pavel Krivitsky. Differentially private exponential random graphs. In *Privacy in Statistical Databases*, pages 143–155. Springer International Publishing, 2014.

40 Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

41 Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. Analyzing graphs with node differential privacy. In Amit Sahai, editor, *10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, volume 7785 of *Lecture Notes in Computer Science*, pages 457–476. Springer, 2013. doi:10.1007/978-3-642-36594-2_26.

42 Shiva Prasad Kasiviswanathan, Mark Rudelson, Adam Smith, and Jonathan Ullman. The price of privately releasing contingency tables and the spectra of random matrices with correlated rows. In *Proceedings of the 42nd ACM STOC*, STOC '10, pages 775–784. ACM, 2010.

43 Shiva Prasad Kasiviswanathan, Mark Rudelson, and Adam D. Smith. The power of linear reconstruction attacks. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1415–1433. SIAM, 2013. doi:10.1137/1.9781611973105.102.

44 Andrew McGregor and Sofya Vorotnikova. Triangle and four cycle counting in the data stream model. In *ACM SIGMOD-SIGACT-SIGART PODS*, pages 445–456, 2020.

45 Andrew McGregor, Sofya Vorotnikova, and Hoa T Vu. Better algorithms for counting triangles in data streams. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI PODS*, pages 401–411, 2016.

46 Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

47 Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. Smooth sensitivity and sampling in private data analysis. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 75–84. ACM, 2007. Full paper: `http://www.cse.psu.edu/~asmith/pubs/NRS07`. `doi:10.1145/1250790.1250803`.

48 Rasmus Pagh and Charalampos E Tsourakakis. Colorful triangle counting and a mapreduce implementation. *Information Processing Letters*, 112(7):277–281, 2012.

49 Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

50 Ha-Myung Park, Francesco Silvestri, U Kang, and Rasmus Pagh. Mapreduce triangle enumeration with guarantees. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1739–1748, 2014.

51 Arnau Prat-Pérez, David Dominguez-Sal, Josep M. Brunat, and Josep Lluís Larriba-Pey. Put three and three together: Triangle-driven community detection. *ACM Trans. Knowl. Discov. Data*, 10(3):22:1–22:42, 2016. `doi:10.1145/2775108`.

52 Zhan Qin, Ting Yu, Yin Yang, Issa Khalil, Xiaokui Xiao, and Kui Ren. Generating synthetic decentralized social graphs with local differential privacy. In *Proceedings of the 2017 ACM SIGSAC CCS*, CCS '17, pages 425–438, New York, NY, USA, 2017.

53 Sofya Raskhodnikova and Adam D. Smith. Differentially private analysis of graphs. In *Encyclopedia of Algorithms*, pages 543–547. Springer, 2016. `doi:10.1007/978-1-4939-2864-4_549`.

54 Haipei Sun, Xiaokui Xiao, Issa Khalil, Yin Yang, Zhan Qin, Hui (Wendy) Wang, and Ting Yu. Analyzing subgraph statistics from extended local views with decentralized differential privacy. In *Proceedings of the 2019 ACM CCS*, CCS '19, pages 703–717, New York, NY, USA, 2019.

55 Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

56 Qingqing Ye, Haibo Hu, Man Ho Au, Xiaofeng Meng, and Xiaokui Xiao. Towards locally differentially private generic graph metric estimation. In *2020 IEEE 36th ICDE*, pages 1922–1925, 2020.

Protecting Single-Hop Radio Networks from Message Drops

Klim Efremenko
Ben-Gurion University, Beer Sheva, Israel

Gillat Kol
Princeton University, NJ, USA

Dmitry Paramonov
Princeton University, NJ, USA

Raghuvansh R. Saxena
Microsoft Research, Cambridge, MA, USA

Abstract

Single-hop radio networks (SHRN) are a well studied abstraction of communication over a *wireless* channel. In this model, in every round, each of the n participating parties may decide to *broadcast* a message to all the others, potentially causing collisions. We consider the SHRN model in the presence of *stochastic message drops* (i.e., *erasures*), where in every round, the message received by each party is *erased* (replaced by $\perp$) with some small constant probability, independently.

Our main result is a *constant rate coding scheme*, allowing one to run protocols designed to work over the (noiseless) SHRN model over the SHRN model with erasures. Our scheme converts any protocol Π of length at most exponential in n over the SHRN model to a protocol Π' that is resilient to constant fraction of erasures and has length linear in the length of Π.

We mention that for the special case where the protocol Π is *non-adaptive*, i.e., the order of communication is fixed in advance, such a scheme was known. Nevertheless, adaptivity is widely used and is known to hugely boost the power of wireless channels, which makes handling the general case of adaptive protocols Π both important and more challenging. Indeed, to the best of our knowledge, our result is the first constant rate scheme that converts adaptive protocols to noise resilient ones in any multi-party model.

2012 ACM Subject Classification Theory of computation → Communication complexity

Keywords and phrases Radio Networks, Interactive Coding, Error Correcting Codes

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.53

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2023/066/`

Funding *Klim Efremenko*: Supported by the Israel Science Foundation (ISF) through grant No. 1456/18 and European Research Council Grant number: 949707.
Gillat Kol: supported by a National Science Foundation CAREER award CCF-1750443 and by a BSF grant No. 2018325.

1 Introduction

Over the last decades, wireless communication found many applications and has transformed technology. On the theoretical side, wireless systems were studied by numerous works, many of which consider the *single-hop radio networks* (SHRN) model of Chlamtac and Kutten [7], which abstracts a simple broadcast channel.

The classical model of SHRN assumes that the communication is *noiseless*, guaranteeing that (if no "collisions" occur) the message broadcast in a round will be received correctly by all the parties. In contrast, recently, Censor-Hillel, Haeupler, Hershkowitz, and Zuzic [6],

initiated the study of the radio networks model under *stochastic message drops* (*a.k.a*, *stochastic erasures*). In their model, each party only gets the message that was broadcast with probability $1-\epsilon$, independently, for some small constant ϵ. Otherwise, the round is "erased" for this party, meaning that it is received as a silent round, as if nothing was broadcast.

While the (noiseless) radio networks model is, by now, mostly well understood, and while noise is inherent in almost all communication systems, the relative power of noisy radio networks is far less explored. In this work we study the power of the SHRN model under the message drop noise of [6].

1.1 Our Result

Our main result is that the model of SHRN with message drops is as powerful as that of (noiseless) SHRN, in the sense that any protocol that was designed to work over the latter can be made to work over the former with a small overhead to the communication. An informal statement of our main result is in Theorem 1 (see Theorem 2 for a formal statement, the assumed model is discussed next).

▶ **Theorem 1.** *Let $n \in \mathbb{N}$ be the number of participants, $\epsilon \in (0,1)$ be the noise rate, and Γ be a non-empty alphabet set. For any protocol Π of length $T \leq 2^n$ over the (n,Γ)-broadcast channel, there is a protocol Π' with $\mathcal{O}(T)$ rounds over the (n,ϵ,Γ)-noisy broadcast channel that simulates[1] Π, and errs with probability polynomially small in T.*

We mention that our scheme works for protocols of length $T \leq 2^n$, as, if T is much larger than 2^n, there will be rounds where the messages received by all parties are erased (see Section 2.4). We also mention that our scheme uses a combinatorial building block called a *tree code* (see Section 2.2), and like other works that use tree codes, it is not computationally efficient, as no efficient tree code construction is known. Whether or not longer protocols can be handled with constant rate, and whether computationally efficient schemes are possible, are two intriguing questions we leave open.

The *collision-as-silence-as-erasures* SHRN model

We next overview the noise model of [6] used by Theorem 1 (for formal definitions, see Section 3): A protocol over the (n,ϵ,Γ)-*noisy broadcast channel* is a communication protocol between n communicating parties that proceeds in synchronous rounds. In each round, each party can decide to either broadcast a symbol from Γ or stay silent. If more than one party broadcasts in a given round (a *collision*), or none of the parties broadcast (a *silent round*), then the "$\perp$" symbol is received by all the parties[2]. Otherwise, exactly one of the parties broadcasts a symbol, and each party receives the broadcast symbol with probability $1-\epsilon$, and $\perp$ with probability ϵ, independently[3]. A protocol over the (n,Γ)-*broadcast channel* is a protocol over the $(n,0,\Gamma)$-noisy broadcast channel, i.e., one where erasures do not occur.

---

[1] By "Π' that simulates Π", we mean that a transcript for Π can be retrieved from a transcript for Π', see Theorem 2.

[2] The name *collision-as-silence* is because the same $\perp$ symbol is received in both collision and silent rounds. This model is, perhaps, the most common model in the literature. Another very popular model is the *collision detection* model, where collision and silence are perceived as different symbols. Theorem 1 is stated for the collision-as-silence model, but applies to the collision detection model as well.

[3] Modeling erasures as the same symbol as collisions/silences only makes our result stronger. As explained in Section 2.3, this makes our erasure model closer to the corruption model.

1.2 Corruption Noise and Adaptivity

The corruption noise model

One of the original motivations for our work was exploring the power of the SHRN model under stochastic *corruption noise*, a noise model that received quite a bit of attention over the last few years (see, e.g., [10, 11]). In this model, in every round, each party receives the correct symbol output by the channel with probability $1 - \epsilon$, and receives one of the other symbols with probability ϵ, independently[4]. Observe that protecting protocols against corruptions is at least as hard as protecting them against message drops.

Adaptivity and the [10] scheme

An encouraging piece of evidence, indicating that it may be possible to make SHRN protocols resilient to corruption noise with small overhead, was recently given by Efremenko, Kol, and Saxena [10], who designed such a scheme for a restricted set of protocols called *non-adaptive* protocols. Still, our initial belief was that such a scheme is impossible in the general case of *adaptive* protocols.

Non-adaptive (*a.k.a*, *oblivious* or *static*) protocols are a restricted set of protocols where it is known ahead of time which party broadcasts in what round, while adaptive protocols allow the parties to decide whether or not they wish to broadcast at a given round based on their input and their received transcript up until the current round.

While non-adaptive protocols are useful, they do not fully utilize the power of the wireless channel, and communication-efficient protocols for some central problems are, in fact, adaptive (e.g., the celebrated Decay protocol for computing the size of a network [3]). This additional power of adaptive protocols is what makes their conversion to noise-resilient ones more challenging, and, indeed, the [10] scheme may fail when applied to adaptive protocols Π.

When starting this project, we identified two inherent reasons (see Section 2.1) for the failure of [10] when applied to adaptive protocols and hoped to show that these must lead to a blowup of $\tilde{\Omega}(\log n)$ in the communication. As most interactive coding lower bounds for multi-party protocols also extend to the message drop model (e.g., [4, 11]), as a first step, we attempted to convince ourselves that no constant rate simulation scheme exists even for the SHRN model with message drop noise.

To our surprise, we were able to overcome both problems in the message drop model and design a scheme that also works for adaptive protocols. As far as we know, the scheme converting noiseless to noise-resilient protocols we construct in our proof of Theorem 1 is the first constant overhead scheme that handles adaptive protocols in any multi-party setting.

We are still very interested in the more general question of making SHRN protocols resilient to corruption noise, as we believe it is a basic and "clean" coding question. Our result can be interpreted as saying that (at least for protocols that are not extremely long) either a high-rate scheme is possible or a novel lower bound approach is required.

1.3 Related Work

Interactive coding. *Interactive error correcting codes* encode interactive communication protocols designed to work over noiseless channels to protocols that also work over noisy channels. The study of interactive codes was initiated by a seminal paper of Schulman [25]

[4] Care needs to be taken while defining an error model for corruptions, as some definitions may allow for signaling-based protocols [20].

that considered two-party protocols, which was also the topic of many follow-up works. Interactive codes for multi-party distributed channels received quite a bit of attention over the last few years. These include codes for *peer-to-peer* channels [24, 21, 20, 1, 4, 16, 17] and codes for various *wireless* channels [5, 10, 6, 11, 12, 2, 8].

Coding for wireless systems. The models of wireless communication considered in the context of noise-resilience differ on a few axes. The first axis is the *adaptivity* of the simulation protocol: in some papers the target simulation protocol is allowed to be adaptive and in others it must be non-adaptive. (Of course, if the noiseless protocols considered are adaptive, the simulation needs to be adaptive. However, simulations of non-adaptive noiseless protocols by adaptive noise-resilient protocols have been considered). The second axis is whether *single or multi-hop* networks are considered. Finally, the last axis is whether the noise is modeled as stochastic *erasures* (message drops) or stochastic *corruptions* (change of symbols).

Non-adaptive simulations. The study of noise in wireless systems can be traced back to [14] that answered an open problem of [15] by giving an $\mathcal{O}(n \log \log n)$ length communication protocol for the *bit exchange* problem (all n parties have an input bit and all parties want to know the input of all the other parties). The underlying model was the *noisy broadcast channel*, which is a non-adaptive, single-hop model with corruption errors. A matching lower bound for this problem was later given by [18]. The communication complexity of other specific n-bit functions, like the *OR*, *majority*, and *parity* functions, were studied under related models by [27, 22, 13, 23, 18]. The non-adaptive single-hop model was studied under erasure noise by [19], where an $\mathcal{O}(n \log^* n)$ protocol is given for the bit exchange problem, breaking the $\Omega(n \log \log n)$ lower bound proved for corruption errors. The general case of simulating *any* non-adaptive protocol by an noise resilient non-adaptive protocol was very recently studied by [9]. Their main result is that, for protocols of length polynomial in n, such a simulation requires $\tilde{\Theta}(\sqrt{\log n})$ multiplicative overhead in the communication complexity.

Adaptive simulations. The work of [10] gave a scheme for converting any non-adaptive noiseless protocol to an adaptive noise-resilient one with only a constant multiplicative overhead, over a single-hop network with corruption errors (in particular, implying an adaptive noise-resilient bit exchange protocol with $\mathcal{O}(n)$ communication).

Multi-hop radio networks. The work of [10] (and our current work) consider the setting where the parties are connected in a *clique* (a *single-hop* network), as it is assumed that when a party transmits, all other parties can hear the transmission. As mentioned above, this topology is the single most extensively studied, as it represents the simplest broadcast channel. However, wireless systems can have arbitrary topologies.

In contrast to [10], in [11] it is shown that such a scheme is impossible over general multi-hop networks, where each of the n communicating parties is associated with a node in the graph, and when a party broadcasts, its message is only received by its neighbors in the graph (if there are no collisions). Specifically, [11] shows that in some networks, the cost of noise-resilience is $\Omega(\log n)$, even for simulating non-adaptive protocols by adaptive protocols. A matching $\mathcal{O}(\log n)$-overhead scheme for converting any noiseless protocol to a noise resilient one over any network is also given by [11].

The recent work of [6], considered general radio networks under message drop noise. They show that any protocol over any network can be converted to a noise resilient one with a multiplicative $\mathcal{O}(\Delta \log^2 \Delta)$ overhead to the communication, where Δ is the maximum degree of a node in the network. For the special case in which the noiseless protocol we wish to convert is non-adaptive, a scheme with an improved overhead of

$\text{poly}(\log\Delta, \log\log n)$ is shown [6]. For networks with small Δ, this implies an efficient simulation of noiseless protocols. However, for networks with large Δ, the [6] simulation can have a huge overhead. This is not for no reason, as the $\Omega(\log n)$ lower bound of [11] mentioned above also applies to the message drop noise and implies that there exist network topologies with large Δ for which an $\Omega(\log\Delta)$ overhead is necessary. Our result shows for the important single-hop topology, these communication overheads can be avoided altogether.

2 Proof Sketch

In this section, we give a detailed sketch of our protocol.

As mentioned in Section 1.2, one of the main motivations for our work was studying the rate of interactive codes over the SHRN model with corruptions. The restricted case where the protocol Π to be simulated is non-adaptive was studied by [10], but their scheme fails for adaptive protocols. We next explain the inherent reasons for this failure and then outline our solutions for erasure noise.

2.1 The [10] Scheme

The rewind-if-error framework

The [10] scheme utilizes the *rewind-if-error* framework, which was initially designed for the two-party setting [25]. Rewind-if-error coding schemes consist of many iterations, where each iteration consists of two phases: a *simulation phase*, where a small number of rounds of the noiseless protocol Π are executed, and a *consistency check phase* where the parties attempt to check if they have the same received transcript or whether an error occurred (e.g., by comparing hashes of their received transcripts). If the check phase passes, parties continue the simulation, otherwise they *rewind* and re-simulate the last few rounds.

A careful examination of the [10] scheme shows that it breaks down when applied to adaptive protocols for the following two fundamental reasons:

Repeated rewinds. The first problem is that with noise rate ϵ, we should expect about ϵn parties to experience message drops in every round of the simulation phase. Since ϵ is constant, $\epsilon n \gg 1$. This implies that the consistency check phase will almost always fail and trigger a rewind, and no progress will ever be made. This situation can be trivially corrected by repeating each broadcast symbol $\mathcal{O}(\log n)$ times, and thereby effectively reducing the noise rate to less than $\frac{1}{n}$. However, this is unaffordable for a constant overhead simulation.

We note that this repeated rewinds problem is avoided by [10] as, although the total number of parties n is large, the assumed non-adaptivity of Π can be used to determine a small subset S of parties that *critically* need to know the simulated transcript. These are the parties that will broadcast in the rounds immediately following the current one. The remaining parties broadcast later in the future and therefore have more time to decode the symbol broadcast in the current round. Then, [10] show that it is enough to make sure that parties in S are not experiencing message drops, which helps reduce overhead down to a constant. Since in the adaptive case, it is possible that *any* of the n parties broadcasts next, this approach cannot be implemented.

Message certification. An even bigger problem we encounter when attempting to run the [10] scheme on adaptive protocols is that it crucially uses the fact that the symbol received from the channel in every round can be *certified* by at least one of the parties: Since Π is

assumed to be non-adaptive, it can also be assumed that a single party broadcasts in every round (collisions and silences can be eliminated ahead of time). Furthermore, this party (and all other parties) knows that it is the only one to broadcast. Therefore, if party i broadcast the symbol σ in round t of Π and some claimed transcript of Π has a symbol different from σ in round t, party i can "object" to this transcript to trigger a rewind.

The adaptive setting is different though. Consider, for example, the case where Π is adaptive and in some rounds has multiple parties broadcasting simultaneously, causing a collision. We call such collisions *intended collisions*. Suppose, however, that in round t, party i was the only one to broadcast, but the claimed transcript for Π has $\perp$ in round t. Since party i may no longer know that it is the only one to broadcast in this round, it may deem it possible that others have broadcast as well, leading to an intended collision, and thus will not object. The other, silent, parties may not object either as they may think that this is a collision or a silent round.

2.2 Avoiding The Repeated Rewinds Problem

A protocol Π exhibiting repeated rewinds

To explain how our scheme handles the first (and easier) repeated rewinds problem described above, consider the following protocol Π that exhibits it (the second, message certification problem, does not occur): The protocol is played over an underlying complete binary tree of depth $T < 2^n$. Each of the n parties gets as input, one symbol $b_v \in \{0, 1, \star\}$ for each vertex v in the tree, where the inputs are sampled as follows: First, we select one of the root-to-leaf paths in the tree uniformly at random and call it the "*correct path*". We assign each of the vertices v on this path to exactly one of the n parties uniformly at random. Here, by "assigning vertex v to party i" we mean that party i gets a bit $b_v \in \{0, 1\}$ for vertex v. If vertex v is not assigned to party i, party i gets $b_v = \star$. Additionally, each of the vertices v outside this path is assigned to many parties, say, to a set of $\frac{n}{2}$ parties selected uniformly at random.

In the noiseless protocol Π, all parties start from the root of the tree, and, upon reaching node v, a party that was not assigned v (has $b_v = \star$) stays silent, and a party that was assigned v broadcasts its bit b_v. Since each of the vertices on the correct path was assigned to exactly one party, exactly one party broadcasts a bit, and all parties then progress to the child of v indicated by this bit (that is, if 0 is broadcast they update v to be the left child of v, otherwise to the right child). This is done until a leaf is reached, which is also the output of the protocol.

Observe that since on every vertex of the correct path a single party broadcasts (and the parties know that this is the case), the message certification problem does not occur. However, since any of the n parties may potentially be the one to broadcast in the next round, the repeated rewinds problem occurs.

The *play-it-safe* simulation scheme

To avoid repeated rewinds in our simulation of Π, we make sure that parties never go off the correct path (i.e., no party ever reaches a vertex v that is not on the correct path) by guaranteeing that the parties never broadcast when it is not their turn to broadcast. To this end, our policy for the parties is that they always *play it safe* and *never broadcast unless they know the entire transcript so far.*

Of course, it may be the case that the received transcript of the party who should broadcast next contains erasures, causing it to refrain from broadcasting. Since no other party broadcasts, this will be a silent round and all parties will receive $\perp$. Upon receiving $\perp$, parties do not update their current node v in the tree. Thus, no progress is made in this round, where progress is measured as the number of steps taken on the correct path (the depth of v in the tree). Note, however, that indeed in this protocol parties never go off the correct path.

To allow progress to resume, we need to ensure that the erasures in the transcript of the party that should broadcast next are resolved (hopefully, within a few rounds). To this end, we pick one of the parties (say, the first party) to be the *leader*. After every communication round, this leader re-broadcasts the symbol it received from the channel on a *tree code* [26]. A tree code is essentially an error correcting code that can be computed "online" and ensures that the messages sent until round t will *eventually* be decoded correctly, where the probability of correct decoding greatly increases with the number of rounds that have passed since round t. Thus, parties that suffer an erasure will be able to recover the missing symbol over the next few iterations by observing what was received from the leader on the tree code. This means that, while progress may pause, it will resume within a few rounds.

2.3 Avoiding The Message Certification Problem

A harder-to-simulate protocol Π

Now let us address the second (and more severe) problem of message certification. Observe that in our simulation of the above protocol Π we did not encounter this problem. The reason is that on every vertex on the correct path a single party is scheduled to broadcast. We now consider the more general case where some of the vertices on the correct path are given to more than one party. For concreteness, say that a quarter of the vertices v on the correct path are given to exactly 2 parties, and an additional quarter is given to $\frac{n}{2}$ parties (that is, in total, there is an intended collision on half of the vertices on the correct path). Additionally, assume that the underlying tree is ternary (instead of binary), and the children of every non-leaf vertex are labeled by $\{0, 1, \perp\}$. In a case of an intended collision, the $\perp$ child of the current vertex should be taken.

Erasures can cause errors

Observe that the play-it-safe simulation protocol we had before has to change: When designing it, we assumed that there are no collisions on the correct path, thus progress was paused when a $\perp$ symbol was received (that is, the parties did not update their current vertex v in the tree). As intended collisions are now possible, we ask that, upon receiving $\perp$, the parties update v to the $\perp$ child of v.

Observe however, that since the parties are *unable to differentiate intended collisions from erasures*, as both are received as $\perp$, they may go off the correct path and will need to eventually detect the error and rewind. We note that working in the erasure model typically means that a party that does not have the correct transcript knows that it does not have the correct transcript. However, as is evident here, this reasoning does not apply to our erasure model. In this sense, our model is *closer to the corruptions model* than other erasure models.

In our simulation, parties can go off the correct path in round t if the party that was supposed to broadcast in round t (say party i) did not do so as it did not know the full transcript so far. By not broadcasting, party i potentially converts the output of the channel in round t from a bit to $\perp$ (this happens when party i was supposed to be the only one

to broadcast) or from $\perp$ to a bit (this happens when one additional party was supposed to broadcast). Recall that, owing to the usage of a tree code, party i eventually learns the complete transcript until the missed round t. When this happens, we can have party i object in the next consistency check in order to trigger a rewind. However, because the rate of erasures is constant and parties broadcast very often (recall that a quarter of the vertices on the correct path are given to $\frac{n}{2}$ parties), there are likely to be too many missed rounds and such objections will once again cause repeated rewinds.

Critical parties

To implement a rewind-if-error mechanism without repeated rewinds, we observe that rewinds are required only when the output symbol was changed due to party i (a party that was scheduled to broadcast in round t) not broadcasting in round t. Note that this only happens if the output symbol in round t is not a collision. In this case, we say that party i is *critical*[5] for round t. We use the policy that party i only objects to round t if it is critical to round t[6]. Note that this policy does not cause repeated rewinds: if many parties were supposed to broadcast in round t, none of them is critical (this round will be a collision round even if one of these parties will not broadcast). Otherwise, if few parties were supposed to broadcast in round t, then there is a good chance that round t is not erased in any of the received transcripts of these parties.

Collision-*not*-as-silence

To be able to implement the policy, party i needs to know if it was critical to the round t that it missed. Observe that if round t was a collision round even without party i broadcasting, then party i is not critical for round t, and no rewind is necessary. It is not hard to see that this is in fact the only case where a party who missed a round is not critical for this round. This means that testing criticality boils down to the ability to differentiate a collision round from a silent round.

To differentiate collision rounds from silent rounds, we use a known radio networks *collision detection* trick. Assume for the purposes of this sketch that there is some player, say the leader, that is known to not broadcast in this round[7]. We "run" the round twice, once in a black-box way (without the leader broadcasting), and once again while having the leader broadcast. If the round was a silent round, then the parties receive a $\perp$ in the first run, and a bit (non-$\perp$ symbol) in the second, while if the round was a collision, they will receive $\perp$ in both the runs. As they receive a different combination of symbols, they can distinguish between collisions and silences[8]. Note that the argument above assumes sender collision-detection, i.e., the parties that are transmitting also receive a symbol in that round. However, this assumption is not needed, see Footnote 10 and Remark 3.

---

[5] We mention that this definition differs slightly from the technical sections, but implements a similar idea.

[6] Observe that a priori, it is not clear if the parties know they are critical. We deal with this later in this section. We also note that the notion of critical parties does not appear in the algorithm description and is used only in the analysis.

[7] This assumption can easily be removed by, *e.g*, running the round an extra time where only the leader will broadcast.

[8] We note that noise can erase the symbol broadcast by the leader in the second run and effectively erase a silence out to look like a collision. We distinguish between these and regular collisions using the method described in Section 2.4.

2.4 Erasures *To* And *From* The Leader

Recall from Section 2.2 that after every round the leader re-transmits the symbol that it received from the channel in this round. We next discuss issues that can arise when the communication to/from the leader is erased.

Erasures *to* the leader

Consider the case where the true output of the channel in a given round is a bit, but the leader receives $\perp$ due to an erasure (re-transmitting this $\perp$ may cause the execution of the protocol to go off the correct path). However, since erasures are assumed to happen independently, then with probability exponentially small in n, at least one of the other parties receives the erased bit and can object in the next consistency check to trigger a rewind. Using the assumption that the length of the protocol is at most exponential in n, we get that all such leader errors will be corrected with high probability. We mention that this is the only place in our proof where we use the bound on the length of the protocol.

Erasures *from* the leader: Collision-as-silence-*not*-as-erasures

Now consider the situation where the leader receives a bit and re-transmits it, but, due to erasures, some parties receive a $\perp$. By updating their current node v using this $\perp$, these parties may fall off the correct path. As mentioned in Section 2.3, this type of error occurs as the channel does not distinguish between erasures and collisions/silences.

To circumvent this problem, we convert our collision-as-silence-as-erasures channel to a collision-as-silence-*not*-as-erasures channel. This is done by having the leader broadcast a special symbol[9] other than 0, 1, and $\perp$, in the case it receives $\perp$. As the other parties know that the leader never broadcasts $\perp$, they can deduce that any $\perp$ they may receive from it is due to an erasure. On the other hand, if they receive the special symbol, they can conclude that the round is a collision/silence.

2.5 Implementing Check Phases

The simulation scheme we discuss so far is in the rewind-if-error framework. In this sketch we attempted to show that whenever the parties go off the correct path due to erasures, at least one of the parties is able to detect the problem and object in the next check phase.

To implement a check phase, we ask parties that wish to object to broadcast a bit (say, 1), and ask all other parties to keep silent. Then, the collision detection subroutine described above allows the parties to tell whether 0, 1, or more than 1 parties were broadcasting, and thus also allows them to tell whether there exists an objecting party and a rewind should take place.

3 The Model

In this paper, we study the broadcast channel with random erasures, assuming the *collision-as-silence-as-erasures* model. To define the model and throughout this paper, we will use the following notation. For a string s, we shall use $|s|$ to denote the length of s. For $i \in [|s|]$, let s_i denote the i^{th} coordinate of s and $s_{<i}, s_{\leq i}$ denote the prefix of the first $i-1$ and i

[9] The actual proof does not require an additional symbol. Rather, we encode every symbol by two symbols.

coordinates of s, respectively. For two strings s, t over the same alphabet, denote by $\Delta(s,t)$ the Hamming distance between s and t, by $\mathsf{LCP}(s,t)$ the longest common prefix of the strings s and t, and by $s \| t$ the concatenation of s and t.

The (n, ϵ, Γ)-*noisy broadcast channel* is defined by a number $n \geq 0$ of parties, an error parameter $\epsilon > 0$, and an alphabet set Γ satisfying $|\Gamma| > 1$. We shall refer to player 1 as the *leader* Ld, use $\perp$ to denote a special symbol not in Γ (this symbol will represent collisions, silences, and deletions), and define $\Gamma_c = \Gamma \cup \{\perp\}$. We also define the (n, Γ)-*broadcast channel* to be the *noiseless* version of this channel, i.e., when $\epsilon = 0$.

Definition of a protocol

A (deterministic) *protocol* Π over the (n, ϵ, Γ)-noisy broadcast channel is defined as:

$$\Pi = \left(T, \{\mathcal{X}^i\}_{i\in[n]}, \mathcal{Y}, \{M_j^i\}_{i\in[n], j\in[T]}, \mathsf{out}\right). \tag{1}$$

Here, $T = \|\Pi\|$ is the number of rounds (or the *length*) of the protocol, $\mathcal{X}^i$ is the input space for player i, $\mathcal{Y}$ is the output space of the protocol, $M_j^i : \mathcal{X}^i \times \Gamma_c^{j-1} \to \Gamma_c$ is the function player i uses to determine what message to send in round j, and $\mathsf{out} : \Gamma_c^T \to \mathcal{Y}$ is the function the leader uses to determine the output from its received transcript. As usual, we define a *randomized protocol* to be a distribution over (deterministic) protocols.

Execution of a protocol

The protocol Π starts with all players $i \in [n]$ having an input $x^i \in \mathcal{X}^i$ and proceeds in T rounds, maintaining the invariant that before round j, for all $j \in [T]$, all players i have a transcript $\pi^i_{<j} \in \Gamma_c^{j-1}$. In round j, player i broadcasts $z^i = M_j^i(x^i, \pi^i_{<j}) \in \Gamma_c$. Define the function:

$$\mathtt{combine}(z^1, \cdots, z^n) = \begin{cases} z^i, & \text{if } \exists \text{ unique } i \in [n] \text{ such that } z^i \neq \perp \\ \perp, & \text{otherwise} \end{cases}. \tag{2}$$

Now, the symbol π^i_j received by player i in round j equals $\mathtt{combine}(z^1, \cdots, z^n)$, with probability $1-\epsilon$, and equals $\perp$, with probability ϵ, independently for all $i \in [n]$ and $j \in [T]$.[10] In the latter case, we say the message to player i in round j was *erased* by the noise. Player i appends π^i_j to $\pi^i_{<j}$ to get a transcript $\pi^i_{\leq j}$ and continues the execution of the protocol.

After T rounds, the leader outputs $\Pi^{\mathsf{Ld}}(X) = \mathsf{out}(\pi^{\mathsf{Ld}}_{\leq T}) \in \mathcal{Y}$. (Note that using only $\mathcal{O}(\max\{T, \log n\})$ additional transmissions, the leader can communicate the output to all the other parties in a reliable manner by encoding with a standard error correcting code.) We shall sometimes omit Ld when the channel is noiseless, as in this case, all the players receive the same transcript and can compute the output.

4 Our Simulation Protocol

We formalize Theorem 1 as Theorem 2 (below). (Note that by having the parties repeat every round of the original protocol Π constantly many times and taking the majority of the outputs, we get the channel noise rate to be smaller than 10^{-10}).

[10] We remark that in the literature (e.g., [6]), the broadcast channel (single-hop radio networks) is often defined such that a player that broadcasts a symbol (other than $\perp$) in a round does not receive any symbol from the channel in that round (in other words, there is no sender collision-detection). However, for simplicity of presentation, in this paper we assume this stronger model. We explain how to make our protocol work with no sender collision-detection in Remark 3.

▶ **Theorem 2** (Formal Version of Theorem 1). *There exists a constant C such that the following holds: Fix $\epsilon = 10^{-10}$, $n > 0$, an alphabet set Γ satisfying $|\Gamma| > 1$. For any protocol Π of length $T \leq 2^n$ in the (n, Γ)-broadcast channel, there is a protocol Π' over the (n, ϵ, Γ)-noisy broadcast channel, with $\|\Pi'\| \leq CT$, and such that for all inputs $X = \left(x^1, x^2, \cdots, x^n\right)$ for the players, we have:*

$$\Pr\left(\Pi'^{\mathsf{Ld}}(X) \neq \Pi(X)\right) \leq 2^{-\min(n,T)},$$

where the probability is over the noise in the channel.

We note that when n is small, so is T, so Π can be simulated by simply repeating each round sufficiently many times. As such, without loss of generality, we may assume that n is large.

The proof of Theorem 2 spans the rest of this paper. In this section we give the simulation protocol Π', and in Appendix B we give its analysis.

Let n, ϵ, Γ be as in the theorem statement and assume without loss of generality that $\Gamma = [|\Gamma|]$. Fix a protocol Π. Observe that fixing Π also fixes $T, \left\{\mathcal{X}^i\right\}_{i\in[n]}, \left\{M_j^i\right\}_{i\in[n],j\in[T]}$, etc. as in Equation (1). As a randomized protocol is simply a distribution over deterministic protocols, we can assume without loss of generality, that the protocol Π is deterministic. We also assume without loss of generality that the output of Π is just its transcript. In order to define the protocol Π', we first set up some notation.

Protocol notation

Define the sets $\mathcal{P}^{\mathsf{Ld}} = [n]$ (all parties including the leader), and $\mathcal{P} = \{2, 3, \ldots, n\}$ (all parties excluding the leader).

As motivated in Section 2, our protocol shall implicitly implement a collision detection model, having two separate symbols for collisions and silences. We shall use a special symbol $\perp_C \notin \Gamma$ to denote a collision and $\perp_S \notin \Gamma$ to denote a silence. Define $\Gamma_{cs} = \Gamma \cup \{\perp_C, \perp_S\}$.

Additionally let $\mathsf{R} \notin \Gamma$ be a special symbol indicating that the leader wants to rewind a round, and denote by $\Gamma_{csr} = \Gamma_{cs} \cup \{\mathsf{R}\}$. We shall treat both $\perp_C$ and $\perp_S$ as $\perp$ in our protocol, and output a string in Γ_{cs}^T. We also redefine the message functions, M_j^i, to take inputs from Γ_{cs}^{j-1} instead of Γ_c^{j-1}, treating both $\perp_C$ and $\perp_S$ as $\perp$, e.g., $M_j^i(x^i, \perp_C \| \perp_S) = M_j^i(x^i, \perp \| \perp)$. For simplicity, we shall pad the protocol Π with $\perp$ infinitely many times and correspondingly define, for all $i \in [n]$, $j > T$, the value $M_j^i(\cdot, \cdot) = \perp$.

Our protocol will use a $(\Gamma_{csr}, \Gamma, R_{\mathsf{TC}}, 0.4)$-tree code TC, where $R_{\mathsf{TC}} \geq \max\left(10^5, 10R\right)$ is a sufficiently large constant and R is as promised by Theorem 5. This tree code will only be written to by the leader, and will be used to log the leader's simulated transcript. In our protocol, when we say *the leader writes $s \in \Gamma_{csr}$ to the tree code*, we mean that it computes and broadcasts $\mathsf{TC}(\rho \| s)$, where ρ is the string of all the symbols it wrote to the tree code before the current s. We shall also use $\mathsf{D\text{-}TC}$ to denote the tree code decoding function from Definition 6.

We give a formal description of our protocol Π' in Algorithm 1.

▶ Remark 3. Recall from Footnote 10 that we are assuming a broadcast model with sender collision-detection. In other words, we assume that players that are talking (broadcasting a symbol other than $\perp$) also receive an output symbol from the channel. We next claim that our simulation protocol Π' can be made to work over the channel with no sender collision-detection, that is, when only players that listen (broadcast $\perp$), get the output symbol from the channel.

Algorithm 1 The simulation protocol Π'.

Input: Each party $i \in \mathcal{P}^{\mathsf{Ld}}$ holds an input $x^i \in \mathcal{X}^i$.
Output: The leader outputs $\pi \in \Gamma_{cs}^T$, that represents a transcript for Π.

1: **for** $t \in \left[10^5 T\right]$ **do**
2: Each player $i \in \mathcal{P}^{\mathsf{Ld}}$ runs parse on τ^i to get output $\left(\pi^i, r^i\right)$, where:
- τ^i, for $i \in \mathcal{P}$, is the concatenation of all messages received by player i at Line 8 up to this point (possibly none).
- τ^{Ld} is the concatenation of all messages broadcast (as opposed to received) by the leader at Line 8 up to this point (possibly none).

3: Each player $i \in \mathcal{P}^{\mathsf{Ld}}$ computes $z^i \leftarrow M^i_{|\pi^i|+1}\left(x^i, \pi^i\right)$. Set $z^i \leftarrow \perp$ if $\pi^i = \mathsf{fail}$.
4: The parties run detect-collisions, using z^i as the input for player $i \in \mathcal{P}$.
Let w^i be the output for player $i \in \mathcal{P}^{\mathsf{Ld}}$.
5: The leader represents $w^{\mathsf{Ld}} \in \Gamma_{cs}$ as an element of Γ^4 and broadcasts it in 4 rounds.
Let $\tilde{w}^i$ be the symbol decoded by player $i \in \mathcal{P}$, or $\perp$ if the player fails to decode.
6: Each player $i \in \mathcal{P}$ sets a flag $e^i \in \{1, \perp\}$ as follows:

$$e^i \leftarrow \begin{cases} 1, & \text{if } r^i = \mathsf{true} \text{ or } \tilde{w}^i = \perp_C \neq w^i \\ \perp, & \text{otherwise} \end{cases}.$$

7: The parties run detect-collisions, using e^i as the input for player $i \in \mathcal{P}$.
Let e^{Ld} be the output for the leader.
8: The leader writes $s^{\mathsf{Ld}} \in \Gamma_{csr}$ to the tree code, where

$$s^{\mathsf{Ld}} \leftarrow \begin{cases} \mathsf{R}, & \text{if } e^{\mathsf{Ld}} \neq \perp_S \\ w^{\mathsf{Ld}}, & \text{else if } z^{\mathsf{Ld}} = \perp \\ z^{\mathsf{Ld}}, & \text{else if } w^{\mathsf{Ld}} = \perp_S \\ \perp_C, & \text{otherwise} \end{cases}.$$

9: **end for**
10: The leader runs parse on τ^{Ld} to get output $\left(\pi^{\mathsf{Ld}}, r^{\mathsf{Ld}}\right)$, where τ^{Ld} is as in Line 2. The leader then outputs $\pi^{\mathsf{Ld}}_{\leq T}$.

Algorithm 2 Algorithm detect-collisions, that distinguishes between collisions and silence.

Input: Each player $i \in \mathcal{P}$ has a symbol $z^i \in \Gamma_c$ that it wishes to broadcast in this round.
Output: Each player $i \in \mathcal{P}^{\mathsf{Ld}}$ outputs a guess $w^i \in \Gamma_{cs}$ for the combined symbol.

11: In one round of communication, each player $i \in \mathcal{P}$ broadcasts z^i and the leader broadcasts $\perp$.
Let u^i be the symbol heard by player $i \in \mathcal{P}^{\mathsf{Ld}}$.
12: In one round of communication, each player $i \in \mathcal{P}$ broadcasts z^i and leader broadcasts 1.
Let $\overline{u}^i$ be the symbol heard by player $i \in \mathcal{P}^{\mathsf{Ld}}$.
13: Each player $i \in \mathcal{P}^{\mathsf{Ld}}$ returns w^i, where

$$w^i \leftarrow \begin{cases} u^i, & \text{if } u^i \neq \perp \\ \perp_S, & \text{else if } \overline{u}^i \neq \perp \\ \perp_C, & \text{otherwise} \end{cases}.$$

Algorithm 3 Algorithm parse, run locally by a player $i \in \mathcal{P}^{\mathsf{Ld}}$ to decode and parse the tree code.

Input: Player i has $\tau \in \Gamma_c^*$, its view of the symbols encoded over the tree code.
Output: Player i outputs a transcript $\pi \in \Gamma_c^*$ or fail if it failed to decode the tree code, and a rewind flag $r \in \{\mathsf{true}, \mathsf{false}\}$ which is true if the player found a problem with π.

14: Initialize π to be the empty string, $\ell \leftarrow \infty$.
15: Let $\rho \leftarrow \mathsf{D\text{-}TC}(\tau)$.
16: If $\rho = \mathsf{fail}$, terminate and return $(\mathsf{fail}, \mathsf{false})$.
17: **for** $k \in [|\rho|]$ **do**
18: **if** $\rho_k = \mathsf{R}$ **then**
19: $\pi \leftarrow \pi_{<|\pi|}$.
20: **if** $|\pi| < \ell$ **then**
21: $\ell \leftarrow \infty$.
22: **end if**
23: **else**
24: $\pi \leftarrow \pi \| \rho_k$.
25: **if** $\mathsf{D\text{-}TC}\left(\tau_{\leq (k-1)R_{\mathsf{TC}}}\right) = \mathsf{fail}$ **and** $M^i_{|\pi|}\left(x^i, \pi_{<|\pi|}\right) \neq \perp$ **and** $\rho_k \neq \perp_C$ **then**
26: $\ell \leftarrow \min(\ell, |\pi|)$.
27: **end if**
28: **end if**
29: **end for**
30: Return $(\pi, \ell \neq \infty)$.

There are two sources of problems if we assume no sender collision-detection. The first is that players $i \in \mathcal{P}$ are expected to get their own w^i at Line 4, which they use to detect erasures experienced by the leader (compute e^i in Line 6). However, as erasures are one-sided, if at least two different players $i \neq i' \in \mathcal{P}$ talk in the same round, the leader and all listening players will receive the correct symbol, i.e., $\perp_C$, as the value of w^i. As such, if an erasure causes the leader to get an incorrect w^{Ld}, there is at most one player $i \in \mathcal{P}$ who is talking. Thus, almost all players in $\mathcal{P}$ are listening, so they will have their own w^i, and this erasure is likely to be detected.

The second issue that arises is that the leader is expected to both talk and listen at Line 12. Recall that the purpose of algorithm detect-collisions is to run a round of the original protocol and essentially tell whether 0, 1, or ≥ 2 players in $\mathcal{P}$ are talking. The leader acts as a "noisemaker" in Line 12 to distinguish the case of 0 talking players from the case of ≥ 2 talking players. However, the role of a noisemaker can be handled by any other player, as long as that player would never have talked in this round otherwise.

This gives rise to the following modification of algorithm detect-collisions: We partition the parties in $\mathcal{P}$ into two non-empty sets $\mathcal{P}_1$ and $\mathcal{P}_2$. We then have parties in $\mathcal{P}_1$ perform algorithm detect-collisions with an arbitrary player in $\mathcal{P}_2$ acting as a noisemaker, and vice versa. This allows the leader to determine whether there were 0, 1, or ≥ 2 players talking in $\mathcal{P}_1$ and in $\mathcal{P}_2$, from which they can tell if there were 0, 1, or ≥ 2 players talking in $\mathcal{P}$.

As there are no other cases in the protocol Π' where a player both talks and uses the value given to it by the channel, these changes are sufficient to make the algorithm work with no sender collision-detection.

References

1 Noga Alon, Mark Braverman, Klim Efremenko, Ran Gelles, and Bernhard Haeupler. Reliable communication over highly connected noisy networks. In *Symposium on Principles of Distributed Computing (DISC)*, pages 165–173. ACM, 2016.

2 Yagel Ashkenazi, Ran Gelles, and Amir Leshem. Brief announcement: Noisy beeping networks. In *Symposium on Principles of Distributed Computing (PODC)*, pages 458–460, 2020.

3 Reuven Bar-Yehuda, Oded Goldreich, and Alon Itai. On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *Journal of Computer and System Sciences*, 45(1):104–126, 1992.

4 Mark Braverman, Klim Efremenko, Ran Gelles, and Bernhard Haeupler. Constant-rate coding for multiparty interactive communication is impossible. In *Symposium on Theory of Computing (STOC)*, pages 999–1010. ACM, 2016.

5 Keren Censor-Hillel, Bernhard Haeupler, D Ellis Hershkowitz, and Goran Zuzic. Broadcasting in noisy radio networks. In *Symposium on Principles of Distributed Computing (PODC)*, pages 33–42, 2017.

6 Keren Censor-Hillel, Bernhard Haeupler, D Ellis Hershkowitz, and Goran Zuzic. Erasure correction for noisy radio networks. In *International Symposium on Distributed Computing (DISC)*, 2019.

7 Imrich Chlamtac and Shay Kutten. On broadcasting in radio networks-problem analysis and protocol design. *IEEE Trans. Communications*, 33(12):1240–1246, 1985.

8 Klim Efremenko, Gillat Kol, Dmitry Paramonov, and Raghuvansh R. Saxena. Computation over the noisy broadcast channel with malicious parties. In *Innovations in Theoretical Computer Science Conference, (ITCS)*, volume 185, pages 82:1–82:19, 2021.

9 Klim Efremenko, Gillat Kol, Dmitry Paramonov, and Raghuvansh R. Saxena. Tight bounds for general computation in noisy broadcast networks. In *Symposium on Foundations of Computer Science (FOCS)*, pages 634–645, 2021.

10 Klim Efremenko, Gillat Kol, and Raghuvansh Saxena. Interactive coding over the noisy broadcast channel. In *Symposium on Theory of Computing (STOC)*, pages 507–520. ACM, 2018.

11 Klim Efremenko, Gillat Kol, and Raghuvansh Saxena. Radio network coding requires logarithmic overhead. In *Foundations of Computer Science (FOCS)*, pages 348–369, 2019.

12 Klim Efremenko, Gillat Kol, and Raghuvansh R. Saxena. Noisy beeps. In Yuval Emek and Christian Cachin, editors, *Symposium on Principles of Distributed Computing (PODC)*, pages 418–427, 2020.

13 Uriel Feige and Joe Kilian. Finding OR in a noisy broadcast network. *Information Processing Letters*, 73(1-2):69–75, 2000.

14 Robert G. Gallager. Finding parity in a simple broadcast network. *IEEE Transactions on Information Theory*, 34(2):176–180, 1988.

15 Abbas El Gamal. Open problems presented at the 1984 workshop on specific problems in communication and computation sponsored by bell communication research. *"Open Problems in Communication and Computation", by Thomas M. Cover and B. Gopinath (editors). Springer-Verlag*, 1987.

16 Ran Gelles and Yael T Kalai. Constant-rate interactive coding is impossible, even in constant-degree networks. *IEEE Transactions on Information Theory*, 65(6):3812–3829, 2019.

17 Ran Gelles, Yael Tauman Kalai, and Govind Ramnarayan. Efficient multiparty interactive coding for insertions, deletions, and substitutions. In *Symposium on Principles of Distributed Computing (PODC)*, pages 137–146, 2019.

18 Navin Goyal, Guy Kindler, and Michael Saks. Lower bounds for the noisy broadcast problem. *SIAM Journal on Computing*, 37(6):1806–1841, 2008.

19 Ofer Grossman, Bernhard Haeupler, and Sidhanth Mohanty. Algorithms for noisy broadcast with erasures. In *Colloquium on Automata, Languages, and Programming (ICALP)*, volume 107 of *LIPIcs*, pages 153:1–153:12, 2018.

20 William M. Hoza and Leonard J. Schulman. The adversarial noise threshold for distributed protocols. In *Symposium on Discrete Algorithms (SODA)*, pages 240–258, 2016.
21 Abhishek Jain, Yael Tauman Kalai, and Allison Bishop Lewko. Interactive coding for multiparty protocols. In *Symposium on Theory of computing (STOC)*, pages 1–10, 2015.
22 Eyal Kushilevitz and Yishay Mansour. Computation in noisy radio networks. *SIAM Journal on Discrete Mathematics (SIDMA)*, 19(1):96–108, 2005.
23 Ilan Newman. Computing in fault tolerance broadcast networks. In *Computational Complexity Conference (CCC)*, pages 113–122, 2004.
24 Sridhar Rajagopalan and Leonard J. Schulman. A coding theorem for distributed computation. In *Symposium on the Theory of Computing (STOC)*, pages 790–799, 1994.
25 Leonard J Schulman. Communication on noisy channels: A coding theorem for computation. In *Foundations of Computer Science (FOCS)*, pages 724–733. IEEE, 1992.
26 Leonard J Schulman. Deterministic coding for interactive communication. In *Symposium on Theory of computing (STOC)*, pages 747–756. ACM, 1993.
27 Andrew Chi-Chih Yao. On the complexity of communication under noise. *invited talk in the 5th ISTCS Conference*, 1997.

A Technical Preliminaries

A.1 Tree Codes

Our algorithms make use of *tree codes*, first introduced in [26].

▶ **Definition 4** (Tree Codes). *Let $\mathcal{X}$ and Γ be two alphabet sets, $R_{\mathsf{TC}} > 0$ be an integer, and $\delta \in (0,1)$. An $(\mathcal{X}, \Gamma, R_{\mathsf{TC}}, \delta)$-*tree code *is a function $\mathsf{TC} : \mathcal{X}^* \to \Gamma^{R_{\mathsf{TC}}}$ such that for any integer $k \geq 0$ and strings $x, x' \in \mathcal{X}^k$, defining $\overline{\mathsf{TC}}(x) = \mathsf{TC}(x_{\leq 1}) \| \mathsf{TC}(x_{\leq 2}) \| \cdots \| \mathsf{TC}(x)$, we have:*

$$\Delta\big(\overline{\mathsf{TC}}(x), \overline{\mathsf{TC}}(x')\big) \geq \delta R_{\mathsf{TC}} \cdot (k - |\mathsf{LCP}(x, x')|).$$

▶ **Theorem 5** ([26]). *There exists a constant $R \geq 0$ such that for any alphabet sets $\mathcal{X}$, Γ and all $R_{\mathsf{TC}} \geq R \cdot \frac{\log|\mathcal{X}|}{\log|\Gamma|}$, there exists an $(\mathcal{X}, \Gamma, R_{\mathsf{TC}}, 0.4)$-tree code.*

We will also need a way of decoding tree codes from erasures. Recall the notation $\Gamma, \perp, \Gamma_c$ from above, and let $w_{i,j} = (w_i)_j$.

▶ **Definition 6** (Decoding from Erasures). *Let TC be an $(\mathcal{X}, \Gamma, R_{\mathsf{TC}}, \delta)$-tree code. The* decoding function of TC, *denoted $\mathsf{D\text{-}TC} : (\Gamma_c^{R_{\mathsf{TC}}})^* \to \mathcal{X}^* \cup \{\mathsf{fail}\}$, is given by the following: For an integer $k \geq 0$ and $w \in (\Gamma_c^{R_{\mathsf{TC}}})^k$,*

$$\mathsf{D\text{-}TC}(w) = \begin{cases} z, & \text{if } \exists \text{ unique } z \in \mathcal{X}^k : \forall i \in [k], j \in [R_{\mathsf{TC}}] : w_{i,j} \in \{\perp, \mathsf{TC}_j(z_{\leq i})\} \\ \mathsf{fail}, & \text{otherwise} \end{cases}.$$

▷ Claim 7. Let TC be an $(\mathcal{X}, \Gamma, R_{\mathsf{TC}}, \delta)$-tree code and let $\mathsf{D\text{-}TC}$ be its decoding function. Let $k \geq 0$ be an integer and let $z \in \mathcal{X}^k$. Then, for any $\tilde{\tau} \in \left(\Gamma_c^{R_{\mathsf{TC}}}\right)^k$ such that $\forall i \in [k], j \in [R_{\mathsf{TC}}] : \tilde{\tau}_{i,j} \in \{\perp, \mathsf{TC}_j(z_{\leq i})\}$, it holds that $\mathsf{D\text{-}TC}(\tilde{\tau}) \in \{z, \mathsf{fail}\}$.

B Analyzing the Protocol

In this section we prove that the simulation protocol Π' given in Section 4 satisfies Theorem 2.

We omit the proofs of lemmas in this section for space. They can be found in the full version of the paper.

Iterations and rounds

Observe that our protocol Π' has $T' = 10^5 T$ *iterations* and each iteration has $R' = R_{\mathsf{TC}} + 8$ *rounds* of communication: 2 rounds in the call to detect-collisions in Line 4, 4 rounds in Line 5, 2 rounds in the call to detect-collisions in Line 7, and R_{TC} rounds in Line 8.

The noise indicator

For $t \in [T']$, $r \in [R']$, and $i \in [n]$, we define the indicator random variable $\mathsf{N}_{t,r,i}$ to be 1 if and only if the message received by player i in the r^{th} round of communication in iteration t is erased due to noise. For a set $S \subseteq [T']$, we shall use N_S to denote the collection $\mathsf{N} = \{\mathsf{N}_{t,r,i}\}_{t\in S, r\in[R'], i\in[n]}$ and sometimes abbreviate $\mathsf{N}_{[T']}$ as N and $\mathsf{N}_{[t]}$ as $\mathsf{N}_{\leq t}$ for all $t \in [T']$. Observe that our definition implies that the variables in N are mutually independent and identically distributed, and take the value 1 with probability ϵ.

Note that fixing any instantiation N of N together with the inputs X to the parties fixes the entire execution of Π'. In fact, for all $t \in [T']$, fixing any instantiation $N_{\leq t}$ of $\mathsf{N}_{\leq t}$ fixes the execution of the first t iterations of Π'. This means that it also fixes the values of all the variables in these iterations.

Variables

For $i \in [n]$ and a variable *var* in Algorithms 1 and 3,[11] we shall use $var_t^i(N)$ to denote the value of variable *var* as seen by player i at the end of iteration t when the noise is N. We shall use $t = 0$ to denote the values at the start of the execution and drop N when it is clear from context. As explained above, these values are determined by $N_{\leq t}$. We also use $\pi^{\mathsf{Ld}}_{T'+1}(N)$ to refer to the leader's π^{Ld} at Line 10.

The collision-not-as-silence model

To help with our analysis, we define a function `combine-CD` that intuitively captures the behavior of a broadcast channel with collision-detection. Formally, we have, for $z^1, z^2, \cdots, z^n \in \Gamma_c^*$,

$$\texttt{combine-CD}\big(z^1, \cdots, z^n\big) = \begin{cases} \perp_S, & \text{if } \forall i \in [n] : z^i = \perp \\ z^i, & \text{if } \exists \text{ unique } i \in [n] \text{ such that } z^i \neq \perp \\ \perp_C, & \text{otherwise} \end{cases}. \tag{3}$$

For the rest of the text, fix inputs $X = \big(x^1, x^2, \cdots, x^n\big)$ for the players. We abuse notation slightly and denote by $\Pi = \Pi(X)$ the transcript of the noiseless protocol Π when the inputs to the parties are as in X and the model uses `combine-CD` in place of `combine` (thus, $\Pi \in \Gamma_{cs}^T$). This is without loss of generality as a transcript in the collision-not-as-silence model only has more information than one in the collision-as-silence model.

B.1 Technical Lemmas and One-Sided Error

A key property of our model is the fact that our noise is one-sided: After collisions are resolved, the resulting symbol will either be received correctly, or will be replaced by a $\perp$. This means that if a player hears a symbol that is not $\perp$, that player will accurately know that that is the "correct" symbol, and that they were not affected by noise.

[11] We do not use this notation for variables in Algorithm 2 as that is invoked twice in every iteration.

This property means that we can make several very useful claims, which we use throughout the rest of this paper.

▶ **Lemma 8.** *For all $t \in [T']$, all $i \in [n]$, and all instantiations N of* N*,*

$$\pi_t^i(N) \in \left\{\mathsf{fail}, \pi_t^{\mathsf{Ld}}(N)\right\}.$$

As a player $i \in [n]$ sets z^i as a deterministic function of x^i and π^i at Line 3, we also directly get the following corollary.

▶ **Corollary 9.** *For all $t \in [T']$, all $i \in [n]$, and all instantiations N of* N*,*

$$z_t^i(N) \in \left\{\perp, M^i_{|\pi_t^{\mathsf{Ld}}(N)|+1}\left(x^i, \pi_t^{\mathsf{Ld}}(N)\right)\right\}.$$

Likewise, we can also analyse the behaviour of Algorithm 2, during the two calls at Line 4 and Line 7, to see the way the noise can affect the executions of this algorithm.

▶ **Lemma 10.** *For all $t \in [T']$, all $i \in [n]$, and all instantiations N of* N*,*

$$w_t^i(N) \in \left\{\perp_C, \texttt{combine-CD}\left(\perp, z_t^2(N), \ldots, z_t^n(N)\right)\right\}.$$

▶ **Lemma 11.** *For $t \in [T']$ and any instantiation N of* N*, we have:*

$$e_t^{\mathsf{Ld}}(N) = \perp_S \implies \texttt{combine-CD}\left(\perp, e_t^2(N), e_t^3(N), \cdots, e_t^n(N)\right) = \perp_S.$$

We also show some properties of the symbol s^{Ld}, and how it relates to the transcript that players maintain.

▶ **Lemma 12.** *For all $t \in [T']$ and any instantiation N of* N *such that $e_t^{\mathsf{Ld}}(N) = \perp_S$ and $w_t^{\mathsf{Ld}}(N) = \texttt{combine-CD}\left(\perp, z_t^2(N), \ldots, z_t^n(N)\right)$, we have*

$$s_t^{\mathsf{Ld}}(N) = \texttt{combine-CD}\left(z_t^{\mathsf{Ld}}(N), z_t^2(N), \ldots, z_t^n(N)\right).$$

We also analyse the behaviour of Algorithm 3, and in particular how π and ρ behave in that algorithm.

▶ **Lemma 13.** *For all $t \in [T']$ and all instantiations N of* N*,*

$$\rho_t^{\mathsf{Ld}}(N) = s_1^{\mathsf{Ld}}(N) \| \cdots \| s_{t-1}^{\mathsf{Ld}}(N).$$

▶ **Lemma 14.** *For all $t \in [T']$, all $i \in [n]$, and all instantiations N of* N*,*

- *If $s_t^{\mathsf{Ld}}(N) \neq \mathsf{R}$, then*

$$\pi_{t+1}^{\mathsf{Ld}}(N) = \pi_t^{\mathsf{Ld}}(N) \| s_t^{\mathsf{Ld}}(N).$$

- *If $s_t^{\mathsf{Ld}}(N) = \mathsf{R}$, then*

$$\pi_{t+1}^{\mathsf{Ld}}(N) = \left(\pi_t^{\mathsf{Ld}}(N)\right)_{<|\pi_t^{\mathsf{Ld}}(N)|}.$$

B.2 Bad Events

B.2.1 Noise Events

Next, we define and analyze some events based on the variable N.

The event $\mathcal{E}^{\mathsf{wo}}_{t,r}$

For $t \in [T']$, $r \in [R']$, the event $\mathcal{E}^{\mathsf{wo}}_{t,r}$ occurs if the communication in round r in iteration t is erased for a significant fraction of the players (it is "wiped out"). Formally, we have:

$$\mathcal{E}^{\mathsf{wo}}_{t,r} \coloneqq \left(\sum_{i \in [n]} \mathsf{N}_{t,r,i} \geq \frac{n}{10} \right). \tag{4}$$

The event $\mathcal{E}^{\mathsf{dc}}_{t,i}$

For $t \in [T']$, $i \in [n]$, the event $\mathcal{E}^{\mathsf{dc}}_{t,i}$ occurs if the communication in the first execution of detect-collisions, i.e., at least one of rounds 1 and 2, in iteration t is erased for player i. Formally, we have:

$$\mathcal{E}^{\mathsf{dc}}_{t,i} \coloneqq (\exists r \in [2] : \mathsf{N}_{t,r,i} = 1). \tag{5}$$

The event $\mathcal{E}^{\mathsf{or}}_{t}$

For $t \in [T']$, we define the event $\mathcal{E}^{\mathsf{or}}_{t}$ to occur if the communication in the second execution of detect-collisions (which effectively computes a logical OR of the e^i's), i.e., in at least one of rounds 7 and 8 in iteration t is erased for the leader. Formally, we have:

$$\mathcal{E}^{\mathsf{or}}_{t} \coloneqq (\exists r \in \{7, 8\} : \mathsf{N}_{t,r,\mathsf{Ld}} = 1). \tag{6}$$

The event $\mathcal{E}^{\mathsf{tc}}_{t',t,i}$

For $0 \leq t' < t \leq T'$ and $i \in [n]$, define the following event concerning the rounds 9 to R' in each iteration, i.e., the rounds where the leader broadcasts on the tree code:

$$\mathcal{E}^{\mathsf{tc}}_{t',t,i} \coloneqq \left(\sum_{s=t'+1}^{t} \sum_{r=9}^{R'} \mathsf{N}_{s,r,i} \geq \frac{2R_{\mathsf{TC}}}{5} \cdot (t - t') \right). \tag{7}$$

B.2.2 Bad Iterations

We now define sets of "bad" iterations for a given execution. Intuitively, these are iterations where our protocol does not make progress. For an instantiation N of N, we have:

$$\begin{aligned} \mathcal{B}^{\mathsf{wo}}(N) &= \left\{ t \in [T'] \mid \exists r \in [R'] : N \in \mathcal{E}^{\mathsf{wo}}_{t,r} \right\}. \\ \mathcal{B}^{\mathsf{dc}}(N) &= \left\{ t \in [T'] \mid N \in \mathcal{E}^{\mathsf{dc}}_{t,\mathsf{Ld}} \right\}. \\ \mathcal{B}^{\mathsf{or}}(N) &= \{ t \in [T'] \mid N \in \mathcal{E}^{\mathsf{or}}_{t} \}. \end{aligned} \tag{8}$$

▶ **Lemma 15.** *It holds that:*

1. $\Pr(\mathcal{B}^{\mathsf{wo}}(\mathsf{N}) \neq \emptyset) \leq 2.25^{-n}$.
2. $\Pr\left(\left| \mathcal{B}^{\mathsf{dc}}(\mathsf{N}) \cup \mathcal{B}^{\mathsf{or}}(\mathsf{N}) \right| \geq \frac{T'}{50} \right) \leq \mathrm{e}^{-\frac{T'}{100}}$.

We note that our assumption that $T \leq 2^n$ is only used in Item 1 of Lemma 15.

B.3 Bad Intervals

B.3.1 Critical Players

We now define and show results about players "critical" to the protocol, i.e., those needed to make sure we make progress in our simulation. For a set S of integers and an integer k define $S_{\leq k}$ to be the set consisting of the k smallest elements of S. If $|S| \leq k$, we define $S_{\leq k} = S$. For a transcript $\pi \in \Gamma^*_{cs}$, we define the set $\mathcal{S}(\pi)$ to be the set of all non-leader players who would broadcast in the noiseless protocol when their received transcript is π. Formally,

$$\mathcal{S}(\pi) = \left\{ i \in \mathcal{P} \mid M^i_{|\pi|+1}(x^i, \pi) \neq \perp \right\}.$$

▶ **Definition 16** (Critical Players). *For $\pi \in \Gamma^*_{cs}$, we define the set of players that are π-*critical *as* $\mathsf{Crit}(\pi) = \mathcal{S}(\mathsf{LCP}(\pi, \Pi))_{\leq 2}$.

We note that this definition is made for analysis purposes and no single player can necessarily compute the set $\mathsf{Crit}(\cdot)$.

B.3.2 Bad Intervals

Next, we define the set of possible augmented transcripts and bad intervals.

▶ **Definition 17.** *For $0 \leq t' \leq t \leq T'$ and an instantiation $N_{\leq t'}$ of $\mathsf{N}_{\leq t'}$, define the set:*

$$\mathsf{Aug}_t(N_{\leq t'}) = \left\{ \pi \in \Gamma^*_{cs} \mid \exists N_{(t',t]} : \pi^{\mathsf{Ld}}_t(N_{\leq t}) = \pi \right\}.$$

▶ **Definition 18.** *Let N be an instantiation of N. We define $\mathcal{B}^\dagger(N)$ to be the set of all intervals $(t', t]$ satisfying $0 \leq t' < t \leq T'$ for which there exists $\pi \in \mathsf{Aug}_t(N_{\leq t'})$ and $i \in \mathsf{Crit}(\pi)$ such that $\mathcal{E}^{\mathsf{tc}}_{t',t,i}$ occurs when $\mathsf{N} = N$. We also define:*

$$\mathcal{B}(N) = \bigcup_{(t',t] \in \mathcal{B}^\dagger(N)} (t', t].$$

▶ **Lemma 19.** *It holds that:*

$$\Pr\left(|\mathcal{B}(\mathsf{N})| \geq \frac{T'}{50} \right) \leq 10^{-\frac{T'}{50}}.$$

To finish this subsection, we show that $\mathcal{B}(\cdot)$ has all the iterations where a critical player fails to decode the tree code.

▶ **Lemma 20.** *For any instantiation N of N and all $t \notin \mathcal{B}(N)$, for all $i \in \mathsf{Crit}(\pi^{\mathsf{Ld}}_t(N))$, we have $\pi^i_t(N) = \pi^{\mathsf{Ld}}_t(N)$.*

B.4 A Potential Function

We now define the potential function that we shall use in the analysis. For $t \in \{0\} \cup [T']$ and an instantiation N of N, we define:

$$\Phi_t(N) = 2 \cdot \left| \mathsf{LCP}\left(\pi^{\mathsf{Ld}}_{t+1}(N), \Pi\right) \right| - \left| \pi^{\mathsf{Ld}}_{t+1}(N) \right|. \tag{9}$$

Our definition clearly implies $\Phi_0(N) = 0$ and $\Phi_t(N) \leq \left| \mathsf{LCP}\left(\pi^{\mathsf{Ld}}_{t+1}(N), \Pi\right) \right|$ for all N. Moreover, as either one symbol is appended to or removed from the end of π^{Ld} in every iteration, we have that $\Phi_t(N) \geq \Phi_{t-1}(N) - 1$ for all N and $t \in [T']$. In Lemma 25 we will now show that if t is not in one of the bad sets defined above, then the potential increases by at least 1. But first, we state some helpful lemmas.

▶ **Lemma 21.** *For any instantiation N of N and any $t \notin \mathcal{B}^{\mathsf{wo}}(N)$, we have:*

$$s_t^{\mathsf{Ld}}(N) \in \left\{\mathsf{R}, \texttt{combine-CD}\left(z_t^{\mathsf{Ld}}(N), z_t^2(N), \ldots, z_t^n(N)\right)\right\}$$

For $i \in [n]$, define the variable $\hat{r}^i$ to be the value of r output by Algorithm 3, when run by player i, with Line 15 replaced[12] by $\rho \leftarrow \mathsf{D\text{-}TC}\left(\tau^{\mathsf{Ld}}\right)$. This value is only used for analysis purposes and cannot be computed by the player during the execution of the protocol (as they may not know τ^{Ld}). We now claim several useful properties of $\hat{r}_t^i$, and how it relates to r_t^i.

▶ **Lemma 22.** *For $t \in [T']$ and any instantiation N of N such that $\mathcal{B}^{\mathsf{wo}}(N) = \emptyset$, we have:*

$$\left(\nexists j \in \left[\left|\pi_t^{\mathsf{Ld}}(N)\right|\right] : \left(\pi_t^{\mathsf{Ld}}(N)\right)_j \neq \Pi_j\right) \implies \left(\nexists i \in \mathcal{P} : \hat{r}_t^i(N) = \mathsf{true}\right).$$

We also prove a modified converse version of the previous lemma.

▶ **Lemma 23.** *For $t \in [T']$ and any instantiation N of N such that $\mathcal{B}^{\mathsf{wo}}(N) = \emptyset$, we have:*

$$\left(\exists j \in \left[\left|\pi_t^{\mathsf{Ld}}(N)\right|\right] : \left(\pi_t^{\mathsf{Ld}}(N)\right)_j \neq \Pi_j\right) \implies \left(\exists i \in \mathsf{Crit}(\pi_t^{\mathsf{Ld}}(N)) : \hat{r}_t^i(N) = \mathsf{true}\right),$$

▶ **Lemma 24.** *For $t \in [T']$, $i \in [n]$ and any instantiation N of N, $r_t^i(N) \in \left\{\hat{r}_t^i(N), \mathsf{false}\right\}$. Furthermore, if $\pi_t^i(N) = \pi_t^{\mathsf{Ld}}(N)$, then $r_t^i(N) = \hat{r}_t^i(N)$.*

▶ **Lemma 25.** *For $t \in [T']$ and any instantiation N of N such that $\mathcal{B}^{\mathsf{wo}}(N) = \emptyset$, we have:*

$$t \notin \mathcal{B}^{\mathsf{dc}}(N) \cup \mathcal{B}^{\mathsf{or}}(N) \cup \mathcal{B}(N) \implies \Phi_t(N) \geq \Phi_{t-1}(N) + 1.$$

B.5 Finishing the proof of Theorem 2

We are now ready to finish the proof of Theorem 2.

Proof of Theorem 2. Let $C \geq 100 R_{\mathsf{TC}}$. Fix ϵ, n and Γ as in the statement of the theorem. We claim that the algorithm provided in Algorithm 1 satisfies all the properties claimed by the theorem. It can be observed that Algorithm 1 takes at most CT rounds of communication, so it just suffices to just show that $\Pr\left(\Pi'^{\mathsf{Ld}}(X) \neq \Pi(X)\right) \leq 2^{-\min(n,T)}$.

By Lemmas 15 and 19 and a union bound, we get that an instantiation N of N satisfies $\left|\mathcal{B}^{\mathsf{dc}}(N) \cup \mathcal{B}^{\mathsf{or}}(N) \cup \mathcal{B}(N)\right| \leq \frac{T'}{25}$ and $\mathcal{B}^{\mathsf{wo}}(N) = \emptyset$ except with probability at most

$$10^{-\frac{1}{50}T'} + \mathrm{e}^{-\frac{1}{100}T'} + 2.25^{-n} \leq 2^{-\min\left(n, \frac{1}{100}T'\right)} \leq 2^{-\min(n,T)}.$$

Lemma 25 then states that for all such N, for all $t \notin \mathcal{B}^{\mathsf{dc}}(N) \cup \mathcal{B}^{\mathsf{or}}(N) \cup \mathcal{B}(N)$, $\Phi_t(N) \geq \Phi_{t-1}(N) + 1$. At the same time, we recall that Equation (9) also gives that for all $t \in [T']$, $\Phi_t(N) \geq \Phi_{t-1}(N) - 1$. Thus, we see that

$$\Phi_{T'}(N) \geq \left(T' - \frac{T'}{25}\right) - \frac{T'}{25} \geq \frac{9}{10}T' \geq T.$$

Furthermore, we consult Equation (9) to get that

$$\left|\mathsf{LCP}\left(\pi_{T'+1}^{\mathsf{Ld}}(N), \Pi\right)\right| \geq \Phi_{T'}(N) \geq T,$$

which implies that $\left(\pi_{T'+1}^{\mathsf{Ld}}(N)\right)_{\leq T} = \Pi_{\leq T}$. so the leader's output at Line 10 is equal to $\Pi_{\leq T}$.

As this happens except with probability at most $2^{-\min(n,T)}$, this concludes the proof. ◀

[12] We stress that Line 25 still uses τ^i and *not* τ^{Ld}.

On the Mixing Time of Glauber Dynamics for the Hard-Core and Related Models on $G(n, d/n)$

Charilaos Efthymiou
Computer Science, University of Warwick, Coventry, UK

Weiming Feng
School of Informatics, University of Edinburgh, Edinburgh, UK

Abstract

We study the single-site Glauber dynamics for the fugacity λ, Hard-Core model on the random graph $G(n, d/n)$. We show that for the typical instances of the random graph $G(n, d/n)$ and for fugacity $\lambda < \frac{d^d}{(d-1)^{d+1}}$, the mixing time of Glauber dynamics is $n^{1+O(1/\log\log n)}$.

Our result improves on the recent elegant algorithm in [Bezáková, Galanis, Goldberg and Štefankovič; ICALP'22]. The algorithm there is an MCMC-based sampling algorithm, but it is not the Glauber dynamics. Our algorithm here is *simpler*, as we use the classic Glauber dynamics. Furthermore, the bounds on mixing time we prove are smaller than those in Bezáková et al. paper, hence our algorithm is also *faster*.

The main challenge in our proof is handling vertices with unbounded degrees. We provide stronger results with regard the spectral independence via branching values and show that the our Gibbs distributions satisfy the approximate tensorisation of the entropy. We conjecture that the bounds we have here are optimal for $G(n, d/n)$.

As corollary of our analysis for the Hard-Core model, we also get bounds on the mixing time of the Glauber dynamics for the Monomer-Dimer model on $G(n, d/n)$. The bounds we get for this model are slightly better than those we have for the Hard-Core model

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Randomness, geometry and discrete structures; Mathematics of computing → Discrete mathematics

Keywords and phrases spin-system, spin-glass, sparse random (hyper)graph, approximate sampling, efficient algorithm

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.54

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2302.06172`

Funding *Charilaos Efthymiou*: EPSRC New Investigator Award (grant no. EP/V050842/1) and Centre of Discrete Mathematics and Applications (DIMAP), The University of Warwick.
Weiming Feng: European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 947778).

Acknowledgements Weiming Feng would like to thank Heng Guo for the helpful discussions.

1 Introduction

The Hard-Core model and the related problem of the geometry of independent sets on the sparse random graph $G(n, d/n)$ is a fundamental area of study in discrete mathematics [17, 11], in computer science they are studied in the context of the random *Constraint Satisfaction Problems* [10, 20], while in statistical physics they are studied as instances of *disordered systems*. Using the so-called *Cavity method* [25, 2], physicists make some impressive

predictions about the independent sets of $G(n, d/n)$, such as higher order replica symmetry breaking etc. Physicists' predictions are (typically) mathematically non-rigorous. Most of these predictions about independent sets still remain open as basic natural objects in the study such as the partition function, or the free energy are extremely challenging to analyse.

The Hard-Core model with fugacity $\lambda > 0$, is a distribution over the *independent sets* of an underlying graph G such that every independent set σ is assigned probability measure $\mu(\sigma)$ which is proportional to $\lambda^{|\sigma|}$, where $|\sigma|$ is the cardinality of σ. Here, we consider the case where the underlying graph is a typical instance of the sparse random graph $G(n, d/n)$. This is the random graph on n vertices, while each edge appears independently with probability $p = d/n$. The quantity $d > 0$ corresponds to the *expected degree*. For us here the expected degree is a bounded constant, i.e., we have $d = \Theta(1)$, hence the graph is sparse.

Our focus is on approximate sampling from the aforementioned distribution using *Glauber dynamics*. This is a classic, very popular, algorithm for approximate sampling. The popularity of this process, mainly, is due to its simplicity and the strong approximation guarantees that provides. The efficiency of Glauber dynamics for sampling is studied by means of the *mixing time*.

Recently, there has been an "explosion" of results about the mixing time of Glauber dynamics for *worst-case* instances the problem, e.g. [1, 9, 8, 12]. Combined with the earlier hardness results in [29, 30, 19] one could claim that for worst-case instances the behaviour of Glauber dynamics for the Hard-Core model, but also the related approximate sampling-counting problem, is well understood. Specifically, for the graphs of maximum degree Δ, Glauber dynamics exhibits $O(n \log n)$ mixing time for any fugacity $\lambda < (\Delta-1)^{\Delta-1}/(\Delta-2)^{\Delta}$, while the hardness results support that this region of λ is best possible.

The aforementioned upper bound on λ coincides with the *critical point* for the uniqueness/non-uniqueness phase transition of the Hard-Core model on the infinite Δ-regular tree [24]. At this point in the discussion, perhaps, it is important to note the dependency of the critical point on the *maximum degree*. This is the point where the situation with the random graph $G(n, d/n)$ differentiates from the worst case one.

For $G(n, d/n)$ and for the range of the expected degree d we consider here, typically, almost all of the vertices in the graph, e.g., say 99%, are of degree very close to d. On the other hand, the maximum degree of $G(n, d/n)$ is as large as $\Theta(\frac{\log n}{\log \log n})$, i.e., it is *unbounded*. In light of this observation, it is natural to expect that the Glauber dynamics on the Hard-Core model mixes fast for values of the fugacity that depend on the *expected degree*, rather the maximum degree. Note that, this implies to use Glauber dynamics to sample from the Hard-Core model with fugacity λ taking *much larger* values than what the worst-case bound implies.

For $d > 1$, let $\lambda_c(d) = \frac{d^d}{(d-1)^{(d+1)}}$. One of the main result in our paper is as follows: we show that for any $d > 1$ and for typical instances of $G(n, d/n)$, the Glauber dynamics on the Hard-Core with any fugacity $\lambda < \lambda_c(d)$, exhibits mixing time which is $n^{1+\frac{C}{\log \log n}} = n^{1+o(1)}$, for some absolute constant $C > 0$ which depends only on λ and d.

It is our *conjecture* that the bound on the mixing time for the hard-core is tight. Furthermore, following intuitions from [10], as well as from *statistical physics* predictions in [2], it is our *conjecture* that the bound $\lambda_c(d)$ on the fugacity λ is also tight, in the following sense: for $\lambda > \lambda_c(d)$ it is not precluded that there is a region where efficient approximate sampling is possible, however, the approximation guarantees are *weaker* than those we have here.

Our result improves on the elegant sampling algorithm that was proposed recently in [3] for the same distribution, i.e., the Hard-Core model on $G(n, d/n)$. That algorithm, similarly to the one we consider here, relies on the Markov Chain Monte Carlo method. The authors

use Spectral Independence [1, 9] to show that the underlying Markov chain exhibits mixing time which is $O\left(n^{1+\theta}\right)$ for any $\lambda < \lambda_c(d)$ and arbitrary small consant $\theta > 0$. The idea that underlies the algorithm in [3] is reminiscent of the *variable marking* technique that was introduced in [26] for approximate counting with the Lovász Local Lemma, and was further exploited in [14, 16, 21, 18]. Here, we use a different, more straightforward, approach and analyse directly the Glauber dynamics.

Note that both algorithms, i.e., here and in [3], allow for the same range for the fugacity λ. On the other hand, the algorithm we study here is the (much simpler) Glauber dynamics, while the running time guarantees we obtain here are asymptotically better.

Previous works in the area, i.e., even before [3], in order to prove their results and avoid the use of maximum degree, have been focusing on various parameters of $G(n, d/n)$ such as the expected degree [13], or the connective constant [28]. Which, as it turns out are not that different with each other. Here, we utilise the notion of *branching value*, which is somehow related to the previous ones.

The notion of the branching value as well as its use for establishing Spectral Independence was introduced in [3]. Unfortunately, the result there were not sufficiently strong to imply rapid mixing of Glauber dynamics. Their analytic tools for Spectral Independence (and others) seems to not be able to handle all that well vertices with unbounded degree. Here we derive stronger results for Spectral independence than those in [3] in the sense that they are more *general* and more *accurate*. Specifically, in our analysis we are able to accommodate vertices of *all degrees*, while we use a more elaborate matrix norm to establish spectral independence, reminiscent of those introduced in [12]. Furthermore, we utilise results from [8] that allow us deal with the unbounded degrees of the graph in order to establish our rapid mixing results.

2 Results

Consider the fixed graph $G = (V, E)$ on n vertices. Given the parameter $\lambda > 0$, which we call *fugacity*, we define the *Hard-Core* model $\mu = \mu_{G,\lambda}$ to be a distribution on the independent sets of the graph G, Specifically, every independent set σ is assigned probability measure $\mu(\sigma)$ defined by

$$\mu(\sigma) \propto \lambda^{|\sigma|} \ , \tag{1}$$

where $|\sigma|$ is equal to the size of the independent set σ.

We use $\{\pm 1\}^V$ to encode the configurations of the Hard-Core model, i.e., the independent sets of G. Particularly, the assignment $+1$ implies that the vertex is in the independent set, while -1 implies the opposite. We often use physics' terminology where the vertices with assignment $+1$ are called "occupied", whereas the vertices with -1 are "unoccupied".

We use the discrete time, (single site) *Glauber dynamics* to approximately sample from the aforementioned distributions. Glauber dynamics is a Markov chain with state space the support of the distribution μ. Typically, we assume that the chain starts from an arbitrary configuration $X_0 \in \{\pm 1\}^V$. For $t \geq 0$, the transition from the state X_t to X_{t+1} is according to the following steps:

1. Choose uniformly at random a vertex v.
2. For every vertex w different than v, set $X_{t+1}(w) = X_t(w)$.
3. Set $X_{t+1}(v)$ according to the marginal of μ at v, conditional on the neighbours of v having the configuration specified by X_{t+1}.

It is standard that when a Markov chain satisfies a set of technical conditions called *ergodicity*, then it converges to a unique stationary distribution. For the cases we consider here, Glauber dynamics is trivially ergodic, while the stationary distribution is the corresponding Hard-Core model μ.

Let P be the transition matrix of an ergodic Markov chain $\{X_t\}$ with a finite state space Ω and equilibrium distribution μ. For $t \geq 0$ and $\sigma \in \Omega$, let $P^t(\sigma, \cdot)$ denote the distribution of X_t when the initial state of the chain satisfies $X_0 = \sigma$. The *mixing time* of the Markov chain $\{X_t\}_{t\geq 0}$ is defined by

$$T_{\text{mix}} = \max_{\sigma\in\Omega} \min\left\{t > 0 \mid \|P^t(\sigma,\cdot) - \mu\|_{\text{TV}} \leq \frac{1}{2\text{e}}\right\} \ .$$

Our focus is on the mixing time of Glauber dynamics for the Hard-Core model for the case where the underlying graph is a typical instance of $G(n, d/n)$, where the expected degree $d > 0$ is a assumed to be a fixed number.

2.1 Mixing Time for Hard-Core Model

For $z > 1$, we let the function $\lambda_c(z) = \frac{z^z}{(z-1)^{(z+1)}}$. It is a well-known result from [24] that the uniqueness region of the Hard-Core model on the k-ary tree, where $k \geq 2$, holds for any λ such that

$$\lambda < \lambda_c(k) \ .$$

The following theorem is the main result of this work.

▶ **Theorem 1.** *For fixed $d > 1$ and any $\lambda < \lambda_c(d)$, there is a constant $C > 0$ such that the following is true:*

Let $\mu_{\boldsymbol{G}}$ be the Hard-Core model with fugacity λ on the graph $\boldsymbol{G} \sim G(n, d/n)$. With probability $1 - o(1)$ over the instances of $\boldsymbol{G}$, Glauber dynamics on $\mu_{\boldsymbol{G}}$ exhibits mixing time

$$T_{\text{mix}} \leq n^{\left(1+\frac{C}{\log\log n}\right)} \ .$$

2.2 Extensions to Monomer-Dimer Model

Utilising the techniques we develop in order to prove Theorem 1, we get mixing time bounds for the Glauber dynamics on the Monomer-Dimer model on $G(n, d/n)$.

Given a fixed graph $G = (V, E)$ and a parameter $\lambda > 0$, which we call *edge weight*, we define the Monomer-Dimer model $\mu = \mu_{G,\lambda}$ to be a distribution on the *matchings* of the graph G such that every matching σ is assigned probability measure $\mu(\sigma)$ defined by

$$\mu(\sigma) \propto \lambda^{|\sigma|} \ , \tag{2}$$

where $|\sigma|$ is equal to the number of edges in the matching σ.

Note that the Hard-Core model considers configurations on the vertices of G, while the Monomer-Dimer model considers configurations on the edges. Similarly to the independent sets, we use $\{\pm 1\}^E$ to encode the matchings of G. Specifically, the assignment $+1$ on the edge e implies that the edge is in matching, while -1 implies the opposite.

For the Monomer-Dimer model the definition of Glauber dynamics $\{X_t\}_{t\geq 0}$ extends in the natural way. That is, assume that the chain starts from an arbitrary configuration $X_0 \in]\pm 1^E$. For $t \geq 0$, the transition from the state X_t to X_{t+1} is according to the following steps:

1. Choose uniformly at random an edge e.
2. For every edge f different than e, set $X_{t+1}(f) = X_t(f)$.
3. Set $X_{t+1}(e)$ according to the marginal of μ at e, conditional on the neighbours of e having the configuration specified by X_{t+1}.

We consider the case of the Monomer-Dimer distribution where the underlying graph is an instance of $G(n, d/n)$. We prove the following result.

▶ **Theorem 2.** *For fixed $d > 1$ and any $\lambda > 0$, there is a constant $C > 0$ such that the following is true:*

Let $\mu_{\boldsymbol{G}}$ be the Monomer-Dimer model with edge weight λ on the graph $\boldsymbol{G} \sim G(n, d/n)$. With probability $1 - o(1)$ over the instances of $\boldsymbol{G}$, Glauber dynamics on $\mu_{\boldsymbol{G}}$ exhibits mixing time

$$T_{\text{mix}} \leq n^{\left(1+C\sqrt{\frac{\log\log n}{\log n}}\right)} .$$

The proof of Theorem 2 can be found in the full version of this paper.

For the Monomer-Dimer model on general graphs, the best-known result is the $\tilde{O}(n^2 m)$ mixing time of the Jerrum-Sinclair chain [23], where $m = |E|$ is the number of edges. For graphs with bounded maximum degree $\Delta = O(1)$, the spectral independence technique proved the $O(n \log n)$ mixing time of Glauber dynamics [9]. However, this result cannot be applied directly to the random graph $G(n, d/n)$, because the maximum degree of a random graph is typically unbounded. For the Monomer-Dimer model on $G(n, d/n)$, [3] gave a sampling algorithm with running time $n^{1+\theta}$, where $\theta > 0$ is an arbitrarily small constant, and [22] also proved the $n^{2+o(1)}$ mixing time of Glauber dynamics in a special case $\lambda = 1$. Our result in Theorem 2 proves the $n^{1+o(1)}$ mixing time of Glauber dynamics, which improves all the previous results for the Monomer-Dimer model on the random graph $G(n, d/n)$ with constant λ. It is an open problem to improve the mixing time in Theorem 2. Moreover, for general graphs, the tight mixing time of Glauber dynamics for the Monomer-Dimer model is also a challenging open problem.

We remark that for the Monomer-Dimer model, we actually proved the $n^{1+o(1)}$ mixing time of Glauber dynamics on *all* graphs satisfying $\Delta \log^2 \Delta = o(\log^2 n)$. See the full version of this paper for a more general result.

This version of the paper focuses on the Hard-Core model, i.e., proving Theorem 1. The proofs for the Monomer-Dimer model is in the full version.

Notation

Suppose that we are given a Gibbs distribution μ on the graph $G = (V, E)$. We denote with Ω the support of μ.

Suppose that Ω is a set of configurations at the vertices of G. Then, for any $\Lambda \subseteq V$ and any $\tau \in \{\pm 1\}^{\Lambda}$, we let $\mu^{\Lambda,\tau}$ (or μ^{τ} if Λ is clear from the context) denote the distribution μ conditional on that the configuration at Λ is τ. Alternatively, we use the notation $\mu(\cdot \mid (\Lambda, \tau))$ for the same conditional distribution. We let $\Omega^{\tau} \subseteq \Omega$ be the support of $\mu^{\Lambda,\tau}$. We call τ *feasible* if Ω^{τ} is nonempty.

For any subset $S \subseteq V$, let μ_S denote the marginal of μ at S, while let Ω_S denote the support of μ_S. In a natural way, we define the conditional marginal. That is, for $\Lambda \subseteq V \setminus S$ and $\sigma \in \{\pm 1\}^{\Lambda}$, we let $\mu_S^{\Lambda,\sigma}$ (or μ_S^{σ} if Λ is clear from the context) denote the marginal at S conditional on the configuration at Λ being σ. Alternatively we use $\mu_S(\cdot \mid (\Lambda, \sigma))$ for μ_S^{σ}. We let Ω_S^{σ} denote the support of μ_S^{σ}.

All the above notation for configurations on the vertices of G can be extended naturally for configurations on the edges of the graph G. We omit presenting it, because it is very similar to the above.

2.3 Hard-Core Model – Entropy Tensorisation for Rapid Mixing

We prove Theorem 1 by exploiting the notion of *approximate tensorisation of the entropy*.

Let μ be a distribution with support $\Omega \subseteq \{\pm 1\}^V$. For any function $f : \Omega \to \mathbb{R}_{\geq 0}$, we let $\mu(f) = \sum_{x\in\Omega} \mu(x)f(x)$, i.e., $\mu(f)$ is the expected value of f with respect to μ. Define the entropy of f with respect to μ by

$$\mathrm{Ent}_\mu(f) = \mu\left(f \log \tfrac{f}{\mu(f)}\right) \ ,$$

where we use the convention that $0 \log 0 = 0$.

Let $\tau \in \Omega_{V\setminus S}$ for some $S \subset V$. Define the function $f_\tau : \Omega_S^\tau \to \mathbb{R}_{\geq 0}$ by having $f_\tau(\sigma) = f(\tau \cup \sigma)$ for all $\sigma \in \Omega_S^\tau$ [1]. Let $\mathrm{Ent}_S^\tau(f_\tau)$ denote the entropy of f_τ with respect to the conditional distribution μ_S^τ. Furthermore, we let

$$\mu(\mathrm{Ent}_S(f)) = \sum_{\tau\in\Omega_{V\setminus S}} \mu_{V\setminus S}(\tau)\mathrm{Ent}_S^\tau(f_\tau) \ ,$$

i.e., $\mu(\mathrm{Ent}_S(f))$ is the average of the entropy $\mathrm{Ent}_S^\tau(f_\tau)$ with respect to the measure $\mu_{V\setminus S}(\cdot)$. When $S = \{v\}$, i.e., the set S is a singleton, we abbreviate $\mu(\mathrm{Ent}_{\{v\}}(f))$ to $\mu(\mathrm{Ent}_v(f))$.

▶ **Definition 3** (Approximate Tensorisation of Entropy). *A distribution μ with support $\Omega \subseteq \{\pm 1\}^V$ satisfies the approximate tensorisation of entropy with constant $C > 0$ if for all $f : \Omega \to \mathbb{R}_{\geq 0}$ we have that*

$$\mathrm{Ent}_\mu(f) \leq C \cdot \sum_{v\in V} \mu\left(\mathrm{Ent}_v(f)\right) \ .$$

On can establish bounds on the mixing time of Glauber dynamics by means of the approximate tensorisation of entropy of the equilibrium distribution μ. Specifically, if μ satisfies the approximate tensorisation of entropy with constant C, then after every transition of Glauber dynamics, the Kullback–Leibler divergence[2] between the current distribution and the stationary distribution decays by a factor which is at least $(1 - C/n)$, where $n = |V|$ is the number of variables.

As far as the mixing time of Glauber dynamics is concerned, if a distribution μ satisfies the approximate tensorisation of entropy with parameter C then we have following well known relation (e.g. see [9, Fact 3.5]),

$$T_{\mathrm{mix}} \leq \left\lceil Cn\left(\log\log\frac{1}{\mu_{\min}} + \log(2) + 2\right)\right\rceil, \qquad \text{where } \mu_{\min} = \min_{x\in\Omega}\mu(x) \ . \tag{3}$$

In light of the above, Theorem 1 follows as a corollary from the following result.

---

[1] With a slight abuse of notation we use $\tau \cup \sigma$ to indicate the configuration what agrees with τ at S and with σ at $V \setminus S$.

[2] For discrete probability distributions P and Q on a discrete space $\mathcal{X}$, the Kullback–Leibler divergence is defined by $D_{\mathrm{KL}}(P||Q) = \sum_{x\in\mathcal{X}} P(x) \log \frac{P(x)}{Q(x)}$.

▶ **Theorem 4** (Hard-Core Model Tensorisation). *For any fixed $d > 1$ and any $\lambda < \lambda_c(d)$, there is a constant $A > 0$ that depends only on d and λ such that the following is true:*

Let $\mu_{\boldsymbol{G}}$ be the Hard-Core model with fugacity λ on the graph $\boldsymbol{G} \sim G(n, d/n)$. With probability $1 - o(1)$ over the instances of $\boldsymbol{G}$, $\mu_{\boldsymbol{G}}$ satisfies the approximate tensorisation of entropy with parameter $n^{A/\log\log n}$.

Proof of Theorem 1. Theorem 1 follows from Theorem 4 and (3).

Specifically, plugging the result from Theorem 4 into (3) we get the following: with probability $1 - o(1)$ over the instances of $\boldsymbol{G}$ we have that

$$\begin{aligned} T_{\text{mix}} &\leq n^{1+\frac{A}{\log\log n}} \left(\log\log \frac{1}{\mu_{\min}} + \log(2) + 2 \right) \\ &\leq n^{1+\frac{A}{\log\log n}} \left(\log\log \left(1 + \lambda + \lambda^{-1}\right)^n + \log(2) + 2 \right) \\ &= n^{1+\frac{A}{\log\log n}} \left(\log n + \log\log(1 + \lambda + \lambda^{-1}) \right) \leq n^{1+\frac{2A}{\log\log n}} \quad . \end{aligned}$$

For the second derivation, we note that for the Hard-Core distribution $\mu = \mu_{\boldsymbol{G}}$, we have that $\mu_{\min}$ is at least $\min\{1, \lambda^n\}/(1+\lambda)^n$, which implies that $\mu_{\min} \geq (1 + \lambda + \lambda^{-1})^{-n}$.

Note that Theorem 1 follows from the above, by setting $C = 2A$. ◀

3 Our Approach & Contributions

In this section we describe our approach towards establishing our results. Our focus is on the Hard-Core model.

3.1 Tensorisation and Block-Factorisation of Entropy

We establish the tensorisation of the entropy, described in Theorem 4, by exploiting the recently introduced notion of *block factorisation of entropy* in [5]. Specifically, we build on the framework introduced in [9] to relate the tensorisation and the block factorisation of the entropy.

The framework in [9] relies on the assumption that the maximum degree of the underlying graph is bounded. Otherwise, the results it implies are not strong. In our setting here, a vanilla application of this approach would not be sufficient to give the desirable bounds on the tensorisation constant due to the fact that the typical instances of $G(n, d/n)$ have unbounded maximum degree. To this end, we employ techniques from [8].

Given the graph $G = (V, E)$, and the integer $\ell \geq 0$, we let $\binom{V}{\ell}$ denote all subsets $S \subseteq V$ with $|S| = \ell$.

▶ **Definition 5** (ℓ-block Factorisation of Entropy). *Let μ be a distribution over $\{\pm 1\}^V$ and $1 \leq \ell \leq |V| = n$ be an integer. The distribution μ satisfies the ℓ block factorisation of entropy with parameter C if for all $f : \Omega \to \mathbb{R}_{\geq 0}$ we have that*

$$\text{Ent}_\mu(f) \leq \frac{C}{\binom{n}{\ell}} \sum_{S \in \binom{V}{\ell}} \mu\left(\text{Ent}_S(f)\right) \quad . \tag{4}$$

The notion of the ℓ block factorisation of entropy generalises that of the approximate tensorisation of entropy. Specifically, a distribution that satisfies the $\ell = 1$ block factorisation of entropy with parameter C, also satisfies the approximate tensorisation of entropy with parameter C/n.

As far as the Hard-Core model on $G(n, d/n)$ is concerned, we show the following theorem via the spectral independence technique, which is one of the main technical results in our paper.

▶ **Theorem 6.** *For fixed $d > 1$ and any $0 < \lambda < \lambda_c(d)$, consider $\boldsymbol{G} \sim G(n, d/n)$ and let $\mu_{\boldsymbol{G}}$ be the Hard-Core model on $\boldsymbol{G}$ with fugacity λ. With probability $1 - o(1)$ over the instances of $\boldsymbol{G}$ the following is true: There is a constant $K = K(d, \lambda) > 0$, such that for*

$$\frac{1}{\alpha} = K \frac{\log n}{\log \log n} \ ,$$

for any $1/\alpha \leq \ell < n$, $\mu_{\boldsymbol{G}}$ satisfies the ℓ-block factorisation of entropy with parameter $C = (\frac{en}{\ell})^{1+1/\alpha}$.

Let us have a high level overview of how we use the ℓ-block factorisation and particularly Theorem 6 to establish our entropy tensorisation result in Theorem 4.

Note that Theorem 6 essentially implies the following: Suppose that $G = (V, E)$ is a *typical instance* of $G(n, d/n)$. Then, the Hard-Core model μ on G, with fugacity $\lambda < \lambda_c(d)$, is such that for any $f : \Omega \to \mathbb{R}_{>0}$ we have

$$\mathrm{Ent}_{\mu}(f) \leq \left(\frac{\mathrm{e}}{\theta}\right)^{1+1/\alpha} \frac{1}{\binom{n}{\ell}} \sum_{S \in \binom{V}{\ell}} \mu\left(\mathrm{Ent}_S(f)\right) \ , \tag{5}$$

where $\ell = \lceil \theta n \rceil$ and $\theta \in (0, 1)$ is a constant satisfying $\lceil \theta n \rceil \geq 1/\alpha = \Omega(\log n / \log \log n)$.

Let $G[S]$ be the subgraph of G that is induced by the vertices in the set S. On the RHS of (5), the entropy is evaluated with respect to conditional distributions μ_S^τ, which is the Hard-Core model on the subgraph $G[S]$ given the boundary condition τ on $V \setminus S$.

We let $C(S)$ denote the set of connected components in $G[S]$. With a slight abuse of notation, we use $U \in C(S)$ to denote the set of vertices in the component U, as well. It is not hard to see that the Hard-Core model μ_S^τ, for $\tau \in \Omega_{V \setminus S}$, factorises as a product distribution over Gibbs marginals at the components $U \in C(S)$, i.e.,

$$\mu_S^\tau = \bigotimes_{U \in C(S)} \mu_U^\tau \ .$$

We use the following result for the factorisation of entropy on product distributions [6, 4, 9].

▶ **Lemma 7** ([9, Lemma 4.1])**.** *For any $S \subseteq V$, any $\tau \in \Omega_{V \setminus S}$, any $f : \Omega_S^\tau \to \mathbb{R}_{\geq 0}$,*

$$\mathrm{Ent}_S^\tau(f) \leq \sum_{U \in C(S)} \mu_S^\tau[\mathrm{Ent}_U(f)] \ .$$

Combining Lemma 7 and (5) we get that

$$\mathrm{Ent}_{\mu}(f) \leq \left(\frac{\mathrm{e}}{\theta}\right)^{1+1/\alpha} \mathbb{E}_{\boldsymbol{S} \sim \binom{V}{\ell}} \left[\textstyle\sum_{U \in C(\boldsymbol{S})} \mu\left(\mathrm{Ent}_U(f)\right)\right] \ , \tag{6}$$

where $\boldsymbol{S} \sim \binom{V}{\ell}$ denotes that $\boldsymbol{S}$ is a uniformly random element from $\binom{V}{\ell}$.

The above step allows us to reduce the proof of approximate tensorisation to that of the components in $C(\boldsymbol{S})$. We choose the parameter $\ell = \lceil \theta n \rceil$ so that the connected components in $C(\boldsymbol{S})$ are typically small.

In light of the above, Theorem 4 follows by establishing two results: The first one is to derive a bound on the constant of the approximate tensorisation of entropy for the components of size k in $C(\boldsymbol{S})$, for each $k > 0$. The second result is to derive tail bounds on the size of the components in $C(\boldsymbol{S})$ for $\boldsymbol{S} \sim \binom{V}{\ell}$. Since the components are small with high probability, the following crude bound on the approximate tensorisation of entropy is enough for our analysis.

▶ **Lemma 8.** *For any fixed $d > 0$, for any $\lambda < \lambda_c(d)$, consider $\boldsymbol{G} \sim G(n, d/n)$. With probability $1 - o(1)$ over the instances of $\boldsymbol{G}$, the following is true:*

For any $k \geq 1$ and $H \subseteq V$ such that $|H| = k$, the Hard-Core model μ_H on $\boldsymbol{G}[H]$ with fugacity λ satisfies the approximate tensorisation of entropy with constant

$$\mathrm{AT}(k) \leq \min\left\{2k^2\left(1+\lambda+1/\lambda\right)^{2k+2},\ 3\log\left(1+\lambda+1/\lambda\right)\cdot\left((1+\lambda)k\right)^{2+2\eta}\right\}\ , \tag{7}$$

where $\eta = B(\log n)^{1/r}$, while $B = B(d,\lambda)$ and $r = r(d) \in (1,2)$ are constants that depend on d, λ.

As far as size of the components in $C(\boldsymbol{S})$ is concerned, we use the following result from [3].

▶ **Lemma 9** ([3])**.** *Let $d > 1$ be a constant. There is a constant $L = L(d)$ such that the following holds with probability at least $1 - o(1)$ over the $\boldsymbol{G} \sim G(n, d/n)$. Let $\boldsymbol{S} \sim \binom{V}{\ell}$, while let $C_v \subseteq \boldsymbol{S}$ be the set of vertices that are in the same component as vertex v in $\boldsymbol{G}[\boldsymbol{S}]$. For any integer $k \geq \log n$, it holds that*

$$\Pr[|C_v| = k] \leq (2\mathrm{e})^{\mathrm{e}Lk}\left(\frac{2\ell}{n}\right)^k \leq (2\mathrm{e})^{\mathrm{e}Lk}\,(2\theta)^k\ .$$

Theorem 4 follows by combining Theorem 6, with Lemmas 9 and 8. For a full proof of Theorem 4, see Section 5.

3.2 Spectral Independence with Branching Values

An important component in our proof of Theorem 6 is to establish Spectral Independence bounds for the Hard-Core model on typical instances of $G(n, d/n)$.

For worst-case graph instances (i.e., non random), typically, we establish Spectral Independence for a region of the parameters of the Gibbs distribution which is expressed in terms of the maximum degree Δ of the underlying graph G. As far as $G(n, d/n)$ is concerned, the maximum degree does not seem to be the appropriate graph parameter to consider for this problem.

Here, we utilise the notion of branching value. The notion of the branching value as well as its use for establishing Spectral Independence was introduced in [3]. Unfortunately, the result there were not sufficiently strong to imply rapid mixing of Glauber dynamics. Here we derive stronger results for Spectral independence than those in [3] in the sense that they are more *general* and more *accurate*. Specifically, in our analysis we are able to accommodate vertices of *all degrees*, while we use a more elaborate matrix norm to establish spectral independence, reminiscent of those introduced in [12]. Furthermore, we utilise results from [8] that allow us deal with the unbounded degrees of the graph in order to establish our rapid mixing results.

Before getting to further details in our discussion, let us first introduce some basic notions. We start with the *pairwise influence matrix* $\mathcal{I}_G^{\Lambda,\tau}$ and the related notion of Spectral Independence. These notions were first introduced in [1]. In this paper, we use the absolute version introduced in [15].

Consider a *fixed* graph $G = (V, E)$. Assume that we are given a Gibbs distribution μ on the configuration space $\{\pm 1\}^V$. We define the pairwise influence matrix $\mathcal{I}_G^{\Lambda,\tau}$ as follows: for a set of vertices $\Lambda \subset V$ and a configuration τ at Λ, the matrix $\mathcal{I}_G^{\Lambda,\tau}$ is indexed by the vertices in $V \setminus \Lambda$, while for any two vertices, different with each other $u, w \in V \setminus \Lambda$, if w can take both values ± 1 given τ, we have that

$$\mathcal{I}_G^{\Lambda,\tau}(w,u) = ||\mu_u(\cdot \mid (\Lambda,\tau),(\{w\},+)) - \mu_u(\cdot \mid (\Lambda,\tau),(\{w\},-))||_{\text{TV}} \ ; \tag{8}$$

if w can only take one value in ± 1 given τ, we have $\mathcal{I}_G^{\Lambda,\tau}(w,u) = 0$. Also, we have that $\mathcal{I}_G^{\Lambda,\tau}(w,w) = 0$ for all $w \in V \setminus \Lambda$. That is, the diagonal of $\mathcal{I}_G^{\Lambda,\tau}$ is always zero.

Recall that, above, $\mu_u(\cdot \mid (\Lambda,\tau),(\{w\},1))$ is the Gibbs marginal that vertex u, conditional that the configuration at Λ is τ and the configuration at w is 1. We have the analogous for $\mu_u(\cdot \mid (\Lambda,\tau),(\{w\},-1))$.

▶ **Definition 10** (Spectral Independence). *For a real number $\eta > 0$, the Gibbs distribution μ_G on $G = (V,E)$ is η-spectrally independent, if for every $0 \le k \le |V| - 2$, $\Lambda \subseteq V$ of size k and $\tau \in \{\pm 1\}^\Lambda$ the spectral radius of $\mathcal{I}_G^{\Lambda,\tau}$ satisfies that $\rho(\mathcal{I}_G^{\Lambda,\tau}) \le \eta$.*

We bound the spectral radius of $\mathcal{I}_G^{\Lambda,\tau}$ by means of matrix norms. Specifically, we use the following norm of $\mathcal{I}_G^{\Lambda,\tau}$

$$\left|\left| D^{-1} \cdot \mathcal{I}_G^{\Lambda,\tau} \cdot D \right|\right|_\infty , \tag{9}$$

where D is the diagonal matrix indexed by the vertices in $V \setminus \Lambda$ such that

$$D(u,u) = \begin{cases} \deg_G(v)^{1/\chi} & \text{if } \deg_G(v) \ge 1 \\ 1 & \text{if } \deg_G(v) = 0 \ , \end{cases} \tag{10}$$

where the parameter χ is being specified later.

Let $G = (V,E)$ be a fixed graph. For any vertex $v \in V$ and integer $\ell \ge 0$, we use $N_{v,\ell}$ to denote the number of simple paths with $\ell + 1$ vertices that start from v in graph G. By definition, we have that $N_{v,0} = 1$.

▶ **Definition 11** (d-branching value). *Let $d \ge 1$ be a real number and $G = (V,E)$ be a graph. For any vertex $v \in V$, the d-branching value S_v is defined by $\sum_{\ell \ge 0} N_{v,\ell}/d^\ell$.*

We establish spectral independence results that utilise the notion of d-branching value that was introduced in [3]. The following theorem is an example of the Spectral Independence results we derive here. In our proof, we actually use the stronger result in Theorem 19 . This analysis of spectral independence is of independent interest.

▶ **Theorem 12.** *Let $d > 1$ be a real number and $G = (V,E)$ be a graph. Let μ_G be the Hard-Core model with fugacity $\lambda < \lambda_c(d)$. For any $\alpha > 0$ such that the d-branching value $S_v \le \alpha$ for all $v \in V$ the following is true: μ_G is η-spectrally independent for*

$$\eta \le C_0 \cdot \alpha^{1/r} \ ,$$

where $C_0 = C_0(d,\lambda)$ and $r = r(d) \in (1,2)$ are constants.

There are a couple of interesting point about Theorem 12 to make. The first one is that the bound on η does not have any dependence on the degrees of the graph G. This is because we utilise the matrix norm $\|D^{-1} \cdot \mathcal{I}_G^{\Lambda,\tau} \cdot D\|_\infty$ instead of $\|\mathcal{I}_G^{\Lambda,\tau}\|_\infty$ that is typically used to establish the bound on the spectral independence. Furthermore, note that Theorem 12 is *not* necessarily about $G(n,d/n)$, i.e., it applies to an arbitrary graph. As a matter of fact in order to use the above result for $G(n,d/n)$ we need to establish bounds on its branching value. To this end, we use the following result from [3] so that we can take $\alpha = \log n$ in Theorem 12.

▶ **Lemma 13** ([3, Lemma 9]). *Let $d \geq 1$. For any fixed $d' > d$, with probability $1 - o(1)$ over $\boldsymbol{G} \sim G(n, d/n)$, the d'-branching factor of every vertex in G is at most $\log n$.*

It is worth mentioning that Lemma 13, here, is a *weaker* version of Lemma 9 in [3], i.e., we do not really need the the full strength of the result there.

In light of the above results, an interesting open problem is to turn the branching-value based spectral independence result in Theorem 12 into rapid mixing bound one for Glauber dynamics on a general graphs with bounded branching value. Note that this is not possible with the techniques we develop here.

Concluding this short introductory section about Spectral Independence, let us remark that for our results we work with the so-called *Complete Spectral Independence* for the Hard-Core model, introduced in [7, 8]. This is more general a notion compared to the (standard) Spectral Independence. For further discussion see Section 4.2.

4 Entropy Factorisation from Stability and Spectral Independence

In this section we establish the ℓ-block factorisation of entropy for the Hard-Core model on $G(n, d/n)$ as it is described in Theorem 6. To this end, we employ techniques from [8]. This means that we study the Hard-Core model on $G(n, d/n)$ in terms of the stability of ratios of the marginals and the so-called Complete Spectral Independence.

4.1 Ratios of Gibbs Marginals & Stability

Consider the *fixed* graph $G = (V, E)$ and a Gibbs distribution μ on this graph. For a vertex $w \in V$, the region $K \subseteq V \setminus \{w\}$ and $\tau \in \{\pm 1\}^K$, we consider the *ratio of marginals* at w denoted as $R^{K,\tau}(w)$ such that

$$R_G^{K,\tau}(w) = \frac{\mu_w(+1 \mid K, \tau)}{\mu_w(-1 \mid K, \tau)} \ . \tag{11}$$

Recall that $\mu_w(\cdot \mid K, \tau)$ denotes the marginal of the Gibbs distribution $\mu(\cdot \mid K, \tau)$ at vertex w. Also, note that the above allows for $R^{K,\tau}(w) = \infty$, e.g., when $\mu_w(-1 \mid K, \tau) = 0$ and $\mu_w(+1 \mid K, \tau) \neq 0$.

▶ **Definition 14** (Marginal stability). *Let $\zeta > 0$ be a real number. The Gibbs distribution μ_G on $G = (V, E)$ is called ζ-marginally stable if for any $\Lambda \subset V$, any $w \in V \setminus \Lambda$, for any configuration τ at Λ and any $S \subseteq \Lambda$ we have that*

$$R_G^{\Lambda,\tau}(w) \leq \zeta \qquad \text{and} \qquad R_G^{\Lambda,\tau}(w) \leq \zeta \cdot R_G^{S,\tau_S}(w) \ . \tag{12}$$

As far as the stability of the Hard-Core marginals at $G(n, d/n)$ is concerned, we prove the following result.

▶ **Theorem 15** (Stability Hard-Core Model). *For any fixed $d > 0$, for any $\lambda < \lambda_c(d)$, consider $\boldsymbol{G} \sim G(n, d/n)$ and let $\mu_{\boldsymbol{G}}$ be the Hard-Core model on $\boldsymbol{G}$ with fugacity λ. With probability $1 - o(1)$ over the instances $\boldsymbol{G}$, $\mu_{\boldsymbol{G}}$ is $2(1+\lambda)^{\frac{2\log n}{\log\log n}}$-marginally stable.*

Proof. Let $\zeta = 2(1+\lambda)^{2\frac{\log n}{\log\log n}}$. Also, let $N(w)$ be the set of the neighbours of w.

For any $\Lambda \subseteq V$ and any $\tau \in \{\pm\}^\Lambda$, we have that $\mu_w(+1 \mid \Lambda, \tau) \leq \frac{\lambda}{1+\lambda}$. One can see that the equality holds if $N(w) \subseteq \Lambda$ and for every $u \in N(w)$ we have that $\tau(w) = -1$. Noting that $R_{\boldsymbol{G}}^{\Lambda,\tau}(w)$ is increasing in the value of the Gibbs marginal $\mu_w(+1 \mid \Lambda, \tau)$, it is immediate that

$$\Pr\left[R_{\boldsymbol{G}}^{\Lambda,\tau}(w) \leq \lambda < \zeta \quad \forall \Lambda \subseteq V,\ \forall w \in V \setminus \Lambda\right] = 1 \ . \tag{13}$$

It remains to show that

$$\Pr\left[R_{\boldsymbol{G}}^{\Lambda,\tau}(w) \leq \zeta \cdot R_{\boldsymbol{G}}^{S,\tau_S}(w) \quad \forall \Lambda \subset V,\ \forall S \subset \Lambda,\ \forall w \in V \setminus \Lambda\right] = 1 - o(1) \ . \tag{14}$$

In light of (13), (14) follows by showing that

$$\Pr\left[R_{\boldsymbol{G}}^{S,\tau_S}(2) > 2\lambda\,(1+\lambda)^{-2\frac{\log n}{\log\log n}} \quad \forall \Lambda \subset V,\ \forall S \subset \Lambda,\ \forall w \in V \setminus \Lambda\right] = 1 - o(1) \ . \tag{15}$$

If there is $u \in N(w)$ such that $\tau(u) = +1$, then $R_{\boldsymbol{G}}^{\Lambda,\tau}(w) = 0$ and (14) holds trivially since $R_{\boldsymbol{G}}^{S,\tau_S}(w) \geq 0$. We focus on the case that all vertices $u \in N(w) \cap \Lambda$ satisfy $\tau(u) = -1$.

Let $\mathcal{E}$ be the event that none of the vertices in $N(w)$ is occupied, while let γ_S be the probability of the event $\mathcal{E}$ under the Gibbs distribution $\mu(\cdot \mid S, \tau_S)$. It is standard to show that

$$R_{\boldsymbol{G}}^{S,\tau_S}(w) = \frac{\frac{\lambda}{1+\lambda}\gamma_S}{1 - \frac{\lambda}{1+\lambda}\gamma_S} \ .$$

Noting that the function $f(x) = \frac{x}{1-x}$ is increasing in $x \in (0,1)$, while $\gamma_S \geq (\frac{1}{1+\lambda})^{\deg_{\boldsymbol{G}}(w)}$, we have that

$$R_{\boldsymbol{G}}^{S,\tau_S}(w) \geq \frac{\frac{\lambda}{1+\lambda}(\frac{1}{1+\lambda})^{\deg_{\boldsymbol{G}}(w)}}{1 - \frac{\lambda}{1+\lambda}(\frac{1}{1+\lambda})^{\deg_{\boldsymbol{G}}(w)}} = \frac{\lambda}{(1+\lambda)^{\deg_{\boldsymbol{G}}(w)+1} - \lambda} \ .$$

From the above, it is immediate to get (15). Specifically, it follows from the above inequality and a standard bound on the maximum degree of random graph which implies that for any fixed number $\epsilon > 0$, the maximum degree in $\boldsymbol{G}$ is less than $(1+\epsilon)\frac{\log n}{\log\log n}$ with probability $1 - o(1)$.

This concludes the proof of Theorem 15. ◀

4.2 (Complete) Spectral Independence

The notions of the pairwise influence matrix $\mathcal{I}_G^{\Lambda,\tau}$ and the Spectral Independence, as we introduce them in Section 3.2, are typically used to establish bounds on the spectral gap for Glauber dynamics and hence derive bounds on the mixing time of the chain.

The authors in [9], make a further use of Spectral Independence to obtain the approximate tensorisation of entropy. Unfortunately, a vanilla application of their technique is not sufficient to prove our tensorisation results, mainly, because of the unbounded degrees we typically have in $G(n, d/n)$.

In this work, we exploit ideas from [9] together with the related notion of the *Complete Spectral Independence*, in order to establish our factorisation results for the entropy in Theorem 6. Specifically, we utilise the connection between complete spectral independence and the ℓ block factorisation of entropy that was established in [8] (see further details in the following section).

Since the notions of the pairwise influence matrix $\mathcal{I}_G^{\Lambda,\tau}$ and the Spectral Independence are so important, let us recall them once more, even though they have already been defined in Section 3.2. Consider a *fixed* graph $G = (V, E)$. Assume that we are given a Gibbs distribution μ on the configuration space $\{\pm 1\}^V$.

We define the pairwise influence matrix $\mathcal{I}_G^{\Lambda,\tau}$ as follows: for a set of vertices $\Lambda \subset V$ and a configuration τ at Λ, the matrix $\mathcal{I}_G^{\Lambda,\tau}$ is indexed by the vertices in $V \setminus \Lambda$, while for any two vertices $v, w \in V \setminus \Lambda$, different with each other, if w can take both values ± 1 given τ, we have that

$$\mathcal{I}_G^{\Lambda,\tau}(w,u) = ||\mu_u(\cdot \mid (\Lambda,\tau),(\{w\},+)) - \mu_u(\cdot \mid (\Lambda,\tau),(\{w\},-))||_{\text{TV}} \quad ; \tag{16}$$

if w can only take one value in ± 1 given τ, we have $\mathcal{I}_G^{\Lambda,\tau}(w,u) = 0$. Also, we have that $\mathcal{I}_G^{\Lambda,\tau}(w,w) = 0$ for all $w \in V \setminus \Lambda$. That is, the diagonal of $\mathcal{I}_G^{\Lambda,\tau}$ is always zero.

Recall that, above, $\mu_u(\cdot \mid (\Lambda,\tau),(\{w\},1))$ is the Gibbs marginal that vertex u, conditional that the configuration at Λ is τ and the configuration at w is 1. We have the analogous for $\mu_u(\cdot \mid (\Lambda,\tau),(\{w\},-1))$.

▶ **Definition 16** (Spectral Independence). *For a real number $\eta > 0$, the Gibbs distribution μ_G on $G = (V,E)$ is η-spectrally independent, if for every $0 \leq k \leq |V| - 2$, $\Lambda \subseteq V$ of size k and $\tau \in \{\pm 1\}^\Lambda$ the spectral radius of $\mathcal{I}_G^{\Lambda,\tau}$ satisfies that $\rho(\mathcal{I}_G^{\Lambda,\tau}) \leq \eta$.*

We proceed to introduce the Complete Spectral Independence. First, consider the notion of the Magnetising operation.

▶ **Definition 17** (Magnetising operation). *Let μ_G be a Gibbs distribution on the graph $G = (V,E)$. For any local fields $\vec{\phi} \in \mathbb{R}_{>0}^V$, the magnetised distribution $\vec{\phi} * \mu$ satisfies*

$$\forall \sigma \in \{\pm 1\}^V, \quad (\vec{\phi} * \mu)(\sigma) \propto \mu(\sigma) \prod_{v \in V : \sigma_v = +1} \phi_v \ .$$

We denote $\vec{\phi} * \mu$ by $\phi * \mu$ if $\vec{\phi}$ is a constant vector with value ϕ.

Suppose that μ is the Hard-Core model on G with fugacity λ. It is immediate that the magnetisied distribution $\vec{\phi} * \mu$ can be viewed as the *non-homogenious* Hard-Core model such that each vertex v has its own fugacity $\lambda_v = \lambda \cdot \phi_v$.

▶ **Definition 18** (Complete Spectral Independence). *For two reals $\eta > 0$ and $s > 0$, the Gibbs distribution μ_G on $G = (V,E)$ is (η, s)-completely spectrally independent, if the magnetised distribution $\vec{\phi} * \mu$ is η-spectrally independent for all $\vec{\phi} \in (0, 1+s]^V$.*

As far as the Hard-Core model on the random graph $G(n, d/n)$ is concerned, we prove the following result.

▶ **Theorem 19.** *For any fixed $d > 1$ and $\lambda < \lambda_c(d)$, there exist bounded constants $r = r(d,\lambda) \in (1,2)$, $B = B(d,\lambda) > 0$ and $s = s(d,\lambda) > 0$ such that the following holds:*

Consider $\boldsymbol{G} \sim G(n, d/n)$ and let $\mu_{\boldsymbol{G}}$ be the Hard-Core model on $\boldsymbol{G}$ with fugacity λ. With probability $1 - o(1)$ over the instances of $\boldsymbol{G}$, $\mu_{\boldsymbol{G}}$ is $(B \cdot (\log n)^{1/r}, s)$-completely spectrally independent.

The proof of Theorem 19 appears in the full version of this paper, where we first relate the influence matrix on the graph to the influence matrix on the self-avoiding walk tree [31, 27] and then use the potential function in [28] to analysis the *weighted* total influence on the self-avoiding walk tree.

4.3 Entropy Block Factorisation - Proof of Theorem 6

The following theorem, from [8], allows us to derive a bound on the ℓ- block factorisation parameter of the entropy by using the result in Theorem 15 for the stability of Gibbs marginals and the result in Theorem 19 for Complete Spectral Independence.

▶ **Theorem 20** ([8, Lemma 2.3]). *Let $\eta > 0, \xi > 0$ and $\zeta > 0$ be parameters. Let μ_G be a Gibbs distribution on $G = (V, E)$. If μ_G is (η, ξ)-completely spectrally independent and ζ-marginally stable, then for any $1/\alpha \leq \ell < n$, μ_G satisfies the ℓ block factorisation of entropy with parameter $C = (\frac{en}{\ell})^{1+1/\alpha}$, where*

$$\alpha = \min\left\{\frac{1}{2\eta}, \frac{\log(1+\xi)}{\log(1+\xi)+\log 2\zeta}\right\} .$$

Proof of Theorem 6. From Theorem 19 we have the following: with probability $1 - o(1)$ over the instances of $\boldsymbol{G}$ we have that $\mu_{\boldsymbol{G}}$ is (η, s)-completely spectrally independent where $s = s(d, \lambda)$ is *constant*, while

$$\eta = B \cdot (\log n)^{1/r} = o\left(\frac{\log n}{\log\log n}\right),$$

where $B = B(d, \lambda)$ and $r = r(d, \lambda) \in (1, 2)$ are constants specified in the statement of Theorem 19. The second equality above follows by noting that $1/r < 1$, bounded away from 1.

Furthermore, from Theorem 15 we have the following: With probability $1 - o(1)$ over the instances of $\boldsymbol{G}$, the distribution $\mu_{\boldsymbol{G}}$ is ζ-marginally stable, where

$$\zeta \leq 2(1+\lambda)^{2\frac{\log n}{\log\log n}} .$$

In light of all the above, the theorem follows by plugging the above values into Theorem 20. ◀

5 Approximate Tensorisation of Entropy

In this section we prove our results related to the approximate tensorisation of the entropy. These are Theorem 4 and Lemma 8.

5.1 Proof of Theorem 4

In this section we give the full proof of Theorem 4. Recall the high level description of the steps we follow towards this endeavour in Section 3.1.

Proof of Theorem 4. From Theorem 6 we have the following: For $d > 1$ and $\lambda < \lambda_c(d)$, consider $\boldsymbol{G} \sim G(n, d/n)$, while let $\mu = \mu_{\boldsymbol{G}}$ be the Hard-Core model on $\boldsymbol{G}$ with fugacity λ. Let the number $\theta = \theta(d, \lambda)$ in the interval $(0, 1)$ be a parameter whose value is going to be specified later. Then, with probability $1 - o(1)$ over the instances of $\boldsymbol{G}$, for $\ell = \lceil \theta n \rceil$ and for any $f : \Omega \to \mathbb{R}_{>0}$ we have that

$$\mathrm{Ent}_\mu(f) \leq \left(\frac{\mathrm{e}}{\theta}\right)^{1+1/\alpha} \frac{1}{\binom{n}{\ell}} \sum_{S \in \binom{V}{\ell}} \mu\left(\mathrm{Ent}_S(f)\right). \tag{17}$$

Recall that $C(S)$ denotes the set of connected components in $\boldsymbol{G}[S]$, the subgraph that is induced by vertices in S. With a slight abuse of notation, we use $U \in C(S)$ to denote the set of vertices in the component U. By the conditional independence property of the Gibbs distribution and Lemma 7, we have

$$\begin{aligned}
\mathrm{Ent}_\mu(f) &\le \left(\frac{\mathrm{e}}{\theta}\right)^{1+1/\alpha} \frac{1}{\binom{n}{\ell}} \sum_{S\in\binom{V}{\ell}} \sum_{U\in C(S)} \mu\left(\mathrm{Ent}_U(f)\right) \\
\text{(by Lemma 8)} \quad &\le \left(\frac{\mathrm{e}}{\theta}\right)^{1+1/\alpha} \frac{1}{\binom{n}{\ell}} \sum_{S\in\binom{V}{\ell}} \sum_{U\in C(S)} \mathrm{AT}(|U|) \sum_{v\in U} \mu[\mathrm{Ent}_v(f)] \\
&\le \left(\frac{\mathrm{e}}{\theta}\right)^{1+1/\alpha} \sum_{v\in V} \mu[\mathrm{Ent}_v(f)] \sum_{k\ge 1} \mathrm{AT}(k)\Pr[|C_v|=k] \ , \qquad (18)
\end{aligned}$$

where C_v is the connected component in $\boldsymbol{G}[S]$, where S is sampled from $\binom{V}{\ell}$ uniformly at random. In order to bound the innermost summation on the R.H.S. of (18) we distinguish two cases for k. For $1 \le k \le \log n$, we use the trivial bound $\Pr[|C_v| = k] \le 1$, while Lemma 8 implies that

$$\begin{aligned}
\sum_{k=1}^{\log n} \mathrm{AT}(k)\Pr[|C_v|=k] \le \sum_{k=1}^{\log n} \mathrm{AT}(k) &= \sum_{k=1}^{\log n} 3\log\left(1+\lambda+\lambda^{-1}\right)\cdot((1+\lambda)k)^{2+2\eta} \\
&\le 3\log\left(1+\lambda+\lambda^{-1}\right)\cdot\log n\cdot((1+\lambda)\log n)^{2+2\eta} \\
&\le 3\log\left(1+\lambda+\lambda^{-1}\right)\cdot((1+\lambda)\log n)^{3+2\eta} \ ,
\end{aligned}$$

where $\eta = B(\log n)^{1/r}$, for constants $B = B(d,\lambda)$ and $r = r(d) \in (1,2)$. Elementary calculations imply that

$$\sum_{k=1}^{\log n} \mathrm{AT}(k)\Pr[|C_v|=k] \le 3\log\left(1+\lambda+\lambda^{-1}\right)\cdot((1+\lambda)\log n)^{3+2\eta} \le n^x \ , \qquad (19)$$

for $x = o\left(\frac{1}{\log\log n}\right)$.

For $k \ge \log n$, we use the bound in Lemma 9 for $\Pr[|C_v| = k]$, while from Lemma 8 we have

$$\sum_{k\ge\log n} \mathrm{AT}(k)\Pr[|C_v|=k] \le 2k^2\left(1+\lambda+\lambda^{-1}\right)^{2k+2}(2\mathrm{e})^{\mathrm{e}Lk}(2\theta)^k \ ,$$

where $L = L(d)$ is the parameter in Lemma 9. We choose sufficiently small $\theta = \theta(d,\lambda)$ such that

$$\forall k \ge 1, \quad 2k^2\left(1+\lambda+\lambda^{-1}\right)^{2k+2}(2\mathrm{e})^{\mathrm{e}Lk}(2\theta)^k \le (1/2)^k \ .$$

This implies that

$$\sum_{k\ge\log n} \mathrm{AT}(k)\Pr[|C_v|=k] \le \sum_{k\ge\log n}\left(\frac{1}{2}\right)^k \le 1 \ . \qquad (20)$$

Plugging (19), (20) into (18), we get the following: With probability $1-o(1)$ over the instances of $\boldsymbol{G}$ we have that

$$\mathrm{Ent}_\mu(f) \le \left(\frac{\mathrm{e}}{\theta}\right)^{1+1/\alpha}\left(n^{\left(\frac{1}{\log\log n}\right)}+1\right)\sum_{v\in V}\mu[\mathrm{Ent}_v(f)] \ .$$

Since, by Theorem 6 we have that $\frac{1}{\alpha} = K(\frac{\log n}{\log\log n})$, for a constant $K = K(d,\lambda)$, and $\theta = \theta(d,\lambda)$ is also a constant, the above inequality can be written as follows: there is a constant $A = A(d,\lambda)$ such that

$$\mathrm{Ent}_\mu(f) \leq n^{\left(\frac{A}{\log\log n}\right)} \sum_{v\in V} \mu[\mathrm{Ent}_v(f)] \ .$$

The above concludes the proof of Theorem 4. ◀

References

1 Nima Anari, Kuikui Liu, and Shayan Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. In *FOCS*, pages 1319–1330, 2020.

2 Jean Barbier, Florent Krzakala, Lenka Zdeborová, and Pan Zhang. The hard-core model on random graphs revisited. *Journal of Physics: Conference Series*, 473(1):012021, December 2013.

3 Ivona Bezáková, Andreas Galanis, Leslie Ann Goldberg, and Daniel Štefankovič. Fast sampling via spectral independence beyond bounded-degree graphs. In *ICALP*, volume 229 of *LIPIcs*, pages 21:1–21:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

4 Pietro Caputo, Georg Menz, and Prasad Tetali. Approximate tensorization of entropy at high temperature. *Ann. Fac. Sci. Toulouse Math. (6)*, 24(4):691–716, 2015.

5 Pietro Caputo and Daniel Parisi. Block factorization of the relative entropy via spatial mixing. *arXiv preprint*, 2020. `arXiv:2004.10574`.

6 Filippo Cesi. Quasi-factorization of the entropy and logarithmic Sobolev inequalities for Gibbs random fields. *Probab. Theory Related Fields*, 120(4):569–584, 2001.

7 Xiaoyu Chen, Weiming Feng, Yitong Yin, and Xinyuan Zhang. Rapid mixing of glauber dynamics via spectral independence for all degrees. In *FOCS*, pages 137–148. IEEE, 2021.

8 Xiaoyu Chen, Weiming Feng, Yitong Yin, and Xinyuan Zhang. Optimal mixing for two-state anti-ferromagnetic spin systems. In *FOCS*, pages 588–599. IEEE, 2022.

9 Zongchen Chen, Kuikui Liu, and Eric Vigoda. Optimal mixing of Glauber dynamics: Entropy factorization via high-dimensional expansion. In *STOC*, 2021. arXiv:2011.02075. `arXiv:2011.02075`.

10 Amin Coja-Oghlan and Charilaos Efthymiou. On independent sets in random graphs. *Random Struct. Algorithms*, 47(3):436–486, 2015.

11 Varsha Dani and Cristopher Moore. Independent sets in random graphs from the weighted second moment method. In *RANDOM*, volume 6845 of *Lecture Notes in Computer Science*, pages 472–482. Springer, 2011.

12 Charilaos Efthymiou. Spectral independence beyond uniqueness using the topological method. *CoRR*, abs/2211.03753, 2022. `arXiv:2211.03753`.

13 Charilaos Efthymiou, Thomas P. Hayes, Daniel Stefankovic, and Eric Vigoda. Sampling random colorings of sparse random graphs. In *SODA*, pages 1759–1771, 2018.

14 Weiming Feng, Heng Guo, Yitong Yin, and Chihao Zhang. Fast sampling and counting k-SAT solutions in the local lemma regime. *J. ACM*, 68(6):40:1–40:42, 2021.

15 Weiming Feng, Heng Guo, Yitong Yin, and Chihao Zhang. Rapid mixing from spectral independence beyond the boolean domain. In *SODA*, pages 1558–1577, 2021.

16 Weiming Feng, Kun He, and Yitong Yin. Sampling constraint satisfaction solutions in the local lemma regime. In *STOC*, pages 1565–1578. ACM, 2021.

17 Alan M. Frieze. On the independence number of random graphs. *Discret. Math.*, 81(2):171–175, 1990.

18 Andreas Galanis, Leslie Ann Goldberg, Heng Guo, and Kuan Yang. Counting solutions to random CNF formulas. *SIAM J. Comput.*, 50(6):1701–1738, 2021.

19 Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Inapproximability of the partition function for the antiferromagnetic Ising and hard-core models. *Combinatorics, Probability and Computing*, 25(04):500–559, 2016.

20 David Gamarnik and Madhu Sudan. Limits of local algorithms over sparse random graphs. In *ITCS*, pages 369–376. ACM, 2014.

21 Vishesh Jain, Huy Tuan Pham, and Thuy-Duong Vuong. On the sampling Lovász local lemma for atomic constraint satisfaction problems. *arXiv preprint*, 2021. `arXiv:2102.08342`.

22 Vishesh Jain, Huy Tuan Pham, and Thuy Duong Vuong. Spectral independence, coupling with the stationary distribution, and the spectral gap of the Glauber dynamics. *arXiv preprint*, 2021. `arXiv:2105.01201`.

23 Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM J. Comput.*, 18(6):1149–1178, 1989.

24 F. P. Kelly. Stochastic models of computer communication systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 47:379–395, 1985.

25 Florent Krzakala, Andrea Montanari, Federico Ricci-Tersenghi, Guilhem Semerjian, and Lenka Zdeborová. Gibbs states and the set of solutions of random constraint satisfaction problems. *Proc. Natl. Acad. Sci. USA*, 104(25):10318–10323, 2007.

26 Ankur Moitra. Approximate counting, the Lovász local lemma, and inference in graphical models. *J. ACM*, 66(2):10:1–10:25, 2019.

27 Jesús Salas and Alan D Sokal. Absence of phase transition for antiferromagnetic Potts models via the Dobrushin uniqueness theorem. *J. Stat. Phys.*, 86(3):551–579, 1997.

28 Alistair Sinclair, Piyush Srivastava, Daniel Štefankovič, and Yitong Yin. Spatial mixing and the connective constant: optimal bounds. *Probab. Theory Related Fields*, 168(1-2):153–197, 2017.

29 Allan Sly. Computational transition at the uniqueness threshold. In *FOCS*, pages 287–296, 2010.

30 Allan Sly and Nike Sun. Counting in two-spin models on d-regular graphs. *Ann. Probab.*, 42(6):2383–2416, 2014.

31 Dror Weitz. Counting independent sets up to the tree threshold. In *STOC*, pages 140–149, 2006.

Broadcasting with Random Matrices

Charilaos Efthymiou
Computer Science, University of Warwick, Coventry, UK

Kostas Zampetakis
Computer Science, University of Warwick, Coventry, UK

Abstract

Motivated by the theory of *spin-glasses* in physics, we study the so-called *reconstruction problem* on the tree, and on the sparse random graph $\boldsymbol{G}(n, d/n)$. Both cases reduce naturally to analysing broadcasting models, where each edge has its own broadcasting matrix, and this matrix is drawn independently from a predefined distribution.

We establish the *reconstruction threshold* for the cases where the broadcasting matrices give rise to symmetric, 2-spin Gibbs distributions. This threshold seems to be a natural extension of the well-known *Kesten-Stigum bound* that manifests in the classic version of the reconstruction problem. Our results determine, as a special case, the reconstruction threshold for the prominent *Edwards–Anderson model* of spin-glasses, on the tree.

Also, we extend our analysis to the setting of the Galton-Watson random tree, and the (sparse) random graph $\boldsymbol{G}(n, d/n)$, where we establish the corresponding thresholds. Interestingly, for the Edwards–Anderson model on the random graph, we show that the *replica symmetry breaking* phase transition, established by Guerra and and Toninelli in [21], coincides with the reconstruction threshold.

Compared to classical Gibbs distributions, spin-glasses have several unique features. In that respect, their study calls for new ideas, e.g. we introduce novel estimators for the reconstruction problem. The main technical challenge in the analysis of such systems, is the presence of (too) many levels of randomness, which we manage to circumvent by utilising recently proposed tools coming from the analysis of Markov chains.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Randomness, geometry and discrete structures; Mathematics of computing → Discrete mathematics

Keywords and phrases spin-system, spin-glass, sparse random graph, reconstruction, phase transitions

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.55

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2302.11657`

Funding *Charilaos Efthymiou*: EPSRC New Investigator Award (grant no. EP/V050842/1) and Centre of Discrete Mathematics and Applications (DIMAP), The University of Warwick.
Kostas Zampetakis: EPSRC New Investigator Award (grant no. EP/V050842/1) and Centre of Discrete Mathematics and Applications (DIMAP), The University of Warwick.

Acknowledgements We are grateful to the anonymous reviewers for their thorough review of our submission, and for their insightful comments and suggestions.

1 Introduction

Motivated by the theory of *spin-glasses* in physics, we study the so-called *reconstruction problem* with respect to the related distributions, on the tree, and on the sparse random graph $\boldsymbol{G}(n, d/n)$.

Spin-glasses are *disordered* magnetic materials that are studied by physicists (not necessarily the theoretical ones). It has been noted that even though they are a type of magnet, actually, "they are not very good at being magnets". Metallic spin-glasses are "unremarkable conductors", and the insulating spin-glasses are "fairly useless as practical insulators ... ", e.g. see [30].

However, the research on spin-glasses has provided tools to analyse some exciting, and extremely challenging, problems in mathematics, physics, but also *real world* ones. Through their study, we have garnered a deep understanding of the nature of complex systems. A case in point is the pioneering work of Giorgio Parisi in '70s on the so-called *Sherrington-Kirkpatrick* spin-glass, which introduces the formulation of the renowned *replica symmetry breaking* [27]. Parisi's ideas were highly influential in physics community, and later, in mathematics, and computer science. The theory of replica symmetry breaking was among the groundbreaking ideas which got Parisi the Nobel Prise in Physics in 2021.

Perhaps one of the most successful, and extensively studied, spin-glass models, is the famous *Edwards-Anderson* model (EA-model for short), introduced back in '70s by Sam Edwards and Philip Anderson in [16]. Few months after the work of Edwards and Anderson, David Sherrington and Scott Kirkpatrick, in [28], introduced their own model of spin-glasses, the well-known in computer science literature, *Sherrington-Kirkpatrick* model (or SK-model for short). As it turns out, the SK-model corresponds to the *mean field* version of the EA-model.

Given a fixed graph $G = (V, E)$, the Edwards-Anderson model with *inverse* temperature $\beta > 0$, is the *random* Gibbs distribution $\boldsymbol{\mu}$ on the configuration space $\{\pm 1\}^V$ defined as follows: let $\{\boldsymbol{J}_e : e \in E\}$ be independent identically distributed (i.i.d.) *standard Gaussians*. Then each configuration $\sigma \in \{\pm 1\}^V$ receives probability mass $\boldsymbol{\mu}(\sigma)$, defined by

$$\boldsymbol{\mu}(\sigma) \propto \exp\left(\beta \cdot \sum_{\{u,w\}\in E} \mathbf{1}\{\sigma(u) = \sigma(w)\} \cdot \boldsymbol{J}_{\{u,w\}}\right) \ , \tag{1}$$

where $\propto$ stands for "proportional to". We usually refer to $\{\boldsymbol{J}_e\}_{e\in E}$ as the *coupling parameters*. Let us comment here that, traditionally, the Gibbs distribution is defined by replacing the indicator $\mathbf{1}\{\sigma(u) = \sigma(w)\}$ in (1), with the product $\sigma(u)\sigma(w)$, in the physics literature. However, the two formulations are equivalent, as a simple transformation converts one to the other (see the full version). We also note that there is a simpler version of the Edwards-Anderson model, in which coupling parameters take independently ± 1 values, uniformly at random.

Apart from its mathematical elegance, and theoretical importance, the Edwards-Anderson model, and the related spin-glass distributions, arise also in applications such as neural networks (e.g. the so-called Hopfield model), protein folding, and conformational dynamics. We refer the interested reader to [30], and references therein.

In this work, we largely study the Edwards-Anderson model on trees, and the (locally tree-like) *random graph* $\boldsymbol{G}(n, d/n)$ with constant expected degree d. This is the random graph on n vertices, such that each edge appears independently with probability d/n.

Since the Edwards-Anderson model on $\boldsymbol{G}(n, d/n)$ shares essential features with random *Constraint Satisfaction Problems* (r-CSPs for short), it is not surprising that has been studied extensively in terms of phase transitions, in physics, e.g. [19, 25], mathematics, e.g. [21, 12], but also in computer science, e.g. for sampling algorithms [17, 2].

In contrast to the standard Gibbs distributions on trees, e.g. the Ising model, the Hard-core model, and the Potts model, the Edwards-Anderson model, despite being the most basic distribution for spin-glasses, has not been sufficiently studied. As a result, several fundamental questions about it still remain open. Here, we consider the tree *reconstruction problem* for the Edwards-Anderson model (and some natural extensions).

The reconstruction problem studies the effect of the configuration at a vertex, r, on that of the vertices at distance h from r, as $h \to \infty$. Specifically, we want to distinguish the region of parameters where the effect is vanishing, from that where the effect is non-vanishing. Typically, the two regions are specified in terms of a *sharp threshold*, i.e., we have an abrupt transition from one region to the other as we vary the parameters of the model. We usually call this phenomenon *reconstruction threshold*, and it has been the subject of intense study, e.g. [26, 1, 22, 7, 29, 10]. In the context of r-CSPs, the onset of reconstruction has been linked to an abrupt deterioration of the performance of algorithms (both searching and counting), e.g. see [1].

In this work, among other results, we establish precisely the reconstruction threshold for the Edwards-Anderson model on the Δ-ary tree, the Galton-Watson tree with general offspring distribution, and the random graph $\boldsymbol{G}(n, d/n)$. Furthermore, as far as the Edwards-Anderson model on $\boldsymbol{G}(n, d/n)$ is concerned, we combine our results with [21, 12], to conclude that the reconstruction threshold coincides with the so-called *Replica Symmetry Breaking* phase transition.

Interestingly, for the Δ-ary tree, we establish the reconstruction threshold, not only for the Edwards-Anderson model, but also for the general version of the Gibbs distribution $\boldsymbol{\mu}$ defined in (1). That is, the coupling parameters are i.i.d. following a *general distribution*, not necessary the standard Normal.

It turns out that the corresponding reconstruction problems on the Galton-Watson tree with Poisson(d) offspring, and on the sparse random graph $\boldsymbol{G}(n, d/n)$, are not too different from each other. Connections have been established between these two Gibbs distributions, e.g. see [4, 15, 11, 14]. We relate the two reconstruction results, i.e., for the tree and the graph, by exploiting the idea of planted-model (Teacher-Student model [31]) and the notion of mutual *contiguity* [12]. In that respect, our basic analysis involves the complete Δ-tree, and the Galton-Watson tree, while, subsequently, we extend these results to the random graph $\boldsymbol{G}(n, d/n)$.

We study the reconstruction problem on trees by means of the broadcasting models. These are abstractions of *noisy transmission* of information over the edges of the tree, i.e., the edges act as noisy channels. To our knowledge, the study of the broadcasting models, and the closely related reconstruction problem, dates back to '60s with the seminal work of Kesten and Stigum [24].

Establishing the reconstruction threshold for the Edwards-Anderson model on the Δ-ary tree, as well as the generalisation of this distribution, turns out to be a challenging problem. The difficulty of these models stems from the manifestation of local *frustration phenomena*, i.e., mixed ferromagnetic and antiferromagnetic interaction in the same neighbourhood, but also from the "many levels of randomness" we need to deal with in their analysis.

To this end, we make an extensive use of various potentials in order to simplify the analysis. To establish non-reconstruction, we employ some newly introduced techniques in the area of Markov chains and Spectral Independence [3, 9], that combine potential functions to analyse tree recursions. To establish reconstruction, we use a carefully crafted potential as an estimator for the root configuration. We call this estimator *flip-majority vote*.

1.1 Broadcasting, Reconstruction and the Kesten-Stigum bound

Consider the Δ-ary tree $T = (V, E)$, of height $h > 0$. Let r be the root of the tree T. Broadcasting on T, is a stochastic process which abstracts noisy transmission of information over the edges of the tree.

There is a finite set of spins $\mathcal{A}$, and an $\mathcal{A} \times \mathcal{A}$ stochastic matrix M, which we call the *broadcasting matrix*, or *transition matrix*. With the broadcasting we obtain a configuration $\boldsymbol{\sigma} \in \mathcal{A}^V$ by working recursively as follows: assume that the configuration at the root r is obtained according to some predefined distribution over $\mathcal{A}$. If for the non-leaf vertex u in T we have $\boldsymbol{\sigma}(u) = i$, then for each vertex w, child of u, we have $\boldsymbol{\sigma}(w) = j$ with probability $M(i, j)$, independently of the other children, i.e.,

$$\Pr[\boldsymbol{\sigma}(w) = j \mid \boldsymbol{\sigma}(u) = i] = M(i, j) \ .$$

Here we assume that $\boldsymbol{\sigma}(r)$ is distributed uniformly at random in $\mathcal{A}$.

A natural problem to study in this setting is the so-called *reconstruction problem*. Suppose that μ_h is the marginal distribution of the configuration of the vertices at distance h from the root. The reconstruction problem amounts to studying the influence of the configuration at the root of the tree to the marginal μ_h. Specifically, we want to compare the two distributions $\mu_h(\cdot \mid \boldsymbol{\sigma}(r) = i)$, and $\mu_h(\cdot \mid \boldsymbol{\sigma}(r) = j)$ for different $i, j \in \mathcal{A}$, i.e., μ_h conditional on the configuration at the root being i and j, respectively. The comparison is by means of the total variation distance, i.e.,

$$||\mu_h(\cdot \mid \boldsymbol{\sigma}(r) = i) - \mu_h(\cdot \mid \boldsymbol{\sigma}(r) = j)||_{\mathrm{TV}} \ .$$

Typically, we focus on the behaviour of the quantity above, as h grows.

▶ **Definition 1.** *We say that the distribution μ exhibits* reconstruction *if there exist spins $i, j \in \mathcal{A}$ such that*

$$\limsup_{h \to \infty} ||\mu_h(\cdot \mid \boldsymbol{\sigma}(r) = i) - \mu_h(\cdot \mid \boldsymbol{\sigma}(r) = j)||_{\mathrm{TV}} > 0 \ .$$

On the other hand, if for all $i, j \in \mathcal{A}$ the above limit is zero, then we have non-reconstruction.

The broadcasting process we describe above gives rise to well-known Gibbs distributions on T such as the *Ising model*, the *Potts model* etc. In terms of the Gibbs distributions on the tree, the reconstruction problem can be formulated as to whether the free-measure on the tree is *extremal*, or not. The extremality here is considered with respect to whether the Gibbs distribution can be expressed as a convex combination of two, or more measures, e.g. see [20]. It is interesting to compare the extremality condition with various spatial mixing conditions of the Gibbs distribution. Perhaps the most interesting case is to compare it with the Gibbs tree *uniqueness*. Then, it is standard to show that the extremality is a *weaker* condition than uniqueness.

The reconstruction problem has been studied since 1960s. Perhaps the most general result in the area is the so-called *Kesten-Stigum bound* [24], or KS-bound (for short). Let $\Delta_{\mathrm{KS}} = \Delta_{\mathrm{KS}}(M)$ be such that

$$\Delta_{\mathrm{KS}} = \lambda_2^{-2}(M) \ , \tag{2}$$

where $\lambda_2(M)$ is the second largest, in magnitude, eigenvalue of the transition matrix M. The result of [24] implies that if $\Delta > \Delta_{\mathrm{KS}}$, then we have reconstruction.

In light of the above, a natural question is whether the condition $\Delta < \Delta_{\mathrm{KS}}$ implies that we have non-reconstruction. In general, the answer to this question is no, e.g. see [5, 29]. However, for several important distributions, including the Ising model, the KS-bound is tight, in the sense that the condition $\Delta < \Delta_{\mathrm{KS}}$ indeed implies non-reconstruction, see [7, 18, 22].

1.2 Broadcasting with random matrices

Here, we consider the natural problem of broadcasting on a tree, where the transition matrix is *random*. In this setting, as before, we consider the Δ-ary tree $T = (V, E)$, of height $h > 0$, rooted at r. Also, we have a finite set of spins $\mathcal{A}$. Rather than using the same matrix for every edge of the tree, each edge has its own matrix, which is an independent sample from a predefined distribution ψ.

More formally, every $\mathcal{A} \times \mathcal{A}$ stochastic matrix can be viewed as a point in the $|\mathcal{A}|^2$ Euclidean space. We endow the set of all $\mathcal{A} \times \mathcal{A}$ stochastic matrices with the σ-algebra induced by the Borel algebra. Then, ψ is a distribution over the set of these matrices.

Once we have a matrix for each edge of T, the broadcasting proceeds with the same rules as in the deterministic case. If for the non-leaf vertex u in T we have $\boldsymbol{\sigma}(u) = i$, then the vertex w, child of u, gets $\boldsymbol{\sigma}(w) = j$ with probability $\boldsymbol{M}_e(i, j)$, independently of the other children of u, i.e.,

$$\Pr[\boldsymbol{\sigma}(w) = j \mid \boldsymbol{\sigma}(u) = i] = \boldsymbol{M}_e(i, j) \ ,$$

where $e = \{u, w\}$.

The above setting gives rise to a *random* probability measure on the set of configurations $\mathcal{A}^V$ which we denote as $\boldsymbol{\mu} = \boldsymbol{\mu}_{T,\psi}$. Hence, the configuration $\boldsymbol{\sigma} \in \mathcal{A}^V$ we get from the broadcasting, consists of *two-levels of randomness*. The first level is due to the fact that the measure $\boldsymbol{\mu}$ is induced by the random instances of the broadcasting matrices $\{\boldsymbol{M}_e\}_{e\in E}$. Once these matrices have been fixed, the second level of randomness emerges from the random choices of the broadcasting process. The above formulation gives rise to well-studied Gibbs distributions, such as the Edwards–Anderson model of spin-glasses, by choosing appropriately the distribution ψ.

In this new setting, we study the reconstruction problem. Here, the definition of reconstruction differs slightly from Definition 1 above. Denote with $\boldsymbol{\mu}_h$ the marginal of $\boldsymbol{\mu}$ on the vertices at distance h from the root of the tree T. Then, the reconstruction problem is defined as follows:

▶ **Definition 2.** *For a distribution ψ on $\mathcal{A} \times \mathcal{A}$ stochastic matrices , we say that the random measure $\boldsymbol{\mu} = \boldsymbol{\mu}_{T,\psi}$ exhibits* reconstruction *if there exist spins $i, j \in \mathcal{A}$ such that*

$$\limsup_{h\to\infty} \mathbb{E}\left[||\boldsymbol{\mu}_h(\cdot \mid \boldsymbol{\sigma}(r) = i) - \boldsymbol{\mu}_h(\cdot \mid \boldsymbol{\sigma}(r) = j)||_{\mathrm{TV}}\right] > 0 \ ,$$

where the expectation is with respect to the randomness of $\boldsymbol{\mu}$.

On the other hand, if for all $i, j \in \mathcal{A}$ the above limit is zero, then we have non-reconstruction*.*

We consider the reconstruction problem in terms of the KS-bound, i.e., we examine whether it is tight, or not. Before addressing this question, we need to specify what the parameter Δ_{KS} might be in this setting.

It turns out that a natural candidate for Δ_{KS} can be defined as follows:

Let $\boldsymbol{M}$ be a matrix sampled from the distribution ψ, and define

$$\Xi = \mathbb{E}\left[\boldsymbol{M} \otimes \boldsymbol{M}\right] \ , \tag{3}$$

i.e., the matrix Ξ is the expectation of the tensor product of the matrix $\boldsymbol{M}$ with itself. Let $\mathbf{1} \in \mathbb{R}^{\mathcal{A}}$ denote the vector whose entries are all equal to one. Also, write

$$\mathcal{E} = \left\{z \in \mathbb{R}^{\mathcal{A}} \otimes \mathbb{R}^{\mathcal{A}} : \forall y \in \mathbb{R}^{\mathcal{A}} \langle z, \mathbf{1} \otimes y\rangle = \langle z, y \otimes \mathbf{1}\rangle = 0\right\} \ ,$$

where $\langle \cdot, \cdot \rangle$ is the standard inner product operation. Then, we define $\Delta_{\mathrm{KS}}(\psi)$ to be such that

$$\Delta_{\mathrm{KS}}(\psi) = \left(\max_{x \in \mathcal{E}^{\cdot} : ||x||=1} \langle \Xi x, x \rangle \right)^{-1} \quad . \tag{4}$$

The above quantity, Δ_{KS}, arises in the study of phases transitions in random CSPs [12]. Specifically, it signifies an *upper bound* on the density of the so-called *Replica Symmetric* phase, of symmetric Gibbs distributions. The value Δ_{KS} is derived in [12] by means of a stability analysis of the so-called *free-energy* functional. Note that the above definition for $\Delta_{\mathrm{KS}}(\psi)$ applies to any set of spins $\mathcal{A}$, and any distribution ψ on $\mathcal{A} \times \mathcal{A}$ matrices.

Here, we prove that the above is indeed the analogue of KS-bound for *symmetric*, 2-spin distributions $\boldsymbol{\mu}$. That is, for any value of the parameter $\beta > 0$, and for any distribution ψ over the broadcasting matrices whose support is comprised of symmetric 2×2 matrices, we prove that the Δ-ary tree T exhibits reconstruction when $\Delta > \Delta_{\mathrm{KS}}(\psi)$, while we have non-reconstruction when $\Delta < \Delta_{\mathrm{KS}}(\psi)$.

Furthermore, we go beyond the basic case of the Δ-ary tree. Firstly, we extend our results to the cases where the underlying graph is the *Galton-Watson* random tree with general offspring distribution. Secondly, we exploit the notion of contiguity of measures to derive non-reconstruction results for the Edwards-Anderson model on the random graph $\boldsymbol{G}(n, d/n)$.

2 Results

We start the presentation of our results on the 2-spin, symmetric distributions, by considering the Δ-ary tree. Specifically, for integers $\Delta > 0$ and $h > 0$, let $T = (V, E)$ be the Δ-ary tree of height h, rooted at vertex r. We let $\mathcal{A} = \{\pm 1\}$ be the set of spins.

Assume that each edge of the tree is equipped with its own broadcasting matrix, each matrix drawn *independently* from the distribution induced by the following experiment: We have two parameters, a real number $\beta > 0$, and a distribution ϕ on the real numbers $\mathbb{R}$, i.e., we have the probability space $(\mathbb{R}, \mathcal{F}, \phi)$ where $\mathcal{F}$ is the σ-algebra induced by the Borel algebra. We generate a matrix $\boldsymbol{M}$ following the two steps below:

Step 1 Draw $\boldsymbol{J} \in \mathbb{R}$ from the distribution ϕ.

Step 2 Generate the $\mathcal{A} \times \mathcal{A}$ matrix $\boldsymbol{M}$ such that

$$\boldsymbol{M} = \frac{1}{\exp(\beta \boldsymbol{J}) + 1} \begin{bmatrix} \exp(\beta \boldsymbol{J}) & 1 \\ 1 & \exp(\beta \boldsymbol{J}) \end{bmatrix} \quad . \tag{5}$$

Note that our broadcasting matrices are always *symmetric*.

The above broadcasting process gives rise to configurations in $\mathcal{A}^V$ following the Gibbs distribution $\boldsymbol{\mu}_{\beta,\phi}$ specified as follows: Let $\{\boldsymbol{J}_e\}_{e \in E}$ be independent, identically distributed (i.i.d.) random variables such that each one of them is distributed as in ϕ (this is the same distribution used to generate matrix $\boldsymbol{M}$). Each $\sigma \in \mathcal{A}^V$ is assigned probability mass $\boldsymbol{\mu}_{\beta,\phi}(\sigma)$ defined by

$$\boldsymbol{\mu}_{\beta,\phi}(\sigma) \propto \exp\left(\beta \textstyle\sum_{\{w,u\} \in E} \mathbf{1}\{\sigma(u) = \sigma(w)\} \cdot \boldsymbol{J}_{\{u,w\}} \right) \quad , \tag{6}$$

where $\propto$ stands for "proportional to".

At this point, it is immediate that by choosing ϕ to be the *standard Gaussian* distribution, we retrieve the Edwards-Anderson model in (1). Note however, that (6) above generates a whole family of "spin-glass" distributions with the EA-model being a special case.

The definition of the distribution of the broadcasting matrix in (5) allows us to derive an explicit formula for the quantity Δ_{KS} in (4). Specifically, for $\boldsymbol{J}$ distributed according to ϕ, it is not hard to prove (see the full version) that

$$\Delta_{\mathrm{KS}}(\beta,\phi)=\left(\mathbb{E}\left[\left(\frac{1-\exp(\beta\boldsymbol{J})}{1+\exp(\beta\boldsymbol{J})}\right)^2\right]\right)^{-1}\ , \tag{7}$$

where the expectation is with respect to the random variable $\boldsymbol{J}$. In light of the above, we prove the following result for the general Gibbs distribution.

▶ **Theorem 3.** *For a real number $\beta>q0$, and a distribution ϕ on the real numbers $\mathbb{R}$ let $\Delta_{\mathrm{KS}}=\Delta_{\mathrm{KS}}(\beta,\phi)$ be defined as in* (7)*.*

For any integer $\Delta>\Delta_{\mathrm{KS}}$, the Gibbs distribution $\boldsymbol{\mu}_{\beta,\phi}$, defined as in (6)*, on the Δ-ary tree exhibits reconstruction. On the other hand, if $\Delta<\Delta_{\mathrm{KS}}$ the distribution $\boldsymbol{\mu}_{\beta,\phi}$ exhibits non-reconstruction.*

The proof of Theorem 3 appears in the full version. Let us state the implications of Theorem 3 for the Edwards-Anderson model on the Δ-ary tree.

▶ **Corollary 4.** *For $\beta>0$ and the standard Gaussian $\boldsymbol{J}$, let*

$$\Delta_{\mathrm{EA}}(\beta)=\left(\mathbb{E}\left[\left(\frac{1-\exp(\beta\boldsymbol{J})}{1+\exp(\beta\boldsymbol{J})}\right)^2\right]\right)^{-1}\ ,$$

where the expectation is with respect to $\boldsymbol{J}$.

For any integer $\Delta>\Delta_{\mathrm{EA}}(\beta)$, the distribution $\boldsymbol{\mu}_{\beta}$, the Edwards-Anderson model with inverse temperature β on the Δ-ary tree, exhibits reconstruction. On the other hand, if $\Delta<\Delta_{\mathrm{EA}}(\beta)$ the distribution $\boldsymbol{\mu}_{\beta}$ exhibits non-reconstruction.

2.1 The case of the Galton-Watson tree

As a further step, we study the reconstruction problem on the Galton-Watson tree. Even though this is a very interesting problem on its own, we make use of our results for the Galton-Watson tree to derive subsequent results for $\boldsymbol{G}(n,d/n)$, see Section 2.2.

Let $\zeta:\mathbb{Z}_{\geq 0}\to[0,1]$ be a distribution over the non-negative integers. Then, the rooted tree $\boldsymbol{T}$ is a Galton-Watson tree with offspring distribution ζ, if the number of children for each vertex in $\boldsymbol{T}$ is distributed according to ζ, *independently* from the other vertices.

Note that broadcasting with random matrices over the Galton-Watson tree $\boldsymbol{T}$, gives rise to configurations that consist of *three* levels of randomness. One of the challenges we circumvent with our analysis, is to disentangle all of three levels of randomness, and make clear the contribution of each one of them. Before getting there, we need to clarify what we mean by (non-)reconstruction in the current setting.

▶ **Definition 5.** *Consider the distributions ϕ over $\mathbb{R}$ and ζ over $\mathbb{Z}_{\geq 0}$, and a real number $\beta\geq 0$. Let the Galton-Watson tree $\boldsymbol{T}$ with offspring distribution ζ, while let the measure $\boldsymbol{\mu}=\boldsymbol{\mu}_{\beta,\phi}$ be defined as in* (6)*, on the tree $\boldsymbol{T}$. We say that $\boldsymbol{\mu}$ exhibits* reconstruction *if*

$$\limsup_{h\to\infty}\mathbb{E}_{\boldsymbol{T}}\left[\ \mathbb{E}_{\boldsymbol{\mu}}\left[||\boldsymbol{\mu}_h(\cdot\mid\boldsymbol{\sigma}(r)=+1)-\boldsymbol{\mu}_h(\cdot\mid\boldsymbol{\sigma}(r)=-1)||_{\mathrm{TV}}\ \mid\boldsymbol{T}\right]\ \right]>0\ .$$

On the other hand, if the above limit is zero, then we have non-reconstruction*.*

For the above, recall that $\boldsymbol{\mu}_h$ is the marginal of $\boldsymbol{\mu}$ on the set of vertices at distance h from the root. Note that if $\boldsymbol{T}$ has no vertex at level h, then the total variation distance above is, degenerately, equal to zero. We use the double expectation in Definition 5 for the sake of clarity: we can just replace it by a single expectation with respect to both the random tree $\boldsymbol{T}$, and the random measure $\boldsymbol{\mu}$.

As far as the reconstruction problem on the Galton-Watson trees is concerned, we have the following result.

▶ **Theorem 6.** *For any real numbers $d > 0, \beta > 0$, for any distribution ϕ on $\mathbb{R}$, for any distribution ζ on $\mathbb{Z}_{\geq 0}$ with expectation d, and* bounded second moment*, let $\boldsymbol{T}$ be the Galton-Watson tree with offspring distribution ζ. Let also $\boldsymbol{\mu}_{\beta,\phi}$ be the Gibbs distribution defined as in* (6)*, on the tree $\boldsymbol{T}$. Finally, let $\Delta_{\mathrm{KS}} = \Delta_{\mathrm{KS}}(\beta, \phi)$ be defined as in* (7)*.*

The distribution $\boldsymbol{\mu}_{\beta,\phi}$ exhibits reconstruction if $d > \Delta_{\mathrm{KS}}$. On the other hand, if $d < \Delta_{\mathrm{KS}}$, the distribution $\boldsymbol{\mu}_{\beta,\phi}$ exhibits non-reconstruction.

Let us now state the implications of Theorem 6 for the Edwards-Anderson model on the Galton-Watson tree.

▶ **Corollary 7.** *For $\beta > 0$, consider the quantity $\Delta_{\mathrm{EA}}(\beta)$ defined in Corollary 4. For any real number $d > 0$, and any distribution $\zeta : \mathbb{Z}_{\geq 0} \to [0, 1]$ with expectation d, and* bounded second moment*, let $\boldsymbol{T}$ be the Galton-Watson tree with offspring distribution ζ.*

Then, for $\boldsymbol{\mu}_\beta$ the Edwards-Anderson model with inverse temperature β, on the tree $\boldsymbol{T}$, the following is true. The distribution $\boldsymbol{\mu}_\beta$ exhibits reconstruction if $d > \Delta_{\mathrm{EA}}(\beta)$. On the other hand, if $d < \Delta_{\mathrm{EA}}(\beta)$, the distribution $\boldsymbol{\mu}_\beta$ exhibits non-reconstruction.

2.2 The Edwards-Anderson model on $G(n, d/n)$

For integer $n \geq 1$, and real $p \in [0, 1]$, let $\boldsymbol{G} = \boldsymbol{G}(n, p)$ be the random graph on $V_n = \{x_1, \ldots, x_n\}$, whose edge set $E(\boldsymbol{G})$ is obtained by including each edge with probability, p *independently*.

The *Edwards-Anderson model* on $\boldsymbol{G}$ at inverse temperature $\beta > 0$, is defined as follows: for $\boldsymbol{J} = \{\boldsymbol{J}_e\}_{e \in E(\boldsymbol{G})}$ a family of independent *standard Gaussians*, we let

$$\mu_{\boldsymbol{G},\boldsymbol{J},\beta}(\sigma) = \tfrac{1}{Z_\beta(\boldsymbol{G},\boldsymbol{J})} \exp\left(\beta \textstyle\sum_{x \sim y} \mathbf{1}\{\sigma(y) = \sigma(x)\} \cdot \boldsymbol{J}_{\{x,y\}}\right) \ , \tag{8}$$

where

$$Z_\beta(\boldsymbol{G}, \boldsymbol{J}) = \textstyle\sum_{\tau \in \{\pm 1\}^{V_n}} \exp\left(\beta \sum_{x \sim y} \mathbf{1}\{\tau(y) = \tau(x)\} \cdot \boldsymbol{J}_{\{x,y\}}\right) \ .$$

Here we assume that $p = \frac{d}{n}$, where $d > 0$ is a fixed number. Typically, we study this distribution as $n \to \infty$. The natural question we ask here is how does the model change as we vary d. According to the physics predictions, for any β there exists a *condensation threshold*, denoted as $d_{\mathrm{cond}}(\beta)$, where the function

$$d \mapsto \lim_{n \to \infty} \frac{1}{n} \mathbb{E}[\ln Z_\beta(\boldsymbol{G}, \boldsymbol{J})]$$

is non-analytic [19]. This conjecture was proved by Guerra and Toninelli [21]. The regime $d < d_{\mathrm{cond}}(\beta)$ is called the *replica symmetric phase*. This region has several interesting properties; here we consider one that seems to be most relevant to our discussion. For any $d < d_{\mathrm{cond}}(\beta)$ the distribution $\mu_{\boldsymbol{G},\boldsymbol{J},\beta}$ satisfies the following property: for $\boldsymbol{\sigma}$ distributed as

in $\mu_{\boldsymbol{G},\boldsymbol{J},\beta}$, for two randomly chosen vertices $\mathbf{x}$ and $\mathbf{y}$, the configurations $\boldsymbol{\sigma}(\mathbf{x})$ and $\boldsymbol{\sigma}(\mathbf{y})$ are asymptotically independent. Formally, the above can be expressed as follows: for $d < d_{\text{cond}}(\beta)$ and any $i, j \in \{\pm 1\}$, we have that

$$\limsup_{n\to\infty} \frac{1}{n^2} \sum_{x,y\in V_n} \mathbb{E}\left[\langle \mathbf{1}\{\boldsymbol{\sigma}(x) = i\} \times \mathbf{1}\{\boldsymbol{\sigma}(y) = j\}\rangle - \langle \mathbf{1}\{\boldsymbol{\sigma}(x) = i\}\rangle \times \langle \mathbf{1}\{\boldsymbol{\sigma}(y) = j\}\rangle\right] = 0 \ ,$$

where $\langle\cdot\rangle$ denotes expectation with respect to the Gibbs distribution $\mu_{\boldsymbol{G},\boldsymbol{J},\beta}$. Note that the above holds not only for pairs of vertices, but also for sets of k vertices, for any fixed integer $k > 0$. Using our notation, the work by Guerra and Toninelli [21] implies the following result.

▶ **Theorem 8** ([21]). *For any $\beta > 0$, for the distribution $\mu_{\boldsymbol{G},\boldsymbol{J},\beta}$ defined as in* (8)*, we have that*

$$d_{\text{cond}}(\beta) = \left(\mathbb{E}\left[\left(\frac{1-\exp(\beta\boldsymbol{J})}{1+\exp(\beta\boldsymbol{J})}\right)^2\right]\right)^{-1} \ ,$$

where $\boldsymbol{J}$ is a standard Gaussian random variable.

Interestingly, one obtains the above by combining our Theorem 6 and using results from [12, 13]. Our main focus is on the reconstruction threshold for the Edwards-Anderson model on $\boldsymbol{G}$. The reconstruction for $\mu_{\boldsymbol{G},\boldsymbol{J},\beta}(\cdot)$ is defined in a slightly different way than what we have for the random tree.

▶ **Definition 9.** *For $d > 0$, for $\beta > 0$, consider the Gibbs distribution $\mu_{\boldsymbol{G},\boldsymbol{J},\beta}$ as this is defined in* (8)*. We say that the measure $\mu = \mu_{\boldsymbol{G},\boldsymbol{J},\beta}$ exhibits* reconstruction *if*

$$\limsup_{h\to\infty} \lim_{n\to\infty} \frac{1}{n} \sum_{x\in V_n} \mathbb{E}\left[||\mu_{x,h}(\cdot \mid \boldsymbol{\sigma}(x) = +1) - \mu_{x,h}(\cdot \mid \boldsymbol{\sigma}(x) = -1)||_{\text{TV}}\right] > 0 \ ,$$

where $\mu_{x,h}$ denote the Gibbs marginal at the vertices at distance h from vertex x. On the other hand, if the above limit is zero, then we have non-reconstruction*.*

Perhaps, it is interesting to notice the order with which we take the double limit in the above definition. We let the reconstruction threshold, denoted as d_{recon}, to be the infimum over $d > 0$ such that

$$\limsup_{h\to\infty} \lim_{n\to\infty} \frac{1}{n} \sum_{x\in V_n} \mathbb{E}\left[||\mu_h(\cdot \mid \boldsymbol{\sigma}(x) = +1) - \mu_h(\cdot \mid \boldsymbol{\sigma}(x) = -1)||_{\text{TV}}\right] > 0 \ .$$

The region of values of d such that $d < d_{\text{recon}}$ is called the *non-reconstruction phase*. It is immediate from Definition 9 that, for any $d < d_{\text{recon}}$, we have that non-reconstruction.

In the following result, we prove that the replica symmetric phase coincides with the non-reconstruction phase of the Edwards-Anderson model on $\boldsymbol{G}$.

▶ **Theorem 10.** *For any $\beta > 0$, for the distribution $\mu_{\boldsymbol{G},\boldsymbol{J},\beta}$ defined as in* (8)*, we have that $d_{\text{recon}}(\beta) = d_{\text{cond}}(\beta)$.*

The above follows from Theorems 8, 7 and [12, Corollary 1.5].

Notation

For the graph $G = (V, E)$ and the Gibbs distribution μ on the set of configurations $\{\pm 1\}^V$. For a configuration σ, we let $\sigma(\Lambda)$ denote the configuration that σ specifies on the set of vertices Λ. We let μ_Λ denote the marginal of μ at the set Λ. We let $\mu(\cdot \mid \Lambda, \sigma)$, denote the distribution μ conditional on the configuration at Λ being σ. Also, we interpret the conditional marginal $\mu_\Lambda(\cdot \mid \Lambda', \sigma)$, for $\Lambda' \subseteq V$, in the natural way.

3 Approach

A major challenge in our setting is that we have to deal with multiple levels of randomness, i.e., we have two levels of randomness in the case of the Δ-ary tree, while the levels increase with the Galton-Watson trees. To circumvent this problem, we follow an analysis that allows us to disentangle the different sources of randomness in our models. In this section, we provide a high-level description of our approach. We restrict our discussion on the Δ-ary tree.

Non-reconstruction

Consider the Δ-ary tree $T = (V, E)$ rooted at r. Suppose that we have a distribution μ as in (6) on T, while assume that each edge $e \in E$ has its own coupling parameter J_e. Assume, for the moment, that the coupling parameters at the edges are fixed, e.g. the reader may assume that are arbitrary real numbers. That is, each J_e can be either positive, or negative. Hence, one might consider the aforementioned distribution as a *non-homogenous* Ising model which involves both ferromagnetic and anti-ferromagnetic interactions. Let us focus on non-reconstruction. We derive an upper bound on

$$||\mu_h(\cdot \mid \boldsymbol{\sigma}(r) = +1) - \mu_h(\cdot \mid \boldsymbol{\sigma}(r) = -1)||_{\text{TV}} \quad ,$$

which is expressed in terms of the *influence* between neighbouring vertices. The notion of influence between vertices is the same as the one developed in the context of *Spectral Independence* technique for establishing rapid mixing of Glauber dynamics [3, 9]. These influences are used in the context of the so-called *down-up* coupling to establish non-reconstruction. This is a coupling approach from [6], which also relies on ideas in [29].

Let us be more specific. For the probability measure μ we consider, let R_r be the *ratio of Gibbs marginals* at the root r defined by

$$R_r = \frac{\mu_r(+1)}{\mu_r(-1)} \quad . \tag{9}$$

Recall that $\mu_r(\cdot)$ denotes the marginal of the Gibbs distribution $\mu(\cdot)$ at the root r. For a vertex $u \in V$, we let T_u be the subtree of T that includes u, and all its descendants. Also, we let R_u be the ratio of marginals at vertex u, where the Gibbs distribution is, now, with respect to the subtree T_u.

Suppose that the vertices $w_1, \ldots, w_\Delta$ are the children of the root r. Our focus is on expressing $\log R_r$ recursively, as a function of $\log R_{w_1}, \ldots, \log R_{w_\Delta}$. Note that we study the logarithm of the ratios involved, which can be viewed as applying the potential function $\log(\cdot)$ to the tree recursions. We have that $\log(R_r) = H(\log R_{w_1}, \ldots, \log R_{w_\Delta})$ where

$$H(x_1, x_2, \ldots, x_\Delta) = \sum_{i=1}^{\Delta} \log\left(\frac{\exp\left(x_i + \beta J_{\{r,w_i\}}\right) + 1}{\exp(x_i) + \exp\left(\beta J_{\{r,w_i\}}\right)}\right) \quad . \tag{10}$$

Note that $J_{\{r,w_i\}}$ is the coupling parameter that corresponds to the edge between the root r with its child w_i. All the above extends naturally in the case where we impose boundary conditions. That is, for a region $K \subseteq V$, and $\tau \in \{\pm 1\}^K$, we define the ratio of marginals $R_r^{K,\tau}$ at the root, where now the ratio is between the conditional marginals $\mu_r(+1 \mid K, \tau)$ and $\mu_r(-1 \mid K, \tau)$. The recursive function H for the conditional ratios is exactly the same as the one above.

Our interest is on the *gradient* of the function H. Specifically, for every $i \in [\Delta]$, we let

$$\Gamma_{\{r,w_i\}} = \sup_{x_1,\ldots,x_\Delta} \left| \frac{\partial}{\partial x_i} H(x_1, x_2, \ldots, x_\Delta) \right| \ . \tag{11}$$

It turns out that, in our case, $\Gamma_{\{r,w_i\}}$ has a simple form

$$\Gamma_{\{r,w_i\}} = \frac{\left|1 - \exp\left(\beta J_{\{r,w\}}\right)\right|}{1 + \exp\left(\beta J_{\{r,w_i\}}\right)} \ .$$

Utilising the idea of down-up coupling from [6], we prove the following:

$$||\mu_h(\cdot \mid \boldsymbol{\sigma}(r) = +1) - \mu_h(\cdot \mid \boldsymbol{\sigma}(r) = -1)||_{TV} \leq \sqrt{\sum_{v \in \Lambda} \prod_{e \in \text{path}(r,v)} \Gamma_e^2} \ , \tag{12}$$

where $\Lambda = \Lambda(h)$ denotes the set of vertices at distance h from the root r. Note that the above provides a bound for the total variation distance of the the marginals for fixed, i.e., non-random, couplings $\{J_e\}_{e \in E}$. Inequality (12), extends naturally when we study reconstruction for the distribution $\boldsymbol{\mu}$ defined in (6), i.e., when the coupling parameters $\boldsymbol{J}_e$ are i.i.d. samples from a distribution ϕ. Indeed, averaging yields

$$\mathbb{E}\left[\left(||\boldsymbol{\mu}_h(\cdot \mid \boldsymbol{\sigma}(r) = +1) - \boldsymbol{\mu}_h(\cdot \mid \boldsymbol{\sigma}(r) = -1)||_{TV}\right)^2\right] \leq \sum_{v \in \Lambda} \prod_{e \in \text{path}(r,v)} \mathbb{E}\left[\boldsymbol{\Gamma}_e^2\right] \ , \tag{13}$$

where we have $\boldsymbol{\Gamma}_e = \frac{|1-\exp(\beta \boldsymbol{J}_e)|}{1+\exp(\beta \boldsymbol{J}_e)}$, for each $e \in E$. Note that the above holds, since each $\boldsymbol{\Gamma}_e$ depends only on $\boldsymbol{J}_e$, while the coupling parameters $\boldsymbol{J}_e$ are assumed to be independent with each other.

At this point, and since the $\boldsymbol{J}_e$'s are identically distributed, we further observe that for any $e \in E$, we have that

$$\Delta_{\text{KS}}(\beta, \phi) = \left(\mathbb{E}\left[\boldsymbol{\Gamma}_e^2\right]\right)^{-1} \ .$$

Since the underlying tree T is Δ-ary, it is immediate to see that for $\Delta < \Delta_{\text{KS}}(\beta, \phi)$, the r.h.s. of (13) tends to zero as $h \to \infty$. From this point on, it is standard to prove non-reconstruction.

Our analysis allows to deal with the randomness of the spin-glass measure $\boldsymbol{\mu}$ by utilising the bound in (12). That is, the upper bound on the total variation distance has a nice *product* form of the quantities Γ_e, which, in turn, expresses the dependence of the total variation distance on the edge couplings $\{J_e\}_{e \in E}$. This product form of the bound, behaves rather nicely when we need to take averages over the randomness of the coupling parameters $\{\boldsymbol{J}_e\}_{e \in E}$ of the the spin-glass measure $\boldsymbol{\mu}$.

Reconstruction

In the reconstruction regime, the configuration at the root has a non-vanishing effect on the configuration of the vertices at distance h, regardless of the height h. Specifically, the corresponding leaf configurations from the measure conditioned on root's spin being $+1$, and -1, are so different with each other, that discrepancies cannot be attributed to random fluctuations. Therefore, a question that naturally arises is how can we take advantage of the discrepancies so that we infer the spin of the root.

For the standard ferromagnetic Ising, several approaches have been developed to establish reconstruction (see [18], [8], [23]). Here, we build on an elegant argument in [18]. The authors in this work, show that a simple *majority vote* of the leaf spins, conveys information sufficient to reconstruct root's spin, The majority vote on the leaves is defined by

$$M_h = \sum_{u \in \Lambda} \sigma(u) \ . \tag{14}$$

The estimation rule is to infer that the spin at the root is $\mathrm{sgn}\{M_h\}$, i.e., the sign of M_h. Impressively, it turns out that this estimator is optimal, i.e., it coincides with the *maximum likelihood* one. For the Δ-ary tree, one establishes reconstruction for the ferromagnetic Ising model by employing a second moment argument on the estimator M_h.

For the distributions we consider here, the above estimator is far from sufficient. This is due to various facts. Firstly, we allow for mixed couplings on the edges, i.e., certain edges can be ferromagnetic, and others can be anti-ferromagnetic. Secondly, the strength of the interaction, i.e., the magnitude of $\boldsymbol{J}_e$'s, is expected to vary from one edge to the other. To this end, we introduce a new estimator, and we establish reconstruction by building on the second moment argument from [18]. The starting point towards deriving this estimator, comes from just considering the standard anti-antiferromagnetic Ising. The statistic from (14), clearly does not work for this distribution. However, there is an easy remedy, by taking into account the parity of the height h, i.e., if h is an even, or an odd number. We infer that the spin at the root is equal to $\mathrm{sgn}\left\{\widehat{M}_h\right\}$, where

$$\widehat{M}_h = (-1)^h \sum_{u\in\Lambda} \sigma(u) \ .$$

For the spin-glass distributions we consider here, we need to get the above idea even further. Firstly, in order to accommodate the *mixed* ferromagnetic and anti-ferromagnetic couplings on the edges of the tree. It seems meaningful to use the estimator $\mathrm{sgn}\left\{\widetilde{M}_h\right\}$ for the root configuration, where

$$\widetilde{M}_h = \sum_{u\in\Lambda} \sigma(u) \prod_{e\in\mathrm{path}(r,u)} \mathrm{sign}\{\boldsymbol{J}_e\} \ ,$$

with $\mathrm{path}(r,u)$ denoting the set of edges along the unique path connecting r to u. So that in $\widetilde{M}_h$, for each leaf we essentially examine the parity of the number of antiferromagnetic couplings along the path that connects it to the root. Unfortunately, for the above estimator, our second moment argument does not seem to work all that well.

The estimator we end up using, is a *reweighted* version of $\widetilde{M}_h$, which we call the "flip majority" vote, and is defined by

$$F_h = \sum_{u\in\Lambda} \sigma(u) \prod_{e\in\mathrm{path}(r,u)} \tfrac{1-\exp(\beta\boldsymbol{J}_e)}{1+\exp(\beta\boldsymbol{J}_e)} \ .$$

Note that the absolute value of the weight for the edge e, above, coincides with the quantity $\boldsymbol{\Gamma}_e$ in (13). Naturally, the estimation rule is to infer that the root spin is $\mathrm{sgn}\{F_h\}$.

References

1 Dimitris Achlioptas and Amin Coja-Oghlan. Algorithmic barriers from phase transitions. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 793–802. IEEE, 2008.

2 Ahmed El Alaoui, Andrea Montanari, and Mark Sellke. Sampling from the Sherrington-Kirkpatrick Gibbs measure via algorithmic stochastic localization. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 323–334. IEEE, 2022. doi:10.1109/FOCS54457.2022.00038.

3 Nima Anari, Kuikui Liu, and Shayan Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. *SIAM Journal on Computing*, 0(0):FOCS20–1–FOCS20–37, 2021. doi:10.1137/20M1367696.

4 Victor Bapst, Amin Coja-Oghlan, and Charilaos Efthymiou. Planting colourings silently. *Combinatorics, probability and computing*, 26(3):338–366, 2017.

5 Nayantara Bhatnagar, Allan Sly, and Prasad Tetali. Reconstruction threshold for the hardcore model. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques: 13th International Workshop, APPROX 2010, and 14th International Workshop, RANDOM 2010, Barcelona, Spain, September 1-3, 2010. Proceedings*, pages 434–447. Springer, 2010.

6 Nayantara Bhatnagar, Juan Vera, Eric Vigoda, and Dror Weitz. Reconstruction for colorings on trees. *SIAM Journal on Discrete Mathematics*, 25(2):809–826, 2011.

7 Pavel M Bleher, Jean Ruiz, and Valentin A Zagrebnov. On the purity of the limiting Gibbs state for the Ising model on the Bethe lattice. *Journal of Statistical Physics*, 79:473–482, 1995.

8 Christian Borgs, Jennifer Chayes, Elchanan Mossel, and Sébastien Roch. The Kesten-Stigum reconstruction bound is tight for roughly symmetric binary channels. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 518–530. IEEE, 2006.

9 Zongchen Chen, Kuikui Liu, and Eric Vigoda. Rapid mixing of Glauber dynamics up to uniqueness via contraction. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1307–1318. IEEE, 2020.

10 Amin Coja-Oghlan and Charilaos Efthymiou. On independent sets in random graphs. *Random Structures & Algorithms*, 47(3):436–486, 2015.

11 Amin Coja-Oghlan, Charilaos Efthymiou, and Nor Jaafari. Local convergence of random graph colorings. *Combinatorica*, 38(2):341–380, 2018.

12 Amin Coja-Oghlan, Charilaos Efthymiou, Nor Jaafari, Mihyun Kang, and Tobias Kapetanopoulos. Charting the replica symmetric phase. *Communications in Mathematical Physics*, 359:603–698, 2018.

13 Amin Coja-Oghlan, Andreas Galanis, Leslie Ann Goldberg, Jean Bernoulli Ravelomanana, Daniel Stefankovic, and Eric Vigoda. Metastability of the Potts Ferromagnet on Random Regular Graphs. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 45:1–45:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.45.

14 Amin Coja-Oghlan, Tobias Kapetanopoulos, and Noela Müller. The replica symmetric phase of random constraint satisfaction problems. *Combinatorics, Probability and Computing*, 29(3):346–422, 2020.

15 Amin Coja-Oghlan, Florent Krzakala, Will Perkins, and Lenka Zdeborová. Information-theoretic thresholds from the cavity method. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 146–157, 2017.

16 Samuel Frederick Edwards and Phil W Anderson. Theory of spin glasses. *Journal of Physics F: Metal Physics*, 5(5):965, 1975.

17 Charilaos Efthymiou. On Sampling Symmetric Gibbs Distributions on Sparse Random Graphs and Hypergraphs. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 57:1–57:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.57.

18 William Evans, Claire Kenyon, Yuval Peres, and Leonard J Schulman. Broadcasting on trees and the Ising model. *Annals of Applied Probability*, pages 410–433, 2000.

19 Silvio Franz, Michele Leone, Federico Ricci-Tersenghi, and Riccardo Zecchina. Exact solutions for diluted spin glasses and optimization problems. *Physical review letters*, 87(12):127209, 2001.

20 Hans-Otto Georgii. *Gibbs measures and phase transitions*, volume 9. Walter de Gruyter, 2011.

21 Francesco Guerra and Fabio Lucio Toninelli. The high temperature region of the Viana-Bray diluted spin glass model. *Journal of statistical physics*, 115:531–555, 2004.

22 Yasunari Higuchi. Remarks on the limiting Gibbs states on a (d+ 1)-tree. *Publications of the Research Institute for Mathematical Sciences*, 13(2):335–348, 1977.

23 Dmitry Ioffe. On the extremality of the disordered state for the Ising model on the Bethe lattice. *Letters in Mathematical Physics*, 37:137–143, 1996.

24 Harry Kesten and Bernt P Stigum. Additional limit theorems for indecomposable multidimensional Galton-Watson processes. *The Annals of Mathematical Statistics*, 37(6):1463–1481, 1966.

25 Marc Mézard and Andrea Montanari. Reconstruction on trees and spin glass transition. *Journal of statistical physics*, 124:1317–1350, 2006.

26 Michael Molloy. The freezing threshold for k-colourings of a random graph. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 921–930, 2012.

27 Giorgio Parisi. Infinite number of order parameters for spin-glasses. *Physical Review Letters*, 43(23):1754, 1979.

28 David Sherrington and Scott Kirkpatrick. Solvable model of a spin-glass. *Physical review letters*, 35(26):1792, 1975.

29 Allan Sly. Reconstruction of random colourings. *Communications in Mathematical Physics*, 288(3):943–961, 2009.

30 Daniel L Stein and Charles M Newman. *Spin glasses and complexity*, volume 4. Princeton University Press, 2013.

31 Lenka Zdeborová and Florent Krzakala. Statistical physics of inference: Thresholds and algorithms. *Advances in Physics*, 65(5):453–552, 2016.

Improved Mixing for the Convex Polygon Triangulation Flip Walk

David Eppstein
Department of Computer Science, University of California, Irvine, CA, USA

Daniel Frishberg
Department of Computer Science, University of California, Irvine, CA, USA

Abstract

We prove that the well-studied *triangulation flip walk* on a convex point set mixes in time $O(n^3 \log^3 n)$, the first progress since McShine and Tetali's $O(n^5 \log n)$ bound in 1997. In the process we give lower and upper bounds of respectively $\Omega(1/(\sqrt{n} \log n))$ and $O(1/\sqrt{n})$ – asymptotically tight up to an $O(\log n)$ factor – for the *expansion* of the *associahedron* graph K_n. The upper bound recovers Molloy, Reed, and Steiger's $\Omega(n^{3/2})$ bound on the mixing time of the walk. To obtain these results, we introduce a framework consisting of a set of sufficient conditions under which a given Markov chain mixes rapidly. This framework is a purely combinatorial analogue that in some circumstances gives better results than the *projection-restriction* technique of Jerrum, Son, Tetali, and Vigoda. In particular, in addition to the result for triangulations, we show quasipolynomial mixing for the *k-angulation* flip walk on a convex point set, for fixed $k \geq 4$.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases associahedron, mixing time, mcmc, Markov chains, triangulations, quadrangulations, k-angulations, multicommodity flow, projection-restriction

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.56

Category Track A: Algorithms, Complexity and Games

Related Version This paper includes results from two of our arXiv preprints:
Previous Version: `https://arxiv.org/abs/2207.09972v1`
Previous Version: `https://arxiv.org/abs/2111.03898`
Full Version: `https://arxiv.org/abs/2207.09972`

1 Introduction and background

The study of *mixing times* – the art and science of proving upper and lower bounds on the efficiency of Markov chain Monte Carlo sampling methods – is a well-established area of research, of interest for combinatorial sampling problems, spin systems in statistical physics, probability, and the study of subset systems. Work in this area brings together techniques from spectral graph theory, combinatorics, and probability, and dates back decades; for a comprehensive survey of classic methods, results, and open questions see the canonical text by Levin, Wilmer, and Peres [27]. Recent breakthroughs [1, 2, 3, 8, 9, 10, 24, 26] – incorporating techniques from the theory of abstract simplicial complexes – have led to a recent slew of results for the mixing times of combinatorial chains for sampling independent sets, matchings, Ising model configurations, and a number of other structures in graphs, injecting renewed energy into an already active area.

We focus on a class of *geometric* sampling problems that has received considerable attention from the counting and sampling [4, 22] and mixing time [29, 31, 35, 6] research communities over the last few decades, but for which tight bounds have been elusive: sampling *triangulations*. A triangulation is a maximal set of non-crossing edges connecting pairs of points (see Figure 1) in a given n-point set. Every pair of triangles sharing an edge forms a

quadrilateral. A triangulation *flip* consists of removing such an edge, and replacing it with the only other possible diagonal within the same quadrilateral. Flips give a natural Markov chain (the *flip walk*): one selects a uniformly random diagonal from a given triangulation and (if possible) flips the diagonal.

McShine and Tetali gave a classic result in a 1997 paper [29], showing that in the special case of a convex two-dimensional point set (a convex n-gon), the flip walk *mixes* (converges to approximately uniform) in time $O(n^5 \log n)$, improving on the best-known prior (and first polynomial) upper bound, $O(n^{25})$, by Molloy, Reed, and Steiger [31]. McShine and Tetali applied a Markov chain comparison technique due to Diaconis and Saloff-Coste [12] and to Randall and Tetali [32] to obtain their bound, using a bijection between triangulations and a structure known as *Dyck paths*. They noted that they could not improve on this bound using this bijection. Furthermore, they believed that an earlier *lower* bound of $\Omega(n^{3/2})$, also by Molloy, Reed, and Steiger [31], should be tight. We show the following result (see Section 3 for the precise definition of mixing time):

▶ **Theorem 1.** *The triangulation flip walk on the convex $n+2$-point set mixes in time $O(n^3 \log^3 n)$.*

Prior to the present paper, no progress had been made either on upper or lower bounds for this chain in 25 years – even as new polynomial upper bounds and exponential lower bounds were given for other geometric chains, from lattice point set triangulations [35, 6] to quadrangulations of planar maps [7], and despite many breakthroughs using the newer techniques for other problems.

In addition to this specific result, we give a general decomposition theorem – which we will state as Theorem 13 once we have built up enough preliminaries, for bounding mixing times by recursively decomposing the state space of a Markov chain. This theorem is a purely combinatorial alternative to the spectral result of Jerrum, Son, Tetali, and Vigoda [21].

1.1 Decomposition framework

To prove our result, we develop a general decomposition framework that applies to a broad class of Markov chains, as an alternative to prior work by Jerrum, Son, Tetali, and Vigoda [21] that used spectral methods. We obtain our new mixing result for triangulations, then generalize our technique to obtain the first nontrivial mixing result for k-angulations. In a companion paper [15] we further generalize this work to obtain the first rapid mixing bounds for Markov chains for sampling independent sets, dominating sets, and *b-edge covers* (generalizing edge covers) in graphs of bounded treewidth, and for maximal independent sets, b-matchings, and *maximal b-matchings* in graphs of bounded treewidth and degree. In that work we also strengthen existing results [18, 14] for proper q-colorings in graphs of bounded treewidth and degree.

The key observation that unifies these chains is that, when viewing their state spaces as graphs (exponentially large graphs relative to the input), they all admit a recursive decomposition satisfying key properties. First, each such graph, called a "flip graph," can be partitioned into a small number of induced subgraphs, where each subgraph is a *Cartesian product* of smaller graphs that are structurally similar to the original graph – and thus can be partitioned again into even smaller product graphs. Second, at each level of recursion, pairs of subgraphs are connected by large matchings. Intuitively, we can "slice" a flip graph into subgraphs that are well connected to each other, then "peel" apart the subgraphs using their Cartesian product structure, and repeat the process recursively. Each recursive level of slicing cuts through many edges (the large matchings), and indeed the peeling also

disconnects many mutually well-connected subgraphs from one another. Prior work exists applying this "slicing" and "peeling" paradigm – albeit with spectral methods instead of purely combinatorial methods – using Jerrum, Son, Tetali, and Vigoda's decomposition theorem (Theorem 14) for combinatorial chains [21, 18, 14]. One of our contributions is to unify these applications, along with the geometric chains, into a sufficient set of conditions under which one can apply the existing decomposition theorem: Lemma 15.

A more substantial technical contribution is our Theorem 13, a combinatorial analogue to Jerrum, Son, Tetali, and Vigoda's Theorem 14. One can use our theorem in place of theirs and, in some cases, obtain better mixing bounds. In particular, in the case of triangulations, we obtain polynomial mixing via an adaptation of our (combinatorial) technique (Lemma 19) – and it is not clear how to adapt the existing spectral methods to get even a polynomial bound. In the case of k-angulations, our theorem gives a bound that has better dependence on the parameter k.

1.2 Paper organization

In the remainder of this section we will define the Markov chains we are analyzing and summarize our main results. Then, in Section 2, we will give intuition for the decomposition by describing its application to triangulations. In Section 4 we will present our general decomposition meta-theorems, and compare our contribution to prior work by Jerrum, Son, Tetali, and Vigoda [21]. In particular, we will discuss why our purely combinatorial machinery is needed for obtaining new bounds in the case of triangulations. In the full version of our paper [16] we will prove a general result that gives a coarse bound on triangulation mixing; we will then improve this bound to near tightness in the full paper version, and give a matching upper bound (up to logarithmic factors) in the full version. Also in the full version, we show that general k-angulations admit a decomposition satisfying a relaxation (Lemma 18) of our general theorem that implies quasipolynomial-time mixing. We analyze the particular quasipolynomial bound we obtain, and show that our combinatorial technique (Theorem 13) gives a better dependence on k than one would obtain with the prior decomposition theorem. In the full version of the paper we prove our general combinatorial decomposition theorem, Theorem 13. In the full version we prove a theorem about lattice triangulations, and fill in a few remaining proof details.

1.3 Triangulations of convex point sets and lattice point sets

Let P_n be the regular polygon with n vertices. Every triangulation t of P_{n+2} has $n-1$ diagonals, and every diagonal can be *flipped*: every diagonal D belongs to two triangles forming a convex quadrilateral, so D can be removed and replaced with the diagonal D' lying in the same quadrilateral and crossing D. The set of all triangulations of P_{n+2}, for $n \geq 1$, is the vertex set of a graph that we denote K_n (this notation is standard), whose edges are the flips between adjacent triangulations. The graph K_n is known to be realizable as the 1-skeleton of an $n-1$-dimensional polytope [28] called the *associahedron* (we also use this name for the graph itself). It is also known to be isomorphic to the rotation graph on the set of all binary plane trees with $n+1$ leaves [34], and equivalently the set of all parenthesizations of an algebraic expression with $n+1$ terms, with "flips" defined as applications of the associative property of multiplication.

The structure of this graph depends only on the convexity and the number of vertices of the polygon, and not on its precise geometry. That is, P_{n+2} need not be regular for K_n to be well defined.

McShine and Tetali [29] showed that the *mixing time* (see Section 3) of the uniform random walk on $K_{3,n+2}$ is $O(n^5 \log n)$, following Molloy, Reed, and Steiger's [31] lower bound of $\Omega(n^{3/2})$. These bounds together can be shown, using standard inequalities [33], to imply that the *expansion* of $K_{3,n+2}$ is $\Omega(1/(n^4 \log n))$ and $O(n^{1/4})$. It is easy to generalize triangulations to *k-angulations* of a convex polygon $P_{(k-2)n+2}$, and to generalize the definition of a flip between triangulations to a flip between k-angulations: a k-angulation is a maximal division of the polygon into k-gons, and a flip consists of taking a pair of k-gons that share a diagonal, removing that diagonal, and replacing it with one of the other diagonals in the resulting $2k-2$-gon. One can then define the *k-angulation flip walk* on the k-angulations of $P_{(k-2)n+2}$. An analogous graph to the associahedron is defined over the triangulations of the integer lattice (grid) point set with n rows of points and n columns. Substantial prior work has been done on bounds for the number of triangulations in this graph ([4, 22]), as well as characterizing the mixing time of random walks on the graph, when the walks are weighted by a function of the lengths of the edges in a triangulation ([6, 5]).

1.4 Convex triangulation flip walk and mixing time

Consider the following random walk on the triangulations of the convex $n+2$-gon:

for $t = 1, 2, \ldots$ **do**
  Begin with an arbitrary triangulation t.
  Flip a fair coin.
  If the result is tails, do nothing.
  Else, select a diagonal in t uniformly at random, and flip the diagonal.
end for

(The "do nothing" step is a standard MCMC step that enforces a technical condition known as *laziness*, required for the arguments that bound mixing time.) At any given time step, this walk induces a probability distribution π over the triangulations of the $n+2$-gon. Standard spectral graph theory shows that π converges to the uniform distribution in the limit. Formally, what McShine and Tetali showed [29] is that the number of steps before π is within *total variation distance* $1/4$ of the uniform distribution is bounded by $O(n^5 \log n)$ – in other words, that the *mixing time* is $O(n^5 \log n)$. Any polynomial bound means the walk *mixes rapidly*. We formally define total variation distance:

The *total variation distance* between two probability distributions μ and ν over the same set Ω is defined as

$$d(\mu, \nu) = \frac{1}{2} \sum_{S \in \Omega} |\pi(S) - \pi^*(S)|.$$

Consider a Markov chain with state space Ω. Given a starting state $S \in \Omega$, the chain induces a probability distribution π_t at each time step t. Under certain mild conditions, all of which are satisfied by the k-angulation flip walk, this distribution is known to converge in the limit to a *stationary* distribution π^*, which for the k-angulation flip walk is the uniform distribution on the k-angulations of the convex polygon. The *mixing time* is defined as follows: Given an arbitrary $\varepsilon > 0$, the *mixing time*, $\tau(\varepsilon)$, of a Markov chain with state space Ω and stationary distribution π^* is the minimum time t such that, regardless of starting state, we always have

$$d(\pi_t, \pi^*) < \varepsilon.$$

Suppose that the chain belongs to a family of chains, whose size is parameterized by a value n. (It may be that Ω is exponential in n.) If $\tau(\varepsilon)$ is upper bounded by a function that is polynomial in $\log(1/\varepsilon)$ and in n, say that the chain is *rapidly mixing*. It is common to omit the parameter ε, assuming its value to be the arbitrary constant $1/4$.

1.5 Main results

We show the following result for the expansion of the associahedron:

▶ **Theorem 2.** *The expansion of the associahedron $K_{3,n+2}$ is $\Omega(1/(\sqrt{n}\log n))$ and $O(1/\sqrt{n})$.*

We will prove the lower bound in the full paper version [16] using the *multicommodity flow*-based machinery we introduce in Section 4, after giving intuition in Section 2. Combining this result with the connection between flows and mixing [33] – with some additional effort in the full version – gives our new $O(n^3 \log^3 n)$ bound (Theorem 1) for triangulation mixing.

Although the expansion lower bound is more interesting for the sake of rapid mixing, the upper bound in Theorem 2 – which we prove in the full version – recovers Molloy, Reed, and Steiger's $\Omega(n^{3/2})$ mixing lower bound [31]. It is also the first result showing that the associahedron has combinatorial expansion $o(1)$. By contrast, Anari, Liu, Oveis Gharan, and Vinzant recently proved [3, 2], settling a conjecture of Mihail and Vazirani [30], that matroids have expansion one. (Mihail and Vazirani in fact conjectured that all graphs realizable as the 1-skeleton of a *0-1 polytope* have expansion one.) Although the set of convex n-gon triangulations is not a matroid, it is an important subset system – and this work shows that it does not have expansion one. More generally, we give the following quasipolynomial bound for k-angulations:

▶ **Theorem 3.** *For every fixed $k \geq 3$, the k-angulation flip walk on the convex $(k-2)n+2$-point set mixes in time $n^{O(k \log n)}$.*

In the full version of the paper [16], we give a lower bound on the *treewidth* of the $n \times n$ integer lattice point set triangulation flip graph:

▶ **Theorem 4.** *The treewidth of the triangulation flip graph F_n on the $n \times n$ integer lattice point set is $\Omega(N^{1-o(1)})$, where $N = |V(F_n)|$.*

2 Decomposing the convex point set triangulation flip graph

2.1 Bounding mixing via expansion

We have a Markov chain that is in fact a random walk on the associahedron K_n. We wish to bound the mixing time of this walk. It turns out that one way to do this is by lower-bounding the *expansion* of the same graph K_n. Intuitively, expansion concerns the extent to which "bottlenecks" exist in a graph. More precisely, it measures the "sparsest" cut – the minimum ratio of the number of edges in a cut divided by the number of vertices on the smaller side of the cut:

The *edge expansion* (or simply *expansion*), $h(G)$, of a graph $G = (V, E)$ is the quantity

$$\min_{S \subseteq V : |S| \leq |V|/2} |\partial S|/|S|,$$

where $\partial S = \{(s,t) | s \in S, t \notin S\}$ is the set of edges across the $(S, V \setminus S)$ cut. It is known [20, 33] that a lower bound on edge expansion leads to an upper bound on mixing:

▶ **Lemma 5.** *The mixing time of the Markov chain whose transition matrix is the normalized adjacency matrix of a Δ-regular graph G is*

$$O\left(\frac{\Delta^2 \log(|V(G)|)}{(h(G))^2}\right).$$

One can do better [13, 33] if the paths in a *multicommodity flow* are not too long (Section 3).

2.2 "Slicing and peeling"

We would like to show that there are many edges in every cut, relative to the number of vertices on one side of the cut. We partition the triangulations $V(K_n)$ into n equivalence classes, each inducing a subgraph of K_n. We show that many edges exist between each pair of the subgraphs. Thus the partitioning "slices" through many edges. After the partitioning, we show that each of the induced subgraphs has large expansion. To do so, we show that each such subgraph decomposes into many copies of a smaller flip graph K_i, $i < n$. This inductive structure lets us assume that K_i has large expansion – then show that the copies of the smaller flip graph are all well connected to one another. We call this "peeling," because one must peel the many K_i copies from one another – removing many edges – to isolate each copy. Molloy, Reed, and Steiger [31] obtained their $O(n^{25})$ mixing *upper* bound via a different decomposition, namely using the *central* triangle, via a non-flow-based method. That decomposition is the one we use for our quasipolynomial bound for general k-angulations in the full paper version. However, we use a different decomposition here, one with a structure that lets us obtain a nearly tight bound, via a multicommodity flow construction. We formalize the slicing step now:

Fix a "special" edge e^* of the convex $n+2$-gon P_{n+2}. For each triangle T having e^* as one of its edges, define the *oriented class* $\mathcal{C}^*(T)$ to be the set of triangulations of P_{n+2} that include T as one of their triangles. Let $\mathcal{T}_n$ be the set of all such triangles; let $\mathcal{S}_n$ be the set of all classes $\{\mathcal{C}^*(T)|T \in \mathcal{T}_n\}$.

Orient P_{n+2} so that e^* is on the bottom. Then say that T (respectively $\mathcal{C}^*(T)$) is to the *left* of T' (respectively $\mathcal{C}^*(T')$) if the topmost vertex of T lies counterclockwise around P_{n+2} from the topmost vertex of T'. Say that T' lies to the *right* of T. Write $T < T'$ and $T' > T$.

See Figure 1.

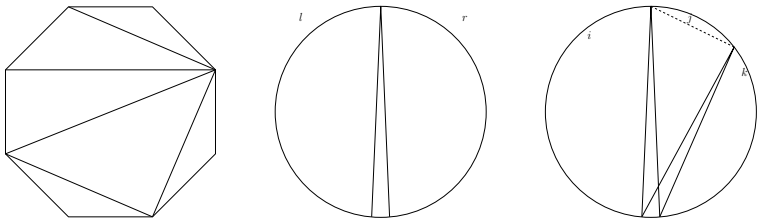


Figure 1 Left: A triangulation of the regular octagon. Center: a class $\mathcal{C}^*(T) \in \mathcal{S}_n$, represented schematically by the triangle T that induces it. We depict the regular $n+2$-gon as a circle (which it approximates as $n \to \infty$), for ease of illustration. Each triangulation $t \in \mathcal{C}^*(T)$ consists of T (the triangle shown), and an arbitrary triangulation of the two polygons on either side of T. Notice that $\mathcal{C}^*(T) \cong K_l \square K_r$, where T partitions the $n+2$-gon into an l-gon and an r-gon. Right: the matching $\mathcal{E}^*(T, T')$ between classes $\mathcal{C}^*(T) \cong K_i \square K_{j+k}$ and $\mathcal{C}^*(T') \cong K_{i+j} \square K_k$, is in bijection with the triangulations in $K_i \square K_j \square K_k$ (induced by the quadrilateral containing T and T'). Therefore, $|\mathcal{E}^*(T, T')| = C_i C_j C_k$.

We make observations about the structure of each class as an induced subgraph of K_n.

The *Cartesian product* graph $G \square H$ of graphs G and H has vertices $V(G) \times V(H)$ and edges

$$\{((u,v),(u',v))|(u,u') \in E(G), v \in V(H)\}$$

$$\cup\{((u,v),(u,v'))|(v,v') \in E(H), u \in V(G)\}.$$

Given a vertex $w = (u,v) \in V(G) \times V(H)$, call u the *projection* of w onto G, and similarly call v the projection of w onto H. (Applying the obvious associativity of the Cartesian product operator, one can naturally define the product $G_1 \square G_2 \square \cdots \square G_k = \square_{i=1}^{k} G_i$.)

We can now characterize the structure of each class as an induced subgraph of K_n:

▶ **Lemma 6.** *Each class $\mathcal{C}^*(T)$ is isomorphic to a Cartesian product of two associahedron graphs K_l and K_r, with $l + r = n - 1$.*

Proof. Each triangle T partitions the $n + 2$-gon into two smaller convex polygons with side lengths $l + 1$ and $r + 1$, such that $l + r = n - 1$. Thus each triangulation in $\mathcal{C}^*(T)$ can be identified with a tuple of triangulations of these smaller polygons. The Cartesian product structure then follows from the fact that every flip between two triangulations in $\mathcal{C}^*(T)$ can be identified with a flip in one of the smaller polygons. ◀

Lemma 6 will be central to the peeling step. For the slicing step, building on the idea in Lemma 6 will help us characterize the edge sets between classes:

Given classes $\mathcal{C}^*(T), \mathcal{C}^*(T') \in \mathcal{S}_n$, denote by $\mathcal{E}^*(T, T')$ the set of edges (flips) between $\mathcal{C}^*(T)$ and $\mathcal{C}^*(T')$. Let $\mathcal{B}^*_{n,T'}(T)$ and $\mathcal{B}^*_{n,T}(T')$ be the *boundary sets* – the sets of endpoints of edges in $\mathcal{E}^*(T, T')$ – that lie respectively in $\mathcal{C}^*(T)$ and $\mathcal{C}^*(T')$.

▶ **Lemma 7.** *For each pair of classes $\mathcal{C}^*(T)$ and $\mathcal{C}^*(T')$, the boundary set $\mathcal{B}^*_{n,T'}(T)$ induces a subgraph of $\mathcal{C}^*(T)$ isomorphic to a Cartesian product of the form $K_i \square K_j \square K_k$, for some $i + j + k = n - 2$.*

Proof. Each flip between triangulations in adjacent classes $\mathcal{C}^*(T)$ involves flipping a diagonal of T to transform the triangulation $t \in \mathcal{C}^*(T)$ into triangulation $t' \in \mathcal{C}^*(T')$. Whenever this is possible, there must exist a quadrilateral Q, sharing two sides with T (the sides that are not flipped), such that both t and t' contain Q. Furthermore, every $t \in \mathcal{C}^*(T)$ containing Q has a flip to a distinct $t' \in \mathcal{C}^*(T')$. The set of all such boundary vertices $t \in \mathcal{C}^*(T)$ can be identified with the Cartesian product described because Q partitions P_{n+2} into three smaller polygons, so that each triangulation in $\mathcal{B}^*_{n,T'}(T)$ consists of a tuple of triangulations in each of these smaller polygons, and such that every flip between triangulations in $\mathcal{B}^*_{n,T'}(T)$ consists of a flip in one of these smaller polygons. ◀

▶ **Lemma 8.** *The set $\mathcal{E}^*(T, T')$ of edges between each pair of classes $\mathcal{C}^*(T)$ and $\mathcal{C}^*(T')$ is a nonempty matching. Furthermore, this edge set is in bijection with the vertices of a Cartesian product $K_i \square K_j \square K_k, i + j + k = n - 2$.*

Proof. The claim follows from the reasoning in Lemma 7 and from the observation that each triangulation in $\mathcal{B}^*_{n,T'}(T)$ has exactly one flip (namely, flipping a side of the triangle T) to a neighbor in $\mathcal{B}^*_{n,T}(T')$. ◀

Lemma 8 characterizes the structure of the edge sets (namely matchings) between classes; we would also like to know the sizes of the matchings. We will use the following formula:

Let C_n be the nth *Catalan number*, defined as $C_n = \frac{1}{n+1}\binom{2n}{n}$.

▶ **Lemma 9** ([25, 19])**.** *The number of vertices in the associahedron K_n is C_n, and this number grows as $\frac{1}{\sqrt{\pi} \cdot n^{3/2}} \cdot 2^{2n}$.*

We will prove the following in the full version of our paper [16]:

▶ **Lemma 10.** *For every $T, T' \in \mathcal{T}_n$,*

$$|\mathcal{E}^*(T, T')| \geq \frac{|\mathcal{C}^*(T)||\mathcal{C}^*(T')|}{C_n}.$$

Lemma 10 – which states that the number of edges between a pair of classes is at least equal to the product of the cardinalities of the classes, divided by the total number of vertices in the graph $|V(K_n)| = C_n$ – is crucial to this paper. To explain why this is, we will need to present our multicommodity flow construction (in the full version of the paper [16]). We will give intuition in Section 4. For now, it suffices to say that Lemma 10 implies that there are many edges between a given pair of classes, justifying (intuitively) the slicing step. For the peeling step, we need the fact that Cartesian graph products preserve the well-connectedness of the graphs in the product [17]:

▶ **Lemma 11.** *Given graphs $G_1, G_2, \ldots, G_k$, Cartesian product $G_1 \square G_2 \square \cdots \square G_k$ satisfies*

$$h(G_1 \square G_2 \square \cdots \square G_k) \geq \frac{1}{2} \min_i h(G_i).$$

Lemma 6 says that each of the classes $\mathcal{C}^*(T) \in \mathcal{S}_n$ is a Cartesian graph product of associahedron graphs K_l, K_r, $l < n, r < n$, allowing us to "peel" (decompose) $\mathcal{C}^*(T)$ into graphs that can then be recursively sliced into classes and peeled. Lemma 11 implies that the peeling must disconnect many edges, as it involves splitting a Cartesian product graph into many subgraphs (copies of K_l).

We will make all of this intuition rigorous in the full paper version by constructing our flow. The choice of paths through which to route flow will closely trace the edges in this recursive "slicing and peeling" decomposition. We will then show that, with this choice of paths, the resulting *congestion* – the maximum amount of flow carried along an edge – is bounded by a suitable polynomial factor. This will provide a lower bound on the expansion.

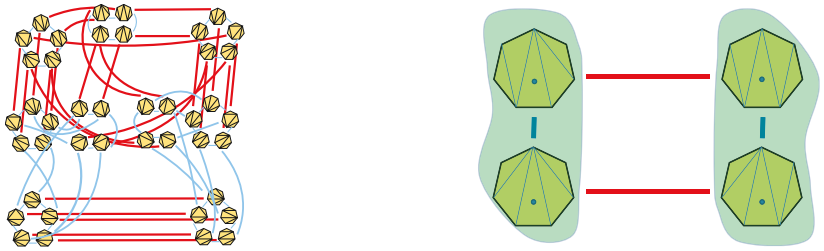


Figure 2 Left: The associahedron graph K_5, with each vertex representing a triangulation of the regular heptagon. Flips are shown with edges (in blue and red). The vertex set $V(K_n)$ is partitioned into a set $\mathcal{S}_n$ of five equivalence classes (of varying sizes). Within each class, all triangulations share the same triangle containing the bottom edge e^*. Flips (edges) between triangulations in the same class are shown in blue. Flips between triangulations in different classes are shown in red. To "slice" K_5 into its subgraphs, one must cut through these red matchings. **Right**: A class $\mathcal{C}^*(T)$ from the graph K_5 on the left-hand side, viewed as an induced subgraph of K_5. The identifying triangle T is marked with a blue dot. This subgraph is isomorphic to a Cartesian product of two K_2 graphs; each copy of K_2 induced by fixing the rightmost diagonal is outlined in green. "Peeling" apart this product requires disconnecting the two red edges connecting the K_2 copies.

3 Bounding expansion via multicommodity flows

The way we will lower-bound expansion is by using *multicommodity flows* [33, 23]. A *multicommodity flow* ϕ in a graph $G = (V, E)$ is a collection of functions $\{f_{st} : A \to \mathbb{R} \mid s, t \in V\}$, where $A = \bigcup_{\{u,v\} \in E} \{(u, v), (v, u)\}$, combined with a *demand function* $D : V \times V \to \mathbb{R}$.

Each f_{st} is a flow sending $D(s,t)$ units of a commodity from vertex s to vertex t through the edges of G. We consider the capacities of all edges to be infinite. Let $f_{st}(u,v)$ be the amount of flow sent by f_{st} across the arc (u,v). (It may be that $f_{st}(u,v) \neq f_{st}(v,u)$.) Let

$$f(u,v) = \frac{1}{|V|} \sum_{s,t \in V \times V} f_{st}(u,v),$$

and let $\rho = \max_{(u,v)\in A} f(u,v)$. Call ρ the *congestion*. Unless we specify otherwise, we will mean by "multicommodity flow" a *uniform multicommodity flow*, i.e. one in which $D(s,t) = 1$ for all s,t. The following is well established and enables the use of multicommodity flows as a powerful lower-bounding technique for expansion:

▶ **Lemma 12.** *Given a uniform multicommodity flow f in a graph $G = (V,E)$ with congestion ρ, the expansion $h(G)$ is at least $1/(2\rho)$.*

Lemma 12, combined with Lemma 5, gives an automatic upper bound on mixing time given a multicommodity flow with an upper bound on congestion – but with a quadratic loss. As we will discuss in the full paper version, one can do better if the paths used in the flow are short [13, 33].

4 Our framework

In addition to the new mixing bounds for triangulations and for general k-angulations, we make general technical contributions, in the form of three meta-theorems, which we present in this section. Our first general technical contribution, Theorem 13, provides a recursive mechanism for analyzing the expansion of a flip graph in terms of the expansion of its subgraphs. Equivalently, viewing the random walk on such a flip graph as a Markov chain, this theorem provides a mechanism for analyzing the mixing time of a chain, in terms of the mixing times of smaller *restriction* chains into which one decomposes the original chain – and analyzing a *projection* chain over these smaller chains. We obtain, in certain circumstances such as the k-angulation walk, better mixing time bounds than one obtains applying similar prior decomposition theorems – which used a different underlying machinery.

The second theorem, Lemma 15, observes and formalizes a set of conditions satisfied by a number of chains (equivalently, flip graphs) under which one can apply either our Theorem 13, or prior decomposition techniques, to obtain rapid mixing reuslts. Depending on the chain, one may then obtain better results either by applying Theorem 13, or by applying the prior techniques. Lemma 15 does not require using our Theorem 13; instead, one can use the spectral gap or log-Sobolev constant as the underlying techincal machinery using Jerrum, Son, Tetali, and Vigoda's Theorem 14. Prior work exists applying these techniques (using Theorem 14) to sampling q-colorings [18] in bounded-treewidth graphs and independent sets in regular trees [21], as well as probabilistic graphical models in machine learning [11] satisfying certain conditions. Lemma 15 amounts to an observation unifying these applications. We apply this observation to general k-angulations, noting that they satisfy a relaxation of this theorem (Lemma 18), giving a quasipolynomial bound. This bound will come from incurring a polynomial loss over logarithmic recursion depth.

The third theorem, Lemma 19, adapts the machinery in Theorem 13 to eliminate this multiplicative loss altogether, assuming that a chain satisfies certain properties. One such key property is the existence large matchings in Lemma 10 in Section 2. Another property, which we will discuss further after presenting Lemma 19, is that the *boundary sets* – the vertices in one class (equivalently, states in a restriction chain) having neighbors in another class –

are well connected to the rest of the first class. When these properties are satisfied, one can apply our flow machinery to overcome the multiplicative loss and obtain a polynomial bound. However, the improvement relies on observations about congestion that do not obviously translate to the spectral setting.

4.1 Markov chain decomposition via multicommodity flow

In this section we state our first general theorem. To place our contribution in context with prior work, we cast our flip graphs in the language of Markov chains. As we discussed in Section 1.4, any Markov chain satisfying certain mild conditions has a *stationary* distribution π^* (which in the case of our triangulation walks is uniform). We can view such a chain as a random walk on a graph $\mathcal{M}$ (an unweighted graph in the case of the chains we consider, which have uniform distributions and regular transition probabilities). In the case of convex polygon triangulations, we have $\mathcal{M} = K_n$.

The flip graph $\mathcal{M}$ has vertex set Ω and (up to normalization by degree) adjacency matrix P – and we abuse notation, identifying the Markov chain $\mathcal{M}$ with this graph. When π^* is not uniform, it is easy to generalize the flip graph to a *weighted* graph, with each vertex (state) t assigned weight $\pi(t)$, and each transition (edge) (t, t') assigned weight $\pi(t)P(t, t') = \pi(t')P(t', t)$. We assume here that this latter equality holds, a condition on the chain $\mathcal{M}$ known as *reversibility*. We then replace a uniform multicommodity flow with one where $D(t, t') = \pi(t)\pi(t')$ (up to normalization factors).

Consider a Markov chain $\mathcal{M}$ with finite state space Ω and probability transition matrix P, and stationary distribution π. Consider a partition of the states of Ω into classes $\Omega_1, \Omega_2, \ldots, \Omega_k$. Let the *restriction* chain, for $i = 1, \ldots, k$, be the chain with state space Ω_i, probability distribution π_i, with $\pi_i(x) = \pi(x)/(\sum_{y\in\Omega_i} \pi(y))$, for $x \in \Omega_i$, and transition probabilities $P_i(x, y) = P(x, y)/(\sum_{z\in\Omega_i} P(x, z))$. Let the *projection* chain be the chain with state space $\bar{\Omega} = \{1, 2, \ldots, k\}$, stationary distribution $\bar{\pi}$, with $\bar{\pi}(i) = \sum_{x\in\Omega_i} \pi(i)$, and transition probabilities $\bar{P}(i, j) = \sum_{x\in\Omega_i, y\in\Omega_j} P(x, y)$.

▶ **Theorem 13.** *Let $\mathcal{M}$ be a reversible Markov chain with finite state space Ω probability transition matrix P, and stationary distribution π^*. Suppose $\mathcal{M}$ is connected (irreducible). Suppose $\mathcal{M}$ can be decomposed into a collection of restriction chains $(\Omega_1, P_1), (\Omega_2, P_2), \ldots, (\Omega_k, P_k)$, and a projection chain $(\bar{\Omega}, \bar{P})$. Suppose each restriction chain admits a multicommodity flow (or canonical paths) construction with congestion at most $\rho_{\max}$. Suppose also that there exists a multicommodity flow construction in the projection chain with congestion at most $\bar{\rho}$. Then there exists a multicommodity flow construction in $\mathcal{M}$ (viewed as a weighted graph in the natural way) with congestion*

$$(1 + 2\bar{\rho}\gamma\Delta)\rho_{\max},$$

where $\gamma = \max_{i\in[k]} \max_{x\in\Omega_i} \sum_{y\notin\Omega_i} P(x, y)$, and Δ is the degree of $\mathcal{M}$.

We give a full proof in the full version of the paper. Jerrum, Son, Tetali, and Vigoda [21] presented an analogous (and classic) decomposition theorem, which we restate below as Theorem 14, and which has become a standard tool in mixing time analysis. The key difference between our theorem and theirs is that our theorem uses multicommodity flows, while their theorem uses the so-called *spectral gap* – another parameter that can use to bound the mixing time of a chain. Often, the spectral gap gives tighter mixing bounds than combinatorial methods. Their Theorem 14 gave bounds analogous to our Theorem 13, but with the multicommodity flow congestion replaced with the *spectral gap* of a chain

– and with a 3γ term in place of our 2γ. (They also gave an analogous version for the *log-Sobolev* constant – yet another parameter for bounding mixing times.) The spectral gap of a chain $\mathcal{M} = (\Omega, P)$, which we denote λ, is the difference between the two largest eigenvalues of the transition matrix P (which we can view as the normalized adjacency matrix of the corresponding weighted graph). The key point is that while on the one hand the mixing time τ satisfies $\tau \leq \lambda^{-1} \log |\Omega|$, the bound on mixing using expansion in Lemma 5 comes from passing through the spectral gap: $\lambda \geq \frac{(h(\mathcal{M}))^2}{2\Delta^2}$, where Δ is the degree of the flip graph and $h(\mathcal{M})$ is the expansion of $\mathcal{M}$. The quadratic loss in passing from expansion to mixing is not incurred when bounding the spectral gap directly, so one can obtain better bounds via the spectral gap. Jerrum, Son, Tetali, and Vigoda gave a mechanism for doing precisely this:

▶ **Theorem 14** ([21]). *Let $\mathcal{M}$ be a reversible Markov chain with finite state space Ω probability transition matrix P, and stationary distribution π^*. Suppose $\mathcal{M}$ is connected (irreducible). Suppose $\mathcal{M}$ can be decomposed into a collection of restriction chains $(\Omega_1, P_1), (\Omega_2, P_2), \ldots, (\Omega_k, P_k)$, and a projection chain $(\bar{\Omega}, \bar{P})$. Suppose each restriction chain has spectral gap at least $\lambda_{\min}$. Suppose also that the projection chain has spectral gap at least $\bar{\lambda}$. Then $\mathcal{M}$ has gap at least*

$$\min\left\{\frac{\lambda_{\min}}{3}, \frac{\bar{\lambda}\lambda_{\min}}{3\gamma + \bar{\lambda}}\right\},$$

where γ is as in Theorem 13.

Our Theorem 13 has a simple, purely combinatorial proof (in the full paper version), and fills a gap in the literature by showing that such a construction can be used in place of the spectral machinery from the earlier technique. We also obtain a tighter bound on expansion than would result from a black-box application of Theorem 14. The cost to our improvement is in passing from expansion to mixing via the spectral gap. Nonetheless, we will show that in the case of triangulations, our Theorem 13 can be adapted to give a new mixing bound whereas, by contrast, it is not clear how to obtain even a polynomial bound adapting Jerrum, Son, Tetali, and Vigoda's spectral machinery. We will also show that for general k-angulations, one can, with our technique, use a combinatorial insight to eliminate the γ factor in our decomposition in favor of a Δ^{-1} factor (for k-angulations we have $\gamma = k/\Delta$) – whereas it is not clear how to do so with the spectral decomposition.

4.2 General pattern for bounding projection chain congestion

Our second decomposition theorem, which we will apply to general k-angulations, states that if one can recursively decompose a chain into restriction chains in a particular fashion, and if the projection chain is well connected, then Theorem 13 gives an expansion bound:

▶ **Lemma 15.** *Let $\mathcal{F} = \{\mathcal{M}_1, \mathcal{M}_2, \ldots\}$ be a family of connected graphs, parameterized by a value n. Suppose that every graph $\mathcal{M}_n = (\mathcal{V}_n, \mathcal{E}_n) \in \mathcal{F}$, for $n \geq 2$, can be partitioned into a set $\mathcal{S}_n$ of classes satisfying the following conditions:*

1. *Each class in $\mathcal{S}_n$ is isomorphic to a Cartesian product of one or more graphs $\mathcal{C}(T) \cong \mathcal{M}_{i_1} \square \cdots \mathcal{M}_{i_k}$, where for each such graph $\mathcal{M}_{i_j} \in \mathcal{F}$, $i_j \leq n/2$.*
2. *The number of classes is $O(1)$.*
3. *For every pair of classes $\mathcal{C}(T), \mathcal{C}(T') \in \mathcal{S}_n$ that share an edge, the number of edges between the two classes is $\Omega(1)$ times the size of each of the two classes.*
4. *The ratio of the sizes of any two classes is $\Theta(1)$.*

Suppose further that $|\mathcal{V}_1| = 1$. Then the expansion of $\mathcal{M}_n$ is $\Omega(n^{-O(1)})$.

Lemma 15 is easy to prove given Theorem 13. An analogue in terms of spectral gap is easy to prove given Theorem 14. Furthermore, as we will prove in the full paper version, a precise statement of the bounds given by Lemma 15 is as follows:

▶ **Lemma 16.** *Suppose a flip graph $\mathcal{M}_n = (\mathcal{V}_n, \mathcal{E}_n)$ belongs to a family $\mathcal{F}$ of graphs satisfying the conditions of Lemma 15. Suppose further that every graph $\mathcal{M}_k = (\mathcal{V}_k, \mathcal{E}_k) \in \mathcal{F}$, $k < n$, satisfies*

$$|\mathcal{V}_k|/|\mathcal{E}_{k,\min}| \leq f(k),$$

for some function $f(k)$, where $\mathcal{E}_{k,\min}$ is the smallest edge set between adjacent classes $\mathcal{C}(T), \mathcal{C}(T') \in \mathcal{S}_k$, where $\mathcal{S}_k$ is as in Lemma 15. Then the expansion of $\mathcal{M}_n$ is

$$\Omega(1/(2f(n))^{\log n})),$$

where γ is as in Theorem 13, and Δ is the degree of $\mathcal{M}_n$.

Proof. Constructing an arbitrary multicommodity flow (or set of canonical paths) in the projection graph at each inductive step gives the result claimed. The term $|\mathcal{V}_k|/|\mathcal{E}_{k,\min}|$ bounds the (normalized) congestion in any such flow because the total amount of flow exchanged by all pairs of vertices (states) combined is $|\mathcal{V}_k|^2$, and the minimum weight of an edge in the projection graph is $|\mathcal{E}_{k,\min}|$.

Notice that we do not incur a $\gamma\Delta$ term here, because even if a state (vertex) in $\Omega_i \subseteq \mathcal{V}_k$ has neighbors $x \in \Omega_j, y \in \Omega_l$, z still only receives no more than $|\mathcal{V}_k|^2/\mathcal{E}_{k,\min}\}$ flow across the edges (z, x) and (z, y) combined. ◀

▶ Remark 17. The $\gamma\Delta$ factor in Theorem 13, which does not appear in Lemma 16, does appear in a straightforward appliation of Jerrum, Son, Tetali, and Vigoda's Theorem 14.

We will show that k-angulations (with fixed $k \geq 4$) satisfy a relaxation of Lemma 15:

▶ **Lemma 18.** *Suppose a family $\mathcal{F}$ of graphs satisfies the conditions of Lemma 15, with the $\Omega(1)$, $O(1)$, and $\Theta(1)$ factors in Conditions 3, 2, and 4 respectively replaced by $\Omega(n^{-O(1)})$, $O(n^{O(1)})$, and $\Theta(n^{O(1)})$. Then for every $\mathcal{M}_n \in \mathcal{F}$, the expansion of $\mathcal{M}_n$ is $\Omega(n^{-O(\log n)})$.*

Lemma 15 enables us to relate a number of chains admitting a certain decomposition process in a black-box fashion, unifying prior work applying Theorem 14 separately to individual chains. Marc Heinrich [18] presented a similar but less general construction for the Glauber dynamics on q-colorings in bounded-treewidth graphs; other precursors exist, including for the hardcore model on certain trees [21] and a general argument for a class of graphical models [11]. In the companion paper [15] we mentioned in Section 1, we apply Lemma 15 to chains for sampling independent sets and dominating sets in bounded-treewidth graphs, as well as chains on q-colorings, maximal independent sets, and several other structures, in graphs whose treewidth and degree are bounded.

4.3 Eliminating inductive loss: nearly tight conductance for triangulations

We now give the meta-theorem that we will apply to triangulations. Lemma 15 – using either Theorem 13 or Theorem 14 – gives a merely quasipolynomial bound when applied straightforwardly to k-angulations, including the case of triangulations – simply because the $f(n)$ term in Lemma 16 is $\omega(1)$ and thus the overall congestion is $\omega(1)^{\log n}$ (not polynomial). However, it turns out that the large matchings given by Lemma 10 between pairs of classes

in the case of triangulations (but not general k-angulations), combined with some additional structure in the triangulation flip walk, satisfy an alternative set of conditions that suffice for rapid mixing. The conditions are:

▶ **Lemma 19.** *Let $\mathcal{F} = \{\mathcal{M}_1, \mathcal{M}_2, \dots\}$ be an infinite family of connected graphs, parameterized by a value n. Suppose that for every graph $\mathcal{M}_n = (\mathcal{V}_n, \mathcal{E}_n) \in \mathcal{F}$, for $n \geq 2$, the vertex set $\mathcal{V}_n$ can be partitioned into a set $\mathcal{S}_n$ of classes inducing subgraphs of $\mathcal{M}_n$ that satisfy the following conditions:*

1. *Each subgraph is isomorphic to a Cartesian product of one or more graphs $\mathcal{C}(T) \cong \mathcal{M}_{i_1} \square \cdots \mathcal{M}_{i_k}$, where for each such graph $\mathcal{M}_{i_j} \in \mathcal{F}$, $i_j < n$.*
2. *The number of classes is $n^{O(1)}$.*
3. *For every pair of classes $\mathcal{C}(T), \mathcal{C}(T') \in \mathcal{S}_n$, the set of edges between the subgraphs induced by the two classes is a matching of size at least $\frac{|\mathcal{C}(T)||\mathcal{C}(T')|}{|\mathcal{V}_n|}$.*
4. *Given a pair of classes $\mathcal{C}(T), \mathcal{C}(T') \in \mathcal{S}_n$, there exists a graph $\mathcal{M}_i$ in the Cartesian product $\mathcal{C}(T)$, and a class $\mathcal{C}(U) \in \mathcal{S}_i$ within the graph $\mathcal{M}_i$, such that the set of vertices in $\mathcal{C}(T)$ having a neighbor in $\mathcal{C}(T')$ is precisely the set of vertices in $\mathcal{C}(T)$ whose projection onto $\mathcal{M}_i$ lies in $\mathcal{C}(U)$. Furthermore, no class $\mathcal{C}(U)$ within $\mathcal{M}_i$ is the projection of more than one such boundary.*

Suppose further that $|\mathcal{V}_1| = 1$. Then the expansion of $\mathcal{M}_n$ is $\Omega(1/(\kappa(n)n))$, where $\kappa(n) = \max_{1 \leq i \leq n} |\mathcal{C}(S_i)|$ is the maximum number of classes in any $\mathcal{M}_i, i \leq n$.

Unlike Lemma 15, this lemma requires a purely combinatorial construction; it is not clear how to apply spectral methods to obtain even a polynomial bound. Condition 4 is crucial. To give more intuition for this condition, we state and prove the following fact about the triangulation flip graph (visualized in Figure 3):

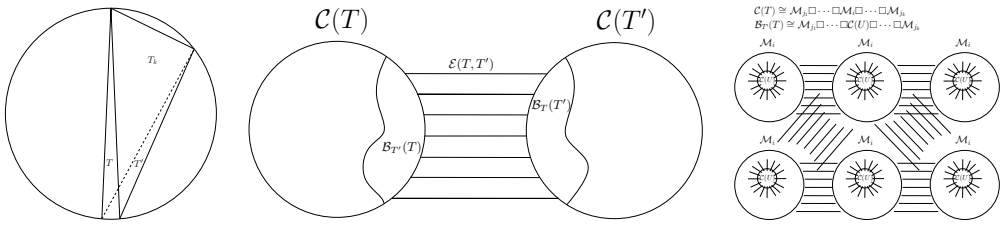


Figure 3 Left: (Lemma 20) The set of edges $\mathcal{E}^*(T, T')$ has $K_i \square \mathcal{C}^*(T_k)$ as its set of boundary vertices in $\mathcal{C}^*(T)$. **Center**: An illustration of Condition 3 in Lemma 19, showing a large matching $\mathcal{E}(T, T')$ between two classes (subgraphs) $\mathcal{C}(T)$ and $\mathcal{C}(T')$. **Right**: An illustration of Conditions 1 and 4 in Lemma 19: $\mathcal{C}(T)$ as a Cartesian product of smaller graphs $\mathcal{M}_{j_1}, \dots, \mathcal{M}_i, \dots, \mathcal{M}_{j_k}$ in the family $\mathcal{F}$. The schematic view shows this Cartesian product as a collection of copies of $\mathcal{M}_i$, connected via perfect matchings between pairs of the copies – with the pairs to connect determined by the structure of the Cartesian product. The boundary $\mathcal{B}_{T'}(T)$ (center) is isomorphic to a class $\mathcal{C}(U)$ (right) within $\mathcal{M}_i$, a graph in the product. Within each copy of $\mathcal{M}_i$, many edges connect $\mathcal{C}(U)$ to the rest of $\mathcal{M}_i$.

▶ **Lemma 20.** *Given $T, T' \in \mathcal{T}_n$, suppose T' lies to the right of T. Then the subgraph of $\mathcal{C}^*(T)$ induced by $\mathcal{B}^*_{n,T'}(T)$ is isomorphic to a Cartesian product $K_l \square \mathcal{C}^*(T_k)$, where $l + r = n - 1$, and where T_k has as an edge the right diagonal of T, and as the vertex opposite this edge the topmost vertex of T'. A symmetric fact holds for $\mathcal{B}^*_{n,T}(T')$.*

Proof. Every triangulation in $\mathcal{B}^*_{n,T'}(T)$ (i) includes the triangle T and (ii) is a single flip away from including the triangle T'. As we observed in the proof of Lemma 7, this implies that $\mathcal{B}^*_{n,T'}(T)$ consists of the set of triangulations in $\mathcal{C}^*(T)$ containing a quadrilateral Q.

Specifically, Q shares two sides with T: one of these is e^*, and the other is the left side of T. One of the other two sides of Q is the right side of $\mathcal{C}^*(T')$. Combining this side with the "top" side of Q and with the right side of T, one obtains the triangle T_k, proving the claim. ◀

Lemma 20 implies that there are many edges between the boundary set $\mathcal{B}^*_{n,T'}(T)$ and the rest of $\mathcal{C}^*(T)$: $\mathcal{C}^*(T) \cong K_l \square K_r$, where K_l and K_r are smaller associahedron graphs, so $\mathcal{C}^*(T)$ is a collection of copies of K_r, with pairs of copies connected by perfect matchings. Each K_r copy can itself be decomposed into a set $\mathcal{S}_r$ of classes, one of which, namely $\mathcal{C}^*(T_k)$, is the intersection of $\mathcal{B}^*_{n,T'}(T)$ with the K_r copy. Applying Condition 3 to the K_r copy implies that there are many edges between boundary vertices in $\mathcal{C}^*(T_k)$ to other subgraphs (classes) in the K_r copy. That is, the boundary set $\mathcal{B}^*_{n,T'}(T)$ is well connected to the rest of $\mathcal{C}^*(T)$.

Figure 3 visualizes this situation in general terms for the framework. We have now proven:

▶ **Lemma 21.** *The associahedron graph K_n, along with the oriented partition we have defined, satisfies the conditions of Lemma 19.*

Proof. The connectedness of K_n is known [29]. Conditions 1 and 3 follow from Lemma 6, Lemma 8, and Lemma 10. Concerning the boundary sets, Condition 4 follows from Lemma 20 and from the discussion leading to this lemma. ◀

Together with Lemma 5 and the easy fact that K_n is a $\Theta(n)$-regular graph, Lemma 21 implies rapid mixing, pending the proof of Lemma 19 – which we prove in the full paper version [16].

4.4 Intuition for the flow construction for triangulations

We will prove Lemma 19 in the full paper version, from which a coarse expansion lower bound for triangulations – and a corresponding coarse (but polynomial) upper bound for mixing – will be immediate by Lemma 21. We give some intuition now for the flow construction we will give in the proof of Lemma 19, and in particular for the centrality of Condition 3 and Condition 4 (corresponding respectively to Lemma 10 and Lemma 20 for triangulations). Consider the case of triangulations, for concreteness. Every $t \in \mathcal{C}^*(T), t' \in \mathcal{C}^*(T')$ must exchange a unit of flow. This means that a total of $|\mathcal{C}^*(T)||\mathcal{C}^*(T')|$ flow must be sent across the matching $\mathcal{E}^*(T,T')$. To minimize congestion, it will be optimal to equally distribute this flow across all of the boundary matching edges. We can decompose the overall problem of routing flow from each $t \in \mathcal{C}^*(T)$ to each $t' \in \mathcal{C}^*(T')$ into three subproblems: (i) *concentrating* flow from every triangulation in $\mathcal{C}^*(T)$ within the boundary set $\mathcal{B}^*_{n,T'}(T)$, (ii) routing flow across the matching edges $\mathcal{E}^*(T,T')$, i.e. from $\mathcal{B}^*_{n,T'}(T) \subseteq \mathcal{C}^*(T)$ to $\mathcal{B}^*_{n,T}(T') \subseteq \mathcal{C}^*(T')$, and (iii) *distributing* flow from the boundary $\mathcal{B}^*_{n,T}(T')$ to each $t' \in \mathcal{C}^*(T')$. Now, the amount of flow that must be concentrated from $\mathcal{C}^*(T)$ at *each* boundary triangulation $u \in \mathcal{B}^*_{n,T'}(T)$ (and symmetrically distributed from *each* $v \in \mathcal{B}^*_{n,T}(T')$ throughout $\mathcal{C}^*(T')$) is equal to

$$\frac{|\mathcal{C}^*(T)||\mathcal{C}^*(T')|}{|\mathcal{B}^*_{n,T'}(T)|} = \frac{|\mathcal{C}^*(T)||\mathcal{C}^*(T')|}{|\mathcal{B}^*_{n,T}(T')|} = \frac{|\mathcal{C}^*(T)||\mathcal{C}^*(T')|}{|\mathcal{E}^*(T,T')|} \leq C_n,$$

where we have used the equality $|\mathcal{B}^*_{n,T'}(T)| = |\mathcal{B}^*_{n,T}(T')| = |\mathcal{E}^*(T,T')|$ by Lemma 7 and Lemma 8, and where the inequality follows from Lemma 10. As a result, in the "concentration" and "distribution" subproblems (i) and (iii), at most C_n flow is concentrated at or distributed from any given triangulation (Figure 4). This bound yields a recursive structure: the concentration (respectively distribution) subproblem decomposes into a flow problem

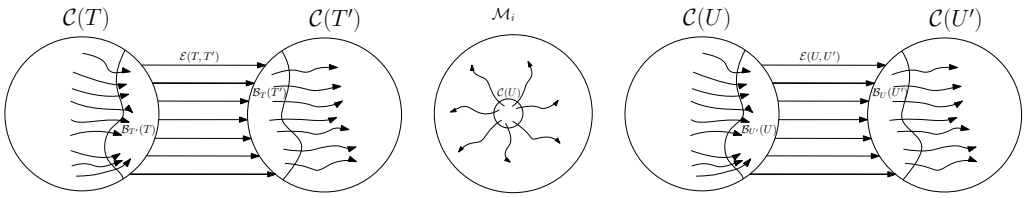


Figure 4 Left: The problem of sending flow from each $t \in \mathcal{C}^*(T)$ to each $t' \in \mathcal{C}^*(T')$, decomposed into subproblems: (i) *concentrating* flow within $\mathcal{B}^*_{n,T'}(T)$, (ii) *transmitting* the flow across the boundary matching $\mathcal{E}^*(T,T')$, and (iii) *distributing* the flow from $\mathcal{B}^*_{n,T}(T')$ throughout $\mathcal{C}^*(T')$. **Center**: Within each copy of $\mathcal{M}_i$ in the product $\mathcal{C}^*(T') \cong \mathcal{M}_{j_1} \square \cdots \square \mathcal{M}_i \square \cdots \square \mathcal{M}_{j_k}$, the distribution problem in Figure 4 induces the problem of distributing flow from a class $\mathcal{C}^*(U)$ – namely the projection of $\mathcal{B}^*_{n,T}(T')$ onto $\mathcal{M}_i$ – throughout the rest of $\mathcal{M}_i$. **Right**: The problem in the center figure induces subproblems in which $\mathcal{C}^*(U) \subseteq \mathcal{M}_i$ must send flow to each $\mathcal{C}^*(U') \subseteq \mathcal{M}_i$. These subproblems are of the same form as the original $\mathcal{C}^*(T), \mathcal{C}^*(T')$ problem (left), and can be solved recursively. The large matchings $\mathcal{E}^*(T,T'), \mathcal{E}^*(U,U')$ guaranteed by Condition 3 prevent any recursive congestion increase.

within $\mathcal{C}^*(T)$ (respectively $\mathcal{C}^*(T')$), in which, by the inequality, each triangulation has C_n total units of flow it must receive (or send). We will then apply Condition 4, observing (see Figure 4) that the concentration (symmetrically) distribution of this flow can be done entirely between pairs of classes $\mathcal{C}^*(U), \mathcal{C}^*(U')$ within copies of a smaller flip graph $\mathcal{M}_i$ in the Cartesian product $\mathcal{C}^*(T') \cong \mathcal{M}_{j_1} \square \cdots \square \mathcal{M}_i \square \cdots \square \mathcal{M}_{j_k}$.

The $\mathcal{C}^*(U), \mathcal{C}^*(U')$ subproblem is of the same form as the original $\mathcal{C}^*(T), \mathcal{C}^*(T')$ problem (Figure 4), and we will show that the C_n bound on the flow (normalizing to congestion one) across the $\mathcal{E}^*(T,T')$ edges will induce the same C_n bound across the $\mathcal{E}^*(U,U')$ edges in the induced subproblem. We further decompose the $\mathcal{C}^*(U), \mathcal{C}^*(U')$ problem into concentration, transmission, and distribution subproblems without any gain in overall congestion. To see this, view the initial flow problem in K_n as though every triangulation $t \in V(K_n)$ is initially "charged" with $|V(K_n)| = C_n$ total units of flow to distribute throughout K_n. Similarly, in the induced distribution subproblem within each copy of $\mathcal{M}_i = K_i$ in the product $\mathcal{C}^*(T')$, each vertex on the boundary $\mathcal{B}^*_{n,T'}(T)$ is initially "charged" with C_n total units to distribute throughout K_i. Just as the original problem in K_n results in each $\mathcal{E}^*(T,T')$ carrying at most C_n flow across each edge, similarly (we will show in the full paper version) the induced problem in K_i results in each $\mathcal{E}^*(U,U')$ carrying at most C_n flow across each edge. This preservation of the bound C_n under the recursion avoids any congestion increase.

One must be cautious, due to the linear recursion depth, not to accrue even a constant-factor loss in the recursive step (the coefficient 2 in Theorem 13). In Theorem 13, it turns out that this loss comes from routing *outbound* flow within a class $\mathcal{C}^*(T)$ – flow that must be sent to other classes – and then also routing *inbound* flow. The combination of these steps involves two "recursive invocations" of a uniform multicommodity flow that is inductively assumed to exist within $\mathcal{C}^*(T)$. We will show in the full paper version that one can avoid the second "invocation" with an initial "shuffling" step: a uniform flow within $\mathcal{C}^*(T)$ in which each triangulation $t \in \mathcal{C}^*(T)$ distributes all of its outbound flow evenly throughout $\mathcal{C}^*(T)$.

It is here that Jerrum, Son, Tetali, and Vigoda's spectral Theorem 14 breaks down, giving a 3-factor loss at each recursion level, due to applying the Cauchy-Schwarz inequality to a *Dirichlet form* that is decomposed into expressions over the restriction chains. Although Jerrum, Son, Tetali, and Vigoda gave circumstances for mitigating or eliminating their multiplicative loss, this chain does not satisfy those conditions in an obvious way.

References

1 Nima Anari, Kuikui Liu, and Shayan Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1319–1330, 2020. doi:10.1109/FOCS46700.2020.00125.

2 Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials II: High-dimensional walks and an FPRAS for counting bases of a matroid. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC 2019)*, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3313276.3316385.

3 Nima Anari, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials, entropy, and a deterministic approximation algorithm for counting bases of matroids. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 35–46, 2018. doi:10.1109/FOCS.2018.00013.

4 Emile E. Anclin. An upper bound for the number of planar lattice triangulations. *Journal of Combinatorial Theory, Series A*, 103(2):383–386, 2003. doi:10.1016/S0097-3165(03)00097-9.

5 Pietro Caputo, Fabio Martinelli, Alistair Sinclair, and Alexandre Stauffer. Dynamics of lattice triangulations on thin rectangles. *Electronic Journal of Probability*, 21, May 2015. doi:10.1214/16-EJP4321.

6 Pietro Caputo, Fabio Martinelli, Alistair Sinclair, and Alexandre Stauffer. Random lattice triangulations: structure and algorithms. *Annals of Applied Probability*, 25:1650–1685, 2015.

7 Alessandra Caraceni and Alexandre Stauffer. Polynomial mixing time of edge flips on quadrangulations. *Probability Theory and Related Fields*, 176(1):35–76, February 2020. doi:10.1007/s00440-019-00913-5.

8 Zongchen Chen, Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Rapid mixing for colorings via spectral independence. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1548–1557, 2021. doi:10.1137/1.9781611976465.94.

9 Zongchen Chen, Kuikui Liu, and Eric Vigoda. Rapid mixing of Glauber dynamics up to uniqueness via contraction, 2020. arXiv:2004.09083.

10 Zongchen Chen, Kuikui Liu, and Eric Vigoda. Optimal mixing of glauber dynamics: Entropy factorization via high-dimensional expansion. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1537–1550, 2021.

11 Christopher M De Sa, Ce Zhang, Kunle Olukotun, and Christopher Ré. Rapidly Mixing Gibbs Sampling for a Class of Factor Graphs Using Hierarchy Width. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper/2015/file/b29eed44276144e4e8103a661f9a78b7-Paper.pdf.

12 Persi Diaconis and Laurent Saloff-Coste. Comparison theorems for reversible markov chains. *The Annals of Applied Probability*, 3(3):696–730, 1993.

13 Persi Diaconis and Daniel Stroock. Geometric Bounds for Eigenvalues of Markov Chains. *The Annals of Applied Probability*, 1(1):36–61, 1991. doi:10.1214/aoap/1177005980.

14 Martin Dyer, Leslie Ann Goldberg, and Mark Jerrum. Matrix norms and rapid mixing for spin systems. *The Annals of Applied Probability*, 19(1):71–107, 2009. URL: http://www.jstor.org/stable/30243572.

15 David Eppstein and Daniel Frishberg. Rapid mixing of the hardcore Glauber dynamics and other Markov chains in bounded-treewidth graphs. *CoRR*, 2021. doi:10.48550/arXiv.2111.03898.

16 David Eppstein and Daniel Frishberg. Improved mixing for the convex polygon triangulation flip walk. *CoRR*, abs/2207.09972, 2022. doi:10.48550/arXiv.2207.09972.

17 F. Graham and P. Tetali. Isoperimetric inequalities for cartesian products of graphs. *Comb. Probab. Comput.*, 7:141–148, 1998.

18 Marc Heinrich. Glauber dynamics for colourings of chordal graphs and graphs of bounded treewidth, 2020. arXiv:2010.16158.

19 Peter J. Hilton and Jean J. Pedersen. Catalan numbers, their generalization, and their uses. *The Mathematical Intelligencer*, 13:64–75, 1991.

20 Mark Jerrum and Alistair Sinclair. Conductance and the rapid mixing property for Markov chains: The approximation of permanent resolved. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, 1988. doi:10.1145/62212.62234.

21 Mark Jerrum, Jung-Bae Son, Prasad Tetali, and Eric Vigoda. Elementary bounds on Poincaré and log-Sobolev constants for decomposable Markov chains. *The Annals of Applied Probability*, 14(4):1741–1765, 2004. URL: http://www.jstor.org/stable/4140446.

22 V. Kaibel and G. Ziegler. Counting lattice triangulations. *arXiv: Combinatorics*, 2002.

23 Volker Kaibel. On the expansion of graphs of 0/1-polytopes. In *The Sharpest Cut: The Impact of Manfred Padberg and His Work*, pages 199–216. SIAM, 2004.

24 Tali Kaufman and Izhar Oppenheim. High order random walks: Beyond spectral gap. *Combinatorica*, 40(2):245–281, 2020.

25 David A. Klarner. Correspondences between plane trees and binary sequences. *Journal of Combinatorial Theory*, 9(4):401–411, 1970. doi:10.1016/S0021-9800(70)80093-X.

26 Vedat Levi Alev and Lap Chi Lau. Improved analysis of higher order random walks and applications. *arXiv e-prints*, 2020. arXiv:2001.02827.

27 David A Levin, Yuval Peres, and Elizabeth Wilmer. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

28 J. Loday. The multiple facets of the associahedron. In *Proc. 2005 Academy Coll. Series*, 2005.

29 Lisa McShine and P. Tetali. On the mixing time of the triangulation walk and other catalan structures. In *Randomization Methods in Algorithm Design*, 1997.

30 Milena Mihail and Umesh Vazirani. On the expansion of 0-1 polytopes. *Journal of Combinatorial Theory, Series B*, 1989.

31 Michael Molloy, Bruce Reed, and William Steiger. On the mixing rate of the triangulation walk. In *Randomization Methods in Algorithm Design*, 1997.

32 Dana Randall and Prasad Tetali. Analyzing glauber dynamics by comparison of markov chains. In *Latin American Symposium on Theoretical Informatics*, pages 292–304. Springer, 1998.

33 Alistair Sinclair. Improved bounds for mixing rates of Markov chains and multicommodity flow. *Combinatorics, Probability and Computing*, 1(4):351–370, 1992. doi:10.1017/S0963548300000390.

34 Daniel D Sleator, Robert E Tarjan, and William P Thurston. Rotation distance, triangulations, and hyperbolic geometry. *Journal of the American Mathematical Society*, 1(3):647–681, 1988.

35 Alexandre Stauffer. A Lyapunov function for Glauber dynamics on lattice triangulations. *Probability Theory and Related Fields*, 169:469–521, 2015.

Optimal Adjacency Labels for Subgraphs of Cartesian Products

Louis Esperet
Laboratoire G-SCOP, Grenoble, France

Nathaniel Harms
EPFL, Lausanne, Switzerland

Viktor Zamaraev
University of Liverpool, UK

Abstract

For any hereditary graph class $\mathcal{F}$, we construct optimal adjacency labeling schemes for the classes of subgraphs and induced subgraphs of Cartesian products of graphs in $\mathcal{F}$. As a consequence, we show that, if $\mathcal{F}$ admits efficient adjacency labels (or, equivalently, small induced-universal graphs) meeting the information-theoretic minimum, then the classes of subgraphs and induced subgraphs of Cartesian products of graphs in $\mathcal{F}$ do too. Our proof uses ideas from randomized communication complexity and hashing, and improves upon recent results of Chepoi, Labourel, and Ratel [Journal of Graph Theory, 2020].

2012 ACM Subject Classification Mathematics of computing → Graph theory; Mathematics of computing → Combinatorics

Keywords and phrases Adjacency labeling schemes, Cartesian product, Hypercubes

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.57

Category Track A: Algorithms, Complexity and Games

Funding *Louis Esperet*: Partially supported by the French ANR Projects GATO (ANR-16-CE40-0009-01), GrR (ANR-18-CE40-0032), TWIN-WIDTH (ANR-21-CE48-0014-01) and by LabEx PERSYVAL-lab (ANR-11-LABX-0025).
Nathaniel Harms: This work was partly funded by NSERC, and was done while the author was a student at the University of Waterloo, visiting Laboratoire G-SCOP and the University of Liverpool.

Acknowledgements We are very grateful to Sebastian Wild, who prevented us trying to reinvent perfect hashing.

1 Introduction

In this paper, we present optimal adjacency labeling schemes (equivalently, induced-universal graph constructions) for subgraphs of Cartesian products, which essentially closes a recent line of work studying these objects [1, 2, 3, 4, 8, 10].

Adjacency labeling

A *class* of graphs is a set $\mathcal{F}$ of graphs closed under isomorphism, where the set $\mathcal{F}_n \subseteq \mathcal{F}$ of graphs on n vertices has vertex set $[n]$. It is *hereditary* if it is also closed under taking induced subgraphs, and *monotone* if it is also closed under taking subgraphs. An *adjacency labeling scheme* for a class $\mathcal{F}$ consists of a *decoder* $D : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ such that for every $G \in \mathcal{F}$ there exists a *labeling* $\ell : V(G) \to \{0,1\}^*$ satisfying

$$\forall x, y \in V(G) : \qquad D(\ell(x), \ell(y)) = 1 \iff xy \in E(G)\,.$$

The *size* of the adjacency labeling scheme (or *labeling scheme* for short) is the function $n \mapsto \max_{G \in \mathcal{F}_n} \max_{x \in V(G)} |\ell(x)|$, where $|\ell(x)|$ is the number of bits of $\ell(x)$. Labeling schemes have been studied extensively since their introduction by Kannan, Naor, & Rudich [13] and Muller [15]. If $\mathcal{F}$ admits a labeling scheme of size $s(n)$, then a graph $G \in \mathcal{F}_n$ can be recovered from the $n \cdot s(n)$ total bits in the adjacency labels of its vertices, so a labeling scheme is an encoding of the graph, distributed among its vertices. The information-theoretic lower bound on any encoding is $\log |\mathcal{F}_n|$, so the question is, when can the distributed adjacency labeling scheme approach this bound? In other words, which classes of graphs admit labeling schemes of size $O(\frac{1}{n} \log |\mathcal{F}_n|)$? We will say that a graph class has an *efficient* labeling scheme if it either has a labeling scheme of size $O(1)$ (i.e. it satisfies $\log |\mathcal{F}_n| = o(n \log n)$ [16]), or $O(\frac{1}{n} \log |\mathcal{F}_n|)$.

Cartesian products

Write $G \square H$ for the Cartesian product of G and H, write G^d for the d-wise Cartesian product of G, and for any class $\mathcal{F}$ write $\mathcal{F}^\square = \{G_1 \square G_2 \square \cdots \square G_d : d \in \mathbb{N}, G_i \in \mathcal{F}\}$ for the class of Cartesian products of graphs in $\mathcal{F}$. A vertex x of $G_1 \square \cdots \square G_d$ can be written $x = (x_1, \ldots, x_d)$ where $x_i \in V(G_i)$ and two vertices x, y are adjacent if and only if they differ on exactly one coordinate $i \in [d]$, and on this coordinate $x_i y_i \in E(G_i)$. Write $\mathsf{mon}(\mathcal{F}^\square)$ and $\mathsf{her}(\mathcal{F}^\square)$, respectively, for the monotone and hereditary closures of this class, which are the sets of all graphs G that are a subgraph (respectively, induced subgraph) of some $H \in \mathcal{F}^\square$.

We will construct optimal labeling schemes for $\mathsf{mon}(\mathcal{F}^\square)$ and $\mathsf{her}(\mathcal{F}^\square)$ from an optimal labeling scheme for $\mathcal{F}$. Cartesian products appear several times independently in the recent literature on labeling schemes [3, 8, 2] (and later in [10, 1, 4]), and are extremely natural for the problem of adjacency labeling for a few reasons.

First, for example, if $\mathcal{F}$ is the class of complete graphs, a labeling scheme for $\mathsf{her}(\mathcal{F}^\square)$ is equivalent to an encoding $\ell : T \to \{0,1\}^*$ of strings $T \subseteq \Sigma^*$, with Σ being an arbitrarily large finite alphabet, such that a decoder who doesn't know T can decide whether $x, y \in T$ have Hamming distance 1, using only the encodings $\ell(x)$ and $\ell(y)$. Replacing complete graphs with, say, paths, one obtains induced subgraphs of grids in arbitrary dimension. Switching to $\mathsf{mon}(\mathcal{F}^\square)$ allows arbitrary edges of these products to be deleted.

Second, Cartesian product graphs admit, by definition, a natural but inefficient "implicit representation", meaning (informally) that the adjacency between two vertices x and y can be verified by examining their representation (in this case, the tuples $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$). Formalizing and quantifying this general notion was the motivation for labeling schemes in [13], who also observed that adjacency labeling schemes are equivalent to *induced-universal graphs* (or simply *universal graphs*). A sequence of graphs $(U_n)_{n \in \mathbb{N}}$ are universal graphs of size $n \mapsto |U_n|$ for a class $\mathcal{F}$ if each n-vertex graph $G \in \mathcal{F}$ is an induced subgraph of U_n. A labeling scheme of size $s(n)$ is equivalent to a universal graph of size $2^{s(n)}$, and Cartesian product graphs admit natural but inefficient universal graphs: if $(U_n)_{n \in \mathbb{N}}$ are universal graphs for $\mathcal{F}$ then for large enough $d = d(n)$, the graphs $(U_n^d)_{n \in \mathbb{N}}$ are universal for $\mathsf{her}(\mathcal{F}^\square)$. In general, this construction has exponential size: the hypercubes K_2^d are themselves universal for $\mathsf{her}(\{K_2\}^\square)$, but a star with $n-1$ leaves cannot be embedded in K_2^d for $d < n-1$, so these universal graphs are of size at least 2^{n-1}. It is not clear *a priori* whether it is possible to use the universal graphs for the base class $\mathcal{F}$ to obtain more efficient universal graphs for $\mathsf{her}(\mathcal{F}^\square)$, and even less clear for $\mathsf{mon}(\mathcal{F}^\square)$, but we will show in this paper how to do so.

Finally, there was the possibility that *subgraphs* of Cartesian products could provide the first explicit counterexample to the Implicit Graph Conjecture (IGC) of [13, 17], which suggested that the condition $\log |\mathcal{F}_n| = O(n \log n)$ was *sufficient* for $\mathcal{F}$ to admit a labeling

scheme of size $O(\log n)$; this was refuted by a non-constructive counting argument in a recent breakthrough of Hatami & Hatami [11]. There is a labeling scheme of size $O(\log^2 n)$ for the subgraphs of hypercubes, due to a folklore bound of $\log n$ on the degeneracy of this class (see [5]) and a general $O(k \log n)$ labeling scheme for classes of degeneracy k [13]. Designing an efficient labeling scheme for *induced* subgraphs of hypercubes (rather, the weaker question of proving bounds on $|\mathcal{F}_n|$ for this family) was an open problem of Alecu, Atminas, & Lozin [2], resolved concurrently and independently in [8]; this also gave an example of a class with an efficient labeling scheme but unbounded *functionality*, answering another open question of [2]. Also independently, Chepoi, Labourel, & Ratel [3] studied the structure of general Cartesian products, motivated by the problem of designing labeling schemes for the classes $\mathsf{mon}(\mathcal{F}^{\square})$. They give upper bounds (via bounds on the degeneracy) for a number of special cases but do not improve on the $O(\log^2 n)$ bound for hypercubes. The following 3 observations then suggested that subgraphs of Cartesian products could give the first explicit counterexample to the IGC (and this was posed as an open problem in [4]):

1. It is shown in [4] that, while *induced* subgraphs of hypercubes have a constant-size *adjacency sketch* (a probabilistic version of a labeling scheme), the *subgraphs* of hypercubes do not, so, with respect to *randomized* labels, subgraphs are more complex than induced subgraphs.
2. The above result shows that the class of subgraphs of hypercubes is a counterexample to a conjecture of [10]. That conjecture was refuted earlier by a construction of [7] that, with some extension, refuted the IGC itself [11].
3. The previous work considering Cartesian products [3, 8, 10, 2, 1] had not improved on the $O(\log^2 n)$ bound for subgraphs.

Alas, a consequence of our main result is that subgraphs of Cartesian products are *not* counterexamples to the IGC.

Results and techniques

We improve the best-known $O(\log^2 n)$ bound for subgraphs of hypercubes to the optimal $O(\log n)$, and in general show how to construct optimal labels for all subgraphs and induced subgraphs of Cartesian products. Our proof is short, and departs significantly from standard techniques in the field of labeling schemes: we do not rely on any structural results, graph width parameters, or decompositions, and instead use communication complexity (as in [8, 10]), encoding, and hashing arguments, which may be useful for future work on labeling schemes. We prove:

▶ **Theorem 1.** *Let $\mathcal{F}$ be a hereditary class with an adjacency labeling scheme of size $s(n)$. Then:*

1. $\mathsf{her}(\mathcal{F}^{\square})$ *has a labeling scheme of size at most* $4s(n) + O(\log n)$.
2. $\mathsf{mon}(\mathcal{F}^{\square})$ *has a labeling scheme where each* $G \in \mathsf{mon}(\mathcal{F}^{\square})$ *on n vertices is given labels of size at most* $4s(n) + O(k(G) + \log n)$*, where $k(G)$ is the degeneracy of G.*

We allow $\mathcal{F}$ to be finite, in which case $s(n) = O(1)$; in particular, setting $\mathcal{F} = \{K_2, K_1\}$, we get the result for hypercubes:

▶ **Corollary 2.** *Let $\mathcal{H}$ be the class of hypercube graphs. Then* $\mathsf{mon}(\mathcal{H})$ *has a labeling scheme of size* $O(\log n)$.

All of the labeling schemes of Chepoi, Labourel, & Ratel [3] are obtained by bounding $k(G)$ and applying the black-box $O(k(G) \cdot \log n)$ bound of [13]. For example, they get labels of size $O(d \log^2 n)$ when the base class $\mathcal{F}$ has degeneracy d, by showing that $\mathsf{mon}(\mathcal{F}^{\square})$ has degeneracy $O(d \log n)$. Our result can be substituted for that black-box, replacing the

multiplicative $O(\log n)$ with an *additive* $O(\log n)$, thereby improving all of the results of [3] when combined with their bounds on $k(G)$; for example, achieving $O(d \log n)$ when $\mathcal{F}$ has degeneracy d.

For subgraphs of hypercubes, [3] observed that a bound of $O(\mathsf{vc}(G) \log n)$ follows from the inequality $k(G) \leq \mathsf{vc}(G)$ due to Haussler [12], where $\mathsf{vc}(G)$ is the VC dimension[1], which can be as large as $\log n$ but is often much smaller; they generalize this inequality in various ways to other Cartesian products. Our result supercedes the VC dimension result for hypercubes.

Theorem 1 is optimal up to constant factors (which we have not tried to optimize), and yields the following corollary (see Section 3 for proofs).

▶ **Corollary 3.** *If a hereditary class $\mathcal{F}$ has an efficient labeling scheme, then so do* $\mathsf{her}(\mathcal{F}^{\square})$ *and* $\mathsf{mon}(\mathcal{F}^{\square})$.

One of our main motivations was to find explicit counterexamples to the IGC; a consequence of the above corollary is that, counterexamples to the IGC cannot be obtained by taking the monotone closure of Cartesian products of some hereditary class $\mathcal{F}$, unless $\mathcal{F}$ itself is already a counterexample. This leaves open the problem of finding an explicit counterexample to the IGC, which would require developing the first lower-bound technique for adjacency labeling schemes.

2 Adjacency Labeling Scheme

Notation

For two binary strings x, y, we write $x \oplus y$ for the bitwise XOR. For two graphs G and H, we will write $G \subset H$ if G is a subgraph of H, and $G \subset_I H$ if G is an *induced* subgraph of H. We will write $V(G)$ and $E(G)$ as the vertex and edge set of a graph G, respectively. All graphs in this paper are simple and undirected. A graph G has *degeneracy* k if all subgraphs of G have a vertex of degree at most k.

Strategy

Suppose $G \subset G_1 \square \cdots \square G_d$ is a subgraph of a Cartesian product. Then $V(G) \subseteq V(G_1) \times \cdots \times V(G_d)$. Let $H \subset_I G_1 \square \cdots \square G_d$ be the subgraph induced by $V(G)$, so that $E(G) \subseteq E(H)$. One may think of G as being obtained from the induced subgraph H by deleting some edges. Then two vertices $x, y \in V(G)$ are adjacent if and only if:

1. There exists exactly one coordinate $i \in [d]$ where $x_i \neq y_i$;
2. On this coordinate, $x_i y_i \in E(G_i)$; and,
3. The edge $xy \in E(H)$ has not been deleted in $E(G)$.

We construct the labels for vertices in G in three phases, which check these conditions in sequence.

2.1 Phase 1: Exactly One Difference

We give two proofs for Phase 1. The first is a reduction to the k-Hamming Distance communication protocol. The second proof is direct and self-contained; it is an extension of the proof of the labeling scheme for induced subgraphs of hypercubes, in the unpublished note [9] (adapted from [8, 10]). In both cases the labels are obtained by the probabilistic method, and are efficiently computable by a randomized algorithm.

[1] See [3] for the definition of VC dimension

For any alphabet Σ and any two strings $x, y \in \Sigma^d$ where $d \in \mathbb{N}$, write $\mathsf{dist}(x, y)$ for the Hamming distance between x and y, i.e. $\mathsf{dist}(x, y) = |\{i \in [d] : x_i \neq y_i\}|$.

For the first proof, we require a result in communication complexity (which we translate into our terminology). A version with two-sided error appears in [18], the one-sided error version below is implicit in [10] (and may appear elsewhere in the literature, which we did not find).

▶ **Theorem 4** ([18, 10]). *There exists a constant $c > 0$ satisfying the following. For any $k \in \mathbb{N}$, there exists a function $D : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ such that, for any $d \in \mathbb{N}$ and set $S \subseteq \{0,1\}^d$ of size $|S| = n$, there exists a probability distribution L over functions $\ell : S \to \{0,1\}^{ck^2}$, where for all $x, y \in S$,*

1. *If $\mathsf{dist}(x, y) \leq k$ then $\mathbb{P}_{\ell \sim L}[D(\ell(x), \ell(y)) = 1] = 1$; and,*
2. *If $\mathsf{dist}(x, y) > k$ then $\mathbb{P}_{\ell \sim L}[D(\ell(x), \ell(y)) = 0] \geq 2/3$.*

We transform these randomized labels into deterministic labels using standard arguments:

▶ **Proposition 5.** *There exists a constant $c > 0$ satisfying the following. For any $k \in \mathbb{N}$, there exists a function $D : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ such that, for any $d \in \mathbb{N}$ and set $S \subseteq \{0,1\}^d$ of size $|S| = n$, there exists a function $\ell : S \to \{0,1\}^{ck^2 \log n}$ where for all $x, y \in S$, $D(\ell(x), \ell(y)) = 1$ if and only if $\mathsf{dist}(x, y) \leq k$.*

Proof. Let $D' : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$, $c > 0$, and L be the function, the constant, and the probability distribution given for S by Theorem 4. Let $q = \lceil 2 \log_3 n \rceil$, and let L' be the distribution over functions defined by choosing $\ell_1, \ldots, \ell_q \sim L$ independently at random, and setting $\ell(x) = (\ell_1(x), \ell_2(x), \ldots, \ell_q(x))$ for each $x \in S$. Define $D : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ such that

$$D(\ell(x), \ell(y)) = \bigwedge_{i=1}^{q} D'(\ell_i(x), \ell_i(y)) \,.$$

Observe that, if $x, y \in S$ have $\mathsf{dist}(x, y) \leq k$ then $\mathbb{P}[D(\ell(x), \ell(y)) = 1] = 1$ since for each $i \in [q]$ we have $\mathbb{P}[D'(\ell_i(x), \ell_i(y)) = 1] = 1$. On the other hand, if $x, y \in S$ have $\mathsf{dist}(x, y) > k$, then

$$\mathbb{P}[D(\ell(x), \ell(y)) = 1] < (1/3)^q \leq 1/n^2 \,.$$

By the union bound, the probability that there exist $x, y \in S$ such that $D(\ell(x), \ell(y))$ takes the incorrect value is strictly less than 1. Therefore there exists a fixed function $\ell : S \to \{0,1\}^{ck^2 q}$ satisfying the required conditions, where $ck^2 q = Ck^2 \log n$ for an appropriate constant C. ◀

We reduce the problem for alphabets Σ to the 2-Hamming Distance labeling problem above.

▶ **Lemma 6.** *There exists a function $D : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ and a constant $c > 0$ such that, for any countable alphabet Σ, any $d \in \mathbb{N}$, and any set $S \subseteq \Sigma^d$ of size $|S| = n$, there exists a function $\ell : S \to \{0,1\}^k$ for $k \leq c \log n$, where $D(\ell(x), \ell(y)) = 1$ if and only if $\mathsf{dist}(x, y) = 1$.*

Proof. Since $\lceil \log n \rceil$ bits can be added to any $\ell(x)$ to ensure that $\ell(x)$ is unique, it suffices to construct functions D, ℓ where $D(\ell(x), \ell(y)) = 1$ if and only if $\mathsf{dist}(x, y) \leq 1$, instead of $\mathsf{dist}(x, y) = 1$ exactly.

Since S has at most n elements, we may assume that Σ has a finite number N of elements, since we may reduce to the set of elements which appear in the strings S. We may then identify Σ with $[N]$ and define an encoding $\mathsf{enc} : [N] \to \{0,1\}^N$ where for any $\sigma \in [N]$, $\mathsf{enc}(\sigma)$ is the string that takes value 1 on coordinate σ, and all other coordinates take value 0.

Abusing notation, for any $x \in \Sigma^d$, we may now define the concatenated encoding $\mathsf{enc}(x) = \mathsf{enc}(x_1) \circ \mathsf{enc}(x_2) \circ \cdots \circ \mathsf{enc}(x_d)$, where $\circ$ denotes concatenation. It is easy to verify that for any $x, y \in \Sigma^d$, $\mathsf{dist}(\mathsf{enc}(x), \mathsf{enc}(y)) = 2 \cdot \mathsf{dist}(x, y)$. We may therefore apply Proposition 5 with $k = 2$ on the set $S' = \{\mathsf{enc}(x) : x \in S\}$ to obtain a function $D : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$, a constant $C > 0$, and a function $\ell' : S' \to \{0,1\}^{C \log n}$ such that for all $x, y \in S$,

$$D(\ell'(\mathsf{enc}(x)), \ell'(\mathsf{enc}(y))) = 1 \iff \mathsf{dist}(\mathsf{enc}(x), \mathsf{enc}(y)) \leq 2 \iff \mathsf{dist}(x, y) \leq 1\,.$$

We may then conclude the proof by setting $\ell(x) = \ell'(\mathsf{enc}(x))$ for each $x \in S$. ◀

Below, we give an alternative, direct proof that does not reduce to k-Hamming Distance.

▶ **Proposition 7.** *For any set $S \subseteq \{0,1\}^d$, there exists a random function $\ell : S \to \{0,1\}^4$ such that, for all $x, y \in S$,*

1. *If $\mathsf{dist}(x, y) \leq 1$ then $\mathop{\mathbb{P}}_{\ell}[\mathsf{dist}(\ell(x), \ell(y)) \leq 1] = 1$, and*
2. *If $\mathsf{dist}(x, y) > 1$ then $\mathop{\mathbb{P}}_{\ell}[\mathsf{dist}(\ell(x), \ell(y)) \leq 1] \leq 3/4$.*

Proof. Choose a uniformly random map $p : [d] \to [4]$ and partition $[d]$ into four sets $P_j = p^{-1}(j)$. For each $i \in [4]$, define $\ell(x)_i := \bigoplus_{j \in P_i} x_j$.

Let $x, y \in S$ and write $w = \ell(x) \oplus \ell(y)$. Note that $\mathsf{dist}(\ell(x), \ell(y)) = |w|$, which is the number of 1s in w. If $\mathsf{dist}(x, y) = 0$ then $\mathsf{dist}(\ell(x), \ell(y)) = 0 \leq 1$. Now suppose $\mathsf{dist}(x, y) = 1$. For any choice of $p : [d] \to [4]$, one of the sets P_i contains the differing coordinate and will have $w_i = 1$, while the other three sets P_j will have $w_j = 0$, so $\mathop{\mathbb{P}}_{\ell}[\mathsf{dist}(\ell(x), \ell(y)) \leq 1] = 1$.

Now suppose $\mathsf{dist}(x, y) = t \geq 2$. We will show that $|w| \leq 1$ with probability at most $3/4$. Note that w is obtained by the random process where $\vec{0} = w^{(0)}, w = w^{(t)}$, and $w^{(i)}$ is obtained from $w^{(i-1)}$ by flipping a uniformly random coordinate.

Observe that, for $i \geq 1$, $\mathbb{P}\left[w^{(i)} = \vec{0}\right] \leq 1/4$. This is because $w^{(i)} = \vec{0}$ can occur only if $|w^{(i-1)}| = 1$, so the probability of flipping the 1-valued coordinate is $1/4$. If $|w^{(i-1)}| \geq 1$ then $\mathbb{P}\left[|w^{(i)}| \leq 1 \;\middle|\; |w^{(i-1)}| \geq 1\right] \leq 1/2$ since either $|w^{(i-1)}| = 1$ and then $|w^{(i)}| = 0 \leq 1$ with probability $1/4$, or $|w^{(i-1)}| \geq 2$ and $|w^{(i)}| = 1$ with probability at most $1/2$. Then, for $t \geq 2$,

$$\begin{aligned}\mathbb{P}\left[|w^{(t)}| \leq 1\right] &= \mathbb{P}\left[w^{(t-1)} = \vec{0}\right] + \mathbb{P}\left[|w^{(t-1)}| \geq 1\right] \cdot \mathbb{P}\left[|w^{(t)}| = 1 \;\middle|\; |w^{(t-1)}| \geq 1\right] \\ &\leq \frac{1}{4} + \frac{1}{2} = \frac{3}{4}\,.\end{aligned}$$

◀

▶ **Proposition 8.** *There exists a function $D : \{0,1\}^4 \times \{0,1\}^4 \to \{0,1\}$ such that, for any countable alphabet, Σ, any $d \in \mathbb{N}$, and any $S \subseteq \Sigma^d$ of size $n = |S|$, there exists a random function $\ell : S \to \{0,1\}^4$ such that, for all $x, y \in S$,*

1. *If $\mathsf{dist}(x, y) \leq 1$, then $\mathop{\mathbb{P}}_{\ell}[D(\ell(x), \ell(y)) = 1] = 1$, and*
2. *If $\mathsf{dist}(x, y) > 1$, then $\mathop{\mathbb{P}}_{\ell}[D(\ell(x), \ell(y)) = 1] \leq 15/16$.*

Proof. For each $\sigma \in \Sigma$ and $i \in [d]$, generate an independently and uniformly random bit $q_i(\sigma) \sim \{0,1\}$. Then for each $x \in S$ define $p(x) = (q_1(x_1), \ldots, q_d(x_d)) \in \{0,1\}^d$ and $S' = \{p(x) : x \in S\}$, and let ℓ' be the random function $S' \to \{0,1\}^4$ guaranteed to exist by Proposition 7. We define the random function $\ell : S \to \{0,1\}^4$ as $\ell(x) = \ell'(p(x))$. We define $D(\ell(x), \ell(y)) = 1$ if and only if $\mathsf{dist}(\ell'(p(x)), \ell'(p(y))) \leq 1$.

Let $x, y \in S$. If $\mathsf{dist}(x,y) \le 1$, so there is a unique $i \in [d]$ with $x_i \neq y_i$, then

$$\mathbb{P}\left[\mathsf{dist}(p(x),p(y)) = 1\right] = \mathbb{P}\left[q_i(x_i) \neq q_i(y_i)\right] = \mathbb{P}\left[\mathsf{dist}(p(x),p(y)) = 0\right] = 1/2\,,$$

so $\mathbb{P}\left[\mathsf{dist}(p(x),p(y)) \le 1\right] = 1$. Then by Proposition 7,

$$\mathbb{P}\left[D(\ell(x),\ell(y)) = 1\right] = \mathbb{P}\left[\mathsf{dist}(\ell'(p(x)),\ell'(p(y))) \le 1\right] = 1\,.$$

If $\mathsf{dist}(x,y) > 1$ so that there are distinct $i, i' \in [d]$ such that $x_i \neq y_i$ and $x_{i'} \neq y_{i'}$, then

$$\mathbb{P}\left[\mathsf{dist}(p(x),p(y)) \ge 2\right] \ge \mathbb{P}\left[q_i(x_i) \neq q_i(y_i) \wedge q_{i'}(x_{i'}) \neq q_{i'}(y_{i'})\right] = 1/4\,.$$

Then by Proposition 7,

$$\begin{aligned}\mathbb{P}\left[D(\ell(x),\ell(y)) = 1\right] &= \mathbb{P}\left[\mathsf{dist}(\ell'(p(x)),\ell'(p(y))) \le 1\right] \\ &= \mathbb{P}\left[\mathsf{dist}(p(x),p(y)) \le 1 \vee \mathsf{dist}(\ell'(p(x)),\ell'(p(y))) \le 1\right] \\ &\le 3/4 + (1-3/4)(3/4) = 15/16\,.\end{aligned}$$

◀

The alternative proof of Lemma 6 now concludes by using Proposition 8 with a nearly identical derandomization argument as in Proposition 5.

2.2 Phase 2: Induced Subgraphs

After the first phase, we are guaranteed that there is a unique coordinate $i \in [d]$ where $x_i \neq y_i$. In the second phase we wish to determine whether $x_i y_i \in E(G_i)$. It is convenient to have labeling schemes for the factors $G_1, \dots, G_d$ where we can XOR the labels together while retaining the ability to compute adjacency. Define an *XOR-labeling scheme* the same as an adjacency labeling scheme, with the restriction that for each $s \in \mathbb{N}$ there is some function $g_s : \{0,1\}^s \to \{0,1\}$ such that on any two labels $\ell(x), \ell(y)$ of size s, the decoder outputs $D(\ell(x),\ell(y)) = g_s(\ell(x) \oplus \ell(y))$. Any labeling scheme can be transformed into an XOR-labeling scheme with at most a constant-factor loss:

▶ **Lemma 9.** *Let $\mathcal{F}$ be any class of graphs with an adjacency labeling scheme of size $s(n)$. Then $\mathcal{F}$ admits an XOR-labeling scheme of size at most $4s(n)$.*

Proof. Let $D : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ be the decoder of the adjacency labeling scheme for $\mathcal{F}$, fix any $n \in \mathbb{N}$, and write $s = s(n)$. Note that D must be symmetric, so $D(a,b) = D(b,a)$ for any $a, b \in \{0,1\}^s$. Let $\phi : \{0,1\}^s \to \{0,1\}^{4s}$ be uniformly randomly chosen, so that for every $z \in \{0,1\}^s$, $\phi(z) \sim \{0,1\}^{4s}$ is a uniform and independently random variable. For any two distinct pairs $\{z_1, z_2\}, \{z_1', z_2'\} \in \binom{\{0,1\}^s}{2}$ where $z_1 \neq z_2$, $z_1' \neq z_2'$, and $\{z_1, z_2\} \neq \{z_1', z_2'\}$, the probability that $\phi(z_1) \oplus \phi(z_2) = \phi(z_1') \oplus \phi(z_2')$ is at most 2^{-4s}, since at least one of the variables $\phi(z_1), \phi(z_2), \phi(z_1'), \phi(z_2')$ is independent of the other ones. Therefore, by the union bound,

$$\mathbb{P}\left[\exists \{z_1,z_2\},\{z_1',z_2'\} : \phi(z_1) \oplus \phi(z_2) = \phi(z_1') \oplus \phi(z_2')\right] \le \binom{2^s}{2}^2 2^{-4s} \le \frac{1}{4}\,.$$

Then there is $\phi : \{0,1\}^s \to \{0,1\}^{4s}$ such that each distinct pair $\{z_1, z_2\} \in \binom{\{0,1\}^s}{2}$ is assigned has a distinct unique value $\phi(z_1) \oplus \phi(z_2)$. So the function $\Phi(\{z_1, z_2\}) := \phi(z_1) \oplus \phi(z_2)$ is a one-to-one map $\binom{\{0,1\}^s}{2} \to \{0,1\}^{4s}$. Then for any graph $G \in \mathcal{F}$ on n vertices, with labeling $\ell : V(G) \to \{0,1\}^s$, we may assign the new label $\phi(\ell(x))$ to each vertex x. On labels $\phi(\ell(x)), \phi(\ell(y)) \in \{0,1\}^s$, the decoder for the XOR-labeling scheme simply computes $\{\ell(x), \ell(y)\} = \Phi^{-1}(\phi(\ell(x)) \oplus \phi(\ell(y)))$ and outputs $D(\ell(x),\ell(y))$, where we are using the fact that $D(\ell(x),\ell(y)) = D(\ell(y),\ell(x))$, so that the ordering of the pair $\{\ell(x),\ell(y)\}$ does not matter. ◀

We can now prove the first part of Theorem 1.

▶ **Lemma 10.** *Let $\mathcal{F}$ be a hereditary class of graphs that admits an adjacency labeling scheme of size $s(n)$. Then $\mathsf{her}(\mathcal{F}^{\square})$ admits an adjacency labeling scheme of size $4s(n) + O(\log n)$.*

Proof. By Lemma 9, there is an XOR-labeling scheme for $\mathcal{F}$ with labels of size $4s(n)$. Let $D : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ be the decoder for this scheme, with $D(a,b) = g(a \oplus b)$ for some function g. Design the labels for $\mathsf{her}(\mathcal{F}^{\square})$ as follows. Consider a graph $G \in \mathsf{her}(\mathcal{F}^{\square})$, so that $G \subset_I G_1 \square G_2 \square \cdots \square G_d$ for some $d \in \mathbb{N}$ and $G_i \in \mathcal{F}$ for each $i \in [d]$. Since $\mathcal{F}$ is hereditary, we may assume that each G_i has at most n vertices; otherwise we could simply replace it with the subgraph of G_i induced by the vertices $\{x_i : x \in V(G)\}$. For each $x = (x_1, \ldots, x_d) \in V(G)$, construct the label as follows:

1. Treating the vertices in each G_i as characters of the alphabet $[n]$, use $O(\log n)$ bits to assign the label given to $x = (x_1, \ldots, x_d) \in [n]^d$ by Lemma 6.
2. Using $4s(n)$ bits, append the vector $\bigoplus_{i \in [d]} \ell_i(x_i)$, where $\ell_i(x_i)$ is the label of $x_i \in V(G_i)$ in graph G_i, according to the XOR-labeling scheme for $\mathcal{F}$.

The decoder operates as follows. Given the labels for $x, y \in V(G)$:

1. If x and y differ on exactly one coordinate, as determined by the first part of the label, continue to the next step. Otherwise output "not adjacent."
2. Now guaranteed that there is a unique $i \in [d]$ such that $x_i \neq y_i$, output "adjacent" if and only if the following is 1:

$$\begin{aligned} D\left(\bigoplus_{j\in[d]} \ell_j(x_j), \bigoplus_{j\in[d]} \ell_j(y_j)\right) &= g\left(\bigoplus_{j\in[d]} \ell_j(x_j) \oplus \bigoplus \ell_j(y_j)\right) \\ &= g\left(\ell_i(x_i) \oplus \ell_i(y_i) \oplus \bigoplus_{j\neq i} \ell_j(x_j) \oplus \ell_j(y_j)\right) \\ &= g(\ell_i(x_i) \oplus \ell_i(y_i)), \end{aligned}$$

   where the final equality holds because $x_j = y_j$ for all $j \neq i$, so $\ell_j(x_j) = \ell_j(y_j)$. Then the output value is 1 if and only $x_i y_i$ is an edge of G_i; equivalently, xy is an edge of G.

This concludes the proof. ◀

The XOR-labeling trick can also be used to simplify the proof of [10] for adjacency *sketches* of Cartesian products. That proof is similar to the one above, except it uses a two-level hashing scheme and some other tricks to avoid destroying the labels of x_i and y_i with the XOR (with sufficiently large probability of success). This two-level hashing approach does not succeed in our current setting, and we avoid it with XOR-labeling.

2.3 Phase 3: Subgraphs

Finally, we must check whether the edge $xy \in E(H)$ in the *induced* subgraph $H \subset_I G_1 \square \cdots \square G_d$ has been deleted in $E(G)$. There is a minimal and perfect tool for this task:

▶ **Theorem 11** (Minimal Perfect Hashing)**.** *For every $m, k \in \mathbb{N}$, there is a family $\mathcal{P}_{m,k}$ of hash functions $[m] \to [k]$ such that, for any $S \subseteq [m]$ of size k, there exists $h \in \mathcal{P}_{m,k}$ where the image of S under h is $[k]$ and for every distinct $i, j \in S$ we have $h(i) \neq h(j)$. The function h can be stored in $k \ln e + \log\log m + o(k + \log\log m)$ bits of space and it can be computed by a randomized algorithm in expected time $O(k + \log\log m)$.*

Minimal perfect hashing has been well-studied. A proof of the space bound appears in [14] and significant effort has been applied to improving the construction and evaluation time. We take the above statement from [6]. We now conclude the proof of Theorem 1 by applying the next lemma to the class $\mathcal{G} = \mathsf{her}(\mathcal{F}^{\square})$, using the labeling scheme for $\mathsf{her}(\mathcal{F}^{\square})$ obtained in Lemma 10 (note that $\mathsf{mon}(\mathsf{her}(\mathcal{F}^{\square})) = \mathsf{mon}(\mathcal{F}^{\square})$).

▶ **Lemma 12.** *Let $\mathcal{G}$ be any graph class which admits an adjacency labeling scheme of size $s(n)$. Then $\mathsf{mon}(\mathcal{G})$ admits an adjacency labeling scheme where each $G \in \mathsf{mon}(\mathcal{G})$ on n vertices has labels of size $s(n) + O(k(G) + \log n)$, where $k(G)$ is the degeneracy of G.*

Proof. Let $G \in \mathsf{mon}(\mathcal{G})$ have n vertices, so that it is a subgraph of $H \in \mathcal{G}$ on n vertices. The labeling scheme is as follows.

1. Fix a total order $\prec$ on $V(H)$ such that each vertex x has at most $k = k(G)$ neighbors y in H such that $x \prec y$; this exists by definition. We will identify each vertex x with its position in the order.
2. For each vertex x, assign the label as follows:
   a. Use $s(n)$ bits for the adjacency label of x in H.
   b. Use $\log n$ bits to indicate x (the position in the order).
   c. Let $N^+(x)$ be the set of neighbors $x \prec y$. Construct a perfect hash function $h_x : N^+(x) \to [k]$ and store it, using $O(k + \log\log n)$ bits.
   d. Use k bits to write the function $\mathsf{edge}_x : [k] \to \{0,1\}$ which takes value 1 on $i \in [k]$ if and only if xy is an edge of G, where y is the unique vertex in $N^+(x)$ satisfying $h_x(y) = i$.

Given the labels for x and y, the decoder performs the following:

1. If xy are not adjacent in H, output "not adjacent".
2. Otherwise xy are adjacent. If $x \prec y$, we are guaranteed that y is in the domain of h_x, so output "adjacent" if and only if $\mathsf{edge}_x(h_x(y)) = 1$. If $y \prec x$, output "adjacent" if and only if $\mathsf{edge}_y(h_y(x)) = 1$.

This concludes the proof. ◀

3 Optimality

We now prove the optimality of our labeling schemes, and Corollary 3. We require:

▶ **Proposition 13.** *For any hereditary class $\mathcal{F}$, let $k(n)$ be the maximum degeneracy of an n-vertex graph $G \in \mathsf{her}(\mathcal{F}^{\square})$. Then $\mathsf{her}(\mathcal{F}^{\square})$ contains a graph H on n vertices with at least $n \cdot k(n)/4$ edges, so $\mathsf{mon}(\mathcal{F}^{\square})$ contains all $2^{n \cdot k(n)/4}$ spanning subgraphs of H.*

Proof. Since G has degeneracy $k = k(n)$, it contains an induced subgraph $G' \subset_I G$ with minimum degree k and $n_1 \leq n$ vertices. If $n_1 \geq n/2$ then G itself has at least $kn_1/2 \geq kn/4$ edges, and we are done. Now assume $n_1 < n/2$. Since $G \in \mathsf{her}(\mathcal{F}^{\square})$, $G \subset_I H_1 \square \cdots \square H_t$ for some $t \in \mathbb{N}$ and $H_i \in \mathcal{F}$. So for any $d \in \mathbb{N}$, the graph $(G')^d \subset_I (H_1 \square \cdots \square H_t)^d$ belongs to $\mathsf{her}(\mathcal{F}^{\square})$. Consider the graph $H \subset_I (G')^d$ defined as follows. Choose any $w \in V(G')$, and for each $i \in [d]$ let

$$V_i = \{(v_1, v_2, \ldots, v_d) : v_i \in V(G') \text{ and } \forall j \neq i, v_j = w\},$$

and let H be the graph induced by vertices $V_1 \cup \cdots \cup V_d$. Then H has dn_1 vertices, each of degree at least k, since each $v \in V_i$ is adjacent to k other vertices in V_i. Set $d = \lceil n/n_1 \rceil$, so that H has at least n vertices, and let $m = dn_1 - n$, which satisfies $m < n_1$. Remove any m vertices of V_1. The remaining graph H' has n vertices, and at least $(d-1)n_1 \geq n - n_1 > n/2$ vertices of degree k. Then H' has at least $kn/4$ edges. ◀

The next proposition shows that Theorem 1 is optimal up to constant factors. It is straightforward to check that this proposition implies Corollary 3.

▶ **Proposition 14.** *Let $\mathcal{F}$ be a hereditary class whose optimal adjacency labeling scheme has size $s(n)$ and which contains a graph with at least one edge. Then any adjacency labeling scheme for $\mathsf{her}(\mathcal{F}^{\square})$ has size at least $\Omega(s(n) + \log n)$, and any adjacency labeling scheme for $\mathsf{mon}(\mathcal{F}^{\square})$ has size at least $\Omega(s(n) + k(n) + \log n)$, where $k(n)$ is the maximum degeneracy of any n-vertex graph in $\mathsf{mon}(\mathcal{F}^{\square})$.*

Proof. Since $\mathcal{F} \subseteq \mathsf{her}(\mathcal{F}^{\square})$ and $\mathcal{F} \subseteq \mathsf{mon}(\mathcal{F}^{\square})$, we have a lower bound of $s(n)$ for the labeling schemes for both of these classes. Since $\mathcal{F}$ contains a graph G with at least one edge, the Cartesian products contain the class of hypercubes: $\mathsf{her}(\{K_2\}^{\square}) \subseteq \mathsf{her}(\mathcal{F}^{\square}) \subseteq \mathsf{mon}(\mathcal{F}^{\square})$. A labeling scheme for $\mathsf{her}(\{K_2\}^{\square})$ must have size $\Omega(\log n)$ (which can be seen since each vertex of K_2^d has a unique neighborhood and thus requires a unique label). This establishes the lower bound for $\mathsf{her}(\mathcal{F}^{\square})$, since the labels must have size $\max\{s(n), \Omega(\log n)\} = \Omega(s(n) + \log n)$. Finally, by Proposition 13, the number of n-vertex graphs in $\mathsf{mon}(\mathcal{F}^{\square})$ is at least $2^{\Omega(nk(n))}$, so there is a lower bound on the label size of $\Omega(k(n))$, which implies a lower bound of $\max\{s(n), \Omega(\log n), \Omega(k(n))\} = \Omega(s(n) + k(n) + \log n)$ for $\mathsf{mon}(\mathcal{F}^{\square})$. ◀

References

1. Bogdan Alecu, Vladimir E. Alekseev, Aistis Atminas, Vadim Lozin, and Viktor Zamaraev. Graph parameters, implicit representations and factorial properties. In submission, 2022.
2. Bogdan Alecu, Aistis Atminas, and Vadim Lozin. Graph functionality. *Journal of Combinatorial Theory, Series B*, 147:139–158, 2021.
3. Victor Chepoi, Arnaud Labourel, and Sébastien Ratel. On density of subgraphs of Cartesian products. *Journal of Graph Theory*, 93(1):64–87, 2020.
4. Louis Esperet, Nathaniel Harms, and Andrey Kupavskii. Sketching distances in monotone graph classes. In *International Conference on Randomization and Computation (RANDOM 2022)*, 2022.
5. Ron L Graham. On primitive graphs and optimal vertex assignments. *Annals of the New York academy of sciences*, 175(1):170–186, 1970.
6. Torben Hagerup and Torsten Tholey. Efficient minimal perfect hashing in nearly minimal space. In *Annual Symposium on Theoretical Aspects of Computer Science (STACS 2001)*, pages 317–326. Springer, 2001.
7. Lianna Hambardzumyan, Hamed Hatami, and Pooya Hatami. A counter-example to the probabilistic universal graph conjecture via randomized communication complexity. *Discrete Applied Math.*, 2022.
8. Nathaniel Harms. Universal communication, universal graphs, and graph labeling. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151, page 33. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020.
9. Nathaniel Harms. Adjacency labeling and sketching for induced subgraphs of the hypercube. `https://cs.uwaterloo.ca/~nharms/downloads/hypercube_sketch.pdf`, 2022. URL: `https://cs.uwaterloo.ca/~nharms/downloads/hypercube_sketch.pdf`.
10. Nathaniel Harms, Sebastian Wild, and Viktor Zamaraev. Randomized communication and implicit graph representations. In *54th Annual Symposium on Theory of Computing (STOC 2022)*, 2022.
11. Hamed Hatami and Pooya Hatami. The implicit graph conjecture is false. In *63rd IEEE Symposium on Foundations of Computer Science (FOCS 2022)*, 2022.
12. David Haussler. Sphere packing numbers for subsets of the Boolean n-cube with bounded Vapnik-Chervonenkis dimension. *Journal of Combinatorial Theory, Series A*, 69(2):217–232, 1995.

13 Sampath Kannan, Moni Naor, and Steven Rudich. Implicit representation of graphs. *SIAM Journal on Discrete Mathematics*, 5(4):596–603, 1992.

14 Kurt Mehlhorn. *Data Structures and Algorithms 1 Sorting and Searching*. Monographs in Theoretical Computer Science. An EATCS Series, 1. Springer Berlin Heidelberg, Berlin, Heidelberg, 1st ed. 1984. edition, 1984.

15 John H Muller. Local structure in graph classes, 1989.

16 Edward R Scheinerman. Local representations using very short labels. *Discrete mathematics*, 203(1-3):287–290, 1999.

17 Jeremy P Spinrad. *Efficient graph representations*. American Mathematical Society, 2003.

18 Andrew Chi-Chih Yao. On the power of quantum fingerprinting. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing (STOC)*, pages 77–81, 2003.

Truthful Matching with Online Items and Offline Agents

Michal Feldman
Blavatnik School of Computer Science, Tel Aviv University, Israel
Microsoft Research, Herzliya, Israel

Federico Fusco
Department of Computer, Control and Management Engineering "Antonio Ruberti", Sapienza University of Rome, Italy

Simon Mauras
Blavatnik School of Computer Science, Tel Aviv University, Israel

Rebecca Reiffenhäuser
Institute for Logic, Language and Computation, University of Amsterdam, The Netherlands

Abstract

We study truthful mechanisms for welfare maximization in online bipartite matching. In our (multi-parameter) setting, every buyer is associated with a (possibly private) desired set of items, and has a private value for being assigned an item in her desired set. Unlike most online matching settings, where agents arrive online, in our setting the items arrive online in an adversarial order while the buyers are present for the entire duration of the process. This poses a significant challenge to the design of truthful mechanisms, due to the ability of buyers to strategize over future rounds. We provide an almost full picture of the competitive ratios in different scenarios, including myopic vs. non-myopic agents, tardy vs. prompt payments, and private vs. public desired sets. Among other results, we identify the frontier up to which the celebrated $e/(e-1)$ competitive ratio for the vertex-weighted online matching of Karp, Vazirani and Vazirani extends to truthful agents and online items.

2012 ACM Subject Classification Theory of computation → Online algorithms; Applied computing → Online auctions

Keywords and phrases Online matching, Karp-Vazirani-Vazirani, truthfulness

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.58

Category Track A: Algorithms, Complexity and Games

Funding Michal Feldman was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 866132) and by the NSF-BSF (grant number 2020788). Federico Fusco and Rebecca Reiffenhäuser were supported by the ERC Advanced Grant 788893 AMDROMA "Algorithmic and Mechanism Design Research in Online Markets" and MIUR PRIN grant Algorithms, Games, and Digital Markets (ALGADIMAR). Federico Fusco was also partially supported by PNRR MUR project PE0000013-FAIR" and PNRR MUR project IR0000013-SoBigData.it. Part of the work of Federico Fusco was done while visiting Michal Feldman at Tel Aviv University.

Acknowledgements The authors are grateful to Amos Fiat and Stefano Leonardi for many useful conversations that have tremendously contributed to this paper.

1 Introduction

Matching in bipartite graphs is a fundamental model that has gained massive importance in numerous applications with the growth of the Internet. Some examples include items and buyers in e-commerce, drivers and passengers in ride-sharing platforms, ad slots and

advertisers in online ad auctions, and jobs and workers in online labor markets. In these applications, it is common that vertices on one side are known from the outset, while vertices from the other side arrive one-by-one in an online fashion. Upon the arrival of an online vertex, its information is revealed (containing, e.g., its set of adjacent edges, and their weights), and the algorithm has to immediately and irrevocably decide either to match it with an available offline partner or leave it unmatched forever. The goal is to maximize the sum of the weights along the matched edges.

A celebrated result in online matching by Karp, Vazirani, and Vazirani [18] shows that, in the unweighted setting, a simple randomized strategy called RANKING achieves a competitive ratio of $e/(e-1)$, and this is optimal. This result extends to the setting where the vertices on the offline side are weighted and the objective is to maximize the sum of the weights of the matched vertices. Although the original algorithm for this problem, PERTURBED-GREEDY [1], was designed for non-strategic settings, online matching problems have also been studied in the presence of *strategic* agents e.g., [21, 25, 11, 7]. This is not a mere theoretical exercise: in many applications of online matching the parties involved are interested in misreporting their true valuations to obtain a better outcome, e.g., combinatorial and ad-auctions, kidney exchange, school-student matching, and house allocation. In the presence of strategic agents, an agent's value is her private information, and is not directly available to the mechanism designer. The main challenge here is to design *incentive-compatible* or *truthful* mechanisms which, besides finding a good matching, also ensure that it is in the agents' best interest to report their true values. In addition to making decisions regarding the matching itself, such mechanisms can also charge some payment from the agents in order to incentivize them to truthfully report their values. Here, each agent strives to maximize her *quasi-linear utility*, i.e. the value she obtains from her assigned item, minus the payment she has to make.

In almost all previous studies, the agents are represented by the vertices on the online side, while the items they are competing over are available offline. In many natural internet applications, e.g. selling advertising opportunities via repeated auctions, the agents are fixed and observe a stream of items arriving online. This motivates the study of a *reversed* online matching problem, where each vertex on the offline side is strategic on her value, and her set of desired items that arrive online. This variant has been considered thus far only in very restricted settings [8, 9]. This is not a coincidence: when agents are present throughout the entire matching process, many new manipulation opportunities arise, and incentivizing truthful behavior is significantly more challenging. Indeed, the online nature of the problem forces any mechanism to repeatedly make irrevocable decisions upon the arrival of goods, lacking knowledge about future opportunities that might arise to the participating agents. The agents – possibly aware of those future opportunities – may strategize to gain benefits in the future, defying standard tools applicable when agents arrive online.

Our work provides a systematic analysis of this scenario, and gives (almost) tight competitive ratios under a rich variation of natural assumptions. We study the problem along different dimensions, as follows. First, we consider two types of agents – myopic and non-myopic – that are characterized by the different information they have on the instance. Myopic agents make strategic considerations that are limited to the current time step, without looking forward into the future (see, e.g., Deng, Panigrahi and Zhang [9]), whereas non-myopic agents optimize across multiple time steps, using the up-front knowledge of the underlying (online) graph. The assumption of myopic agents clearly eradicates some of the difficulties of designing (almost tight) online mechanisms with offline strategic agents, thus allowing us to derive efficient mechanisms from known online matching algorithms, e.g., from Aggarwal et al. [1]. Second, we consider two types of private information. In the first scenario we consider, an

Table 1 Summary of our results, with $\nu = \min(m, n)$, where n is the number of agents and m the number of items.

	deterministic	randomized
prompt	2 (Theorem 12)	$e/(e-1)$ (Theorem 13)

(a) Myopic agents.

	deterministic	randomized
tardy	2 (Theorem 14)	$e/(e-1)$ (Theorem 14)
prompt	$\geq \nu$ (Theorem 3)	$\Omega(\log \nu / \log\log \nu)$ (Theorem 4)

(b) Non-myopic agents with *public* graph edges.

	deterministic	randomized
tardy	equiv. to prompt	equiv. to prompt
prompt	$\leq \nu$ (Theorem 9)	$\mathcal{O}(\log \nu)$ (Theorem 10)

(c) Non-myopic agents with *private* graph edges.

agent's private information consists only of her private value for her desired items, but the set of desired items is publicly known. In the second scenario, both the value and the set of desired items are private information. Notably, in both cases the graph structure is revealed to the mechanism step-by-step, upon the arrival of every item. Finally, we distinguish between *prompt* and *tardy* mechanisms. Both types of mechanisms make allocation decisions immediately. However, they differ in the time at which they make payment decisions. Prompt mechanisms make payment decisions immediately upon allocation, while tardy mechanisms may delay payment decisions to the end of the entire process.

1.1 Our Results and Techniques

We conduct a systematic study of online bipartite matching with online items and offline agents, in a variety of scenarios and we provide (almost) tight bounds for the settings of interest, as summarized in Table 1.

Myopic agents. The simpler setting we investigate is that of myopic agents. These agents care only about their instantaneous utility, and do not strategize over the future. As such, we only consider prompt mechanisms for this type of agents. By exploiting the myopic nature of the agents, it is not difficult to turn the best (non-truthful) algorithms into (truthful) mechanisms. In particular, we construct a deterministic prompt mechanism based on the greedy matching algorithm that is guaranteed to achieve at least a half of the optimal welfare. We also give a randomized prompt mechanism based on the algorithm for weighted online matching [1], which is $e/(e-1)$-competitive. This shows that the transition from non-strategic agents to strategic myopic agents does not lead to a deterioration in efficiency guarantees. Notably, for the special case we study, our bounds for myopic online matching improve vastly over those obtained by Deng, Panigrahi and Zhang [9] for general XOS valuations. The results for myopic agents are reported in Appendix A.

Non-myopic agents with public graph edges. In a more general setting, we consider non-myopic agents who can strategize about their values, but not about their desired items: upon the arrival of an item, the set of agents interested in it is revealed (no strategizing involved), but the agent values are reported by the agents themselves. This variant is single-parameter, for which Myerson's lemma applies [24]. We prove that, if the mechanism is allowed to wait until the end of the online phase to set prices (i.e., tardy mechanism), it is possible to

achieve the same bounds as in the myopic case – by showing that our algorithms Greedy and Perturbed-Greedy maintain a certain form of *global monotonicity*. In contrast, when prices have to be fixed upon item arrival, an agent might hope to receive an item, i.e. one over which there is not as much competition, for a better price later if she waits instead of truthfully reporting her interest in the current item. To avoid this, prompt prices need to be non-decreasing throughout the mechanism, which allows us to show a sharp deterioration from tardy to prompt mechanisms: for deterministic mechanisms, we prove a $\nu = \min(m, n)$ competitive lower bound, where n and m denote the number of agents and items, respectively. For *randomized* prompt mechanisms obtaining such a lower bound is much more challenging, but as a central result we manage to establish an $\Omega(\log \nu / \log \log \nu)$ lower bound, using Yao's minimax principle. Starting from a carefully designed distribution of problem instances with exponentially increasing agent valuations, we employ a primal-dual approach together with our previous observations on the behavior of *deterministic* truthful mechanisms to bound the achievable competitive ratio. Almost matching deterministic and randomized upper bounds for prompt mechanisms are inherited from non-myopic prompt mechanisms with private graph edges. See Section 3 for our results for public graph edges.

Non-myopic agents with private graph edges. We finally consider non-myopic agents when both valuations and the set of desired items are private information. For deterministic prompt mechanisms, the ν lower bound from the case of public graph edges applies. Moreover, we show that in the case of private edges, every deterministic truthful mechanism is essentially prompt. Thus, tardy mechanisms for this case retain the ν lower bound, exhibiting a large gap between tardy mechanisms for public vs. private edges. We then provide a prompt truthful deterministic mechanism that is ν-competitive, matching the lower bound. For randomized prompt truthful mechanisms, the $\Omega(\log \nu / \log \log \nu)$ lower bound from the case of public edges applies to tardy randomized mechanisms as well, since these are probability distributions over deterministic mechanisms and, as stated above, all deterministic truthful mechanisms for private edges are prompt. On the positive side, we provide a randomized prompt truthful mechanism that gives an almost matching competitive ratio of $O(\log \nu)$. This algorithm is based on a tailored explore-exploit approach. Our results for private graph edges are reported in Section 4.

Ex-post vs. ex-ante truthfulness. Finally, we explore the notion of *ex-ante* truthfulness, as opposed to *ex-post* truthfulness, where agents' true declarations maximize their expected utility instead of their utility in *any realization* of the random choices of the mechanism. Clearly, ex-post truthfulness implies ex-ante truthfulness. In the setting with myopic buyers, we only need to consider ex-post truthfulness as we obtain tight approximation in this stronger model that closes the problem also for the ex-ante analogue. In the setting of non-myopic buyers, we show that the additional hardness introduced by truthfulness cannot be fully attributed to the fact that we require ex-post truthfulness. Specifically, we establish a lower bound of 2 for the competitive ratio of ex-ante truthful mechanisms for this setting (even with respect to randomized tardy ones), exhibiting a gap to the corresponding $e/(e-1)$ upper bound for myopic buyers. Our proof utilizes an instance for which we establish lower bounds on the expected utility of various types of agents. We then employ these to show a contradiction to the mechanism's correctness. Our results for ex-ante truthfulness are reported in Section 6.

Remark. Throughout the paper, we assume that weights are assigned to vertices (agents) rather than edges; note, it is well known that for the more general case of edge weights even the algorithmic problem is hopeless (see, e.g., Appendix G of [1]). One may also wonder why we do not study non-myopic agents with public valuations but private edges. The reason is that in the case of public valuations, it is easy to see that agents cannot benefit from misreporting their edges, implying that Greedy and Perturbed-Greedy are truthful.

1.2 Further Related Work

Karp, Vazirani and Vazirani [18] introduce the online matching problem, and study it under one-sided bipartite arrivals. They observe that the trivial 1/2-competitive greedy algorithm (which matches any arriving vertex to an arbitrary unmatched neighbor, if one exists) is optimal among deterministic algorithms for this problem. They also provide a groundbreaking and elegant randomized algorithm for this problem, called Ranking, which achieves an optimal $e/(e-1)$ competitive ratio. The work of Karp, Vazirani and Vazirani [18] was extended to vertex weighted settings by Aggarwal et al. [1], who give an optimal $e/(e-1)$-competitive, randomized algorithm using *random perturbations* of weights by appropriate multiplicative factors. The same bound has been re-proven over the years [6, 10, 14, 12]. Various extensions of one sided online matching and its economic applications (e.g., display ads) have been widely studied, see e.g. the excellent survey of Mehta [22] for further reference. Online matching has also been studied under edge and general vertex arrivals, as well as in different stochastic settings (see e.g., [19, 20, 13, 17, 15, 16]).

An important generalization of assignment problems in the form of matchings are combinatorial auctions, where buyers can obtain a *subset* of the available items, instead of just one. Combinatorial auctions with offline strategic buyers and online items have been recently studied by [9] for submodular and XOS valuations in the case of myopic buyers – considered also in this work – and in the less constrained setting of items that must not be irrevocably assigned at time of arrival. Deng, Panigrahi and Zhang [9] show (for myopic buyers) a sharp separation between submodular valuations, which admit a logarithmic competitive ratio, and XOS valuations, for which a polynomial lower bound is proven. In our work, we prove tight constant bounds for myopic buyers in the important special case of a unit-demand matching.

Cole, Dobzinski and Fleischer [8] formally introduced the notions of *prompt* and *tardy* for mechanisms, after observing the severe negative aspects of many existing (tardy) methods. They study prompt truthful mechanisms for an online problem that is related to ours, but with some restrictions: while agents are still on the offline side of the graph, their items of interest are restricted to form an interval over the online steps (which corresponds to the interval buyers are present). Further, agents report their departure time (which can be public/private) once they arrive, and their arrival time is public knowledge. Babaioff, Blumrosen and Roth [4] later investigated truthful prompt mechanisms for allocating an unknown number of identical items arriving online, which can be phrased in our model as having all desired sets equal to the same prefix of the sequence of items. Both of these works [8, 4] are close to ours in spirit. They present logarithmic-competitive prompt mechanisms in restricted settings, and prove lower bounds using Yao's pinciple (≥ 2 in [8], and $\Omega(\log\log n)$ in [4]). The notions of tardy and prompt mechanisms have since been adopted in the literature, see e.g. [3, 28]. The model of offline agents and online items has been the subject of extensive investigation in economic theory in dynamic mechanism design. Despite this obvious relation to our setting, there are fundamental differences (see for example [23, 2, 5]). In dynamic mechanism design, a strategic buyer learns her valuations at time of arrival of each item. Opposed to our setting, priors on agents' valuations for each online item are usually known beforehand. Finally,

in our matching setting the agents' valuations can assume only two values, v_i and 0, and we consider unit demand buyers instead of additive valuation agents as it is customary in dynamic mechanism design.

2 Preliminaries

We are given a bipartite graph $G = (B, I; E)$, where B is a set of n vertices, corresponding to buyers, I is a set of m vertices, corresponding to items, and $E \subseteq B \times I$ is the set of edges. We denote by ν the smallest between the number of buyers n and the items m. The set of buyers is known beforehand, while the items arrive one by one in some unknown, possibly adversarial, order. Without loss of generality we assume that item j arrives at time j. Each buyer i has two pieces of private information: the set of items she is interested in, and her value v_i if she gets at least one of them (the value for other items is 0). Upon the arrival of a new item, every buyer declares if she is interested in the current item and, if yes, her value. Let $b_{i,j}$ denote the bid of buyer i for item j (with the convention that $b_{i,j} = 0$ if buyer i is not interested in item j). Without loss of generality, we may assume that buyers cannot change their declared valuation after they have declared it once[1], i.e. every nonzero bid of the same buyer is the same value b_i, and that every buyer is assigned at most one item.

A mechanism $\mathcal{M}$ is composed of an *allocation* scheme and a *payment* scheme. Upon the arrival of every item, and based on buyer bids, the mechanism decides immediately and irrevocably to either assign the new item to some buyer who has not been assigned an item yet, or leave it unassigned forever. Thus, the resulting allocation is a matching in G: every buyer receives at most one item, and every item is allocated to at most one buyer. We denote by μ the induced matching, so that μ_j denotes the buyer to whom item j is assigned (we assume that an item j can only be assigned to a buyer who declares interest in j). If j is unassigned, we write $\mu_j = \emptyset$. We also write μ_i^{-1} to denote the item assigned to buyer i, with the convention that $\mu_i^{-1} = \emptyset$ if i is left unassigned. The *allocation* is computed online; i.e., μ_j is determined using only the bids on items up to j. In addition to the allocation, the mechanism decides how much each buyer should pay. A payment scheme is denoted by p, where p_i denotes the non-negative payment of buyer i. We distinguish between two types of *payment* schemes, according to the time at which the mechanism determines the payment. A *tardy* mechanism is one where the payment vector p is computed in the end of the process. A *prompt* mechanism is one where the payment p_i of every buyer i is determined upon the assignment of buyer i (i.e., upon the arrival of item μ_i^{-1}). The mechanism's objective is to maximize the *social welfare* of the allocation μ, which is the sum of the buyer values for their assigned items. The social welfare is given by $\mathrm{SW}(\mu) = \sum_{i \in B} v_i \cdot \mathbb{1}_{\{(i,\mu_i^{-1}) \in E\}}$. Note that a mechanism can also be randomized, so that its allocation is a distribution over matchings. In case of a randomized mechanism, we measure its efficiency by the expected social welfare. We say that a mechanism gives an α approximation, or is α-competitive (where $\alpha \geq 1$), if its (expected) social welfare is at least an $1/\alpha$ fraction of the welfare of a maximum weight matching. That is, μ is α-competitive if $\mathrm{OPT} = \mathrm{SW}(\mu^\star) \leq \alpha \cdot \mathbb{E}\left[\mathrm{SW}(\mu)\right]$, where $\mu^\star$ is the maximum weight matching in G.

A *bidding strategy* $\mathcal{B}_i$ for buyer i is a sequence of bids $b_{i,j}$ that specifies, every time a new item j arrives, whether to declare interest in it and which value to report. The bid $\mathcal{B}_i$ might depend on the bids of the other agents, the actions of the mechanism, and the knowledge the buyers have on the sequence of items. Recall that once an agent declares a positive valuation

[1] Mechanisms can "punish" such behavior by discarding the buyer from further consideration

$b_{i,j} = b_i > 0$ for some item j, she cannot change her value thereafter; namely, all bids for future items j' can take the value of either b_i or 0. Let $\mathcal{B}$ denote the profile of buyer bidding strategies, and $\mathcal{B}_{-i}$ denote the profile of all buyer strategies excluding buyer i. We assume that every buyer has a quasi-linear utility function: $u_i(\mathcal{M}, \mathcal{B}_i, \mathcal{B}_{-i}) = v_i \cdot \mathbb{1}_{\{(i,\mu_i^{-1})\in E\}} - p_i$.

A buyer is called *myopic* if upon the arrival of every item j, she cares only about maximizing her utility in that round, without considering its effect on future rounds. I.e., upon the arrival of item j, she maximizes the utility function $u_{i,j} = v_i \cdot \mathbb{1}_{\{\mu_j=i,\,(i,j)\in E\}} - p_i$. We consider myopic agents only in the context of prompt mechanisms, where the price p_i is determined immediately. We study the following ex-post notion of truthfulness: (i) A mechanism for myopic agents is *truthful* if it is always in the best interest of a myopic buyer to declare her value truthfully. (ii) A mechanism for non-myopic agents is *truthful* if an agent maximizes her utility for every realization of the mechanism by declaring her value truthfully. Finally, we only consider mechanisms that are *ex-post individually rational*, meaning that all agents (myopic or not) have non-negative utility, for every realization of the mechanism.

3 Prompt mechanisms with public graph edges

We start with the setting where agents are assumed to know, and strategize about, the whole sequence of items arriving. Note that this is a strong information asymmetry between agents and mechanism, as the latter only discovers the items as they are revealed online and has no information on the future. As a first step in this challenging model, in this section we study the case where agents may only lie on their valuations. Our main focus here is on establishing lower bounds, which will naturally extend to the case where the edges of the graph are private information.

3.1 Deterministic truthful mechanisms

When mechanisms are required to be prompt, the problem becomes much harder despite the fact that each agent's private information is just a single value. This is due to the online nature of the problem versus the possibly universal knowledge of the buyers, as outlined before. We first concentrate on deterministic prompt truthful mechanisms, and prove that the scope of these is quite limited. The critical item property is also used in [4] to prove a lower bound analogous to Theorem 3.

▶ **Definition 1** (critical item property). *We say that a deterministic mechanism satisfies the* critical item property *if and only if for every buyer i, there exists some $j \in I$ such that for any reported value b_i of i, the mechanism assigns i with item j, or none at all. Note that j may depend on the edges of the graph, and on the values of other buyers.*

▶ **Lemma 2.** *Prompt deterministic truthful mechanisms for the problem with public graph edges satisfy the critical item property.*

Proof. For the sake of contradiction, assume that there is a buyer i who gets item j_1 at price p_1 if she reports a value β_1 and gets item j_2 at price p_2 if she reports a value β_2. Without loss of generality, let $j_1 < j_2$. By truthfulness, the mechanism must give item j_1 to buyer i if she reports a value $\geq p_1$ (as far as the mechanism knows, i might not like items after j_1, and she would have incentive to lie and report β_1 if she is not given j_1). Thus, we have $p_2 \leq \beta_2 < p_1$, where the first inequality comes from individual rationality. But now, buyer i has incentive to report β_2, in order to get j_2 and pay p_2 which is less than p_1. ◀

▶ **Theorem 3.** *Any prompt deterministic truthful mechanism for the problem with public graph edges has competitive ratio of at most* $\nu = \min(m, n)$.

Proof. Consider an instance with n buyers with value 1 that are all interested in the first item. If there is a buyer i who will never get item 1 no matter what she reports, then we change the instance so that i has an arbitrary large value and is only interested in item 1, in which case i will get nothing and the mechanism does not even approximate the optimal social welfare. Conversely, if there is no such buyer, then the critical item property states that no other item can be allocated, which gives an approximation ratio of $\min(m, n)$. ◀

3.2 Randomized truthful mechanisms

Somewhat surprisingly, the previous result has revealed a large gap between tardy and prompt deterministic mechanisms, when the topology of the graph is public knowledge: while tardy mechanisms can be implemented *for free*, i.e., maintaining the efficiency guarantees of (non-strategic) combinatorial algorithms, for prompt mechanisms the story is different. After showing that deterministic mechanisms cannot achieve anything better than ν, we turn our focus towards impossibility results for randomized mechanisms. We utilize a well-known property of randomized truthful mechanisms, which (by definition) make truthful reports utility-maximizing for any outcome of a mechanism's random decisions, even in hindsight: this implies that they are lotteries over deterministic truthful mechanisms, which satisfy the properties shown in the previous section. By Yao's minimax principle [29], it is then enough to construct a distribution over instances, such that the optimal solutions have welfare $\Omega(\log n)$, and a best-possible deterministic mechanism $\mathcal{M}$, since it satisfies the critical item property, outputs solutions with expected value $\mathcal{O}(\log \log n)$.

▶ **Theorem 4.** *Any prompt randomized truthful mechanism for the problem with public graph edges has competitive ratio of at least* $\Omega(\log \nu / \log \log \nu)$.

Proof. Fix any prompt randomized ex-post truthful mechanism for public graph edges. We argue by Yao's principle [29] that its competitive ratio is at least $\Omega(\log \nu / \log \log \nu)$. This holds due to the upcoming Lemma 5, which shows that there exists a distribution over instances, such that the optimal solutions have welfare at least $n \log(n)/2$ with high probability, and such that any deterministic mechanism (since it satisfies the critical item property) outputs solutions with expected value $\mathcal{O}(n \log \log n)$. More precisely, given a random instance r and a mechanism $\mathcal{M}_s$ with random coin flips s, recall that Yao's principle states that:

$$\min_r \left[\frac{\mathbb{E}_s[\mathcal{M}_s(r)]}{\mathrm{OPT}(r)}\right] \leq \mathbb{E}_r \left[\frac{\mathbb{E}_s[\mathcal{M}_s(r)]}{\mathrm{OPT}(r)}\right] = \mathbb{E}_s \left[\mathbb{E}_r \left[\frac{\mathcal{M}_s(r)}{\mathrm{OPT}(r)}\right]\right] \leq \max_s \left[\mathbb{E}_r \left[\frac{\mathcal{M}_s(r)}{\mathrm{OPT}(r)}\right]\right]$$

In particular, fixing the coin flips s, the mechanism $\mathcal{M}_s$ is deterministic and truthful. Hence, Lemma 5 bounds its expected approximation ratio over the random instance r, with

$$\mathbb{E}_r \left[\frac{\mathcal{M}_s(r)}{\mathrm{OPT}(r)}\right] \leq \mathbb{E}_r \left[\frac{\mathcal{M}_s(r)}{\log(n)/2} + \mathbb{1}_{\{\mathrm{OPT}(r) \leq \log(n)/2\}}\right] \leq \frac{\mathcal{O}(\log \log n)}{\log(n)/2} + \mathcal{O}(1/\log^2 n),$$

where the first inequality holds by the disjunction of whether or not $\mathrm{OPT}(r) \leq \log(n)/2$ for a given r. Combining the two inequalities concludes the proof. ◀

▶ **Lemma 5.** *There is a distribution over instances with n buyers and n items, for which any deterministic mechanism satisfying the critical item property outputs solutions with expected value $\mathcal{O}(n \log \log n)$, and such that the optimal solution has value $\geq n \log(n)/2$ with probability at least $1 - O(1/\log^2 n)$.*

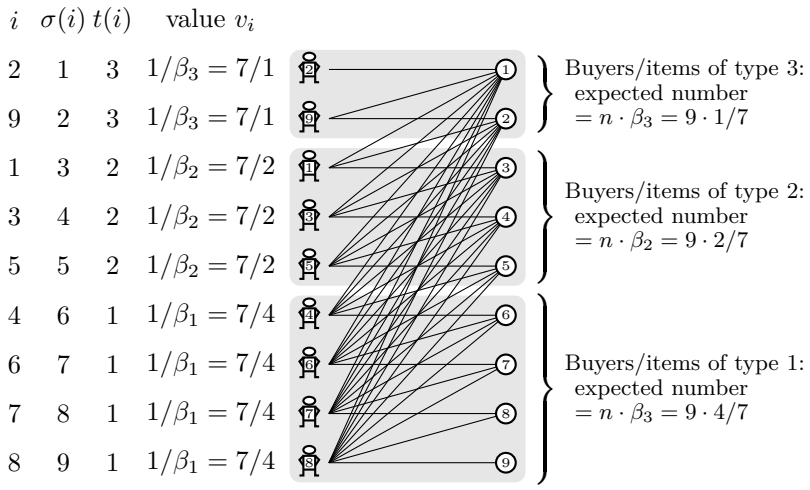


Figure 1 The instance from Lemma 5 with $k = 3$ and $n = 9$. Items are ordered (from top to bottom) according to their arrival times, and buyers are ordered (from top to bottom) according to σ (sort by decreasing types, breaking ties with indices). Preferences of buyers are given by the edges of the graph.

Proof. Let $k \geq 1$ be a parameter, which corresponds to the number of types of buyers, and let $\beta_1 > \cdots > \beta_k > 0$ be the probabilities of each type ($\beta_1 + \cdots + \beta_k = 1$). We choose $\beta_t = 2^{-t}/(1 - 2^{-k})$ for all t, and we set $n = 1 + 2^k$. Consider the following distribution over instances, with n buyers and n items. Each buyer i draws independently a type $t(i) \in \{1, \ldots, k\}$ with probability $\beta_{t(i)}$, and we set her value to $v_i = 1/\beta_{t(i)}$. Then, we sort buyers by decreasing $t(i)$, breaking ties using indices, and call $\sigma(i) \in \{1, \ldots, n\}$ the rank of buyer i in this ordering. We decide that buyer i is interested in all items up to the $\sigma(i)$-th one. To visualize this procedure, we refer to Figure 1. It is easy to find the optimal allocation: it consists in assigning each buyer of rank $\sigma(i)$ the $\sigma(i)$-th item, in a perfect matching. Thus the expected optimal social welfare is equal to

$$\mathbb{E}\left[\mathrm{OPT}\right] = \sum_{i=1}^{n} \sum_{t=1}^{k} \beta_t \cdot 1/\beta_t = n \cdot k.$$

Moreover, because each type is drawn independently the variance of OPTis

$$\mathrm{Var}(\mathrm{OPT}) = \sum_{i=1}^{n} \mathrm{Var}(v_i) \leq \sum_{i=1}^{n} \mathbb{E}[v_i^2] = n \cdot \sum_{t=1}^{k} \frac{1}{\beta_t} \leq 2n^2.$$

In particular, if we apply Chebyshev's inequality, we obtain

$$\mathbb{P}\left[\mathrm{OPT} \leq \frac{nk}{2}\right] \leq \mathbb{P}\left[|\mathrm{OPT} - nk| \geq \frac{nk}{2}\right] \leq \frac{\mathrm{Var}(\mathrm{OPT})}{(nk/2)^2} \leq \frac{8}{k^2}.$$

We now define the type $s(j) = t(\sigma^{-1}(j))$ of an item j as the type of the j-th buyer in the ordering σ, which corresponds to the type of its buyer in the abovementioned optimal matching. Observe that of each type, there are as many items as buyers, and that buyer i cannot be allocated an item j of type $s(j) < t(i)$. For each buyer i and for all types $t \leq s$, let $x^i_{s,t}$ be the probability (over the randomness of the types of all buyers except i) that i gets an item of type s, conditioning on the fact that i has type t. Let $x_{s,t} = \sum_i x^i_{s,t}/n$, that is, the average probability that a type t buyer will be assigned a type s item. The expected social welfare of our deterministic mechanism is equal to

$$\mathbb{E}\left[\mathrm{SW}(\mu)\right] = \sum_{i=1}^{n}\sum_{t=1}^{k} \beta_t \cdot 1/\beta_t \cdot \sum_{s=t}^{k} x_{s,t}^i = n\sum_{t=1}^{k}\sum_{s=t}^{k} x_{s,t}.$$

In expectation, the mechanism sells $\sum_i \sum_{t=1}^{s} \beta_t \cdot x_{s,t}^i$ items of type s. Because there are equally many items and buyers of each type, the expected number of items of type s is $\beta_s \cdot n$. Thus, we have the linear constraint

$$\forall 1 \leq s \leq k, \qquad \sum_{t=1}^{s} \beta_t \cdot x_{s,t} \leq \beta_s.$$

We are now going to use the critical item property. Fix a buyer i, and condition on the types of all buyers except her. We show that there exists an item $j(i) \in \{1, \ldots, n\}$, such that for every type $t(i)$, either i gets item $j(i)$, or she gets nothing. Denote as I_t the instance given by the fixed types of all buyers except i, together with buyer i who has type t. Using the critical item property with instance I_1, where i instead is of type 1 (meaning that i is interested in maximally many items), there is an item $j(i)$ such that buyer i either gets $j(i)$ or nothing. From the perspective of the mechanism, any other instance I_t (defined analogously) is identical to instance I_1 up to the point when i stops being interested in items. At this point, if buyer i has already been allocated an item, then it must be $j(i)$. Otherwise, she will not get anything.

Now that $j(i)$ is well-defined (and only depends on types of other buyers), let y_s^i be the probability (over the randomness of the types of all buyers except i) that there exists some type t such that if t is the type of i, then item $j(i)$ has type s. Let $y_s = \sum_i y_s^i/n$. Because buyer i can only get item $j(i)$, and because $j(i)$ is independent from $t(i)$, we have $x_{s,t}^i \leq y_s^i$. Thus, summing over all buyers, we have the linear constraint $x_{s,t} \leq y_s$, for all $1 \leq t \leq s \leq k$. Finally, conditioning on the types of all buyers except i, we show that there is only a small number of types that $j(i)$ can take. Recall that $s(j(i)) = t(\sigma^{-1}(j(i)))$, that is, the type of item $j(i)$ is by definition the type of the $j(i)$-th buyer in the ordering σ, where σ was obtained by sorting buyers in decreasing order of type. Consider the ordering induced by σ after excluding buyer i, and denote i_1 and i_2 the buyers of rank $j(i) - 1$ and $j(i)$. In the original ordering σ, either i comes before i_1 (in which case $s(j(i)) = t(i_1)$), or i comes after i_2 (in which case $s(j(i)) = t(i_2)$), or i comes between i_1 and i_2 (in which case $s(j(i)) = t(i)$). In any case, $t(i_1) \geq s(j(i)) \geq t(i_2)$. This shows that there are at most $2+z$ possible values for $s(j(i))$, where z denotes the number of types not seen among other buyers. By a standard computation, the expected value of z is smaller than $\sum_{t=1}^{k}(1-\beta_t)^{n-1}$. Recall that y_s denotes the average probability over i that there exists a type for i which can make $j(i)$ have type s, where the randomness is over the instance without i. Since for every *fixed* such instance, $j(i)$ can only possibly take two of the types seen in buyers except i, for any fixed i, it holds that $\sum_{s=1}^{k} y_s^i \leq \alpha$, where $\alpha = 2 + \sum_{t=1}^{k}(1-\beta_t)^{n-1}$, and therefore, the same holds also on average, i.e. for the y_s. Thus, averaging over possible types for the other buyers, and summing over i, we have the linear constraint $\sum_{s=1}^{k} y_s \leq \alpha$. If we choose $n = 1 + 2^k$ and $\beta_t = 2^{-t}/(1-2^{-k})$, we have

$$\sum_{t=1}^{k}(1-\beta_t)^{n-1} \leq \sum_{t=1}^{k} e^{-2^{k-t}/(1-2^{-k})} \leq \sum_{t=0}^{+\infty} e^{-2^t} \leq 1,$$

and thus $\alpha \leq 3$. To conclude the proof, we use the linear constraints obtained to define a linear program (P) whose objective function is the expected value of the social welfare

$$\begin{aligned}
\max \sum_{t=1}^{k}\sum_{s=t}^{k} x_{s,t} \qquad & \text{(P)} \\
\text{s.t. } & x_{s,t} \le y_s \\
& \textstyle\sum_{t=1}^{s} \beta_t \cdot x_{s,t} \le \beta_s \\
& \textstyle\sum_{s=1}^{k} y_s \le \alpha \\
& x_{s,t}, y_s \ge 0
\end{aligned}
\qquad\qquad
\begin{aligned}
\min\ \alpha \cdot w + \sum_{s=1}^{k} \beta_s \cdot v_s \qquad & \text{(D)} \\
\text{s.t. } & u_{s,t} + \beta_t \cdot v_s \ge 1 \\
& w \ge \textstyle\sum_{t=1}^{s} u_{s,t} \\
& u_{s,t}, v_s, w \ge 0
\end{aligned}$$

obtained by a deterministic truthful mechanism. We want to show that the objective function of our linear program is at most $\mathcal{O}(n \log k)$. To this end, Lemma 6 builds a solution for the dual linear program (D), whose value is an upper bound on the value of the primal linear program (for convenience, the objective function is divided by n). ◀

▶ **Lemma 6.** *Consider the linear program (P), parameterized by $\alpha > 0$ and $\beta_1 > \cdots > \beta_k > 0$. If $\beta_t = 2^{-t}/(1-2^{-k})$ for all $1 \le t \le k$, then the dual (D) has a feasible solution of value $\mathcal{O}(\alpha \log k)$.*

Proof. Set $\delta = \lceil \log_2 k \rceil$, then following solution of the dual is feasible and yields the desired objective value: $w = \delta$, $v_s = 0$ if $s < \delta$ and $2^{s-\delta}$ otherwise, while the $u_{s,t}$ are defined as:

$$\forall 1 \le t \le s \le k, \quad u_{s,t} = \begin{cases} 1 & \text{if } s < \delta \\ 1 - 2^{s-\delta-t} & \text{if } 0 \le s - \delta \le t \\ 0 & \text{otherwise} \end{cases}$$ ◀

4 Mechanisms with private graph edges

We move to the (harder) case where the graph edges are private information of the agents. The additional hardness, interestingly, severely affects the competitive guarantees for tardy truthful mechanisms. We begin by characterizing deterministic mechanisms, and then move on to results for randomized mechanisms.

4.1 Deterministic truthful mechanisms

In the previous section we assumed that the agents could not misreport their interest in items, thus reducing the problem to a single-parameter one. We now lift this assumption, and investigate the effect on the competitive ratio of deterministic truthful mechanisms. We show that deterministic truthful mechanisms can always be implemented in a prompt manner. Then, we give matching upper and lower bounds on the best approximation ratio for the social welfare.

▶ **Lemma 7.** *Tardy deterministic truthful mechanisms for the problem with private graph edges satisfy the critical item property (see Definition 1).*

Proof. For the sake of contradiction, assume that there is a buyer i who gets item j_1 at price p_1 if she reports a value β_1, and gets item j_2 at price p_2 if she reports a value β_2. Without loss of generality, we assume that $j_1 < j_2$. First, we argue that $p_1 = p_2$. Indeed, if $p_1 > p_2$ then i with value β_1 has incentive to lie and report β_2; whereas if $p_1 < p_2$ then i with value β_2 has incentive to lie and report β_1. Second, we slightly change the instance,

such that buyer i has value β_2 and is not interested in items after j_1. When allocating j_1, the mechanism has not seen any difference to the original instance, hence i has incentive to lie and report β_1 to get j_1, then lie and pretend she was interested in subsequent items to make sure she is charged p_1. ◀

▶ **Lemma 8.** *Tardy deterministic truthful mechanisms for the problem with private graph edges are prompt.*

Proof. Assume that our mechanism assigns an item j to buyer i, who reports value b_i. By Lemma 7, the mechanism satisfies the critical item property, and j is the only item which can be assigned to i. Let π be the minimum value that i could have reported and still be assigned j. By truthfulness, i must be charged exactly π. Indeed, if she is charged $p > \pi$ then i with value b_i has incentive to lie and report π; whereas if she is charged $p < \pi$ then i with value p would have incentives to lie and report b_i. Now, when the mechanism assigns j to i, it can retrospectively compute π, which proves that the mechanism is prompt. ◀

▶ **Theorem 9.** *There exists a deterministic truthful mechanism that achieves an $\nu = \min(m, n)$ approximation of the offline optimum. This result is tight in the class of deterministic truthful mechanisms, when graph edges are private.*

Proof. Consider the simple mechanism which only assigns an item to a buyer if she has the highest value seen so far (breaking ties arbitrarily), charging her the second highest value seen so far. This is a ν-competitive deterministic truthful mechanism. For the tightness, Lemma 8 shows that deterministic tardy mechanisms are in fact prompt, thus the lower bound from Theorem 4 (public graph edges) applies to this setting. ◀

5 Randomized truthful mechanisms

Recall that randomized (ex-post) truthful mechanisms are lotteries over deterministic truthful mechanisms, which in turn satisfy the characterizing properties we obtain for the deterministic case. The proof of our lower bound in Theorem 4 was based on this fact. This same argument also applies to mechanisms for private edges, even when they are tardy. On the positive side, we construct a prompt randomized truthful mechanism, the Explore-Exploit Mechanism, that yields a logarithmic approximation. The Explore-Exploit Mechanism divides the buyers into two types: "explore" buyers will not receive any item but are used to set the price for the "exploit" buyers. To guarantee truthfulness, we enforce monotonicity of the prices proposed by the seller during the routine: with prices always increasing, there is no way a buyer can benefit from withholding information in previous stages of the process to get something at a cheaper price later.

▶ **Theorem 10.** *The Explore-Exploit Mechanism is truthful, and computes a $O(\log n)$ approximation to the optimal social welfare. This result is nearly tight (up to $\log \log n$) in the class of randomized truthful mechanisms when the edges are private information, even for tardy mechanisms.*

Proof. Buyers of type *Explore* will not get any item, and thus have no incentive to lie. Buyers of type *Exploit* only need to say if they are interested to buy an item at a given price. Because prices are non-decreasing, they have no incentive to misreport their value or their interest in an item. For each item j, we define x_j as the maximum value seen among buyers interested in items up to j.

$$\forall j \in I, \qquad x_j = \max\{v_i \text{ with } i \in B \text{ such that } \exists j' \leq j, (i, j') \in E\}$$

Algorithm 1 Explore-Exploit Mechanism.

Initialization:
  Set $p \leftarrow 0$ and draw $k \leftarrow \text{Unif}(\{0, 1, \ldots, \lceil \log_2 n \rceil\})$
  For each buyer i, draw type $t_i \leftarrow \text{Unif}(\{Explore, Exploit\})$.
When an item arrives:
  Buyers report if they are interested in the item.
  For each buyer i of type $t_i = Explore$ who is interested in the item**, do**
    Set $p \leftarrow \max(p, v_i/2^k)$
  Sell the item at price p to a buyer i of type $t_i = Exploit$, who is interested
  in the item and does not yet have an item, chosen arbitrarily (e.g. lowest index).

For the sake of analysis, we look at a maximum weight matching $\mu \subseteq E$, having a total value of OPT. Each edge $(i, j) \in \mu$ from the optimal solution is assigned to a bucket $\ell_{(i,j)} = \lceil \log_2(x_j/v_i) \rceil \in \mathbb{N}$. Then for each $\ell \in \mathbb{N}$ we define OPT_ℓ as the total weight of the restriction of the optimal solution to bucket ℓ.

$$\text{OPT} = \sum_{\ell \geq 0} \text{OPT}_\ell \qquad \text{where } \forall \ell \geq 0, \quad \text{OPT}_\ell = \sum_{(i,j) \in \mu} v_i \cdot \mathbb{1}_{\{\ell_{(i,j)} = \ell\}}$$

Let V be maximum value among buyers who are interested in at least one item. By optimality of μ, the corresponding buyer must be given an item, and thus $\text{OPT}_0 \geq V$. Now observe that for each $(i, j) \in \mu$ such that $\ell_{(i,j)} > \lceil \log_2 n \rceil$, we have $v_i < x_j/n \leq V/n \leq \text{OPT}_0/n$. Thus, the sum of OPT_ℓ for $\ell > \lceil \log_2 n \rceil$ is smaller than OPT_0. Therefore, buckets $0, 1, \ldots, \lceil \log_2 n \rceil$ contain at least half of OPT, that is

$$\frac{\text{OPT}}{2} \leq \sum_{\ell=0}^{\lceil \log_2 n \rceil} \text{OPT}_\ell$$

For all $\ell \in \{0, 1 \ldots, \lceil \log_2 n \rceil\}$, we will now show that if $k = \ell$ then the Explore-Exploit Mechanism gives a solution of expected cost at least $\Omega(\text{OPT}_\ell)$. Then we will conclude the proof using the law of total probability: summing over k shows that the Explore-Exploit Mechanism computes a solution of expected cost at least $\Omega(\text{OPT}/\log n)$. First, assume that $k = 0$. For each edge $(i, j) \in \mu$ in bucket $\ell_{(i,j)} = 0$, then i is the best buyer seen by the time j arrives. With probability $1/4$, buyer i has type *Exploit* and the second best buyer has type *Explore*. In that case, the Explore-Exploit Mechanism gives buyer i an item (either j or one of the previous items). Using linearity of expectation, the Explore-Exploit Mechanism outputs a solution of expected value at least $\text{OPT}_0/4$. Second, assume that $k = \ell$ with $\ell \in \{1, \ldots, \lceil \log_2 n \rceil\}$. This case requires an amortized analysis: for each buyer i, denote X_i the random variable equal to v_i if i gets an item and 0 otherwise; and for each item j, denote Y_j the random variable equal to the value of the buyer to whom j is assigned, and 0 if j is unassigned. Notice that the Explore-Exploit Mechanism outputs a solution of value $= \sum_{i \in B} X_i = \sum_{j \in I} Y_j$. Let $(i, j) \in \mu$ be an edge from bucket $\ell_{(i,j)} = \ell$. We are going to show that

$$\mathbb{E}\left[X_i + 4Y_j \mid k = \ell \text{ and } t_i = Exploit\right] \geq v_i.$$

We condition on the fact that $k = \ell$ and $t_i = Exploit$. If buyer i already has an item when item j arrives, then $X_i = v_i$. Otherwise, the best buyer seen so far has type *Explore* with probability $1/2$, in which case the Explore-Exploit Mechanism gives item j to a

buyer of value $\geq x_j/2^\ell \geq v_i/2$. Buyer i has type $t_i = \textit{Exploit}$ with probability 1/2, thus $v_i \leq E[2X_i + 8Y_j \mid k = \ell]$. Summing this last inequality over edges from bucket ℓ shows that the Explore-Exploit Mechanism outputs a solution of expected value at least $\mathrm{OPT}_\ell/10$.

Let's move our attention to the lower bound . Fix all random decisions of an ex-post truthful randomized mechanism. This yields a deterministic algorithm, that together with the original mechanism's payment scheme yields a (tardy) mechanism. This mechanism is deterministic, and truthful due to the definition of truthfulness. Also, such a mechanism fulfills the critical item property (Lemma 7), and can even be made prompt (Lemma 8). With this, we can follow the original proof of the lower bound. ◀

6 Ex-ante truthfulness

One might wonder if the hardness of truthful mechanisms for our problem is mainly due to the very restrictive notion of ex-post truthfulness. We state here that also for the much looser ex-ante truthfulness, the setting of non-myopic buyers separates clearly from the myopic case. The proof is via a nontrivial construction allowing bounds on agents' expected utilities.

▶ **Theorem 11.** *There exists no randomized ex-ante truthful mechanism that yields an α-approximation to the optimal social welfare, for the problem with private edges and any $\alpha < 2$. This is true even for tardy mechanisms.*

Proof. Fix $\alpha < 2$ and assume mechanism M guarantees an expected approximation ratio of α. Consider the following problem instance: there are n' buyers and $m = n' + 1$ items. Every item j has exactly one interested buyer, i_j, and all i_j have some small value $v_{i_j} = \epsilon > 0$. There exist some additional buyers $B_1 \subseteq B$ with different values who are interested only in item 1, and one buyer, i, whom we fix for our considerations. Note that $|B| = n' + n_1$, with $n_1 = |B_1|$. For n' large enough, clearly, $n'\epsilon > \max_{i' \in B_1} v_{i'}$ and the contribution of item 1 to the optimum becomes negligible with growing n'. Therefore, for M to guarantee an α-approximation, there must exist $j \in \{2, \ldots, n' + 1\}$ such that i_j is assigned the according item with probability at least $\frac{1}{\alpha}$, or in case item 1 is worth more than ϵ, at least probability $\frac{1}{\alpha} - \Delta_1$, where Δ_1 arbitrarily small for large n'.

Now, if we choose $i = i_j$, then M will assign item j to i_j w.pr. $\geq \frac{1}{\alpha} - \Delta_1$, and charge an expected price of at most ϵ. The latter is because the price cannot depend on i's bid due to incentive compatibility, and it needs to be below i's value. Assume we replace i's valuation by some $v > \epsilon$, and call this new buyer $i^{(1)}$. Since M is ex-ante truthful, still, the exp. utility $u_{i^{(1)}}$ achieved with a truthful report must be at least as large as when reporting ϵ instead of v, i.e. at least $(v - \epsilon)(\frac{1}{\alpha} - \Delta_1) > \frac{1}{2}v$, which is at least half of v because α is < 2 and ϵ, Δ_1 can be chosen arbitrarily small. We replace $i^{(1)}$ again by a different buyer $i = i^{(2)}$. She still has valuation v, however, she is now interested in items 1 *and* j. We consider the first step of M, i.e. the assignment decision made for item 1. Assuming that v is the largest value bid on item 1, and given the fact that M has no idea if any additional value will present itself in the later steps, the probability that M assigns item 1 to $i^{(2)}$ is at least $\frac{1}{\alpha} - \Delta_2$, where Δ_2 approaches 0 since the other bids on item 1 might be, in comparison, too small to matter. Note again that the assignment decision cannot depend on v itself, but only on the fact that it is the largest value bid on item 1.

We know that $i^{(2)}$ can get utility larger than $\frac{v}{2}$ by simply reporting type $i^{(1)}$ instead. We also know that since she is assigned item 1 w.pr. $> \frac{1}{2}$, she is assigned item j w.pr. $< \frac{1}{2}$. This, intuitively, means that not all of the guaranteed utility is generated by item j, not even if the price of j is always 0 – but some must be generated because her expected price paid when

item 1 is assigned is bounded away from v, i.e. $p_{i^{(2)}}(1) = v - \Delta_3$. In fact, the exp. price M charges from $i^{(2)}$ when assigning item 1 cannot be smaller if $i^{(2)}$ later reports interest in item j, since this would give a buyer of type $i^{(1)}$ incentive to also report interest in j. Also, the price charged from $i^{(2)}$ when assigning item j cannot be less than 0, and when there is no item assigned, $i^{(2)}$ is not charged anything (see preliminaries). This implies that, for $P_k(i)$ denoting the assignment probability of item k to buyer i,

$$\begin{aligned} u_{i^{(2)}} &= (v - p_{i^{(2)}}(1)) \cdot P_1(i^{(2)}) + (v - p_{i^{(2)}}(j)) \cdot P_j(i^{(2)}) \\ &= \Delta_3 \cdot P_1(i^{(2)}) + (v - p_{i^{(2)}}(j)) \cdot P_j(i^{(2)}) > \frac{v}{2} \end{aligned}$$

Otherwise, we would have a contradiction on the utility being larger than $\frac{v}{2}$, i.e. it would be beneficial for $i^{(2)}$ to only report interest in item j. In consequence, it also holds

$$u_{i^{(2)}} = \Delta_3 \cdot P_1(i^{(2)}) + (v - p_{i^{(2)}}(j)) \cdot P_j(i^{(2)}) \geq \Delta_3 \cdot P_1(i^{(2)}) + (v - v) \cdot P_j(i^{(2)}) > 0.$$

This is true because the exp. price when receiving item j can be no more than v, and $P_j(i^{(2)}) < \frac{1}{2}$. Therefore, there exists some $v^- < v$ for which it holds that

$$u_{i^-}(1) = u_{i^{(2)}}(1) - P_1(i^{(2)})(v - v^-) = (\Delta_3 - (v - v^-))P_1(i^{(2)})$$

Here, $u_{i^-}(1)$ denotes the utility obtained from being assigned item 1 of some buyer with valuation v^- for item 1, and 0 otherwise, when she reports $i^{(2)}$ as her type. Note that if buyer i^- reports value v for item 1 and 0 for all others, she will also obtain $u_{i^-}(1)$ from being assigned the first item: the assignment decision is made before the algorithm can know the difference, and the expected price paid cannot depend on the buyer's later reports due to truthfulness.

We use this to show a contradiction to the approximation ratio of M. Assume there exists, in absence of $i^{(2)}$, such a buyer i^- with smaller value v^- and utility of $u^-(1) > 0$ when reporting to have value v, who is interested in purchasing item 1, i.e. $i^- \in B_1$. Since M is ex-ante truthful, a truthful report for her will also result in positive expected utility of at least $u^-(1)$. As a direct consequence, it holds also that the probability $P_1(i^-)$ for assigning item 1 to i^- (when she reports truthfully) is lower bounded, in order to achieve above expected utility, as follows: $P_1(i^-) \geq \frac{u_{i^-}(1)}{v^-}$. Finally, we copy buyer i^- at least $\frac{v^-}{u_{i^-}(1)} + 1$ times. If necessary for tie-breaking, we distort their values a bit. Our conclusions about $i^{(2)}$'s utility hold once $i^{(2)}$ reports the largest value for item 1, regardless of other values. This means, if either of our copied v^- should decide to deviate and report to be valued like $i^{(2)}$ instead, they can recover utility $u_{i^-}(1)$. As a result, each one of the copies, when reporting truthfully, has at least the same utility, and therefore an assignment probability of at least $P_1(i^-)$. This, in sum, results in a probability of more than 1 for assigning item 1, i.e., a contradiction. ◀

7 Conclusions

We have studied vertex-weighted bipartite online matching with offline agents in various settings, obtaining an almost-complete picture of the competitive ratios achievable by mechanisms under different truthfulness notions. Our results encompass that for myopic truthfulness, the best algorithmic results [18, 1] transfer to the online agents setting. This showcases that the very general myopic bounds of [9] are far from tight for restricted settings like ours. On the other hand, we also show that equally near-optimal approximations are impossible under the assumption of classic truthfulness, even ex-ante; and for ex-post

truthfulness our seemingly simple problem already exhibits lower bounds almost matching the myopic, logarithmic competitive ratio for submodular combinatorial auctions in [9]. We leave open to what extent this additional hardness (moving from a tight $e/(e-1)$ myopic to $\Omega(\log n/\log\log n)$ truthful) already happens when imposing ex-ante truthfulness. This is an interesting subject of investigation, also for different scenarios than the one of our ≥ 2 lower bound (private edges).

References

1 Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *SODA*, pages 1253–1264. SIAM, 2011.

2 Susan Athey and Ilya Segal. An efficient dynamic mechanism. *Econometrica*, 81(6):2463–2485, 2013.

3 Yossi Azar and Ety Khaitzin. Prompt mechanism for ad placement over time. In *International Symposium on Algorithmic Game Theory SAGT*, pages 19–30. Springer, 2011.

4 Moshe Babaioff, Liad Blumrosen, and Aaron Roth. Auctions with online supply. *Games Econ. Behav.*, 90:227–246, 2015.

5 Dirk Bergemann and Juuso Välimäki. Dynamic mechanism design: An introduction. *Journal of Economic Literature*, 57(2):235–74, June 2019.

6 Benjamin Birnbaum and Claire Mathieu. On-line bipartite matching made simple. *Acm Sigact News*, 39(1):80–87, 2008.

7 Constantine Caramanis, Paul Dütting, Matthew Faw, Federico Fusco, Philip Lazos, Stefano Leonardi, Orestis Papadigenopoulos, Emmanouil Pountourakis, and Rebecca Reiffenhäuser. Single-sample prophet inequalities via greedy-ordered selection. In *SODA*, pages 1298–1325. SIAM, 2022.

8 Richard Cole, Shahar Dobzinski, and Lisa Fleischer. Prompt mechanisms for online auctions. In *International Symposium on Algorithmic Game Theory*, pages 170–181. Springer, 2008.

9 Yuan Deng, Debmalya Panigrahi, and Hanrui Zhang. Online combinatorial auctions. In *SODA*, pages 1131–1149. SIAM, 2021.

10 Nikhil R. Devanur, Kamal Jain, and Robert D. Kleinberg. Randomized primal-dual analysis of RANKING for online bipartite matching. In *SODA*, pages 101–107. SIAM, 2013.

11 Paul Dütting, Federico Fusco, Philip Lazos, Stefano Leonardi, and Rebecca Reiffenhäuser. Efficient two-sided markets with limited information. In *STOC*, pages 1452–1465. ACM, 2021.

12 Alon Eden, Michal Feldman, Amos Fiat, and Kineret Segal. An economics-based analysis of RANKING for online bipartite matching. In *SOSA*, pages 107–110. SIAM, 2021.

13 Tomer Ezra, Michal Feldman, Nick Gravin, and Zhihao Gavin Tang. Online stochastic max-weight matching: Prophet inequality for vertex and edge arrival models. In *EC*, pages 769–787. ACM, 2020.

14 Uriel Feige. Tighter bounds for online bipartite matching. In *Building Bridges II*, pages 235–255. Springer, 2019.

15 Buddhima Gamlath, Sagar Kale, and Ola Svensson. Beating greedy for stochastic bipartite matching. In *SODA*, pages 2841–2854. SIAM, 2019.

16 Buddhima Gamlath, Michael Kapralov, Andreas Maggiori, Ola Svensson, and David Wajc. Online matching with general arrivals. In *FOCS*, pages 26–37. IEEE Computer Society, 2019.

17 Nick Gravin, Zhihao Gavin Tang, and Kangning Wang. Online stochastic matching with edge arrivals. In *ICALP*, volume 198 of *LIPIcs*, pages 74:1–74:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

18 Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC*, pages 352–358. ACM, 1990.

19 Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions. In *ESA*, volume 8125 of *Lecture Notes in Computer Science*, pages 589–600. Springer, 2013.

20 Nitish Korula and Martin Pál. Algorithms for secretary problems on graphs and hypergraphs. In *ICALP (2)*, volume 5556 of *Lecture Notes in Computer Science*, pages 508–520. Springer, 2009.

21 Piotr Krysta and Berthold Vöcking. Online mechanism design (randomized rounding on the fly). In *ICALP (2)*, volume 7392 of *Lecture Notes in Computer Science*, pages 636–647. Springer, 2012.

22 Aranyak Mehta. Online matching and ad allocation. *Found. Trends Theor. Comput. Sci.*, 8(4):265–368, 2013.

23 Vahab S. Mirrokni, Renato Paes Leme, Pingzhong Tang, and Song Zuo. Non-clairvoyant dynamic mechanism design. In *EC*, page 169. ACM, 2018.

24 Roger B Myerson. Optimal auction design. *Mathematics of operations research*, 6(1):58–73, 1981.

25 Rebecca Reiffenhäuser. An optimal truthful mechanism for the online weighted bipartite matching problem. In *SODA*, pages 1982–1993. SIAM, 2019.

26 Tim Roughgarden. *Twenty lectures on algorithmic game theory*. Cambridge University Press, 2016.

27 William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 16(1):8–37, 1961.

28 Xiangzhong Xiang. Prompt mechanism for online auctions with multi-unit demands. *Journal of Combinatorial Optimization*, 30(2):335–346, 2015.

29 Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *FOCS*, pages 222–227. IEEE Computer Society, 1977.

A Mechanisms for Myopic Buyers

In this section we study myopic buyers. We show that for this class of agents, one can obtain strategy proof versions of the best (non-truthful) algorithms [18]. Formally, we construct a deterministic prompt mechanism that achieves at least half of the welfare of the best offline matching, and a randomized prompt mechanism that is (in expectation) $e/(e-1)$-competitive with the best offline matching. We start describing our deterministic mechanism HonestGreedy, which mimics the classical Greedy algorithm for online weighted matching in a way that is robust to strategic bidding. Every time a new item arrives, HonestGreedy runs a second price auction [27] to allocate it between the remaining (interested) buyers. Since the buyers are myopic, every time a new item arrives, they behave like if it was the last: clearly there is no point in lying about being interested in an item. Moreover, the truthfulness in each step (as well as the individual rationality) is guaranteed by the well-known properties of the second price auction. Note that the mechanism sets the price for item i immediately, so it is prompt. The analysis of the approximation guarantee is also quite simple: the allocation output by HonestGreedy is the same one that the standard Greedy algorithm would have computed on the same input. It is well known that Greedy is 2-competitive with respect to the best offline matching (see, e.g., Appendix B of [1]), and that this approximation is tight in the class of deterministic algorithms [18]. We summarize these observations.

▶ **Theorem 12.** *The deterministic prompt mechanism HonestGreedy is truthful for myopic agents and guarantees a 2 approximation to the best offline matching. The approximation is tight even for (non-truthful) deterministic algorithms.*

We complement this deterministic 2-competitive, simple mechanism with an optimal, randomized $e/(e-1)$−competitive alternative, HonestPerturbedGreedy, based on Perturbed-Greedy of Aggarwal et al. [1]. There, each offline vertex is associated with a random multiplier; then, every time one of the online vertices arrives, it is matched to the

Algorithm 2 HonestPerturbedGreedy.

1: **For** each buyer i**, do**
2: Draw x_i uniformly at random from $[0, 1]$
3: Let $y_i = 1 - e^{x_i - 1}$
4: Reveal publicly all x_i and y_i
5: **For** item j arriving online**, do**
6: Receive bids for j and let $N(j)$ be the set of agents interested in j
7: Allocate j to $i^\star \in \arg\max\{b_i \cdot y_i \mid i \in N(j)\}$ ▷ Allocation Rule
8: Charge $i^\star$ with $p_{i^\star} = \max\left\{\frac{y_i}{y_{i^\star}} b_i \mid i \in N(j) \setminus \{i^\star\}\right\}$ ▷ Payment Rule
9: Discard for further consideration $i^\star$

free neighbor with largest multiplier-value product. To protect from the strategic behavior of agents, HonestPerturbedGreedy declares – before the beginning of the online phase – publicly all random multipliers, and then implements Myerson's payment rule [24] for every round. For a formal description we refer to the pseudocode of HonestPerturbedGreedy, where we maintain the convention that the max of an empty set is 0 and thus if $N(j)$ is empty in line 7, then j is discarded and the mechanism passes to the next item. The properties of HonestPerturbedGreedy are summarized in the following Theorem, whose formal proof is deferred to Appendix C.

▶ **Theorem 13.** *The randomized prompt mechanism HonestPerturbedGreedy is truthful for myopic agents and achieves (in expectation) a $e/(e-1)$ approximation to the best offline matching. The approximation is tight even for (non-truthful) randomized algorithms.*

B Tardy mechanisms with public graph edges

When the graph edges are public knowledge, we can turn once again to using the algorithmic approaches outlined in the previous Section, i.e. Greedy and Perturbed-Greedy. Now that agents cannot strategically withhold or misreport the existence of edges, a tardy truthful mechanism can use the whole graph structure (but of course still not the reported value b_i) when computing the price charged from any buyer i. The prompt, round-wise payment rules we designed for myopic buyers, however, do not guarantee non-myopic truthfulness. What remains to prove therefore is that these algorithms can be augmented by a different (tardy) payment rule to be made truthful. This is formally done in two steps: first, it is established that the allocations produced are monotone, and then Myerson's Lemma is employed to enforce truthfulness. All in all, we have the following.

▶ **Theorem 14.** *There exists a deterministic, respectively randomized, tardy mechanism that is truthful for non-myopic agents with public graph edges and guarantees a 2, resp. $e/(e-1)$, approximation to the best offline matching. The approximation is tight even for (non-truthful) deterministic, resp. randomized, algorithms.*

Note that the allocation computed by the mechanisms we just described are analogous to the ones computed by HonestGreedy and HonestPerturbedGreedy, but the payments are different! We are still using Myerson's Lemma, but the critical prices[2] are clearly different, as they are computed considering the whole run of the algorithm. To see this, consider the following example. There are two buyers, b_1 and b_2, and two items i_1 and i_2. b_1 is interested

[2] The critical price is the smallest bid that would have still resulted in the item being allocated to the agent. See the Appendix C for a formal definition

in both the items and has a value of 1, while b_2 only cares about i_1, with a value of 0.9. Assume also for the sake of simplicy that the perturbations y_1 and y_2 of PERTURBED-GREEDY are both 1. Both versions of PERTURBED-GREEDY would only allocate i_1 to b_1, but at two different prices: the mechanism for myopic agents would charge 0.9, while the tardy one for non-myopic agents would wait until the end of the second round and charge 0.

C Proofs of Theorems 13 and 14

In this section we prove the properties of HONESTPERTURBEDGREEDY and of the tardy versions of GREEDY and PERTURBED-GREEDY presented in Appendices A and B. Starting from the guarantees of their non-strategic counterparts it is immediate to see that the approximation factor claimed are indeed correct. The only property to show is incentive compatibility. A crucial ingredient to prove incentive compatibility is Myerson's Lemma, that we recall here for the sake of completeness. The Lemma has been proved in Myerson's seminal paper [24]; here we follow the more modern approach by Roughgarden [26]. Since in this paper we study unit-demand agents, we restrict to consider only this type of agents.

We start introducing the notion of single-parameter environments. In such environments, there are n agents and a set X of feasible allocations of items to agents. Each agents is characterized by a private valuation to get an item and strives to maximize her quasi-linear utility. To familiarize with this notion consider the model of non-myopic buyers with public graph edges studied in the paper: those agents are indeed single-parameters, as their valuations is their only private information. At the same time, note that the "edge compatibility" is implicitly modeled by the following set of feasible allocations of items to agents: an allocation $\mathbf{x} \in \{0,1\}^n$ is feasible if and only if it is corresponds with a matching in the underlying buyers-items bipartite graph. As already mentioned in the main body, a mechanism $\mathcal{M}$ is characterized by two features: an allocation $\mathbf{x} \in X$ and a payment rule $\mathbf{p}$. While the allocation specifies who gets what, the payment rule defines how much each agent pays. Allocation and payments are functions of the bids; in particular, we use the notation $x_i(b_i, b_{-i}) \in \{0,1\}$ to specify whether the i^{th} agent is allocated an item, given her bid b_i and the $n-1$ bids b_{-i} of the other agents. We are ready for the following crucial definitions.

▶ **Definition 15** (Monotone allocation). *An allocation rule* $\mathbf{x}$ *for a single-parameter environment is monotone if for every bidder* i *and bids* b_{-i} *by the other bidders, the allocation* $x_i(z, b_{-i})$ *to* i *is nondecreasing in its bid* z*.*

▶ **Definition 16** (critical prices). *Fix and agent* i *and bids* b_{-i} *of the other agents. Then the critical price for* i *is defined as the smallest bid* z_i *such that* i *is allocated an item, if any. Formally, if we use the convention that the* inf *of an empty set is* 0*, we have* $z_i = \inf\{z \mid x_i(z, b_{-i}) = 1\}$

Clearly, the critical prices enforce ex-post individual rationality. Myerson showed that they also induce (ex-post) truthfulness; we report here a version of Lemma 2 of Myerson [24] that is tailored to our problem and then show the two Theorems.

▶ **Theorem 17** (Myerson's Lemma). *Fix a single-parameter mechanism. Given any monotone allocation* $\mathbf{x}$*, it is possible to compute a payment scheme* $\mathbf{p}$ *such that the resulting mechanism is truthful and individually rational. In particular, in* $\mathbf{p}$*, each agents that receives an item pays its critical price and* 0 *otherwise.*

▶ **Theorem 13.** *The randomized prompt mechanism* HONESTPERTURBEDGREEDY *is truthful for myopic agents and achieves (in expectation) a* $e/(e-1)$ *approximation to the best offline matching. The approximation is tight even for (non-truthful) randomized algorithms.*

Proof. We start the proof by arguing that HonestPerturbedGreedy is truthful and individually rational for myopic agents. First, note that when any item j arrives, there is no point for the buyers still unallocated to lie about their interest for it: if they are not interested and they bid, they would risk to get j and lose future opportunity to get allocated to something they are interested in, while if they are interested they do not want to lose the opportunity (since they have no information on the future, and the prices charged never exceed their valuations). If we restrict to consider the buyers $N(j)$ interested in item j, we see that the problem reduces to a single-parameter auction: the agents are myopic and just want to maximize their utility by getting j at a small price. All y_i are public knowledge and non-negative, so our allocation rule (line 7 of HonestPerturbedGreedy), fixing these values, is clearly monotone (the more an agent i bids, the more likely she is to exhibit the largest $y_i \cdot b_i$). The allocation is therefore implementable using the Myerson payment rule (line 8 of HonestPerturbedGreedy). We can conclude, by Myerson's Lemma, that our mechanism is truthful for myopic buyers. Moreover, it is easy to verify that the payment rule enforces individual rationality. Once we have settled the truthfulness, we can assume that all buyers declare their true bids and thus the allocation output by HonestPerturbedGreedy is the same as Perturbed-Greedy *for any realization* of the perturbations x_i and inherits the same approximation: HonestPerturbedGreedy is $e/(e-1)$-competitive in expectation. ◀

▶ **Theorem 14.** *There exists a deterministic, respectively randomized, tardy mechanism that is truthful for non-myopic agents with public graph edges and guarantees a* 2*, resp.* $e/(e-1)$*, approximation to the best offline matching. The approximation is tight even for (non-truthful) deterministic, resp. randomized, algorithms.*

Proof. It is easy to see how the two mechanisms are monotone, thus it is possible to employ directly Myerson's Lemma, as the problem is single-parameter (i.e., the only private information of buyer i is the single value v_i). Therefore, Greedy or Perturbed-Greedy (with fixed perturbation factors) together with the critical payments defined in Myerson's Lemma result in a truthful mechanism. Note that the greedy algorithm clearly respects our ex-post notion of truthfulness, since no randomization is involved. For the Perturbed-Greedy algorithm, this is also true since we fix all random decisions (perturbation) up front, and choose the payment rule accordingly. ◀

Completely Reachable Automata: A Polynomial Algorithm and Quadratic Upper Bounds

Robert Ferens
University of Wrocław, Poland

Marek Szykuła
University of Wrocław, Poland

Abstract

A complete deterministic finite (semi)automaton (DFA) with a set of states Q is *completely reachable* if every non-empty subset of Q can be obtained as the image of the action of some word applied to Q. The concept of completely reachable automata appeared several times, in particular, in connection with synchronizing automata; the class contains the Černý automata and covers a few separately investigated subclasses. The notion was introduced by Bondar and Volkov (2016), who also raised the question about the complexity of deciding if an automaton is completely reachable. We develop a polynomial-time algorithm for this problem, which is based on a new complement-intersecting technique for finding an extending word for a subset of states. The algorithm works in $\mathcal{O}(|\Sigma| \cdot n^3)$ time, where $n = |Q|$ is the number of states and $|\Sigma|$ is the size of the input alphabet. Finally, we prove a weak Don's conjecture for this class of automata: a subset of size k is reachable with a word of length smaller than $2n(n-k)$. This implies a quadratic upper bound in n on the length of the shortest synchronizing words (reset threshold) for the class of completely reachable automata and generalizes earlier upper bounds derived for its subclasses.

2012 ACM Subject Classification Theory of computation → Algorithm design techniques; Mathematics of computing → Combinatorics; Theory of computation → Formal languages and automata theory

Keywords and phrases Černý conjecture, complete reachability, DFA, extending word, reachability, reset threshold, reset word, simple idempotent, synchronizing automaton, synchronizing word

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.59

Category Track A: Algorithms, Complexity and Games

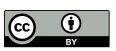
Related Version *Full Version*: `https://arxiv.org/abs/2208.05956`

Funding *Robert Ferens*: Supported in part by the National Science Centre, Poland under project number 2017/26/E/ST6/00191.
Marek Szykuła: Supported in part by the National Science Centre, Poland under project number 2021/41/B/ST6/03691.

Acknowledgements We thank the anonymous reviewer for careful proofreading and his suggestions.

1 Introduction

The concept of completely reachable automata origins from the theory of synchronizing automata. A deterministic finite automaton is *synchronizing* if starting from the set of all states, after reading a suitable *reset* word, we can narrow (reach) the set of possible states to a singleton. On the other hand, an automaton is *completely reachable* if starting from the set of all states, we can reach every non-empty subset of states. Thus, every completely reachable automaton is synchronizing, and the class of completely reachable automata forms a remarkable subclass of (synchronizing) automata.



Synchronizing automata are most famous due to a longstanding open problem: the Černý conjecture, which states that every synchronizing n-state automaton admits a reset word of length at most $(n-1)^2$. Applications of synchronizing automata include testing of reactive systems [20] and synchronization of codes [2, 4]. The currently best upper bound for the Černý problem is cubic in n [18, 21, 22]. Finding better bounds for particular cases was a topic of extensive study. Some of the related computational problems such as checking if an automaton is synchronizing are solved in polynomial time [11]. Other ones, such as finding a reset word of the smallest length, are hard, but practical (heuristic and optimizing) methods are developed [23]. Most of the research on the topic of synchronizing automata was collected and comprehensively described in the recent survey [25]; we also refer to older ones [16, 24].

Other studies where completely reachable automata appear include the descriptional complexity of formal languages. Here, the complete reachability of an automaton is often juxtaposed with the maximality of the state complexity or the syntactic complexity of languages recognized by automata [5, 15, 17].

The notion of completely reachable automata was first introduced in 2016 by Bondar and Volkov [6], who also asked about the complexity of the computational problem of deciding whether a given automaton is completely reachable. The analogous decision problem for synchronizing automata is easily solvable in quadratic time in the number of states. Later studies revealed a connection to the so-called Rystsov graph, whose generalization can be used to characterize completely reachable automata [5, 7]. However, this does not yet lead to an effective algorithm, as it is unknown how to compute these graphs. Recently, the case of binary automata was solved with a quasilinear-time algorithm [8], which strongly relies on the specificity of both letters when they ensure the complete reachability of the automaton.

The class of completely reachable automata contains several previously studied cases. It contains the Černý automata [9], which meet the conjectured bound $(n-1)^2$ for the Černý problem, and some of the so-called *slowly synchronizing series* [1]. The class of automata with the full transition monoid [13], synchronizing automata with simple idempotents [19] and aperiodically 1-contracting automata [10] are proper subclasses of completely reachable automata. For the first two subclasses and a special case of the third one quadratic upper bounds on the length of the shortest reset words instead of a cubic one were established for the Černý problem. However, for the whole class of completely reachable automata, only cubic bounds were known (though better than in the general case) [5].

In connection to the bounds, a remarkable conjecture and a generalization of the Černý one is Don's conjecture [10, Conjecture 18]. It states that for an n-state automaton, for every $1 \leq k < n$, if a subset of states of size k is reachable, then it can be reached with a word of length at most $n(n-k)$. This conjecture was disproved in general [14] but weaker versions were proposed: one restricting to completely reachable automata [14, Problem 4] and another general one, in relation to avoiding words [12, Conjecture 15].

1.1 Contribution

We design a polynomial-time algorithm for the problem of deciding whether an automaton is completely reachable, thus solving the 7-year-old open question [6]. For this solution, we develop a complement-intersecting technique, which allows finding a short extending word for a given subset of states (i.e., that gives a larger preimage). We optimize the complexity of the algorithm to work in $\mathcal{O}(|\Sigma| \cdot n^3)$ time.

Based on the discovered properties, we prove that every non-empty subset of $k < n$ states in a completely reachable n-state automaton is reachable with a word of length smaller than $2n(n-k)$. This is a weaker version (by the factor of 2) of Don's conjecture stated for completely reachable automata [14, Problem 4].

It follows that a completely reachable n-state automaton has a reset word of length at most $2n^2 - n\ln n - 4n + 2$ (for $n \geq 3$). This generalizes and improves the previous bounds obtained with different techniques for known proper subclasses of completely reachable automata: automata with a full transition monoid (with the previous upper bound $2n^2 - 6n + 5$) [13], synchronizing automata with simple idempotents (with the previous upper bound $2n^2 - 4n + 2$) [19], and 1-contracting automata (only a special case was solved) [10].

2 Solving Complete Reachability in Polynomial Time

2.1 Reachability

A *complete deterministic finite semiautomaton* (called simply *automaton*) is a 3-tuple (Q, Σ, δ), where Q is a finite set of *states*, Σ is an *input alphabet*, and $\delta\colon Q \times \Sigma \to Q$ is the *transition function*, which is extended to a function $Q \times \Sigma^* \to Q$ in the usual way. Throughout the paper, by n we always denote the number of states in Q.

Given a subset $S \subseteq Q$, the *image* of S under the action of a word $w \in \Sigma^*$ is $\delta(S, w) = \{\delta(q, w) \mid q \in S\}$. The *preimage* of S under the action of w is $\delta^{-1}(S, w) = \{q \in Q \mid \delta(q, w) \in S\}$. For a state q, we also simplify $\delta^{-1}(q, w) = \delta^{-1}(\{q\}, w)$. Note that $q \in \delta(Q, w)$ if and only if $\delta^{-1}(q, w) \neq \emptyset$. For a subset $S \subseteq Q$, by $\overline{S}$ we denote its complement $Q \setminus S$.

For two subsets $S, T \subseteq Q$, if there exists a word $w \in \Sigma^*$ such that $\delta(T, w) = S$, then we say that S is *reachable from* T *with* the word w. Then we also say that T is a *w-predecessor* of S. It is simply a *predecessor* of S if it is a w-predecessor for some word w. One set can have many w-predecessors, but there is at most one maximal with respect to inclusion (and size).

▶ Remark 1. For $S \subseteq Q$ and $w \in \Sigma^*$, the preimage $\delta^{-1}(S, w)$ is a w-predecessor of S if and only if $\delta^{-1}(q, w) \neq \emptyset$ for every state $q \in S$. Equivalently, we have $\delta(\delta^{-1}(S, w)) = S$.

▶ Remark 2. For $S \subseteq Q$ and $w \in \Sigma^*$, if $\delta^{-1}(S, w)$ is a w-predecessor of S, then all w-predecessors of S are contained in it, thus $\delta^{-1}(S, w)$ is maximal in terms of inclusion and size. If $\delta^{-1}(S, w)$ is not a w-predecessor of S, then S does not have any w-predecessors.

A word w is called *extending* for a subset S, or w *extends* S, if $|\delta^{-1}(S, w)| > |S|$. It is called *properly extending* if additionally $\delta^{-1}(S, w)$ is a w-predecessor of S, i.e., $\delta^{-1}(q, w) \neq \emptyset$ for all $q \in S$.

A subset $S \subseteq Q$ is *reachable* in the automaton if S is reachable from Q with any word. An automaton is *completely reachable* if all non-empty subsets $S \subseteq Q$ are reachable. Equivalently, Q is a predecessor of all its non-empty subsets. The latter leads to an alternative characterization of completely reachable automata:

▶ Remark 3. An automaton (Q, Σ, δ) is completely reachable if and only if for every non-empty proper subset of Q, there is a properly extending word.

The decision problem Completely Reachable is the following: Given an automaton (Q, Σ, δ), is it completely reachable?

2.2 Witnesses

For an automaton (Q, Σ, δ), we consider unreachable sets that have the maximal size among all unreachable subsets of Q. They play the role of witnesses for the non-complete reachability of the automaton (or counterexamples for its complete reachability).

▶ **Definition 4** (Witness). *A non-empty set $S \subsetneq Q$ is a* witness *if it is unreachable and has the maximal size of all unreachable subsets of Q.*

Since Q is trivially reachable, the maximal size of unreachable subsets of Q is in $\{0, \ldots, n-1\}$. It equals 0 (i.e., there is no witness) if and only if the automaton is completely reachable.

Although any non-empty unreachable set is evidence that the automaton is not completely reachable, it turns out that verifying if a set is a witness is computationally easier – later we show that we can verify the complete reachability and also find a witness if it exists in polynomial time. Verifying whether a given set is reachable (or unreachable) in general is PSPACE-complete [6] and it remains hard in many variations of the problem [3].

We start with a simpler solution in co-NP, where we just guess a candidate for a witness and verify it. A witness obviously cannot have a larger predecessor, as this predecessor or maybe some other larger set would be unreachable and so a witness instead. Still, a set can have exponentially many predecessors of the same size. The following observation allows us to infer the existence of a larger predecessor indirectly.

▶ **Lemma 5.** *Let $S, T \subseteq Q$ be distinct. If a word $u \in \Sigma^*$ is properly extending $S \cup T$, then u is also properly extending S or T.*

Proof. Let u be a properly extending word for $U = S \cup T \subsetneq Q$, thus there is a larger maximal u-predecessor U' of U, i.e., $\delta(U', u) = U$, $\delta^{-1}(U, u) = U'$, and $|U'| > |U|$. By Remark 1, we have $|\delta^{-1}(q, u)| \geq 1$ for all $q \in U$. As this holds for all the states of S and T, they also have their u-predecessors $\delta^{-1}(S, u)$ and $\delta^{-1}(T, u)$, respectively. Suppose that $|\delta^{-1}(S, u)| = |S|$ and $|\delta^{-1}(T, u)| = |T|$. Then $|\delta^{-1}(q, u)| = 1$ for all $q \in U$, which gives a contradiction with $|U'| > |U|$. ◀

▶ **Corollary 6.** *Let $S, T \subsetneq Q$ be two distinct witnesses. Then $S \cup T = Q$.*

Proof. Since $S \neq T$, the union $S \cup T$ is larger than the witness size $|S| = |T|$, thus it must be reachable. If $S \cup T \neq Q$, then $S \cup T$ has a larger predecessor. By Lemma 5, either S or T also has a larger predecessor, which contradicts that they both are witnesses. ◀

Now, we focus on detecting sets that are potential witnesses. We relax the required property for being a witness and introduce an auxiliary definition:

▶ **Definition 7** (Witness candidate)**.** *A non-empty set $S \subsetneq Q$ is a* witness candidate *if it does not have a larger predecessor and all its predecessors (which are of the same size) have pairwise disjoint complements.*

▶ Remark 8. For two sets $T, T' \subseteq Q$, the condition of disjoint complements $\overline{T} \cap \overline{T'} = \emptyset$ is equivalent to $T \cup T' = Q$.

▶ **Lemma 9.** *A witness is a witness candidate.*

Proof. Let S be a witness and let T and T' be two distinct predecessors of S. As they are of the same size $|T| = |T'| = |S|$ and S is reachable from both T and T', they also must be witnesses. By Corollary 6, $T \cup T' = Q$. Therefore, S meets the definition of a witness candidate. ◀

Thus, every witness is a witness candidate and clearly, every witness candidate is unreachable (moreover, it is unreachable from every larger set). However, both converses do not necessarily hold.

▶ **Example 10.** The automaton from Figure 1 (left) is not completely reachable, and:

- All sets of size 5 are witnesses.
- The sets $Q \setminus \{q_i, q_{(i+1) \bmod 6}\}$ for $i \in \{0, \ldots, 5\}$ are reachable.

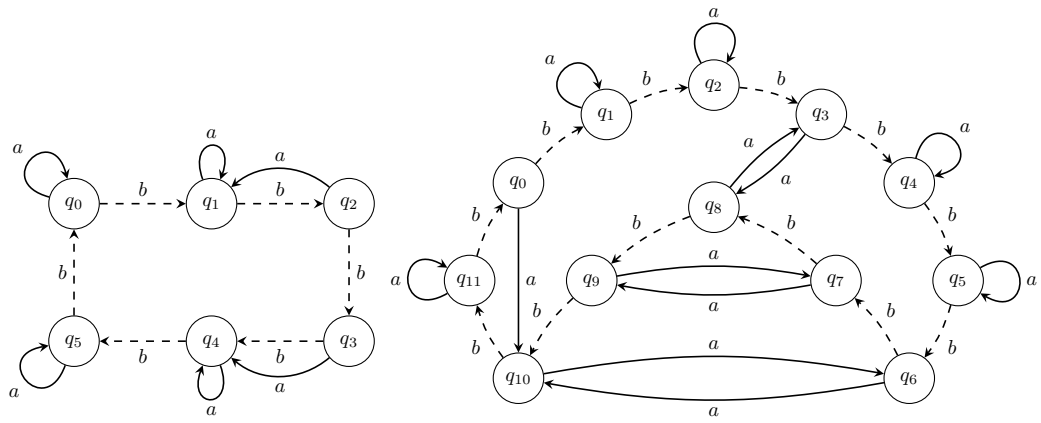


Figure 1 Left: an example of an automaton that is not completely reachable. **Right:** an example of a completely reachable automaton [8, Fig. 3]. (a's transitions are solid, b's transitions are dashed.)

- The sets $Q \setminus \{q_i, q_{(i+2) \bmod 6}\}$ for $i \in \{0, \ldots, 5\}$ are unreachable (only the action of b yields a predecessor). They do not admit a properly extending word but they are not witness candidates, since their complements are not disjoint.
- The set $\{q_0, q_2, q_4\}$ and its complement $\{q_1, q_3, q_5\}$ are witness candidates but not witnesses, as they have size 3.
- All non-empty proper subsets of Q are extensible, e.g., $\delta^{-1}(q_1, (ab)^3, a) = Q$, but not necessarily properly extensible.

The following remark will be useful for efficient checking if a set is a witness candidate:

▶ Remark 11. A witness candidate S has at most $\lfloor n/(n - |S|) \rfloor$ predecessors.

Proof. The complements of the predecessors of S are pairwise disjoint, so each state is contained in at most one complement of size $n - |S|$. ◀

2.3 An Algorithm in co-NP

We build a polynomial procedure that checks whether a given set S is a witness candidate. It is shown in Algorithm 1. Starting from S, we process all its predecessors in a breath-first search manner; for this, a FIFO queue *Process* is used. A next set T is taken in line 6. Then, we verify whether $\overline{T} \cap \overline{T'} \neq \emptyset$, for some previously processed set T'. For this, we maintain *Absent* array, which for every state q indicates whether q occurred in the complement of some previously processed set, and if so, this set is stored as $Absent[q]$. We additionally use this array to check whether the same set T has been processed previously (line 9); if so, then it is ignored. Otherwise, S is not a witness candidate as the complements of its two predecessors have a non-empty intersection. We update this array in lines 12–13. Finally, in lines 14–19, we add one-letter predecessors of T to the queue. If one of the predecessors is larger than T, then this immediately implies that S is not a witness candidate. Since predecessors are never smaller, this means that all processed sets that are put into the queue are of the same size $|S|$. When all predecessors of S have been considered and neither of the two conditions occurred, the function reports that S is a witness candidate.

The function is a base for our next algorithms. Hence, even though we do not need to optimize it here, in the next lemma, we describe the technical details for achieving optimal time complexity. In particular, an important optimization that lowers the time complexity

Algorithm 1 Verifying whether a given subset is a witness candidate from Definition 7.

Input: An n-state automaton $\mathscr{A} = (Q, \Sigma, \delta)$ and a non-empty $S \subsetneq Q$.
Output: **true** if S is a witness candidate; **false** otherwise.

```
1:  function IsWitnessCandidate(𝒜, S)
2:      Process ← EmptyFifoQueue()        ▷ It contains predecessors of S to be processed
3:      Process.Push(S)
4:      Absent ← Array indexed by q ∈ Q initialized with none
5:      while not Process.Empty() do
6:          T ← Process.Pop()
7:          if Absent[q] ≠ none for some q ∈ T̄ then          ▷ Then T ∪ T′ ≠ Q
8:              T′ ← Absent[q]
9:              if T = T′ then
10:                 continue                    ▷ T has been processed previously
11:             return false    ▷ T and T′ are predecessors with non-disjoint complements
12:         for all q ∈ T̄ do
13:             Absent[q] ← T
14:         for all a ∈ Σ do
15:             if T has a-predecessor then
16:                 T′ = δ^{-1}(T, a)
17:                 if |T′| > |S| then
18:                     return false
19:                 Process.Push(T′)
20:     return true                          ▷ All predecessors of S were checked
```

is storing each processed set T as a list of states in its complement. Then we can perform all operations on a set T in $\mathcal{O}(n - |S|)$ time, which compensates the number of iterations, which is at most $\mathcal{O}(n/(n - |S|))$.

▶ **Lemma 12.** *Function* IsWitnessCandidate *from Algorithm 1 is correct and can be implemented to work in* $\mathcal{O}(|\Sigma| \cdot n)$ *time.*

Proof. *Correctness:* The function can end in line 11, 18, or 20. The first case means that S has two distinct predecessors, T and T' such that $\overline{T} \cap \overline{T'} \neq \emptyset$, which implies that S is not a witness candidate. The second case (line 18) means that we have found a larger predecessor of S, thus S is not a witness candidate. The last possibility (line 20), where the function ends with a positive answer, occurs when there are no more predecessors of S to consider (*Process* becomes empty), so all predecessors of S have been checked and the two previous cases have not occurred. Thus, S satisfies the conditions from Definition 7.
Running time: We separately consider the time complexity of two types of iterations of the main loop: *full* and *extra* iterations. An extra iteration is where the case from line 7 holds; then the iteration ends either in line 10 or 11. Otherwise, we count it as a full iteration.

In each iteration, T has size $|S|$, because a predecessor cannot be smaller than the set and we check in line 18 if it is larger. The number of distinct processed predecessors (including S itself) of the same size $|S| < n$ with pairwise disjoint complements is at most $\lfloor n/(n - |S|) \rfloor$. This bounds the number of full iterations. In extra iterations, the same sets can be repeated. As these sets are added in full iterations, and one full iteration adds at most $|\Sigma|$ sets to the queue, the number of extra iterations is at most $|\Sigma| \cdot (\lfloor n/(n - |S|) \rfloor)$.

If we store T (and all sets in *Process*) as a list of states in its complement, then full one iteration takes $\mathcal{O}(|\Sigma| \cdot (n - |S|))$ time: Lines 6–11 trivially take $\mathcal{O}(n - |S|)$ time. Updating *Absent* in lines 12–13 also takes $\mathcal{O}(n-|S|)$ time, if we store a pointer/reference to T (line 14 in $\mathcal{O}(1)$) instead of copying. Computing one-letter predecessors in lines 15–19 can be performed in $\mathcal{O}(|\Sigma| \cdot (n - |S|))$ time as follows. At the beginning of the function, we additionally do some preprocessing of the automaton. For each $a \in \Sigma$ and $q \in Q$, we compute the list of states $\delta^{-1}(q, a)$; note that for the same a, these lists are always disjoint. For each $a \in \Sigma$, we also count the states $q \in Q$ such that $|\delta^{-1}(q, a)| = 0$; let z_a be their number. Then in line 15, we check if $\delta^{-1}(T, a)$ is a-predecessor by counting the states $p \in \overline{T}$ such that $|\delta^{-1}(p, a)| = 0$. The set $\delta^{-1}(T, a)$ is a-predecessor if and only if the number of such states p equals z_a, because if it is smaller, then some state $q \in T$ has an empty preimage under the action of a. Next, we compute $\delta^{-1}(\overline{T}, a)$ by joining the preprocessed lists for a for each $q \in \overline{T}$, and $T' = Q \setminus \delta^{-1}(\overline{T}, a)$ is also stored in the form of a list of states in the complement. Since a predecessor is never smaller than the set, we have $|T'| \geq |T| = |S|$, thus the length of this list is at most $|S|$, so we do this computation in $\mathcal{O}(n - |S|)$ time.

Also, the running time of an extra iteration (lines 6 up to 11) is easily bounded by $\mathcal{O}(n - |S|)$ time.

Summarizing, full iterations take $\mathcal{O}(n/(n - |S|) \cdot |\Sigma| \cdot (n - |S|))$ time and extra iterations take $\mathcal{O}(|\Sigma| \cdot n/(n - |S|) \cdot (n - |S|))$ time, which gives the same upper bound on the total. The aforementioned preprocessing can be done in $\mathcal{O}(|\Sigma| \cdot n)$ time as well. ◀

▶ Remark 13. The asymptotic running time of IsWitnessCandidate in terms of $|\Sigma|$ and n is the best possible because $|\Sigma| \cdot n$ is the number of automaton's transitions that we have to read in the worst case.

▶ **Theorem 14.** *Problem Completely Reachable can be solved in co-NP.*

Proof. To certify that a given automaton $\mathscr{A}$ is not completely reachable, we can guess a witness candidate $S \subsetneq Q$ and call IsWitnessCandidate$(\mathscr{A}, S)$ to verify it. If the automaton is not completely reachable, then there exists some witness, which is a witness candidate. Otherwise, there are no unreachable non-empty sets, thus no witness candidates. ◀

2.4 A Polynomial-Time Algorithm

The overall idea to make the algorithm work in deterministic polynomial time is as follows. We replace guessing a witness with a constructive procedure. We extend the function from Algorithm 1 so that instead of a Boolean answer, it finds a properly extending word for S. This works under a certain assumption that S is not a witness candidate and there are no witness candidates of size larger than $|S|$. When S is a witness candidate, the function returns **none**.

Then, we use this function to hunt for a witness. We iteratively reduce each set of size $n - 1$ to smaller sets S in the way that if some witness is a subset of the initial set, then this witness is also a subset of set S. As we process the sets from the largest size, we keep the assumption that there are no witness candidates larger than the currently processed set. Hence, the first found witness candidate will be a witness.

2.4.1 Finding Properly Extending Words

The function for finding properly extending words is shown in Algorithm 2. Here, together with predecessor sets T of S, we also keep track of the words w such that $\delta(T, w) = S$. The main difference with IsWitnessCandidate is the case in line 7. For the union $T \cup T'$,

Algorithm 2 An algorithm finding a properly extending word (a larger predecessor) for a given set (recursive version).

Input: An n-state automaton $\mathscr{A} = (Q, \Sigma, \delta)$ and a non-empty $S \subsetneq Q$.
Output: A properly extending word for S or **none** . If S is a witness candidate, then always returns **none** . If S is not a witness candidate and there are no witness candidates of size $> |S|$, then always returns a word.

```
1:  function FindProperlyExtendingWord(𝒜, S)
2:      Process ← EmptyFifoQueue()      ▷ It contains pairs (T, w) such that δ(T, w) = S
3:      Process.Push((S, ε))
4:      Absent ← Array indexed by q ∈ Q initialized with none
5:      while not Process.Empty() do
6:          (T, w) ← Process.Pop()
7:          if Absent[q] ≠ none for some q ∈ T̄ then          ▷ Then T ∪ T′ ≠ Q
8:              (T′, w′) ← Absent[q]
9:              if T′ = T then
10:                 continue                                  ▷ T′ has been processed previously
11:             u ← FindProperlyExtendingWord(𝒜, T ∪ T′)
12:             if u = none then
13:                 return none
14:             else                          ▷ u is a properly extending word for T ∪ T′
15:                 if |δ⁻¹(T, u)| > |T| then                 ▷ u properly extends T
16:                     return uw
17:                 else                                      ▷ u properly extends T′
18:                     return uw′
19:         for all q ∈ T̄ do
20:             Absent[q] ← (T, w)
21:         for all a ∈ Σ do
22:             if T has a-predecessor then
23:                 T′ = δ⁻¹(T, a)
24:                 if |T′| > |S| then
25:                     return aw
26:                 Process.Push((T′, aw))
27:     return none                                           ▷ S is a witness candidate
```

we aim at finding a properly extending word for it, which then turns out to be properly extending either for T or T' by Lemma 5; this is done by a recursive call in line 11. This call can return **none** instead, which means that a witness candidate larger than S was found.

▶ **Example 15.** For the automaton from Figure 1 (left) and $S = \{q_0, q_2, q_4, q_5\}$ (this is an unreachable and not properly extensible set, but not a witness candidate), FindProperlyExtendingWord returns **none** using one recursive call.

Proof. Since letter a cannot be applied, the second iteration is with $T = \delta^{-1}(S, b) = \{q_5, q_1, q_3, q_4\}$, and similarly, the third one is with $T = \delta^{-1}(S, b^2) = \{q_4, q_0, q_2, q_3\}$. Now, since the complements of $\{q_4, q_0, q_2, q_3\}$ and S contain the common state q_1, we recursively call the function for the union $Q \setminus \{q_1\}$. This union set is a witness and has 6 predecessors (including itself). After processing all these predecessors, the function returns **none**. ◀

▶ **Example 16.** For the automaton from Figure 1 (right) and let $S = \{q_0, q_{10}\}$, FindProperlyExtendingWord returns the word ab^2 using 8 recursive calls.

▶ **Lemma 17.** *Function FindProperlyExtendingWord is correct and works in at most $\mathcal{O}(|\Sigma| \cdot n^2 \log n)$ time. Moreover, if a word is returned, then it has length at most $\mathcal{O}(n \log n)$.*

Proof sketch. The correctness can be proved by descending induction on the set size, using similar arguments as for Lemma 12 and Lemma 5

Without counting the recursive call, the running time is $\mathcal{O}(|\Sigma| \cdot n^2/(n - |S|))$; for this, to avoid higher complexity by copying potentially long words, we need to concatenate words by storing pointers to both parts and maintain the induced transformations $Q \to Q$ along them. If we consider the recursive calls, in the worst case $n - |S|$ we call the function on sets of sizes $n - |S|, n - |S| + 1, \ldots, n - 1$; in the calculation we get the harmonic series what is bounded by the logarithm, giving the final running time and word length bounds. ◀

2.4.2 Set Reduction for Witness Containment

Suppose that given a subset $S \subsetneq Q$, we search for a witness that is contained within it. Having a properly extending word w for S, we can reduce S to its proper subset by excluding some states which surely cannot be in any witness contained in S. The next lemma shows the criterion.

▶ **Lemma 18.** *Let $S \subseteq Q$ and $\delta^{-1}(S, w)$ be a w-predecessor of S for some w. Then for every witness candidate $S' \subseteq S$, every state $q \in S'$ is such that $|\delta^{-1}(q, w)| = 1$.*

Proof. Since $\delta^{-1}(S, w)$ is a w-predecessor of S, by Remark 1, we have $\delta^{-1}(q, w) \neq \emptyset$ for every state $q \in S$. Suppose for a contradiction that there exists $S' \subseteq S$ that is a witness candidate and contains a state $p \in S'$ such that $|\delta^{-1}(p, w)| > 1$. Note that the set $\delta^{-1}(S', w)$ is a w-predecessor of S', since its superset S has a w-predecessor, and:

$$|\delta^{-1}(S', w)| = |\delta^{-1}(p, w)| + \sum_{s \in S \setminus \{p\}} |\delta^{-1}(s, w)| > 1 + (|S| - 1) = |S|.$$

Thus, S' cannot be a witness candidate, since it has a larger predecessor. ◀

As a witness is also a witness candidate, the lemma also applies to witnesses. From the lemma, we know that by having a properly extending word for S, we can remove at least one state from S and all the witnesses will still be contained in the resulting set. Function Reduce$(\mathscr{A}, S, w)$ in Algorithm 3 realizes this reduction and returns a set of states that can be removed. If w is given as a transformation, the function trivially works in $\mathcal{O}(n)$ time.

Algorithm 3 Reducing a set for possible witness containment.

Input: An n-state automaton $\mathscr{A} = (Q, \Sigma, \delta)$, a non-empty $S \subsetneq Q$, and a properly extending word w for S.

Output: A non-empty subset $R \subseteq S$ such that all the witnesses contained in S are also contained in $S \setminus R$.

```
1: function Reduce(𝒜, S, w)
2:     return {p ∈ S : |δ⁻¹(p, w)| > 1}
```

2.4.3 Finding a Witness

We have all ingredients to build a polynomial algorithm that solves the decision problem Completely Reachable. It is shown in Algorithm 4. If the automaton is not completely reachable, the algorithm also finds a witness. Starting from all sets of size $n-1$, we process sets in descending order by size. Processing a set consists of finding a properly extending word for it and reducing the set for a witness containment by this word.

Algorithm 4 A polynomial-time algorithm verifying the complete reachability of an automaton or finding a witness.

Input: An n-state automaton $\mathscr{A} = (Q, \Sigma, \delta)$.
Output: **none** if $\mathscr{A}$ is completely reachable; a witness otherwise.

```
1:  function FindWitness(𝒜)
2:      Queue ← EmptyPriorityQueue()   ▷ Ordered by set size; the largest sets go first
3:      for all q ∈ Q do
4:          Queue.Push(Q \ {q})                ▷ Initialize with sets of size n − 1
5:      while not Queue.Empty() do
6:          S ← Queue.Pop()                    ▷ Get a set of the largest size
7:          w ← FindProperlyExtendingWord(𝒜, S)
8:          if w = none then
9:              return S                       ▷ Found witness
10:         R ← Reduce(𝒜, S, w)                ▷ R is non-empty
11:         S′ ← S \ R                         ▷ S′ ⊊ S
12:         if S′ ≠ ∅ then
13:             Queue.Push(S′)
14:     return none                            ▷ No witnesses
```

▶ **Lemma 19.** *Function FindWitness is correct and works in $\mathcal{O}(|\Sigma| \cdot n^4 \log n)$ time.*

▶ Remark 20. The number of witnesses of size k is at most $\lfloor n/k \rfloor$, as their complements must be pairwise disjoint – otherwise, we could properly extend at least one of them by Lemma 5. Function FindWitness can be easily modified to find all the witnesses: after finding the first witness of size k, we can continue the main loop until all sets of size k are processed.

2.5 Improving Running Time

The main idea for the improvement is to use already computed reductions instead of finding a properly extending word recursively (Algorithm 2, line 13), which is required when we encounter the case of non-disjoint complements. For lowering the time complexity by this optimization, using adequate set representations is also crucial.

2.5.1 Reduction History

A *reduction history* RED is an array of size n of lists of states. For a state $q \in Q$, $RED[q]$ denotes the list of states assigned to q. Let $|RED[q]|$ denote the length of this list, and let $RED[q][i]$ denote the i-th state for $1 \leq i \leq |RED[q]|$. The states in the list must be pairwise different and distinct from q.

A reduction history represents our current knowledge about possible witness containment and will be progressively filled out in the algorithm, starting from the empty lists. For each $q \in Q$, the reduction history defines a *reduction chain* that is a sequence of reduced sets

$R^q_0, R^q_1, \ldots, R^q_{|RED[q]|}$. The first set $R^q_0 = Q \setminus \{q\}$, and each next set is obtained from the previous set by removing the corresponding state in the list: for $1 \leq i \leq |RED[q]|$, we define $R^q_i = R^q_{i-1} \setminus \{RED[q][i]\}$ (equivalently, $R^q_i = Q \setminus (\{q\} \cup \bigcup_{i \in \{1,\ldots,|RED[q]|\}} \{RED[q][i]\})$).

A reduction history RED is *valid* (for an automaton) if for each $q \in Q$ and each $1 \leq i \leq |RED[q]|$, there exists a properly extending word w for R^q_{i-1} such that $|\delta^{-1}(RED[q][i], w)| > 1$. This means that the reduction for witness containment of R^q_{i-1} to R^q_i by removing the state $RED[q][i]$ is justified by Lemma 18, i.e., all witnesses contained in R^q_{i-1} are also contained in R^q_i. However, note that the reduction can be not exhaustive with respect to w, i.e., there may exist other states $p \in R^q_{i-1} \setminus \{RED[q][i]\}$ such that $|\delta^{-1}(p, w)| > 1$, which also could be removed by Lemma 18. In our algorithm, this situation will be possible because we do not always compute properly extending words directly but infer their existence based on the reductions computed for other sets, tracing only one state to remove.

Besides being valid, we need that our reduction history is sufficiently filled out. For $q \in Q$, the *deficiency* of $RED[q]$ is the length $|RED[q]|$ thus equals the number of states to be removed from $Q \setminus \{q\}$. The *deficiency* of the whole reduction history RED is its minimum deficiency over $q \in Q$. Hence, a valid history reduction of deficiency d stores information for reducing every set of size $n-1$ to a set of size at most $n-1-d$.

Algorithm 5 A fast reduction retrieval for witness containment for a given set from the past stored reductions.

Input: An n-state automaton $\mathscr{A} = (Q, \Sigma, \delta)$, a non-empty set $S \subsetneq Q$, and a reduction history RED.
Require: RED is a valid reduction history of deficiency at least $n - |S|$.
Output: A state $p \in S$ such that there exists a properly extending word w for S such that $|\delta^{-1}(p, w)| > 1$.

```
1: function GetStoredReduction(𝒜, S, RED)
2:     q ← any state from S̄
3:     for i ← 1, 2, ... do
4:         assert(i ≤ |RED[q]|)
5:         p ← RED[q][i]
6:         if p ∈ S then
7:             return p
```

Function GetStoredReduction from Algorithm 5 quickly finds a reduction of a given set S, provided that a valid reduction history with a large enough deficiency is available. The function starts from picking any q outside of S, and it repeats the reduction chain starting from $Q \setminus \{q\} \supseteq S$. It seeks the first removed state that belongs to S. In this way, since the same reduction was applied previously to a superset of S, it can be correctly applied to S as well, i.e., the same word is properly extending word for both the superset and S. We note that this may not hold for the states removed later, since then a properly extending word inferred from the reduction history may not be properly extending for S (may not give a predecessor of S).

Since the found state $p \in S$ has the property that there exists a properly extending word w for S such that $|\delta^{-1}(p, w)| > 1$, we can apply Lemma 18 and get a smaller $S' = S \setminus \{p\} \subsetneq S$ such that every witness in S is also contained in S'.

▶ **Lemma 21.** *Function* GetStoredReduction *is correct and can be implemented to work in* $\mathcal{O}(n)$ *time.*

2.5.2 Finding a Reduction

We redesign earlier Algorithm 2 for finding a properly extending word so that we return a reduction (here, one state to remove) directly for the given S. GetStoredReduction can be used for computing the reduction fast, but only if our reduction history has a large enough deficiency, i.e., in each of the reduction chains, $Q \setminus \{q\}$ is reduced to a set smaller than the set that we want to reduce. We cannot ensure this for S, but if in the main algorithm, we reduce the sets in descending order by their size, then we can fulfil the weaker requirement that the reduction chains end with sets of size at most $|S|$. Then, to reduce S, we perform as the previous algorithm until encountering the case of a non-empty complements intersection. Then we can use GetStoredReduction for the obtained set of size at least $|S| + 1$.

Function FindReduction from Algorithm 6 for a given S finds a state to remove or **none** if S is a witness candidate. The most important difference with previous FindProperlyExtendingWord is the use of GetStoredReduction in line 11 and processing its result p.

To keep the time complexity low, as before, we need to store the sets T in the form of a list of states in the complement. This time, we do not maintain the induced transformations for words (they are too costly here). Instead, we compute $\delta(p, w)$ applying the letters one by one, but only for this state, which is doable in $\mathcal{O}(n/(n - |S|))$ time, avoiding quadratic time complexity.

▶ **Lemma 22.** *Function FindReduction is correct and can be implemented to work in $\mathcal{O}(|\Sigma| \cdot n)$ time.*

2.5.3 The Optimized Algorithm

The optimized algorithm FindWitnessFaster is shown in Algorithm 7. For reducing sets, it relies on FindReduction, which returns a state p such that there exists a properly extending word w for S and $|\delta^{-1}(p, w)| > 1$. We can remove p from S by Lemma 18, i.e., every witness contained in S must be also contained in $S' = S \setminus \{p\}$.

Additionally, the sets in *Queue* are stored together with their initially removed state q, to know where to store their reductions in the reduction history. Note that in an iteration, the set S taken from the queue is the currently last set $R^q_{|RED[q]|}$ from the reduction chain for q, and S' will be a next set in this chain. Updating the reduction history with state p keeps it valid, which also follows from the guaranteed existence of a properly extending word by FindReduction. As we process the sets in descending order by size, our reduction history always has the required deficiency when used in line 8.

▶ **Lemma 23.** *Function FindWitnessFaster is correct and works in $\mathcal{O}(|\Sigma| \cdot n^3)$ time.*

3 An Upper Bound on Reset Threshold

3.1 Synchronization

A *reset word* is a word w such that $|\delta(Q, w)| = 1$. Equivalently, we have $\delta^{-1}(q, w) = Q$ for some $q \in Q$. If an automaton admits a reset word, then it is called *synchronizing* and its *reset threshold* is the length of the shortest reset words.

The central problem in the theory of synchronizing automata is the famous Černý conjecture, which states that every synchronizing n-state automaton has its reset threshold at most $(n - 1)^2$. For the subclass of completely reachable automata, the previously known upper bound on the reset threshold was $7/48n^3 + \mathcal{O}(n^2)$ [5], which has been obtained through the technique of avoiding [22], that is, it follows in particular from the fact that every set of size $n - 1$ is reachable with a word of length at most n.

Algorithm 6 An algorithm finding a reduction for witness containment using a reduction history.

Input: An n-state automaton $\mathscr{A} = (Q, \Sigma, \delta)$, a non-empty $S \subsetneq Q$, and a reduction history *RED*.
Require: *RED* is a valid reduction history of deficiency at least $n - |S| - 1$.
Output: If S is not a witness candidate: a state $p \in S$ such that there exists a properly extending word w for S such that $|\delta^{-1}(p, w)| > 1$. Otherwise: **none**.

```
1:  function FindReduction(𝒜, S, RED)
2:      Process ← EmptyFifoQueue()
3:      Process.Push((S, ε))
4:      Absent ← Array indexed by q ∈ Q initialized with none
5:      while not Process.Empty() do
6:          (T, w) ← Process.Pop()
7:          if Absent[q] ≠ none for some q ∈ T̄ then          ▷ Then T ∪ T′ ≠ Q
8:              (T′, w′) ← Absent[q]
9:              if T′ = T then
10:                 continue                                  ▷ T′ has been processed previously
11:             p ← GetStoredReduction(𝒜, T ∪ T′, RED)
12:             if p ∈ T then
13:                 return δ(p, w)
14:             else                                          ▷ Then p ∈ T′
15:                 return δ(p, w′)
16:         for all q ∈ T̄ do
17:             Absent[q] ← (T, w)
18:         for all a ∈ Σ do
19:             if T has a-predecessor then
20:                 T′ = δ^{-1}(T, a)
21:                 if |T′| > |S| then
22:                     p ← any state such that |δ^{-1}(T, a)| > 1
23:                     return δ(p, w)
24:                 Process.Push((T′, aw))
25:     return none                                           ▷ S is a witness candidate
```

3.2 Finding Short Properly Extending Words

For a completely reachable automaton, function FindProperlyExtendingWord from Algorithm 8 always finds a properly extending word. Therefore, using the well-known *extension* method (e.g., [24]), we can construct a synchronizing word starting from some singleton $\{q\}$ and iteratively increasing the set by at least one in at most $n - 1$ iterations, finally obtaining Q. This is an easy way to get the upper bound of order $\mathcal{O}(n^2 \log n)$ by Lemma 17. However, it is not enough to prove a quadratic upper bound, so we are going to further adapt the algorithm for that.

The idea is to keep track of all subsets for an intersection of complements, instead of starting an independent search for a properly extending word recursively. This modification is shown in Algorithm 8.

The function keeps *Trace* map, which stores for a given predecessor set, by the application of what letter it has been obtained or, in the second case, of what two sets it is a union. It also stores S' as the current origin set, which is the set for which we are going to find a larger predecessor currently.

Algorithm 7 A faster algorithm for finding a witness.

Input: An n-state automaton $\mathscr{A} = (Q, \Sigma, \delta)$.
Output: **none** if $\mathscr{A}$ is completely reachable; a witness otherwise.

```
 1: function FindWitnessFaster(𝒜)
 2:     RED[q] ← EmptyList  for all q ∈ Q          ▷ Empty reduction history
 3:     Queue ← EmptyPriorityQueue()               ▷ Contains pairs (S, q); ordered by |S|
 4:     for all q ∈ Q do
 5:         Queue.Push((Q \ {q}, q))
 6:     while not Queue.Empty() do
 7:         (S, q) ← Queue.Pop()                   ▷ Get a set of the largest size
 8:         p ← FindReduction(𝒜, S, RED)
 9:         if p = none then
10:             return S                           ▷ Found witness
11:         RED[q].append(p)                       ▷ Store the removed state
12:         S' ← S \ {p}
13:         if S' ≠ ∅ then
14:             Queue.Push((S', q))
15:     return none                                ▷ No witnesses
```

The function has two phases. First, it searches for a larger predecessor or a non-empty complement intersection as the previous variants. In the second case, it only changes the origin set S' (line 15), notes this in the map *Trace* and initiates a fresh search for S'. In the second phase (from line 25), a properly extending word is reconstructed (like in FindProperlyExtendingWord after a recursive call) until we reach our original S.

▶ **Lemma 24.** *Function FindShortProperlyExtendingWord is correct.*

The remaining effort is to prove an upper bound on the length of the returned word.

3.2.1 Nested Boxes

To bound the length of the found word, we consider an auxiliary combinatorial problem. Consider an n-element universe Q. Two subsets $S, T \subseteq Q$ are *colliding* if $S \cap T \notin \{\emptyset, S, T\}$. Thus, colliding sets have a non-trivial intersection. A family of non-empty subsets of Q is called *non-colliding* if all the subsets are pairwise non-colliding.

▶ **Definition 25.** *For an $n \geq 1$, the number $MaxNestedBoxes(n)$ is the maximum size of a non-colliding family for an n-element universe.*

The problem is equivalent to, e.g., the maximum number of boxes for n items, such that a box must contain either an item or at least two boxes.

▶ **Lemma 26.** $MaxNestedBoxes(n) = 2n - 1$.

The generalized version of the problem limits the maximum size of the subsets:

▶ **Definition 27.** *For an $n \geq 1$, the number $MaxNestedBoxes(n, k)$ is the maximum size of a non-colliding family for an n-element universe where each subset from the family has size at most k.*

▶ **Lemma 28.** $MaxNestedBoxes(n, k) = 2n - \lceil n/k \rceil$.

Algorithm 8 An algorithm finding a short properly extending word.

Input: An n-state automaton $\mathscr{A} = (Q, \Sigma, \delta)$ and a non-empty $S \subsetneq Q$.
Require: $\mathscr{A}$ is completely reachable.
Output: A properly extending word w of S.

1: **function** FindShortProperlyExtendingWord($\mathscr{A}, S$)
2: $Trace \leftarrow$ EmptyMap() ▷ For a processed set, it stores how this set was obtained; for not yet processed sets, it gives **none**
3: $Process \leftarrow$ EmptyFifoQueue()
4: $S' \leftarrow S$ ▷ Current origin set
5: $Trace[S'] \leftarrow \varepsilon$
6: $Process$.Push(S')
7: **while true do**
8: **assert**(**not** $Process$.IsEmpty()) ▷ Otherwise S' is a witness candidate
9: $T \leftarrow Process$.Pop()
10: **if** $|T| > |S'|$ **then** ▷ Found a properly extending word for S'
11: $S' \leftarrow T$
12: **break**
13: **if** there is T' such that $Trace[T'] \neq$ **none** and $T \subsetneq T \cup T' \subsetneq Q$ **then**
14: $S' \leftarrow T \cup T'$ ▷ New origin set; $|S'| > |S|$
15: $Trace[S'] \leftarrow (T, T')$
16: $Process$.Clear() ▷ Continue only for the new origin
17: $Process$.Push(S')
18: **else**
19: **for all** $a \in \Sigma$ **do**
20: **if** a is properly extending T **then**
21: $T' \leftarrow \delta^{-1}(T, a)$
22: **if** $Trace[T'] =$ **none then** ▷ A not yet processed set
23: $Trace[T'] \leftarrow a$ ▷ $\delta(T', a) = T$
24: $Process$.Push(T')
25: $w \leftarrow \varepsilon$ ▷ Word reconstruction starts with the empty word
26: **while** $S' \neq S$ **do**
27: **assert**($Trace[S'] \neq$ **none**)
28: **if** $Trace[S']$ is a letter **then**
29: $a \leftarrow Trace[S']$
30: $S' \leftarrow \delta(S', a)$
31: $w \leftarrow wa$
32: **else**
33: $(T, T') \leftarrow Trace[S']$ ▷ w properly extends $T \cup T'$
34: **if** $|\delta^{-1}(T, w)| > |T|$ **then** ▷ w properly extends T
35: $S' \leftarrow T$
36: **else** ▷ w properly extends T'
37: $S' \leftarrow T'$
38: **return** w

3.2.2 Final Bounding

We apply the above combinatorial problem to derive an upper bound on the length of a word found by FindShortProperlyExtendingWord. The crucial property is that the family of the complements of all subsets T that are processed in the block of lines 19–24 is

non-colliding, since for these subsets T, the condition in line 13 does not hold. The upper bound follows since all the letters in the final word are added in line 31, where S' is always the complement of one of the sets from the family.

▶ **Lemma 29.** *For a completely reachable automaton and a non-empty proper subset $S \subsetneq Q$, the word returned by* FindShortProperlyExtendingWord *from Algorithm 8 has length at most $MaxNestedBoxes(n, n - |S|) = 2n - \lceil n/(n - |S|) \rceil$.*

Finally, using the standard extension method (starting from a subset S and iteratively extending it to Q) and some calculations, we obtain a weaker Don's conjecture (cf. [14, Problem 4]):

▶ **Theorem 30.** *For a completely reachable n-state automaton (Q, Σ, δ), every non-empty proper subset $S \subseteq Q$ is reachable with a word of length at most*

$$(n - |S|)2n - n\ln(n - |S|) - n/(n - |S|) < 2n(n - |S|).$$

▶ **Corollary 31.** *The reset threshold of a completely reachable automaton with $n \geq 3$ states is at most*

$$(n - 2)2n - n\ln(n - 2) - n/(n - 2) < 2n^2 - n\ln n - 4n + 2.$$

References

1. D. S. Ananichev, M. V. Volkov, and V. V. Gusev. Primitive digraphs with large exponents and slowly synchronizing automata. *Journal of Mathematical Sciences*, 192(3):263–278, 2013.
2. M. V. Berlinkov, R. Ferens, A. Ryzhikov, and M. Szykuła. Synchronizing Strongly Connected Partial DFAs. In *STACS*, volume 187 of *LIPIcs*, pages 12:1–12:16. Schloss Dagstuhl, 2021.
3. M. V. Berlinkov, R. Ferens, and M. Szykuła. Preimage problems for deterministic finite automata. *Journal of Computer and System Sciences*, 115:214–234, 2021.
4. J. Berstel, D. Perrin, and C. Reutenauer. *Codes and Automata*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2009.
5. E. A. Bondar, D. Casas, and M. V. Volkov. Completely reachable automata: an interplay between automata, graphs, and trees, 2022. `arXiv:2201.05075`.
6. E. A. Bondar and M. V. Volkov. Completely Reachable Automata. In Cezar Câmpeanu, Florin Manea, and Jeffrey Shallit, editors, *DCFS*, pages 1–17. Springer, 2016.
7. E. A. Bondar and M. V. Volkov. A Characterization of Completely Reachable Automata. In Mizuho Hoshi and Shinnosuke Seki, editors, *DLT*, pages 145–155. Springer, 2018.
8. D. Casas and M. V. Volkov. Binary completely reachable automata. In Armando Castañeda and Francisco Rodríguez-Henríquez, editors, *LATIN 2022: Theoretical Informatics*, pages 345–358. Springer, 2022. Full version at `arXiv:2205.09404`.
9. J. Černý. Poznámka k homogénnym experimentom s konečnými automatami. *Matematicko-fyzikálny Časopis Slovenskej Akadémie Vied*, 14(3):208–216, 1964. In Slovak.
10. H. Don. The Černý Conjecture and 1-Contracting Automata. *Electronic Journal of Combinatorics*, 23(3):P3.12, 2016.
11. D. Eppstein. Reset sequences for monotonic automata. *SIAM Journal on Computing*, 19:500–510, 1990.
12. R. Ferens, M. Szykuła, and V. Vorel. Lower Bounds on Avoiding Thresholds. In *MFCS*, volume 202 of *LIPIcs*, pages 46:1–46:14. Schloss Dagstuhl, 2021.
13. F. Gonze, V. V. Gusev, B. Gerencser, R. M. Jungers, and M. V. Volkov. On the interplay between Babai and Černý's conjectures. In *DLT*, volume 10396 of *LNCS*, pages 185–197. Springer, 2017.

14 F. Gonze and R. M. Jungers. Hardly reachable subsets and completely reachable automata with 1-deficient words. *Journal of Automata, Languages and Combinatorics*, 24(2–4):321–342, 2019.

15 S. Hoffmann. Completely Reachable Automata, Primitive Groups and the State Complexity of the Set of Synchronizing Words. In Alberto Leporati, Carlos Martín-Vide, Dana Shapira, and Claudio Zandron, editors, *LATA*, LNCS, pages 305–317. Springer, 2021.

16 J. Kari and M. V. Volkov. Černý conjecture and the road colouring problem. In *Handbook of automata*, volume 1, pages 525–565. European Mathematical Society Publishing House, 2021.

17 M. Maslennikova. Reset complexity of ideal languages over a binary alphabet. *International Journal of Foundations of Computer Science*, 30(06n07):1177–1196, 2019.

18 J.-E. Pin. On two combinatorial problems arising from automata theory. In *Proceedings of the International Colloquium on Graph Theory and Combinatorics*, volume 75 of *North-Holland Mathematics Studies*, pages 535–548, 1983.

19 I.K. Rystsov. Estimation of the length of reset words for automata with simple idempotents. *Cybern. Syst. Anal. 36*, pages 339–344, 2000.

20 S. Sandberg. Homing and synchronizing sequences. In *Model-Based Testing of Reactive Systems*, volume 3472 of *LNCS*, pages 5–33. Springer, 2005.

21 Y. Shitov. An Improvement to a Recent Upper Bound for Synchronizing Words of Finite Automata. *Journal of Automata, Languages and Combinatorics*, 24(2–4):367–373, 2019.

22 M. Szykuła. Improving the Upper Bound on the Length of the Shortest Reset Word. In *STACS 2018*, LIPIcs, pages 56:1–56:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

23 M. Szykuła and A. Zyzik. An Improved Algorithm for Finding the Shortest Synchronizing Words. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *30th Annual European Symposium on Algorithms (ESA 2022)*, volume 244 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 85:1–85:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

24 M. V. Volkov. Synchronizing automata and the Černý conjecture. In *Language and Automata Theory and Applications*, volume 5196 of *LNCS*, pages 11–27. Springer, 2008.

25 M. V. Volkov. Synchronization of finite automata. *Uspekhi Matematicheskikh Nauk*, 77:53–130, 2022. in Russian.

Approximating Long Cycle Above Dirac's Guarantee

Fedor V. Fomin
Department of Informatics, University of Bergen, Norway

Petr A. Golovach
Department of Informatics, University of Bergen, Norway

Danil Sagunov
St. Petersburg Department of V.A. Steklov Institute of Mathematics, Russia

Kirill Simonov
Hasso Plattner Institute, Universität Potsdam, Germany

Abstract

Parameterization above (or below) a guarantee is a successful concept in parameterized algorithms. The idea is that many computational problems admit "natural" guarantees bringing to algorithmic questions whether a better solution (above the guarantee) could be obtained efficiently. For example, for every boolean CNF formula on m clauses, there is an assignment that satisfies at least $m/2$ clauses. How difficult is it to decide whether there is an assignment satisfying more than $m/2 + k$ clauses? Or, if an n-vertex graph has a perfect matching, then its vertex cover is at least $n/2$. Is there a vertex cover of size at least $n/2 + k$ for some $k \geq 1$ and how difficult is it to find such a vertex cover?

The above guarantee paradigm has led to several exciting discoveries in the areas of parameterized algorithms and kernelization. We argue that this paradigm could bring forth fresh perspectives on well-studied problems in approximation algorithms. Our example is the longest cycle problem. One of the oldest results in extremal combinatorics is the celebrated Dirac's theorem from 1952. Dirac's theorem provides the following guarantee on the length of the longest cycle: for every 2-connected n-vertex graph G with minimum degree $\delta(G) \leq n/2$, the length of the longest cycle L is at least $2\delta(G)$. Thus the "essential" part of finding the longest cycle is in approximating the "offset" $k = L - 2\delta(G)$. The main result of this paper is the above-guarantee approximation theorem for k. Informally, the theorem says that approximating the offset k is not harder than approximating the total length L of a cycle. In other words, for any (reasonably well-behaved) function f, a polynomial time algorithm constructing a cycle of length $f(L)$ in an undirected graph with a cycle of length L, yields a polynomial time algorithm constructing a cycle of length $2\delta(G) + \Omega(f(k))$.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Approximation algorithms analysis; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Longest path, longest cycle, approximation algorithms, above guarantee parameterization, minimum degree, Dirac theorem

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.60

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: http://arxiv.org/abs/2305.02011 [26]

Funding Supported by the Research Council of Norway via the project BWCA (grant no. 314528) and DFG Research Group ADYN via grant DFG 411362735.

1 Introduction

One of the concepts that had a strong impact on the development of parameterized algorithms and kernelization is the idea of the above guarantee parameterization. Above guarantee parameterization grounds on the following observation: *the natural parameterization of a maximization/minimization problem by the solution size is not satisfactory if there is a lower bound for the solution size that is sufficiently large* [23]. To make this discussion concrete, consider the example of the classical NP-complete problem Max Cut. Observe that in any graph with m edges there is always a cut containing at least $m/2$ edges. (Actually, slightly better bounds are known in the literature [18, 10].) Thus Max Cut is trivially fixed-parameter tractable (FPT) parameterized by the size of the max-cut. Indeed, the following simple algorithm shows that the problem is FPT: If $k \leq m/2$, then return yes; else $m \leq 2k$ and any brute-force algorithm will do the job. However, the question about Max Cut becomes much more meaningful and interesting, when one seeks a cut above the "guaranteed" lower bound $m/2$.

The above guarantee approach was introduced by Mahajan and Raman [46] and it was successfully applied in the study of several fundamental problems in parameterized complexity and kernelization. For illustrative examples, we refer to [1, 4, 14, 23, 25, 33, 34, 35, 37, 38, 45], see also the recent survey of Gutin and Mnich [36]. Quite surprisingly, the theory of the above (or below) guarantee *approximation* remains unexplored. (Notable exceptions are the works of Mishra et al. [47] on approximating the minimum vertex cover beyond the size of a maximum matching and of Bollobás and Scott on approximating max-cut beyond the $m/2 + \sqrt{m/8}$ bound [10].)

In this paper, we bring the philosophy of the above guarantee parameterization into the realm of approximation algorithms. In particular,

> The goal of this paper is to study the approximability of the classical problems of finding a longest cycle and a longest (s,t)-path in a graph from the viewpoint of the above guarantee parameterization.

Our results. Approximating the length of a longest cycle in a graph enjoys a lengthy and rich history [6, 8, 21, 20, 29, 30, 49]. There are several fundamental results in extremal combinatorics providing lower bounds on the length of a longest cycle in a graph. The oldest of these bounds is given by Dirac's Theorem from 1952 [17]. Dirac's Theorem states that a 2-connected graph G with the minimum vertex degree $\delta(G)$ contains a cycle of length $L \geq \min\{2\delta(G), |V(G)|\}$. Since every longest cycle in a graph G with $\delta(G) < \frac{1}{2}|V(G)|$ (otherwise, G is Hamiltonian and a longest cycle can be found in polynomial time) always has a "complementary" part of length $2\delta(G)$, the essence of the problem is in computing the "offset" $k = L - 2\delta(G)$. Informally, the first main finding of our paper is that Dirac's theorem is well-compatible with approximation. We prove that approximating the offset k is essentially not more difficult than approximating the length L.

More precisely. Recall that f is subadditive if for all x, y it holds that $f(x+y) \leq f(x)+f(y)$. Our main result is the following theorem.

▶ **Theorem 1.** *Let $f : \mathbb{R}_+ \to \mathbb{R}_+$ be a non-decreasing subadditive function and suppose that we are given a polynomial-time algorithm finding a cycle of length at least $f(L)$ in graphs with the longest cycle length L. Then there exists a polynomial time algorithm that finds a cycle of length at least $2\delta(G) + \Omega(f(L - 2\delta(G)))$ in a 2-connected graph G with $\delta(G) \leq \frac{1}{2}|V(G)|$ and the longest cycle length L.*

The 2-connectivity condition is important. As was noted in [24], deciding whether a connected graph G contains a cycle of length at least $2\delta(G)$ is NP-complete. Theorem 1 trivially extends to approximating the longest path problem above $2\delta(G)$. For the longest path, the requirement on 2-connectivity of a graph can be relaxed to connectivity. This can be done by a standard reduction of adding an apex vertex v to the connected graph G, see e.g. [24]. The minimum vertex degree in the new graph $G+v$, which is 2-connected, is equal to $\delta(G)+1$, and G has a path of length at least L if and only if $G+v$ has a cycle of length at least $L+2$. Thus approximation of the longest cycle (by making use of Theorem 1) in $G+v$, is also the approximation of the longest path in G.

Related work. The first approximation algorithms for longest paths and cycles followed the development of exact parameterized algorithms. Monien [48] and Bodlaender [9] gave parameterized algorithms computing a path of length L in times $\mathcal{O}(L!2^L n)$ and $\mathcal{O}(L!nm)$ respectively. These algorithms imply also approximation algorithms constructing in polynomial time a path of length $\Omega(\log L/\log\log L)$, where L is the longest path length in graph G. In their celebrated work on color coding, Alon, Yuster, and, Zwick [2] obtained an algorithm that in time $\mathcal{O}(5.44^L n)$ finds a path/cycle of length L. The algorithm of Alon et al. implies constructing in polynomial time a path of length $\Omega(\log L)$. A significant amount of the consecutive work targets to improve the base of the exponent c^L in the running times of the parameterized algorithms for longest paths and cycles [43, 50, 28, 5, 7]. The surveys [27, 44], and [15, Chapter 10] provide an overview of ideas and methods in this research direction. The exponential dependence in L in the running times of these algorithms is asymptotically optimal: An algorithm finding a path (or cycle) of length L in time $2^{o(L)}n^{\mathcal{O}(1)}$ would fail the Exponential Time Hypothesis (ETH) of Impagliazzo, Paturi, and Zane [39]. Thus none of the further improvements in the running times of parameterized algorithms for longest cycle or path, would lead to a better than $\Omega(\log L)$ approximation bound.

Björklund and Husfeldt [6] made the first step "beyond color-coding" in approximating the longest path. They gave a polynomial-time algorithm that finds a path of length $\Omega(\log L/\log\log L)^2$ in a graph with the longest path length L. Gabow in [30] enhanced and extended this result to approximating the longest cycle. His algorithm computes a cycle of length $2^{\Omega(\sqrt{\log L/\log\log L})}$ in a graph with a cycle of length L. Gabow and Nie [32] observed that a refinement of Gabow's algorithm leads to a polynomial-time algorithm constructing cycles of length $2^{\Omega(\sqrt{\log L})}$. This is better than $(\log(L))^{\mathcal{O}(1)}$ but worse than L^{ε}. Pipelining the algorithm of Gabow and Nie with Theorem 1 yields a polynomial time algorithm constructing in a 2-connected graph G a cycle of length $2\delta(G)+\Omega(c^{\sqrt{\log k}})$. For graphs of bounded vertex degrees, better approximation algorithms are known [13, 21].

The gap between the upper and lower bounds for the longest path approximation is still big. Karger, Motwani, and Ramkumar [41] proved that the longest path problem does not belong to APX unless P = NP. They also show that for any $\varepsilon>0$, it cannot be approximated within $2^{\log^{1-\varepsilon} n}$ unless NP $\subseteq$ DTIME$(2^{O(\log^{1/\varepsilon} n)})$. Bazgan, Santha, and Tuza [3] extended these lower bounds to cubic Hamiltonian graphs. For directed graphs the gap between the upper and lower bounds is narrower [8, 31].

Our approximation algorithms are inspired by the recent work Fomin, Golovach, Sagunov, and Simonov [24] on the parameterized complexity of the longest cycle beyond Dirac's bound. Fomin et al. were interested in computing the "offset" beyond $2\delta(G)$ exactly. Their parameterizes algorithm decides whether G contains a cycle of length at least $2\delta(G)+k$ in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$, and thus in polynomial time computes a cycle of length $2\delta(G)+\Omega(\log k)$. However, the tools developed in [24] are not sufficient to go beyond $\Omega(\log k)$-bound on the

offset. The main combinatorial tools from [24] are Erdős-Gallai decomposition and Dirac decomposition of graphs. For the needs of approximation, we have to develop novel ("nested") variants or prove additional structural properties of these decompositions.

Dirac's theorem is one of the central pillars of Extremal Graph Theory. The excellent surveys [12] and [11] provide an introduction to this fundamental subarea of graph theory. Besides [24], the algorithmic applications of Dirac's theorem from the perspective of parameterized complexity were studied by Jansen, Kozma, and Nederlof in [40].

Paper structure. Section 2 provides an overview of the techniques employed to achieve our results. Then, Section 3 introduces notations and lists auxiliary results. Section 4 guides through the proof of the approximation result for (s,t)-paths, which is the key ingredient required for Theorem 1. Finally, we conclude with a summary and some open questions in Section 5. Note that the proofs of technical statements are omitted from this extended abstract due to space constraints. Detailed proofs of all results can be found in the full version of the paper [26].

2 Overview of the proofs

In this section, we provide a high-level strategy of the proof of Theorem 1, as well as key technical ideas needed along the way. The central concept of our work is an approximation algorithm for the Longest Cycle problem. Formally, such an algorithm should run in polynomial time for a given graph G and should output a cycle of length at least $f(L)$, where L is the length of the longest cycle in G. The function f here is the *approximation guarantee* of the algorithm. In our work, we allow it to be an arbitrary non-decreasing function $f : \mathbb{R}_+ \to \mathbb{R}_+$ that is also subadditive (i.e., $f(x) + f(y) \geq f(x+y)$ for arbitrary x, y). We also note that an $f(L)$-approximation algorithm for Longest Cycle immediately gives a $\frac{1}{2}f(2L)$-approximation algorithm for Long (s,t)-Path in 2-connected graphs (by Menger's theorem, see Lemma 4 for details).

Our two main contributions assume that we are given such an f-approximation algorithm as a black box. In fact, we only require to run this algorithm on an arbitrary graph as an oracle and receive its output. We do not need to modify or know the algorithm routine.

While the basis of our algorithm comes from the structural results of Fomin et al. [24], in the first part of this section we do not provide the details on how it is used.

The first of our contributions is a polynomial-time algorithm that finds a long (s,t)-path in a given 2-connected graph G with two vertices $s, t \in V(G)$. The longest (s,t)-path in G always has length $\delta(G - \{s,t\}) + k$ for $k \geq 0$ by Erdős-Gallai theorem, and the goal of the algorithm is to find an (s,t)-path of length at least $\delta(G - \{s,t\}) + \Omega(f(k))$ in G. To find such a path, this algorithm first recursively decomposes the graph G in a specific technical way. As a result, it outputs several triples (H_i, s_i, t_i) in polynomial time, where H_i is a 2-connected minor of G and $s_i, t_i \in V(H_i)$. For each triple, the algorithm runs the black box to find a f-approximation of the longest (s_i, t_i)-path in H_i. In the second round, our algorithm cleverly uses constructed approximations to construct a path of length at least $\delta(G - \{s,t\}) + \Omega(f(k))$ in the initial graph G. This is summarized as the following theorem.

▶ **Theorem 2.** *Let $f : \mathbb{R}_+ \to \mathbb{R}_+$ be a non-decreasing subadditive function and suppose that we are given a polynomial-time algorithm computing an (s,t)-path of length at least $f(L)$ in graphs with given two vertices s and t having the longest (s,t)-path of length L. Then there is a polynomial-time algorithm that outputs an (s,t)-path of length at least $\delta(G - \{s,t\}) + \Omega(f(L - \delta(G - \{s,t\})))$ in a 2-connected graph G with two given vertices s and t having the longest (s,t)-path length L.*

The second (and main) contribution of this paper is the polynomial-time algorithm that approximates the longest cycle in a given 2-connected graph G such that $2\delta(G) \leq |V(G)|$. It employs the black-box f-approximation algorithm for Longest Cycle to find a cycle of length $2\delta(G) + \Omega(f(k))$, where $2\delta(G) + k$ is the length of the longest cycle in G. By Dirac's theorem applied to G, k is always at least 0.

To achieve that, our algorithm first tries to decompose the graph G. However, in contrast to the first contributed algorithm, here the decomposition process is much simpler. In fact, the decomposition routine is never applied recursively, as the decomposition itself needs not to be used: its existence is sufficient to apply another, simple, procedure.

Similarly to the first contribution, the algorithm then outputs a series of triples (H_i, s_i, t_i), where H_i is a 2-connected minor of G and $s_i, t_i \in V(H_i)$. The difference here is that for each triple the algorithm runs not the initial black-box f-approximation algorithm, but the algorithm of the first contribution, i.e. the algorithm of Theorem 2. Thus, the output of each run is an (s_i, t_i)-path of length $\delta(H_i - \{s_i, t_i\}) + \Omega(f(k_i))$ in H_i, where $\delta(H_i - \{s_i, t_i\}) + k_i$ is the length of the longest (s_i, t_i)-path in H_i.

Finally, from each approximation, our algorithm constructs a cycle of length at least $2\delta(G) + \Omega(f(k_i))$. It is guaranteed that $k_i = \Omega(k)$ for at least one i, so the longest of all constructed cycles is of length at least $2\delta(G) + \Omega(f(k))$. The following theorem is in order.

▶ **Theorem 1.** *Let $f : \mathbb{R}_+ \to \mathbb{R}_+$ be a non-decreasing subadditive function and suppose that we are given a polynomial-time algorithm finding a cycle of length at least $f(L)$ in graphs with the longest cycle length L. Then there exists a polynomial time algorithm that finds a cycle of length at least $2\delta(G) + \Omega(f(L - 2\delta(G)))$ in a 2-connected graph G with $\delta(G) \leq \frac{1}{2}|V(G)|$ and the longest cycle length L.*

One may note that Theorem 2 actually follows from Theorem 1 (again, by Menger's theorem, see Lemma 4). However, as described above, the algorithm in Theorem 1 employs the algorithm of Theorem 2, so we have to prove the latter before the former.

In the remaining part of this section, we provide more detailed proof overviews of both theorems, in particular, we explain how the algorithms employ the structural results of [24]. In both proofs, we complement these results by showing useful properties of specific graph decompositions. For clarity, we start with Theorem 1, as its proof is less involved.

2.1 Approximating long cycles

The basis of our algorithm is the structural result due to Fomin et al. [24]. In that work, the authors show the following: There is an algorithm that, given a cycle in a 2-connected graph, either finds a longer cycle or finds that G is of a "particular structure". This algorithm can be applied to any cycle of length less than $(2 + \sigma_1) \cdot \delta(G)$ (to be specific, we use $\sigma_1 = \frac{1}{24}$, see [26] for details).

To see how this stuctural result is important, recall that we aim to find a cycle of length at least $2\delta(G) + \Omega(f(k))$ in a 2-connected graph G with the longest cycle length $2\delta(G) + k$. Our algorithm simply starts with some cycle in G and applies the result of [24] to enlarge it exhaustively. It stops when either a cycle is of length at least $(2+\sigma_1)\cdot\delta(G)$, or the particular structure of G is found.

The crucial observation here is that if a long cycle is found, we can trivially find a good approximation. If $\sigma_1 \cdot \delta(G)$ is, e.g., less than $\sigma_1/10 \cdot f(k)$, then $10\delta(G) < f(k)$. If we just apply the blackbox f-approximation algorithm for the Longest Cycle problem, we get a cycle of length at least $f(2\delta(G) + k) \geq f(k) \geq 2\delta(G) + 4/5 \cdot f(k)$. Hence, by taking the longest of the cycles of length $(2 + \sigma_1) \cdot f(k)$ and of length $f(2\delta(G) + k)$ we always achieve a good approximation guarantee on k.

The most important part of the algorithm is employed when the "particular structure" outcome is received from the structural lemma applied on G and the current cycle C. Here we need to be specific about this structure, and the outcome can be of two types. The first outcome is a bounded vertex cover of the graph. This vertex cover is of size at most $\delta(G) + 2(k' + 1)$, where $k' \geq 0$ is such that $|V(C)| = 2\delta(G) + k'$. Such vertex cover is a guarantee that C is not much shorter than the longest cycle in G: the length of the longest cycle is bounded by twice the vertex cover size, so $k \leq 4(k' + 1)$. Hence, $k' = \Omega(k)$ and C is a sufficient approximation.

The second, and last, structural outcome is the Dirac decomposition, defined in [24]. Basically, this decomposition is obtained by finding a small separator of G (that consists of just two subpaths P_1, P_2 of the cycle C), and the parts of this decomposition are the connected components of G after the separation. The main result on Dirac decomposition proved in [24] is that there always exists a longest cycle that contains an edge in at least one of these parts.

While the definition and properties of Dirac decomposition may seem quite involved, our algorithm does not even require the Dirac decomposition of G to be found. In fact, we show a new nice property of Dirac decomposition. It guarantees that if a Dirac decomposition for G exists, then there also exists a 2-vertex separator $\{u, v\}$ of G that also divides the longest cycle in G into almost even parts. Our contribution is formulated in the following lemma.

▶ **Lemma 3.** *Let G be a 2-connected graph and P_1, P_2 induce a Dirac decomposition for a cycle C of length at most $2\delta(G) + \kappa$ in G such that $2\kappa \leq \delta(G)$. If there exists a cycle of length at least $2\delta(G) + k$ in G, then there exist $u, v \in V(G)$ such that*

- *$G - \{u, v\}$ is not connected, and*
- *there is an (u, v)-path of length at least $\delta(G) + (k - 2)/4$ in G.*

Our algorithm employs Lemma 3 in the following way. Since there are $\mathcal{O}(|V(G)|^2)$ vertex pairs in G, our algorithm iterates over all vertex pairs. If a pair u, v separates the graph into at least two parts, then our algorithm finds a long (u, v)-path that contains vertices in only one of the parts. Formally, it iterates over all connected components in $G - \{u, v\}$. For a fixed connected component H, our algorithm applies the algorithm of Theorem 2 to the graph $G[V(H) \cup \{u, v\}] + uv$ (the edge uv is added to ensure 2-connectivity), to find an approximation of the longest (u, v)-path. By Lemma 3, if u, v is the required separating pair, then for at least one H the length of the found (u, v)-path should be $\delta(G) + \Omega(k)$. And if such a path is found, a sufficiently long (u, v)-path outside H in G is guaranteed by Erdős-Gallai theorem. Together, these two paths form the required cycle of length $2\delta(G) + \Omega(k)$.

With that, the proof overview of Theorem 1 is finished. The formal proof is presented in [26].

2.2 Approximating long (s, t)-paths

While the algorithm of Theorem 1 does not use the underlying Dirac decomposition explicitly, in the case of finding (s, t)-paths (and to prove Theorem 2), we require deeper usage of the obtained graph decomposition. While the Dirac decomposition of Fomin et al. was originally used in [24] to find long cycles above $2\delta(G)$, for finding (s, t)-paths above $\delta(G - \{s, t\})$ the authors introduced the Erdős-Gallai decomposition.

In the formal proof of Theorem 2 in Section 4, we give a complete definition of Erdős-Gallai decomposition. In this overview, we aim to avoid the most technical details in order to provide an intuition of the structure of the decomposition and how our algorithm employs it.

Similarly to Dirac decomposition, the Erdős-Gallai decomposition is obtained through the routine that, given a graph G and an (s,t)-path inside it, either enlarges the path or reports that two subpaths P_1 (that starts with s) and P_2 (that starts with t) of the given path induce (when deleted) an Erdős-Gallai decomposition in G. This routine can be applied to an (s,t)-path until it reaches $(1+\sigma_2)\cdot\delta(G-\{s,t\})$ in length (specifically, $\sigma_2=\frac{1}{4}$, see Lemma 13; in this overview, we also skip the case of a Hamiltonian (s,t)-path for brevity). Note that, in contrast to the cycle enlargement routine of the Dirac decomposition, here the bounded vertex cover outcome is not possible. Similarly to the algorithm of the previous subsection, the only non-trivial part of the algorithm is dealing with the Erdős-Gallai decomposition outcome. In the other case, a single run of the black-box f-approximation algorithm for Longest Cycle provides the desired approximation immediately.

The main property of this decomposition due to [24] is as follows: If an (s,t)-path of length at least $\delta(G-\{s,t\})+k$ exists in G, then there necessarily exists the path of length at least $\delta(G-\{s,t\})+k$ that goes through one of the connected components in the decomposition. Moreover, for each of the connected components G_i there is exactly one pair of distinct entrypoints s_i,t_i: if an (s,t)-path in G goes through G_i, it should necessary enter G_i in s_i (or t_i) once and leave G_i in t_i (or s_i) exactly once as well.

Additionally to that, we have that the degree of each G_i is not much different from G: $\delta(G_i-\{s_i,t_i\})\geq\delta(G-\{s,t\})-2$ holds true. And this constant difference is always compensated by paths from s and t to s_i and t_i: if we succeed to find an (s_i,t_i)-path of length at least $\delta(G_i-\{s_i,t_i\})+k_i$ inside G_i, we can always complete it with *any* pair of disjoint paths from $\{s,t\}$ to $\{s_i,t_i\}$ into an (s,t)-path of length $\delta(G-\{s,t\})+k_i$ in G. Should this pair be longer than the trivial lower bound of 2, it grants the additional length above $\delta(G-\{s,t\})+k_i$.

The previous paragraph suggests the following approach for our approximation algorithm: for each G_i,s_i,t_i, our algorithm applies itself recursively to find an (s_i,t_i)-path of length $\delta(G_i-\{s_i,t_i\}+\Omega(f(k_i))$, where k_i comes from the longest (s_i,t_i)-path length in G_i. Since the other part of the additional length comes from two disjoint paths between $\{s,t\}$, and $\{s_i,t_i\}$, we would like to employ the black-box f-approximation algorithm to find the f-approximation of this pair of paths.

Unfortunately, finding such pair of paths reduces only to finding a long cycle through a given pair of vertices (it is enough to glue s with t and s_i with t_i in G, and ask to find the long cycle through the resulting pair of vertices). In their work, Fomin et al. have shown that the problem of finding such a cycle of length at least k can be done in $2^{\mathcal{O}(k)}\cdot n^{\mathcal{O}(1)}$ time. However, this is of little use to us, as k is only bounded by $\mathcal{O}(\delta(G))$, but we require polynomial time. Simultaneously, we do not know of any way to force the black-box algorithm to find an f-approximation for a cycle through the given pair of vertices.

These arguments bring us away from the idea of a recursive approximation algorithm. Instead, our approximation algorithm will apply the black-box algorithm to a single "complete-picture" graph that is obtained according to the structure brought by the Erdős-Gallai decomposition. However, the recursion here remains in the sense that we apply the path-enlarging routine to each component of the decomposition. This brings us to the idea of the recursive decomposition, which we define as the *nested Erdős-Gallai decomposition* in Section 4. This decomposition can be seen as a tree, where the root is the initial triple (G,s,t), the children of a node represent the triples (G_i,s_i,t_i) given by the Erdős-Gallai decomposition, and the leaves of this decomposition are the graphs G_i where sufficient approximations of long (s_i,t_i)-paths are found (by taking the longest of $(1+\sigma_2)\cdot\delta(G-\{s_i,t_i\})$-long path from the enlarging routine and the approximation obtained from the blackbox algorithm). A schematic picture of this novel decomposition is present in Figure 1.

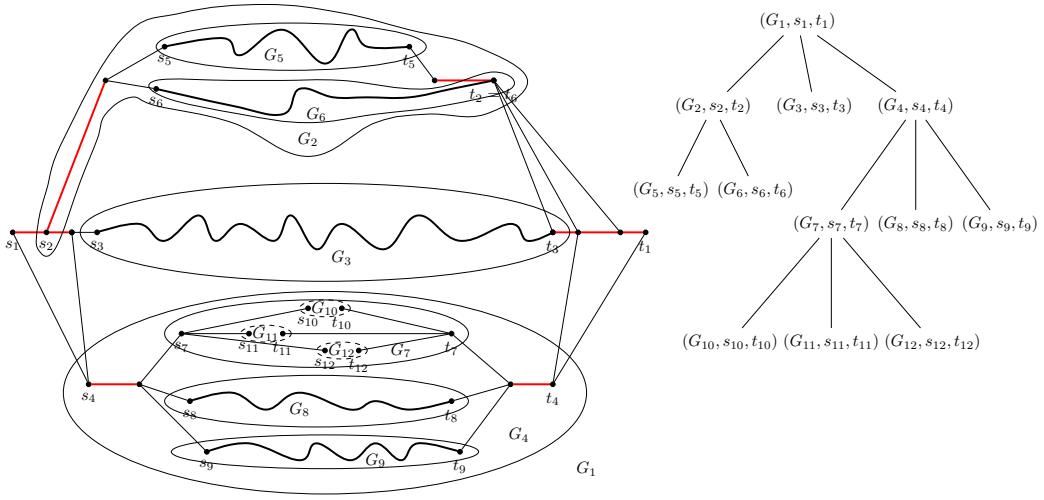


Figure 1 A schematic example of a nested Erdős-Gallai decomposition (left) and the corresponding recursion tree (right). Red straight paths inside G_i denote the pair of paths inducing an Erdős-Gallai decomposition in G_i. Bold (s_i, t_i)-paths are sufficient approximations of the longest (s_i, t_i)-paths in G_i. Dashed contours correspond to G_i with constant $\delta(G_i - \{s_i, t_i\})$, which is one of a few technical cases in the proof.

In Section 4, we show that a long path found inside a leaf (G_i, s_i, t_i) of the decomposition can be contracted into a single edge $s_i t_i$. Moreover, if (G_j, s_j, t_j) is a child of a (G_i, s_i, t_i) in the decomposition, and the longest pair of paths from $\{s_i, t_i\}$ to $\{s_j, t_j\}$ is just a pair of edges (so it does not grant any additional length as described before), we contract these edges. The crucial in our proof is the claim that after such a contraction, if an (s, t)-path of length $\delta(G - \{s, t\}) + k$ exists in the initial graph, an (s, t)-path of length at least $\Omega(k)$ exists in the graph obtained with described contractions. After doing all the contractions, the algorithm applies the black-box algorithm to the transformed graph and finds an (s, t)-path of length $f(\Omega(k))$ (which is $\Omega(f(k))$ by subadditivity) inside it.

The final part of our algorithm (and the proof of Theorem 2) is the routine that *transforms* this (s, t)-path inside the *contracted* graph G into a path of length $\delta(G - \{s, t\}) + \Omega(f(k))$ in the *initial* graph G. In this part, we prove that it is always possible to transform an (s, t)-path of length r in the contracted graph into a path of length $\Omega(r)$ that goes through at least one edge corresponding to a leaf of the nested Erdős-Gallai decomposition (hence, to a good approximation of (s_i, t_i)-path inside G_i). Finally, we observe that reversing the contractions in G transforms this path into the required approximation.

This finishes the overview of the proof of Theorem 2. Section 4 outlines the proof in detail, providing the sequence of intermediate technical results leading to the proof of the theorem.

3 Preliminaries

In this section, we define the notation used throughout the paper and provide some auxiliary results. We use $[n]$ to denote the set of positive integers $\{1, \ldots, n\}$. We remind that a function $f\colon D \to \mathbb{R}$ is *subadditive* if $f(x + y) \leq f(x) + f(y)$ for all $x, y \in D \subseteq \mathbb{R}$. We denote the set of all nonnegative real numbers by $\mathbb{R}_+$.

Recall that our main theorems are stated for arbitrary nondecreasing subadditive functions $f : \mathbb{R}_+ \to \mathbb{R}$, such that an algorithm achieving the respective approximation exists. Throughout the proofs we will, additionally assume that $f(x) \leq x$ for every $x \in \mathbb{R}_+$. For any integer $x \geq 3$, this is already implied by the statement, since a consistent approximation algorithm cannot output an (s,t)-path (respectively, cycle) of length greater than x in a graph where the longest (s,t)-path (respectively, cycle) has length x. However, for a general function $f(\cdot)$ this does not necessarily hold on the whole $\mathbb{R}_+$. If this is the case, for clarity of the proofs we redefine $f(x) := \min\{x, f(x)\}$ for every $x \in \mathbb{R}_+$. Clearly, f remains subadditive and non-decreasing, while also imposing exactly the same guarantee on the approximation algorithm.

Graphs. We consider only finite simple undirected graphs and use the standard notation (see, e.g., the book of Diestel [16]). The following useful observation follows immediately from Menger's theorem (see, e.g., [16, 42]).

▶ **Lemma 4.** *For any 2-connected graph G with a cycle of length L, there is a path of length at least $L/2$ between any pair of vertices in G. Moreover, given a cycle C and two distinct vertices s and t, an (s,t)-path of length at least $|V(C)|/2$ can be constructed in polynomial time.*

We observe that given an approximation algorithm for a longest cycle, we can use it as a black box to approximate a longest path between any two vertices.

▶ **Lemma 5.** *Let $\mathcal{A}$ be a polynomial-time algorithm that finds a cycle of length at least $f(L)$ in a graph with the longest cycle length L. Then there is a polynomial-time algorithm using $\mathcal{A}$ as a subroutine that, given a graph G and two distinct vertices s and t, finds an (s,t)-path of length at least $\frac{1}{2}f(2L)$, where L is the length of a longest (s,t)-path in G.*

We will use as a subroutine an algorithm finding two disjoint paths between two pairs of vertices of total length at least k, where k is the given parameter. For us, constant values of k suffice, though in fact there exists an FPT algorithm for this problem parameterized by the total length. It follows as an easy corollary from the following result of [24] about Long $(s,\ t)$-Cycle, the problem of finding a cycle of length at least k through the given two vertices s and t.

▶ **Theorem 6** (Theorem 4 in [24]). *There exists an FPT algorithm for Long $(s,\ t)$-Cycle parameterized by k.*

For completeness, we state the corollary next.

▶ **Corollary 7.** *There is an FPT algorithm that, given a graph G with two pairs of vertices $\{s,t\}$ and $\{s',t'\}$, and a parameter k, finds two disjoint paths between $\{s,t\}$ and $\{s',t'\}$ in G of total length at least k, or correctly determines that such paths do not exist.*

Finally, it is convenient to use the following corollary, which generalizes the theorem of Erdős and Gallai [19, Theorem 1.16].

▶ **Corollary 8** (Corollary 3 in [24]). *Let G be a 2-connected graph and let s,t be a pair of distinct vertices in G. For any $B \subseteq V(G)$ there exists a path of length at least $\delta(G-B)$ between s and t in G. Moreover, there is a polynomial time algorithm constructing a path of such length.*

4 Approximating (s,t)-path

In this section, we outline the proof of Theorem 2, stating that any guarantee for approximating the longest cycle in a 2-connected graph can be transferred to approximating the longest (s,t)-path above minimum degree. For the convenience of the reader, we recall the precise statement next.

▶ **Theorem 2.** *Let $f : \mathbb{R}_+ \to \mathbb{R}_+$ be a non-decreasing subadditive function and suppose that we are given a polynomial-time algorithm computing an (s,t)-path of length at least $f(L)$ in graphs with given two vertices s and t having the longest (s,t)-path of length L. Then there is a polynomial-time algorithm that outputs an (s,t)-path of length at least $\delta(G-\{s,t\}) + \Omega(f(L-\delta(G-\{s,t\})))$ in a 2-connected graph G with two given vertices s and t having the longest (s,t)-path length L.*

In order to obtain this result, we first recall the concept of Erdős-Gallai decomposition introduced in [24] together with a few of its helpful properties established there. Then we introduce the recursive generalization of this concept, called nested Erdős-Gallai decomposition, and show how to obtain with its help the compression of the graph such that a long (s,t)-path in the compressed graph can be lifted to an (s,t)-path in the original graph with a large offset.

4.1 Erdős-Gallai decomposition

This subsection encompasses the properties of an Erdős-Gallai decomposition, defined next. The definition itself and most of the technical results presented here are due to [24]. Some of the results from [24] need to be modified in order to be used for our purposes, we provide the proofs of such results in [26]. Note that the statements in [24] hold in the more general case where there is also a low-degree vertex subset in the graph, here while recalling the results we automatically simplify the statements. Next, we recall the definition of an Erdős-Gallai decomposition.

▶ **Definition 9** (Erdős-Gallai decomposition and Erdős-Gallai component, Definition 2 in [24])**.** *Let P be a path in a 2-connected graph G. We say that two disjoint paths P_1 and P_2 in G induce* an Erdős-Gallai decomposition for P *in G if*

- *Path P is of the form $P = P_1P'P_2$, where the inner path P' has at least $\delta(G-\{s,t\})$ edges.*
- *There are at least two connected components in $G - V(P_1 \cup P_2)$, and for every connected component H, it holds that $|V(H)| \geq 3$ and one of the following is fulfilled:*
  1. *H is 2-connected and the maximum size of a matching in G between $V(H)$ and $V(P_1)$ is one, and between $V(H)$ and $V(P_2)$ is also one;*
  2. *H is not 2-connected, exactly one vertex of P_1 has neighbors in H, that is $|N_G(V(H)) \cap V(P_1)| = 1$, and no inner vertex from a leaf-block of H has a neighbor in P_2;*
  3. *The same as 2, but with P_1 and P_2 interchanged. That is, H is not 2-connected, $|N_G(V(H)) \cap V(P_2)| = 1$, and no inner vertex from a leaf-block of H has a neighbor in P_1.*

The set of Erdős-Gallai component*s for an Erdős-Gallai decomposition is defined as follows. First, for each component H of type 1, H is an Erdős-Gallai component of the Erdős-Gallai decomposition. Second, for each H of type 2, or of type 3, all its leaf-blocks are also Erdős-Gallai components of the Erdős-Gallai decomposition.*

As long as an Erdős-Gallai decomposition is available, Erdős-Gallai components allow us to bound the structure of optimal solutions in a number of ways. First, Fomin et al. [24] observe that the longest (s,t)-path necessarily visits an Erdős-Gallai component.

▶ **Lemma 10** (Lemma 7 in [24]). *Let G be a graph and P_1, P_2 induce an Erdős-Gallai decomposition for an (s,t)-path P in G. Then there is a longest (s,t)-path in G that enters an Erdős-Gallai component.*

Next, since an Erdős-Gallai component has a very restrictive connection to the rest of the graph, it follows that any (s,t)-path has only one chance of entering the component.

▶ **Lemma 11** (Lemma 5 in [24]). *Let G be a 2-connected graph and P be an (s,t)-path in G. Let paths P_1, P_2 induce an Erdős-Gallai decomposition for P in G. Let M be an Erdős-Gallai component. Then for every (s,t)-path P' in G, if P' enters M, then all vertices of $V(M) \cap V(P')$ appear consecutively in P'.*

For the purposes of recursion, it is convenient to enclose an Erdős-Gallai component together with some of its immediate connections, so that this slightly larger subgraph behaves exactly like an (s,t)-path instance. The subgraph K in the next lemma plays this role.

▶ **Lemma 12** (Lemma 8 in [24]). *Let paths P_1, P_2 induce an Erdős-Gallai decomposition for an (s,t)-path P in graph G. Let M be an Erdős-Gallai component in G. Then there is a polynomial time algorithm that outputs a 2-connected subgraph K of G and two vertices $s', t' \in V(K)$, such for that every (s,t)-path P' in G that enters M, the following hold:*

1. *$V(K) = (V(M) \cup \{s', t'\})$;*
2. *$P'[V(K)]$ is an (s', t')-subpath of P' and an (s', t')-path in K;*
3. *$\delta(K - \{s', t'\}) \geq \delta(G - \{s, t, s', t'\})$.*

Most importantly, Erdős-Gallai decompositions capture extremal situations, where the current (s,t)-path cannot be made longer in a "simple" way. The next lemma formalizes that intuition, stating that in polynomial time we can find either a long (s,t)-path or an Erdős-Gallai decomposition. The lemma is largely an analog of Lemma 4 in [24], however our statement here is slightly modified. Next, we recall the statement from Section 2.

▶ **Lemma 13.** *Let G be a 2-connected graph such that $\delta(G - \{s,t\}) \geq 16$. There is a polynomial time algorithm that*

- *either outputs an (s,t)-path P of length at least $\min\{\frac{5}{4}\delta(G - \{s,t\}) - 3, |V(G)| - 1\}$,*
- *or outputs an (s,t)-path P with paths P_1, P_2 that induce an Erdős-Gallai decomposition for P in G. Additionally, there is no (s,t)-path in G that enters at least two Erdős-Gallai components of this Erdős-Gallai decomposition.*

Finally, to deal with (s,t)-paths that do not enter any Erdős-Gallai component, one can observe the following. Intuitively, such a path should be far from optimal, as going through an Erdős-Gallai component would immediately give at least $\delta(G - \{s,t\}) - \mathcal{O}(1)$ additional edges of the path. The final lemma of this subsection establishes how precisely the length of a path avoiding Erdős-Gallai components can be "boosted" in this fashion. To obtain this result, we first need a technical lemma from [24] that yields long paths inside separable components.

▶ **Lemma 14** (Lemma 6 in [24]). *Let H be a connected graph with at least one cut-vertex. Let I be the set of inner vertices of all leaf-blocks of H. Let $S \subseteq V(H) \setminus I$ separate at least one vertex in $V(H) \setminus I$ from I in H. For any vertex v that is not an inner vertex of a leaf-block of H, there is a cut-vertex c of a leaf-block of H and a (c,v)-path of length at least $\frac{1}{2}(\delta(H) - |S|)$ in H. This path can be constructed in polynomial time.*

Now we move to (s,t)-paths that avoid Erdős-Gallai components. The following Lemma 15 has been already stated in Section 2, here we recall the statement.

▶ **Lemma 15.** *Let P be an (s,t)-path of length at most $\delta(G-\{s,t\})+k$ and let two paths P_1, P_2 induce a Erdős-Gallai decomposition for P in G. There is a polynomial time algorithm that, given an (s,t)-path of length at least $4k+5$ in G that does not enter any Erdős-Gallai component, outputs a path of length at least $\min\{\delta(G-\{s,t\})+k-1, \frac{3}{2}\delta(G-\{s,t\})-\frac{5}{2}k-1\}$ in G.*

4.2 Proof of Theorem 2

To deal with the recursive structure of the solution, we introduce the following *nested* generalization of an Erdős-Gallai decomposition. Intuitively, it captures how the structural observations of the previous subsection allow us to recursively construct Erdős-Gallai decompositions with the aim of finding a long (s,t)-path. For an illustration of a nested Erdős-Gallai decomposition, see Figure 1. We recall the formal definition from Section 2.

▶ **Definition 16** (Nested Erdős-Gallai decomposition)**.** *A sequence of triples (G_1,s_1,t_1), (G_2,s_2,t_2), …, (G_ℓ,s_ℓ,t_ℓ) is called a* nested Erdős-Gallai decomposition *for G and two vertices $s,t \in V(G)$ if*

- $(G_1,s_1,t_1)=(G,s,t)$*;*
- *for each $i \in [\ell]$, either*
  - $\delta(G_i-\{s_i,t_i\})<16$*, or*
  - *Lemma 13 applied to G_i,s_i,t_i gives a path P_i of length at least $\min\{\frac{5}{4}\delta(G_i-\{s_i,t_i\})-3, |V(G_i)|-1\}$ in G_i, or*
  - *Lemma 13 applied to G_i,s_i,t_i gives a path P_i and two paths $P_{i,1},P_{i,2}$ that induce an Erdős-Gallai decomposition for P_i in G_i, and for each Erdős-Gallai component M of this decomposition there is $j>i$ such that (G_j,s_j,t_j) is the result of Lemma 12 applied to M in G_i. In this case, we say that G_i is* decomposed*.*
- *for each $i \in \{2,\dots,\ell\}$, there is $e(i)<i$ such that (G_i,s_i,t_i) is a result of Lemma 12 applied to some Erdős-Gallai component of the Erdős-Gallai decomposition of $G_{e(i)}$ for $P_{e(i)}$.*

The proof of Theorem 2 is performed in two steps: first, we show how to obtain a nested Erdős-Gallai decomposition for a given graph G, and then we use the nested Erdős-Gallai decomposition to recursively construct a good approximation to the longest (s,t)-path. The first part is achieved simply by applying Lemma 13 recursively on each Erdős-Gallai component until components are no longer decomposable. The main hurdle is the second part, on which we focus for the rest of the section. For completeness, first we show that a nested Erdős-Gallai decomposition can always be constructed in polynomial time.

▶ **Lemma 17.** *There is a polynomial time algorithm that, given a 2-connected graph G and its two vertices s and t, outputs a nested Erdős-Gallai decomposition for G, s, t.*

Clearly, it follows that the size of a nested Erdős-Gallai decomposition returned by Lemma 17 is also polynomial. Observe also that the construction algorithm invokes Lemma 13 for all sufficiently large G_i, thus in what follows we assume that the corresponding paths P_i are already computed.

Now we focus on using a constructed nested Erdős-Gallai decomposition for approximating the longest (s,t)-path. First of all, we present the algorithm `long_nested_st_path` that, given a nested Erdős-Gallai decomposition of G, computes a long (s,t)-path by going over

the decomposition. The pseudocode of `long_nested_st_path` is present in Algorithm 3. Intuitively, first the algorithm computes a compression H of the graph G that respects the nested Erdős-Gallai decomposition: components that are not decomposed are replaced by single edges, and edges that are "unavoidable" to visit a component are contracted. The computation of this compression is encapsulated in the `nested_compress` function presented in Algorithm 1. As a subroutine, this function uses the `two_long_disjoint_paths` algorithm given by Corollary 7, that finds two disjoint paths of at least the given length between the given pairs of vertices.

Next, the blackbox approximation algorithm `long_st_path_approx` is used to compute an (s,t)-path Q in H. The function `nested_decompress` reconstructs then this path in the original graph G, see Algorithm 2 for the pseudocode. Later we argue (Lemma 19) that any (s,t)-path in H of length r yields in this way an (s,t)-path in G of length at least $\delta(G-\{s,t\})+r/8-3$. Finally, either the length of Q in H was large enough and the reconstructed path provides the desired approximation or a long path can be found inside one of the components in a "simple" way, and then connected arbitrarily to $\{s,t\}$. Specifically, in this component, it suffices to either take an approximation of the longest path computed by `long_st_path_approx`, or a long Erdős–Gallai path returned by the algorithm from Corollary 8, `long_eg_st_path`. Thus, in the final few lines `long_nested_st_path` checks whether any of these paths is longer than the reconstructed path Q. The path from inside the component is extended to an $\{s,t\}$-path in G by using the algorithm `two_long_disjoint_paths`, given by Corollary 7, with the parameter 0.

Algorithm 1 The algorithm compressing a given graph G with a given nested Erdős-Gallai decomposition.

`nested_compress`$((G_1,s_1,t_1),(G_2,s_2,t_2),\ldots,(G_\ell,s_\ell,t_\ell))$
Input: a nested Erdős-Gallai decomposition for G, s and t.
Output: the compressed graph H.
$H \longleftarrow G$;
foreach $i \in \{2,\ldots,\ell\}$ **do**
  $j \longleftarrow e(i)$;
  $d_i \longleftarrow |\{s_j,t_j\} \setminus \{s_i,t_i\}|$;
  if `two_long_disjoint_paths` $(G_i,\{s_j,t_j\},\{s_i,t_i\},d_i+1)$ *is* NO **then**
    contract all edges of a maximum matching between $\{s_j,t_j\}$ and $\{s_i,t_i\}$ in H;
  end
  if G_i *is not decomposed* **then**
    remove all vertices in $V(G_i) \setminus \{s_i,t_i\}$ from H;
    add edge s_it_i to H and mark it with G_i;
  end
end
return H;

Now, our goal is to show that the path that the `long_nested_st_path` algorithm constructs serves indeed as the desired approximation of the longest (s,t)-path in G. For the rest of this section, let $G_1,\ldots,G_\ell$ be the given nested Erdős-Gallai decomposition for G,s,t. An important piece of intuition about nested Erdős-Gallai decomposition is that, as we go deeper into the nested Erdős-Gallai components, the minimum degree of the component $\delta(G_i \setminus \{s_i,t_i\})$ decreases, but we gain more and more edges that we collect while going

Algorithm 2 The algorithm decompressing a path in H into a long path in G.

`nested_decompress`$((G_1, s_1, t_1), (G_2, s_2, t_2), \ldots, (G_\ell, s_\ell, t_\ell), H, Q)$

Input: a nested Erdős-Gallai decomposition for G, s and t, the compressed graph H and an (s,t)-path Q in H of length r.

Output: an (s,t)-path of length at least $\delta(G - \{s,t\}) + r/8 - 3$ in G.

2.1 **foreach** $i \in \{2, \ldots, \ell\}$ *such that* $d_i > 0$ *and* Q *enters* G_i **do**
2.2 $\quad j \longleftarrow e(i)$;
2.3 $\quad$ **if** *an edge between* $\{s_j, t_j\}$ *and* $\{s_i, t_i\}$ *was contracted in* H **then**
2.4 $\quad\quad$ replace s_i and/or t_i in Q with the respective contracted edges;
2.5 $\quad$ **else**
2.6 $\quad\quad$ $S_1, S_2 \longleftarrow$ `two_long_disjoint_paths`$(G, \{s_j, t_j\}, \{s_i, t_i\}, d_i + 1)$;
2.7 $\quad\quad$ replace the two subpaths of Q going from $\{s_j, t_j\}$ to $\{s_i, t_i\}$ with S_1 and S_2 if the length of Q increases;
2.8 $\quad$ **end**
2.9 **end**
2.10 $h \longleftarrow$ largest $h \in [\ell]$ such that Q enters G_h;
2.11 **if** G_h *is not decomposed* **then**
2.12 $\quad$ replace $s_h t_h$ in Q with P_h;
2.13 **else**
2.14 $\quad k' \longleftarrow \lfloor (|E(Q) \cap E(G_h)| - 5)/8 \rfloor$;
2.15 $\quad$ **if** $|E(P_h)| \geq \delta(G_h - \{s_h, t_h\}) + k'$ **then**
2.16 $\quad\quad R \longleftarrow P_h$;
2.17 $\quad$ **else**
2.18 $\quad\quad R \longleftarrow$ result of Lemma 15 applied to G_h, P_h and the (s_h, t_h)-subpath of Q;
2.19 $\quad$ **end**
2.20 $\quad$ **if** (s_h, t_h)*-subpath of* Q *is shorter than* R **then**
2.21 $\quad\quad$ replace the (s_h, t_h)-subpath of Q with R;
2.22 $\quad$ **end**
2.23 **end**
2.24 **return** Q;

from $\{s,t\}$ to $\{s_i, t_i\}$. We introduce values that help us measure this difference between the nested components: for each $i \in [\ell]$, denote $d_i = |\{s_{e(i)}, t_{e(i)}\} \setminus \{s_i, t_i\}|$. In particular, by Lemma 12 we know that for any $i \in [\ell]$, $\delta(G_i) \geq \delta(G_{e(i)}) - d_i$. On the other hand, any pair of disjoint paths that connects $\{s_{e(i)}, t_{e(i)}\}$ to $\{s_i, t_i\}$ contains at least d_i edges. This leads to the following simple observation about extending an (s_j, t_j)-path in a component G_j to an (s,t)-path in G.

▷ Claim 18. For each $j \in [\ell]$, let $G_{j_1}, \ldots, G_{j_c}$ be such that $j_c = j$ and $j_1 = 1$ and $e(j_{i+1}) = j_i$ for each $i \in [c-1]$. Let P be an (s_j, t_j)-path in G_j. Then P combined with any pair of disjoint paths connecting $\{s,t\}$ to $\{s_j, t_j\}$ yields an (s,t)-path in G of length at least $|E(P)| + \sum_{i \in [c-1]} d_{j_{i+1}}$.

However, there might also exist longer paths connecting nested components $G_{e(i)}$ and G_i. When we construct the compressed graph H in Algorithm 1, we distinguish between two cases. Either any pair of such paths have the total length d_i, meaning that the only option is to use the edges of a matching between $\{s_{e(i)}, t_{e(i)}\}$ and $\{s_i, t_i\}$. In that case we simply contract these edges as we know that there is no choice on how to reach G_i from $G_{e(i)}$. Or, there is a pair of disjoint paths of total length at least $d_i + 1$. This situation is

Algorithm 3 The algorithm finding a long (s,t)-path in a 2-connected graph with a given nested Erdős-Gallai decomposition.

`long_nested_st_path`$((G_1,s_1,t_1),(G_2,s_2,t_2),\dots,(G_\ell,s_\ell,t_\ell))$
Input: a nested Erdős-Gallai decomposition for G, s and t.
Output: an (s,t)-path of length at least $\delta(G-\{s,t\})+f(k)/32-3$ in G where $k=L-\delta(G-\{s,t\})$ for the longest (s,t)-path length L in G.
3.1 $H \longleftarrow$ `nested_compress`$((G_1,s_1,t_1),(G_2,s_2,t_2),\dots,(G_\ell,s_\ell,t_\ell))$;
3.2 $Q \longleftarrow$ `long_st_path_approx`(H,s,t);
3.3 $Q \longleftarrow$ `nested_decompress`$((G_1,s_1,t_1),(G_2,s_2,t_2),\dots,(G_\ell,s_\ell,t_\ell),H,Q)$;
3.4 **foreach** $i \in [\ell]$ **do**
3.5 $P_i \longleftarrow$ the longest of $\{$`long_st_path_approx`(G_i,s_i,t_i), `long_eg_st_path`$(G_i,s_i,t_i)\}$;
3.6 $Q \longleftarrow$ the longest of $\{Q,$ `two_long_disjoint_paths`$(G,\{s,t\},\{s_i,t_i\},0)\cup P_i\}$;
3.7 **end**
3.8 **return** Q;

beneficial to us in a different way: since we can find such a pair of paths in polynomial time, we can traverse at least d_i+1 edges going from $G_{e(i)}$ to G_i, while we only lose at most d_i in the minimum degree. This dichotomy on the structure of the "slice" between two nested components is the main leverage that allows us to lift the length of an (s,t)-path in H to an offset above the minimum degree in G. We formally show this crucial property of the compressed graph H and the `nested_decompress` routine in the next lemma.

▶ **Lemma 19.** *The `nested_decompress` routine transforms an (s,t)-path Q in H of length r into an (s,t)-path in G of length at least $\delta(G-\{s,t\})+r/8-3$.*

It will also be helpful to observe that in the "slice" between a decomposed component and the nested components, at most two edges of any path can be contracted. Note that this does not follow immediately, as a pair of edges to *each* of the nested components is potentially contracted.

▷ Claim 20. Let Q be an (s_j,t_j)-path inside a decomposed graph G_j. Then all edges $E(Q)\cap E(G_j)\setminus\bigcup_{e(i)=j}E(G_i)$ are unchanged in H except for, possibly, contraction of the first and the last edge of Q.

Now we are ready to prove the main lemma that bounds the length of the (s,t)-path returned by Algorithm 3.

▶ **Lemma 21.** *`long_nested_st_path` outputs an (s,t)-path in G of length at least $\delta(G-\{s,t\})+f(k)/32-3$, where $k=L-\delta(G-\{s,t\})$ and L is the length of the longest (s,t)-path in G.*

Finally, observe that the running time of Algorithm 3 is polynomial in the size of the given nested Erdős-Gallai decomposition. By Lemma 17, its size is polynomial in the size of the input graph G. This concludes the proof of Theorem 2.

5 Conclusion

In this article, we have shown a general theorem that allows us to leverage all the algorithmic machinery for approximating the length of the longest cycle to approximate the "offset" of the longest cycle provided by the classical Dirac's theorem. As far as one can compute

a cycle of length $f(L)$ in a 2-connected graph G with the longest cycle length L, we can also construct a cycle of length $2\delta(G) + \Omega(f(L - 2\delta(G)))$. In particular, we can use the state-of-the-art approximation algorithm for Longest Cycle due to Gabow and Nie [31]. They achieve an algorithm finding a cycle of length $f(L) = c^{\sqrt{\log L}}$ for some constant $c > 1$ in a graph with the longest cycle length L. Note that f is non-decreasing and subadditive (as f is concave on $[1, +\infty]$, and any concave function is subadditive; we also can formally set $f(x) = \min\{x, c^{\sqrt{\log x}}\}$ for $x \geq 1$ and $f(x) = x$ for $x < 1$ to fit the statement of Theorem 1). By substituting this to Theorem 1, we achieve a polynomial-time algorithm that outputs a cycle of length $2\delta(G) + 2^{\Omega(\sqrt{\log(L-2\delta(G))})}$ in a 2-connected graph G with the longest cycle length $L > 2\delta(G)$.

In the field of parameterized algorithms, there are many results on computing longest cycles or paths above some guarantees. It is a natural question, whether approximation results similar to ours hold for other types of "offsets". To give a few concrete questions, recall that the *degeneracy* $\mathrm{dg}(G)$ of a graph G is the maximum d such that G has an induced subgraph of minimum degree d. By Erdős and Gallai [19], a graph of degeneracy $d \geq 2$ contains a cycle of length at least $d + 1$. It was shown by Fomin et al. in [22] that a cycle of length at least $L = \mathrm{dg}(G) + k$ in a 2-connected graph can be found in $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time. This immediately yields a polynomial-time algorithm for computing a cycle of length at least $\mathrm{dg}(G) + \Omega(\log(L - \mathrm{dg}(G)))$. Is there a better approximation of the longest cycle above the degeneracy?

Another concrete question. Bezáková et al. [4] gave an FPT algorithm that for $s, t \in V(G)$ finds a detour in an undirected graph G. In othere words, they gave an algorithm that finds an (s, t)-path of length at least $L = \mathrm{dist}_G(s, t) + k$ in $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time. Here $\mathrm{dist}_G(s, t)$ is the distance between s and t. Therefore, in undirected graph we can find an (s, t)-path of length $\mathrm{dist}_G(s, t) + \Omega(\log(L - \mathrm{dist}_G(s, t))$ in polynomial time. The existence of any better bound is open. For directed graphs, the question of whether finding a long detour is FPT is widely open [4]. Nothing is known about the (in)approximability of long detours in directed graphs.

References

1 Noga Alon, Gregory Gutin, Eun Jung Kim, Stefan Szeider, and Anders Yeo. Solving MAX-r-SAT above a tight lower bound. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 511–517. SIAM, 2010.

2 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.

3 Cristina Bazgan, Miklos Santha, and Zsolt Tuza. On the approximation of finding a(nother) hamiltonian cycle in cubic hamiltonian graphs. *J. Algorithms*, 31(1):249–268, 1999. doi:10.1006/jagm.1998.0998.

4 Ivona Bezáková, Radu Curticapean, Holger Dell, and Fedor V. Fomin. Finding detours is fixed-parameter tractable. *SIAM J. Discrete Math.*, 33(4):2326–2345, 2019. doi:10.1137/17M1148566.

5 Andreas Björklund. Determinant sums for undirected hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014. doi:10.1137/110839229.

6 Andreas Björklund and Thore Husfeldt. Finding a path of superlogarithmic length. *SIAM J. Comput.*, 32(6):1395–1402, 2003. doi:10.1137/S0097539702416761.

7 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *CoRR*, abs/1007.1161, 2010. arXiv:1007.1161.

8 Andreas Björklund, Thore Husfeldt, and Sanjeev Khanna. Approximating longest directed paths and cycles. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP)*, volume 3142 of *Lecture Notes in Comput. Sci.*, pages 222–233. Springer, 2004. doi:10.1007/978-3-540-27836-8_21.

9 Hans L. Bodlaender. On linear time minor tests with depth-first search. *J. Algorithms*, 14(1):1–23, 1993. doi:10.1006/jagm.1993.1001.

10 B. Bollobás and A. D. Scott. Better bounds for Max Cut. In *Contemporary combinatorics*, volume 10 of *Bolyai Soc. Math. Stud.*, pages 185–246. János Bolyai Math. Soc., Budapest, 2002.

11 Béla Bollobás. Extremal graph theory. In *Handbook of combinatorics, Vol. 1, 2*, pages 1231–1292. Elsevier Sci. B. V., Amsterdam, 1995.

12 J. A. Bondy. Basic graph theory: paths and circuits. In *Handbook of combinatorics, Vol. 1, 2*, pages 3–110. Elsevier Sci. B. V., Amsterdam, 1995.

13 Guantao Chen, Zhicheng Gao, Xingxing Yu, and Wenan Zang. Approximating longest cycles in graphs with bounded degrees. *SIAM Journal on Computing*, 36(3):635–656, 2006.

14 Robert Crowston, Mark Jones, Gabriele Muciaccia, Geevarghese Philip, Ashutosh Rai, and Saket Saurabh. Polynomial kernels for lambda-extendible properties parameterized above the Poljak-Turzik bound. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 24 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43–54, Dagstuhl, Germany, 2013. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

15 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

16 Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 5th edition, 2017.

17 G. A. Dirac. Some theorems on abstract graphs. *Proc. London Math. Soc. (3)*, 2:69–81, 1952.

18 C. S. Edwards. Some extremal properties of bipartite subgraphs. *Canad. J. Math.*, 3:475–485, 1973.

19 P. Erdős and T. Gallai. On maximal paths and circuits of graphs. *Acta Math. Acad. Sci. Hungar*, 10:337–356, 1959.

20 Tomás Feder and Rajeev Motwani. Finding large cycles in Hamiltonian graphs. *Discrete Appl. Math.*, 158(8):882–893, 2010. doi:10.1016/j.dam.2009.12.006.

21 Tomás Feder, Rajeev Motwani, and Carlos Subi. Approximating the longest cycle problem in sparse graphs. *SIAM J. Comput.*, 31(5):1596–1607, 2002. doi:10.1137/S0097539701395486.

22 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Going far from degeneracy. *SIAM J. Discrete Math.*, 34(3):1587–1601, 2020. doi:10.1137/19M1290577.

23 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Multiplicative parameterization above a guarantee. *ACM Trans. Comput. Theory*, 13(3):18:1–18:16, 2021. doi:10.1145/3460956.

24 Fedor V. Fomin, Petr A. Golovach, Danil Sagunov, and Kirill Simonov. Algorithmic extensions of Dirac's theorem. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 406–416, 2022. doi:10.1137/1.9781611977073.20.

25 Fedor V. Fomin, Petr A. Golovach, Danil Sagunov, and Kirill Simonov. Longest cycle above erdős-gallai bound. In *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, volume 244 of *LIPIcs*, pages 55:1–55:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.55.

26 Fedor V. Fomin, Petr A. Golovach, Danil Sagunov, and Kirill Simonov. Approximating long cycle above dirac's guarantee. *CoRR*, abs/2305.02011, 2023. arXiv:2305.02011.

27 Fedor V. Fomin and Petteri Kaski. Exact exponential algorithms. *Commun. ACM*, 56(3):80–88, 2013. doi:10.1145/2428556.2428575.

28 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016. doi:10.1145/2886094.

29 Martin Fürer and Balaji Raghavachari. Approximating the minimum-degree steiner tree to within one of optimal. *J. Algorithms*, 17(3):409–423, 1994. doi:10.1006/jagm.1994.1042.

30 Harold N. Gabow. Finding paths and cycles of superpolylogarithmic length. *SIAM J. Comput.*, 36(6):1648–1671, 2007. doi:10.1137/S0097539704445366.

31 Harold N. Gabow and Shuxin Nie. Finding a long directed cycle. *ACM Transactions on Algorithms*, 4(1), 2008. doi:10.1145/1328911.1328918.

32 Harold N. Gabow and Shuxin Nie. Finding long paths, cycles and circuits. In *Proceedings of the 19th International Symposium on Algorithms and Computation (ISAAC)*, volume 5369 of *Lecture Notes in Comput. Sci.*, pages 752–763. Springer, 2008. doi:10.1007/978-3-540-92182-0_66.

33 Shivam Garg and Geevarghese Philip. Raising the bar for vertex cover: Fixed-parameter tractability above a higher guarantee. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1152–1166. SIAM, 2016. doi:10.1137/1.9781611974331.ch80.

34 Gregory Gutin, Eun Jung Kim, Michael Lampis, and Valia Mitsou. Vertex cover problem parameterized above and below tight bounds. *Theory of Computing Systems*, 48(2):402–410, 2011. doi:10.1007/s00224-010-9262-y.

35 Gregory Gutin, Leo van Iersel, Matthias Mnich, and Anders Yeo. Every ternary permutation constraint satisfaction problem parameterized above average has a kernel with a quadratic number of variables. *J. Computer and System Sciences*, 78(1):151–163, 2012. doi:10.1016/j.jcss.2011.01.004.

36 Gregory Z. Gutin and Matthias Mnich. A survey on graph problems parameterized above and below guaranteed values. *CoRR*, abs/2207.12278, 2022. doi:10.48550/arXiv.2207.12278.

37 Gregory Z. Gutin and Viresh Patel. Parameterized traveling salesman problem: Beating the average. *SIAM J. Discrete Math.*, 30(1):220–238, 2016.

38 Gregory Z. Gutin, Arash Rafiey, Stefan Szeider, and Anders Yeo. The linear arrangement problem parameterized above guaranteed value. *Theory Comput. Syst.*, 41(3):521–538, 2007. doi:10.1007/s00224-007-1330-6.

39 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity. *J. Computer and System Sciences*, 63(4):512–530, 2001.

40 Bart M. P. Jansen, László Kozma, and Jesper Nederlof. Hamiltonicity below Dirac's condition. In *Proceedings of the 45th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 11789 of *Lecture Notes in Computer Science*, pages 27–39. Springer, 2019.

41 David R. Karger, Rajeev Motwani, and G. D. S. Ramkumar. On approximating the longest path in a graph. *Algorithmica*, 18(1):82–98, 1997. doi:10.1007/BF02523689.

42 Jon M. Kleinberg and Éva Tardos. *Algorithm design*. Addison-Wesley, 2006.

43 Ioannis Koutis. Faster algebraic algorithms for path and packing problems. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 5125 of *Lecture Notes in Comput. Sci.*, pages 575–586. Springer, 2008.

44 Ioannis Koutis and Ryan Williams. Algebraic fingerprints for faster algorithms. *Commun. ACM*, 59(1):98–105, 2016. doi:10.1145/2742544.

45 Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Trans. Algorithms*, 11(2):15:1–15:31, 2014. doi:10.1145/2566616.

46 Meena Mahajan, Venkatesh Raman, and Somnath Sikdar. Parameterizing above or below guaranteed values. *J. Computer and System Sciences*, 75(2):137–153, 2009. doi:10.1016/j.jcss.2008.08.004.

47 Sounaka Mishra, Venkatesh Raman, Saket Saurabh, Somnath Sikdar, and C. R. Subramanian. The complexity of König subgraph problems and above-guarantee vertex cover. *Algorithmica*, 61(4):857–881, 2011. doi:10.1007/s00453-010-9412-2.

48 B. Monien. How to find long paths efficiently. In *Analysis and design of algorithms for combinatorial problems (Udine, 1982)*, volume 109 of *North-Holland Math. Stud.*, pages 239–254. North-Holland, Amsterdam, 1985. doi:10.1016/S0304-0208(08)73110-4.

49 Sundar Vishwanathan. An approximation algorithm for finding long paths in hamiltonian graphs. *J. Algorithms*, 50(2):246–256, 2004. doi:10.1016/S0196-6774(03)00093-2.

50 Ryan Williams. Finding paths of length k in $O^*(2^k)$ time. *Inf. Process. Lett.*, 109(6):315–318, 2009.

Compound Logics for Modification Problems

Fedor V. Fomin
Department of Informatics, University of Bergen, Norway

Petr A. Golovach
Department of Informatics, University of Bergen, Norway

Ignasi Sau
LIRMM, Université de Montpellier, CNRS, France

Giannos Stamoulis
LIRMM, Université de Montpellier, CNRS, France

Dimitrios M. Thilikos
LIRMM, Université de Montpellier, CNRS, France

Abstract

We introduce a novel model-theoretic framework inspired from graph modification and based on the interplay between model theory and algorithmic graph minors. The core of our framework is a new *compound logic* operating with two types of sentences, expressing graph modification: the *modulator sentence*, defining some property of the modified part of the graph, and the *target sentence*, defining some property of the resulting graph. In our framework, modulator sentences are in counting monadic second-order logic (CMSOL) and have models of bounded treewidth, while target sentences express first-order logic (FOL) properties along with minor-exclusion. Our logic captures problems that are not definable in first-order logic and, moreover, may have instances of unbounded treewidth. Also, it permits the modeling of wide families of problems involving vertex/edge removals, alternative modulator measures (such as elimination distance or $\mathcal{G}$-treewidth), multistage modifications, and various cut problems. Our main result is that, for this compound logic, model-checking can be done in quadratic time. All derived algorithms are constructive and this, as a byproduct, extends the constructibility horizon of the algorithmic applications of the Graph Minors theorem of Robertson and Seymour. The proposed logic can be seen as a general framework to capitalize on the potential of the *irrelevant vertex technique*. It gives a way to deal with problem instances of unbounded treewidth, for which Courcelle's theorem does not apply.

2012 ACM Subject Classification Theory of computation → Logic; Theory of computation → Parameterized complexity and exact algorithms; Mathematics of computing → Graph algorithms

Keywords and phrases Algorithmic meta-theorems, Graph modification problems, Model-checking, Graph minors, First-order logic, Monadic second-order logic, Flat Wall theorem, Irrelevant vertex technique

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.61

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `http://arxiv.org/abs/2111.02755` [31]

Funding *Fedor V. Fomin*: Supported by the Research Council of Norway via the project BWCA (grant no. 314528).
Petr A. Golovach: Supported by the Research Council of Norway via the project BWCA (grant no. 314528).
Ignasi Sau: Supported by the ANR projects ELIT (ANR-20-CE48-0008-01) and ESIGMA (ANR-17-CE23-0010) and the French-German Collaboration ANR/DFG Project UTMA (ANR-20-CE92-0027).
Giannos Stamoulis: Supported by the ANR project ESIGMA (ANR-17-CE23-0010) and the French-German Collaboration ANR/DFG Project UTMA (ANR-20-CE92-0027).
Dimitrios M. Thilikos: Supported by the ANR project ESIGMA (ANR-17-CE23-0010) and the French-German Collaboration ANR/DFG Project UTMA (ANR-20-CE92-0027).

Acknowledgements We wish to thank the anonymous reviewers for their valuable remarks.

1 Introduction

Our work is kindled by the current algorithmic advances in graph modification. The core of our approach is a novel model-theoretic framework that is based on the interplay between model theory and algorithmic graph minors. Departing from this new perspective, we obtain algorithmic meta-theorems that encompass, unify, and extend all known meta-algorithmic results on minor-closed graph classes.

1.1 State of the art and our contribution

Modification problems. A *graph modification problem* asks whether it is possible to apply a series of modifications to a graph in order to transform it to a graph with some desired target property. Such problems have been the driving force of Parameterized Complexity where parameterization quantifies the concept of "distance from triviality" [48] and measures the amount of the applied modification. Classically, modification operations may be vertex or edge deletions, edge additions/contractions, or combinations of them like taking a minor. In their generality, such problems are NP-complete [60] and much research in Parameterized Complexity is on the design of algorithms in time $f(k) \cdot n^{\mathcal{O}(1)}$, where the parameter k is some measure of the modification operation [20]. The target property may express desired structural properties that respond to certain algorithmic or combinatorial demands. A widely studied family of target properties are minor-closed graph classes such as edgeless graphs [14], forests [13], bounded treewidth graphs [35, 54], planar graphs [50, 62], bounded genus graphs [55], or, most generally, minor-excluding graphs [72, 73]. However, other families of target properties have also been considered, such as those that exclude an odd cycle [29], a topological minor [36], an (induced) subgraph [22, 69], an immersion [40], or an induced minor [41]. A broad class of graph modification problems concerns cuts. In a typical cut problem, one wants to find a minimum-size set of edges or vertices X in a graph G such that in the new graph $G \setminus X$, obtained by deleting X from G, some terminal-connectivity conditions are satisfied. For example, the condition can be that a set of specific terminals becomes separated or that at least one connected component in the new graph is of a specific size. The development of parameterized algorithms for cut problems is a popular trend in parameterized algorithms [21, 61]. More involved modification measures of vertex set removals, related to treewidth or treedepth, have been considered very recently [1, 12, 27, 49].

Algorithmic meta-theorems. A vibrant line of research in Logic and Algorithms is the development of *algorithmic meta-theorems*. According to Grohe and Kreutzer [45], algorithmic meta-theorems state that certain families of algorithmic problems, typically defined by some logical and some combinatorial condition, can be solved "efficiently", under some suitable definition of this term. Algorithmic meta-theorems play an important role in the theory of algorithms as they reveal deep interplays between Algorithms, Logic, and Combinatorics. One of the most celebrated meta-theorems is Courcelle's theorem asserting that graph properties definable in CMSOL (counting monadic second-order logic) are decidable in linear time on graphs of bounded treewidth [15]; see also [2, 10]. Another stream of research concerns identifying wide combinatorial structures where model-checking for FOL (first-order logic) can be done in polynomial time. This includes graph classes of bounded degree [76], graph classes of bounded local treewidth [37], *minor-closed graph classes* [30], graph classes locally excluding a minor [23], and more powerful concepts of sparsity, such as having bounded expansion [26, 63], nowhere denseness [46], or having bounded twin-width [9]. (See [44, 58] for surveys. Also for results on the combinatorial horizon of FOL and CMSOL (and its variants) see [8, 9, 46] and [57] respectively.)

Another line of research, already mentioned in [44], is to prove algorithmic meta-theorems for extensions of FOL of greater expressibility. Two such extensions have been recently presented. The first one consists in enhancing FOL with predicates that can express k-connectivity for every $k \geq 1$. This extension of FOL was introduced independently by Schirrmacher, Siebertz, and Vigny in [75] (under the name FOL+conn) and by Bojańczyk in [6] (under the name *separator logic*). The second and more expressive extension, also introduced by Schirrmacher, Siebertz, and Vigny in [75], is FOL+DP, that enhances FOL with predicates expressing the existence of disjoint paths between certain pairs of vertices. For FOL+conn, an algorithmic meta-theorem for model-checking on graphs excluding a topological minor has been very recently given by Pilipczuk, Schirrmacher, Siebertz, Torunczyk, and Vigny [66]. For the more expressive FOL+DP, an algorithmic meta-theorem for model-checking on graphs excluding a minor has been very recently given by Golovach, Stamoulis, and Thilikos in [43] (see [42] for the full version).

Research on the meta-algorithmics of FOL is quite active and has moved to several directions such as the study of FOL-interpretability [7, 39, 64, 65] or the enhancement of FOL with counting/numerical predicates [25, 47, 59].

In this paper, we initiate an alternative approach consisting in combining the expressive power of FOL and CMSOL. A typical family of problems where such an approach becomes relevant is the one of modification problems. Courcelle's theorem implies that if the target property corresponds to a class of bounded treewidth and the modification conditions are definable in CMSOL, then such modification problems are fixed-parameter tractable when parameterized by the length of the sentence and the treewidth of the graph. However, when the target class graph is of unbounded treewidth, none of the aforementioned algorithmic meta-theorems encompasses broad families of modification problems. As an illustrative example, consider the Planarization problem, which consists in deciding whether at most k vertices can be removed from an input graph to make it planar (or equivalently, minor-excluding K_5 and $K_{3,3}$). While this problem is definable in CMSOL, Courcelle's theorem cannot be applied as we cannot assume that yes-instances are of bounded treewidth. On the other hand, we can easily assume that yes-instances minor-exclude K_{k+6}. However, all known meta-theorems whose combinatorial condition encompasses the minor-exclusion are about FOL, and FOL cannot express the Planarization problem. On the positive side, an algorithm in time $f(k) \cdot n^2$ for Planarization is an algorithmic consequence of Robertson-Seymour's theorem [68] (combined with [51, 67]). This automatic implication follows directly (albeit non-constructively) for a wide family of modification problems whose yes-instances are minor-closed. There is a long line of research in parameterized algorithms towards providing constructive and reasonable estimations of $f(k)$ [50, 62, 72, 73]. Note that Robertson-Seymour's theorem, besides not being constructive in general, automatically offers results only for problems whose yes-instances are minor-closed.

Our contribution. We introduce a *compound logic* that models computational problems through the lens of the "modulator vs target" duality of graph modification problems. Each sentence of this logic is a composition of two types of sentences. The first one, called the *modulator sentence*, models a modification operation, while the second one, called the *target sentence*, models a target property. Informally, our result, in its simplest form, asserts that if some appropriate version of the modulator sentence meets the meta-algorithmic assumptions of Courcelle's theorem [15] (i.e., CMSOL-definability and bounded treewidth) and the target sentence meets the meta-algorithmic assumptions of the theorem of Flum and Grohe [30] (i.e., FOL-definability and minor-exclusion), then model-checking for the composed compound

sentence can be done, constructively, in quadratic time. Our main result (Theorem 5) can be seen as a "two-dimensional product" of the two aforementioned meta-algorithmic results, contains both of them as special cases, and automatically implies the tractability of wide families of problems that *neither* are FOL-definable *nor* have instances of bounded treewidth.

1.2 Our results

In this subsection we give formal statements of our results. We need first some definitions.

Preliminaries on graphs. Given a graph G, we denote by $\mathsf{cc}(G)$ the set of all connected components of G. For a graph G and a set $X \subseteq V(G)$, the *stellation* of X in G is the graph $\mathsf{stell}(G, X)$ obtained from G if, for every $C \in \mathsf{cc}(G \setminus X)$, we contract all edges of C to a single vertex v_C. The *torso* of X in G is the graph $\mathsf{torso}(G, X)$ obtained from $\mathsf{stell}(G, X)$ if, for every v_C where $C \in \mathsf{cc}(G \setminus X)$, we add all edges between neighbors of v_C and finally remove all v_C's from the resulting graph. Given a family of graphs $\mathcal{H}$, we define $\mathsf{excl}(\mathcal{H})$ as the class of all graphs minor-excluding the graphs in $\mathcal{H}$ and note that $\mathsf{excl}(\mathcal{H})$ is minor-closed. The *Hadwiger number* of a graph G, denoted by $\mathbf{hw}(G)$, is the minimum k where $G \in \mathsf{excl}(\{K_k\})$ and K_k is the complete graph on k vertices. We also use the well-known parameter of *treewidth* of a graph G, denoted by $\mathbf{tw}(G)$. Given a graph class $\mathcal{G}$, we define $\mathbf{tw}(\mathcal{G}) = \max\{\mathbf{tw}(G) \mid G \in \mathcal{G}\}$. We define $\mathbf{hw}(\mathcal{G})$ analogously. We use $\mathcal{G}_{\mathrm{all}}$ for the set of all graphs.

Preliminaries on logic. We use CMSOL (resp. FOL) for the set of sentences in counting monadic second-order logic (resp. first-order logic). Given some vocabulary τ and a sentence $\varphi \in \mathsf{CMSOL}[\tau]$, we denote by $\mathrm{Mod}(\varphi)$ the set of all finite models of φ, i.e., all structures that are models of φ. In this introduction, in order to simplify our presentation, all structures that we consider are either graphs or annotated graphs, i.e., pairs (G, X) where G is a graph and $X \subseteq V(G)$. In the first case $\tau = \{\mathsf{E}\}$, and in the second $\tau = \{\mathsf{E}, \mathsf{X}\}$.

Given a $\varphi \in \mathsf{CMSOL}[\{\mathsf{E}\}]$, we define the *connectivity extension* $\varphi^{(\mathsf{c})}$ of φ so that $G \models \varphi^{(\mathsf{c})}$ if $\forall C \in \mathsf{cc}(G), C \models \varphi$. Similarly, for every $\mathcal{L} \subseteq \mathsf{CMSOL}[\{\mathsf{E}\}]$, we define $\mathcal{L}^{(\mathsf{c})} = \mathcal{L} \cup \{\varphi^{(\mathsf{c})} \mid \varphi \in \mathcal{L}\}$. Notice that $\{\varphi\}^{(\mathsf{c})} = \{\varphi, \varphi^{(\mathsf{c})}\}$. Also by $\mathbf{PB}(\mathcal{L})$ we denote the set of all positive Boolean combinations (i.e., using only the Boolean connectives $\vee$ and $\wedge$) of sentences in $\mathcal{L}$. We next define the following sets of sentences:

- The set $\mathsf{CMSOL}^{\mathsf{tw}}[\{\mathsf{E}, \mathsf{X}\}]$ contains every sentence $\beta \in \mathsf{CMSOL}[\{\mathsf{E}, \mathsf{X}\}]$ for which there exists some c_β such that the torsos of all the models of β have treewidth at most c_β. Formally,
  $\mathsf{CMSOL}^{\mathsf{tw}}[\{\mathsf{E}, \mathsf{X}\}] = \{\beta \in \mathsf{CMSOL}[\{\mathsf{E}, \mathsf{X}\}] \mid \exists c_\beta : \mathbf{tw}\{\mathsf{torso}(G, X) \mid (G, X) \models \beta\} \leq c_\beta\}$.
- The set $\mathsf{EM}[\{\mathsf{E}\}]$ is the set of all sentences in $\mathsf{CMSOL}[\{\mathsf{E}\}]$ that express the minor-exclusion of a non-empty set of graphs. Formally,
  $\mathsf{EM}[\{\mathsf{E}\}] = \{\mu \in \mathsf{CMSOL}[\{\mathsf{E}\}] \mid \exists \mathcal{H} \subseteq \mathcal{G}_{\mathrm{all}}, \mathcal{H} \neq \emptyset : \mathrm{Mod}(\mu) = \mathsf{excl}(\mathcal{H})\}$.
- $\Theta_0[\{\mathsf{E}\}]$ contains every sentence $\sigma \wedge \mu$ where $\sigma \in \mathsf{FOL}[\{\mathsf{E}\}]$ and $\mu \in \mathsf{EM}[\{\mathsf{E}\}]$.

For simplicity, we use $\mathsf{CMSOL}^{\mathsf{tw}}$, EM, and Θ_0 as shortcuts for $\mathsf{CMSOL}^{\mathsf{tw}}[\{\mathsf{E}, \mathsf{X}\}]$, $\mathsf{EM}[\{\mathsf{E}\}]$, and $\Theta_0[\{\mathsf{E}\}]$, respectively. Note that both $\mathsf{CMSOL}^{\mathsf{tw}}$ and Θ_0 are undecidable.

Algorithmic meta-theorems. We are now in position to restate three major meta-algorithmic results that were mentioned in the previous subsection.

▶ **Proposition 1** (Courcelle [15]). *For every $\beta \in \mathsf{CMSOL}^{\mathsf{tw}}$, there is an algorithm deciding $\mathrm{Mod}(\beta)$ in linear time.*

▶ **Proposition 2** (Robertson and Seymour [67,68] and Kawarabayashi, Kobayashi, and Reed [51]). *For every minor-closed graph class $\mathcal{G}$, deciding membership in $\mathcal{G}$ can be done in quadratic time.*

▶ **Proposition 3** (Flum and Grohe [30]). *For every $\gamma \in \Theta_0$, there is an algorithm deciding* $\mathrm{Mod}(\gamma)$ *in quadratic time.*

Some comments are in order. The statements of Proposition 1 and Proposition 3 have been adapted so to incorporate the combinatorial demands in the logical condition. While they can both be stated for structures, we state Proposition 1 for annotated graphs and Proposition 3 for graphs in order to facilitate our presentation. In the classic formulation of Courcelle's theorem, we are given a sentence $\beta \in \mathsf{CMSOL}$ and a tree decomposition of bounded treewidth. As such a decomposition can be found in linear time, using e.g., [4,56], the linearity in the running time of Courcelle's theorem is preserved when it is stated in the form of Proposition 1. For the theorem of Flum and Grohe, the situation is different as the combinatorial demand is minor-exclusion of a clique, which is not definable is FOL. For this reason we state Proposition 3 using the logic Θ_0 that contains *compound* sentences of the form $\sigma \wedge \mu$, where $\sigma \in \mathsf{FOL}$ and μ expresses minor-exclusion. For the running time of the algorithm of Proposition 3, we also need to take into account Proposition 2. As we already mentioned, Proposition 1 and Proposition 3 cannot deal, in general, with modification problems to properties of unbounded treewidth. Moreover, recall that Proposition 2 applies only to problems whose yes-instances are minor-closed.

We stress that Proposition 1, Proposition 2, and Proposition 3 are non-constructive. In order to construct the algorithms promised by Proposition 1, one should also know the bound c_β on the treewidth of the models of $\beta \in \mathsf{CMSOL}^{\mathsf{tw}}$ (note that bounded treewidth is also CMSOL-definable since it is characterized by a finite set of forbidden minors) and this appears in the hidden constants in the running time in Proposition 1. Similarly, for Proposition 2 (resp. Proposition 3), one should have an upper bound on the Hadwiger number of the graphs in $\mathcal{G}$ (resp. the models of γ).

A logic for modification problems. As a key ingredient of our result, we define the following operation between sentences. Let $\beta \in \mathsf{CMSOL}[\{\mathsf{E},\mathsf{X}\}]$ and $\gamma \in \mathsf{CMSOL}[\{\mathsf{E}\}]$. We refer to β as the *modulator* sentence on annotated graphs and to γ as the *target* sentence on graphs. We define $\beta \triangleright \gamma$ so that

$$G \models \beta \triangleright \gamma \text{ if there is } X \subseteq V(G) \text{ such that } (\mathsf{stell}(G,X),X) \models \beta \text{ and } G \setminus X \models \gamma. \tag{1}$$

In other words, $G \models \beta \triangleright \gamma$ means that the stellation of X in G, along with X, is a model of the modulator sentence β and the $G \setminus X$ is a model of the target sentence γ. That way, β implies the modification operation and γ expresses the target graph property. It is easy to see that $\beta \triangleright \gamma \in \mathsf{CMSOL}[\{\mathsf{E}\}]$. This will allow us to apply the operation $\triangleright$ iteratively.

As an example, the problem of removing a set X of k vertices so that $G \setminus X$ is a triangle-free planar graph can be expressed by $\beta \triangleright \gamma$ if β asks that X has k vertices and $\gamma = \sigma \wedge \mu$, where σ expresses triangle-freeness and μ expresses planarity by the exclusion of $K_{3,3}$ and K_5.

Before we present our result in full generality, we give first the following indicative special case, which already expresses the conditions of Proposition 1 and Proposition 3.

▶ **Theorem 4.** *For every $\beta \in \mathsf{CMSOL}^{\mathsf{tw}}$ and every $\gamma \in \Theta_0$, there is an algorithm deciding* $\mathrm{Mod}(\beta \triangleright \gamma)$ *in quadratic time.*

Indeed, Proposition 1 follows if β expresses that $X = V(G)$ and γ demands that $G \setminus X$ is the empty graph (in particular, Theorem 4 contains Proposition 1 as a linear-time black-box procedure for deciding models of bounded treewidth) and Proposition 3 follows if β demands that $X = \emptyset$. In other words, Proposition 1 follows if the target sentence becomes void while Proposition 3 follows if the modulator sentence is void.

As a first step towards a more general statement, Theorem 4 also holds if we replace $\gamma \in \Theta_0$ by $\gamma \in \Theta_0^{(c)}$ or even by positive Boolean combinations of sentences in $\Theta_0^{(c)}$, i.e., $\gamma \in \mathbf{PB}(\Theta_0^{(c)})$. Moreover, in order to present our result in full generality, we recursively define, for every $i \geq 1$,

$$\Theta_i \quad = \quad \{\beta \triangleright \gamma \mid \beta \in \mathsf{CMSOL}^{\mathsf{tw}} \text{ and } \gamma \in \mathbf{PB}(\Theta_{i-1}^{(c)})\}. \tag{2}$$

Notice that the sentences of Theorem 4 (hence also of Proposition 1 and Proposition 3) are already contained in Θ_1. We set $\Theta = \bigcup_{i \geq 1} \Theta_i$. The full strength of our results, stated in the vocabulary of graphs, is given by our main theorem.

▶ **Theorem 5.** *For every $\theta \in \Theta$, model-checking for θ can be done in quadratic time.*

An alternative statement. Our results can also be seen under the typical meta-algorithmic framework where a logical and a combinatorial condition are given. For this, consider an alternative of Θ, called $\tilde{\Theta}$, that is defined as in (2) by taking $\tilde{\Theta}_0 = \mathsf{FOL}$ as the base case, i.e., by discarding the minor-exclusion from the definition of Θ_0. Notice that $\tilde{\Theta}$ contains FOL and can be seen as a natural extension of it. A direct consequence of Theorem 5 is the following.

▶ **Theorem 6.** *For every $\tilde{\theta} \in \tilde{\Theta}$, model-checking for $\tilde{\theta}$ can be done in quadratic time on every graph class of bounded Hadwiger number.*

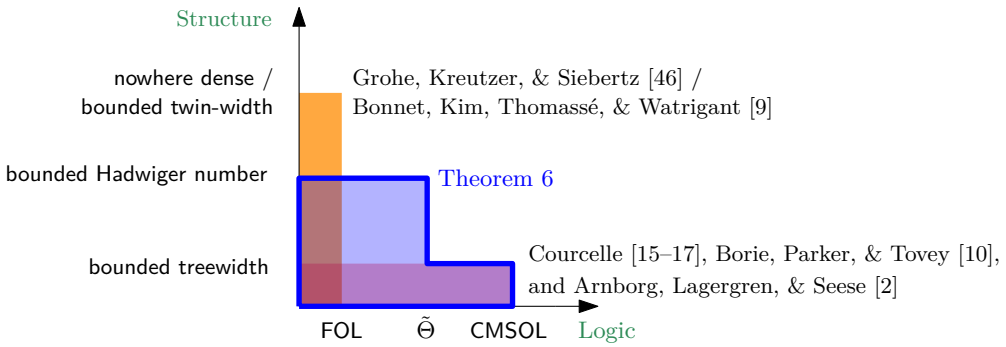


Figure 1 Theorem 6 in the current meta-algorithmic landscape. The vertical axis is the combinatorial one and is marked by four different types of (structural) sparsity, while the horizontal one is the logical one and is marked with FOL, $\tilde{\Theta}$, and CMSOL.

Theorem 6 is a corollary of Theorem 5 and provides an alternative meta-algorithmic set up between the logical and the combinatorial condition (see Figure 1): for each sentence θ in Θ, one may consider a sentence $\tilde{\theta}$ in $\tilde{\Theta}$ where we discard minor-exclusion from all its target sentences and then consider the problem of deciding $\mathrm{Mod}(\theta)$ on some minor-excluding graph class. This correspondence is many-to-one, as many different $\theta \in \Theta$ correspond to the same $\tilde{\theta} \in \tilde{\Theta}$. We opted for presenting and proving our results in the form of Theorem 5, as it is more general and more versatile in expressing modification problems. In the full version of the paper [31], we define we define Θ on general structures.

Compound logics based on FOL+DP. In the full version of the paper [31], by combining our proofs with the meta-algorithmic results of [42, 43], we extend Theorem 5 (resp. Theorem 6) in the cases of the logic Θ^{DP} (resp. $\tilde{\Theta}^{\mathsf{DP}}$) that are obtained if in the definition of Θ (resp. $\tilde{\Theta}$) we now consider the (more expressive) logic FOL+DP instead of FOL in the target sentences. That way, the derived extensions of Theorem 5 and Theorem 6 (that is, Theorem 8 and Theorem 9) encompass, as special cases, all results and applications in [42, 43] (see Figure 3 for a visualization of the overall state-of-the-art on the related algorithmic meta-theorems on subgraph-closed graph classes). While presenting our results and techniques, for the sake of simplicity, we chose to focus on the statement and the proof of our meta-theorems for Θ (Theorem 5) and $\tilde{\Theta}$ (Theorem 6) and then, in the full version of the paper [31], present the modifications that should be applied in order to extend them for Θ^{DP} and $\tilde{\Theta}^{\mathsf{DP}}$.

A parametric variant of our results. A *graph parameter* is a function $\mathbf{p} : \mathcal{G}_{\text{all}} \to \mathbb{N}$. We say that $\mathbf{p}$ is *treewidth-bounded* if there is a function $f : \mathbb{N} \to \mathbb{N}$ such that for each $G \in \mathcal{G}_{\text{all}}$, $\mathbf{p}(G) \leq f(\mathsf{tw}(G))$. We say that $\mathbf{p}$ is CMSOL-*definable* if for every $k \in \mathbb{N}$ there is a CMSOL-sentence (on graphs) β_k such that the set of all models of β_k is $\text{Mod}(\beta_k) = \{G \mid \mathbf{p}(G) \leq k\}$. Clearly, if $\mathbf{p}$ is treewidth-bounded then we can also assume that each β_k is a sentence in $\mathsf{CMSOL}^{\mathsf{tw}}$ and in this case we say that $\mathbf{p}$ is $\mathsf{CMSOL}^{\mathsf{tw}}$-definable. There are several known graph parameters that are $\mathsf{CMSOL}^{\mathsf{tw}}$-definable, such as treewidth, pathwidth, tree-depth, bridge-depth, block tree-depth, vertex cover, feedback vertex set, branch-width, carving-width, or cutwidth.

For a graph parameter $\mathbf{p}$ and a graph class $\mathcal{G}$, we define the new graph parameter $\mathbf{p}_{\mathcal{G}} : \mathcal{G}_{\text{all}} \to \mathbb{N}$ such that

$$\mathbf{p}_{\mathcal{G}}(G) = \min\{k \mid \exists X \subseteq V(G) \mid \mathbf{p}(\mathsf{torso}(G, X)) \leq k \ \wedge \ G \setminus X \in \mathcal{G}\}.$$

Thus $\mathbf{p}_{\mathcal{G}}$ measures by $\mathbf{p}$ the quality of a modulator X to property $\mathcal{G}$. For example, when $\mathbf{p}$ is the size of the modulator, then this is just the *vertex deletion distance to* $\mathcal{G}$, that is, the minimum number of vertices X such that $G \setminus X \in \mathcal{G}$. When $\mathbf{p}$ is the tree-depth of a graph, then $\mathbf{p}_{\mathcal{G}}$ is the elimination distance to $\mathcal{G}$. Or when $\mathbf{p}$ is the treewidth of a graph, then $\mathbf{p}_{\mathcal{G}}$ corresponds to $\mathcal{G}$-treewidth. We consider the general setting where $\mathbf{p}$ is a $\mathsf{CMSOL}^{\mathsf{tw}}$-definable graph parameter and $\mathcal{G}$ is a Θ-definable graph class, that is, $\text{Mod}(\theta) = \mathcal{G}$ for some $\theta \in \Theta$. By setting $\theta_k = \beta_k \triangleright \theta \in \Theta$, we have that $\text{Mod}(\theta_k) = \{G \mid \mathbf{p}_{\mathcal{G}}(G) \leq k\}$. Then the following theorem is a direct consequence of Theorem 5 and Theorem 6.

▶ **Theorem 7.** *Let* $\mathbf{p}$ *be a* $\mathsf{CMSOL}^{\mathsf{tw}}$*-definable graph parameter and* $\mathcal{G} = \text{Mod}(\theta)$ *for some* $\theta \in \Theta$*. Then there is an algorithm that, with input a graph* G *and* $k \in \mathbb{N}$*, checks whether* $\mathbf{p}_{\mathcal{G}}(G) \leq k$ *in time* $\mathcal{O}_{k,|\theta|}(n^2)$*. Moreover, if* $\mathcal{G} = \text{Mod}(\tilde{\theta})$ *for some* $\tilde{\theta} \in \tilde{\Theta}$*, then there is an algorithm that, with the same input, checks whether* $\mathbf{p}_{\mathcal{G}}(G) \leq k$ *in time* $\mathcal{O}_{k,|\theta|,\mathsf{hw}(G)}(n^2)$*.*

All the results mentioned in this subsection, in what concerns minor-excluded graphs, are subsumed by Theorem 7. Moreover, by allowing FOL-definability in the target sentence and $\mathsf{CMSOL}^{\mathsf{tw}}$-definability in the modulator sentence, we vastly extend Proposition 2 to graph classes and parameters that are not necessarily minor-closed or hereditary. We stress that none of the results in [43, 66] is able to deal with the problems captured by Theorem 7 in their full generality.

Constructibility. While Robertson-Seymour's theorem (Proposition 2) implies the existence of an algorithm, its proof is not constructive and cannot be used to construct such an algorithm [28]. An extra feature of the proof of Theorem 5 (as well as of its corollary Theorem 6) is that it is *constructive*, in the sense that the implied algorithms can be constructed

if we are given some bound on the Hadwiger number of the models of θ. This considerably extends the constructibility horizon of Proposition 2 for graph classes that are not necessarily minor-closed or even hereditary. See the full version of the paper [31] for more details.

Techniques. The algorithm and the proofs of Theorem 5 use as departure point core techniques from the proofs of Propositions 1, 3, and 2 such as Courcelle's theorem for dealing with CMSOL-sentences, the use of Gaifman's theorem for dealing with FOL-sentences, and an extended version of the *irrelevant vertex technique*, introduced by Robertson and Seymour in [67], along with some suitable version of the Flat Wall theorem which appeared recently in [53, 71] (see also [3, 70, 72, 73]). The algorithm produces equivalent and gradually "strictly simpler" instances of an annotated version of the problem. Each equivalent instance is produced in linear time and this simplification is repeated until the graph has bounded treewidth (here we may apply Courcelle's theorem, that is Proposition 1). This yields a (constructive) quadratic-time algorithm. We stress that our approach avoids techniques that have been recently used for this type of problems such as *recursive understanding* (in [1]) or the use of *important separators* (in [49]) that give worst running times in n.

Natural limitations. We wish to comment on why the three basic ingredients of the definition of our logic Θ are necessary for the statement and the proof of our meta-algorithmic results.

The first ingredient of Θ is that the modulator sentences belong in $\mathsf{CMSOL}^{\mathsf{tw}}[\{\mathsf{E},\mathsf{X}\}]$ which is defined so that the treewidth of $\mathsf{torso}(G,X)$ is bounded. While it is known that bounding the treewidth is necessary for CMSOL-model-checking [19, 58], one may ask why it is not enough to just bound the treewidth of $G[X]$. To see why this unavoidable, consider a graph G and let G' be the graph obtained from G by subdividing each edge once. Then, asking whether G is Hamiltonian, which is a well-known NP-complete problem, is equivalent to asking whether G' has a vertex set S' such that $G'[S']$ is a cycle and such that $G' \setminus S'$ is an edgeless graph, that is, a K_2-minor-free graph. Notice that, while $\mathbf{tw}(G'[S']) = 2$, $\mathsf{torso}(G', S') = G$ has unbounded treewidth.

The second ingredient of Θ is minor-exclusion, that is materialized by the conjunction with μ in the definition of Θ_0. Notice first that expressing whether a graph G contains a clique on k vertices can be done by a FOL-sentence, while the k-Clique problem is W[1]-hard [20]. Therefore, the minor-exclusion condition cannot be dropped. Moreover, even if we consider a *fixed* target FOL-sentence, it was proved in [33] that there exists a FOL-sentence σ such that checking whether a graph G has a set $S \subseteq V(G)$ with $|S| = k$ such that $G \setminus S \models \sigma$ is a W[1]-hard problem, when parameterized by k. This implies that, even for this restricted problem where the FOL-sentence σ is fixed, an algorithm running in time $f(k) \cdot n^{\mathcal{O}(1)}$ cannot be expected.

The third ingredient of Θ is the FOL demand, that is materialized by the conjunction with σ in the definition of Θ_0. This is also necessary, as otherwise we may choose some property σ not definable in FOL, such as Hamiltonicity, which is CMSOL-definable and NP-complete on planar graphs. Without the restriction that σ needs to be FOL-definable, a void modulator and a sentence μ expressing planarity would be able to model this NP-complete problem. Nevertheless, we may consider extensions of FOL in the target sentence, as done in Section 3.

2 Overview of the proof

In this section we summarize some of the main ideas involved in the proof of Theorem 5, while keeping the description at an intuitive level. We would like to stress that some of the informal definitions given in this section are deliberately *imprecise*, since providing the

precise ones would result in a huge overload of technicalities that would hinder the flow of the proof. Our algorithms consider as input a general structure $\mathfrak{A}$ (not necessarily a graph), and most of the arguments in the proofs concern its Gaifman graph $G_{\mathfrak{A}}$. Dealing with general structures, besides making our results more versatile, turns out to be useful in the proofs, in particular for using tools such as the *Backwards Translation Theorem* [18, Theorem 1.40], or for extending our results to other modification operations beyond vertex removal (see the full version of the paper [31]). Since the Gaifman graph of a graph is the graph itself, in this overview we will assume for simplicity that the input of our algorithms is a graph G, instead of a general structure $\mathfrak{A}$. In Subsection 2.1 we present the general scheme of the algorithm. In Subsection 2.2 we present a simplified and illustrative setting, where the input sentence θ belongs to the fragment $\bar{\Theta}_1$. This (very) particular case of Theorem 5 is helpful to illustrate our main conceptual ideas. For a more detailed proof-overview and formal proofs (up to the general compound logic Θ considered in Theorem 5), we refer the reader to the full version of the paper [31].

2.1 General scheme of the algorithm

We use the *irrelevant vertex technique* introduced by Robertson and Seymour [67]. Our overall strategy is the "typical" one when using this technique: if the treewidth of the input graph G is bounded by an appropriately chosen function, depending only on the sentence $\theta \in \Theta$, then we use Courcelle's theorem [15] and solve the problem in linear time, using the fact that our compound logic Θ is a fragment of counting monadic second-order logic. Otherwise, we identify an irrelevant vertex in linear time, that is, a vertex whose removal produces an equivalent instance. Naturally, the latter case concentrates all our efforts and, in what follows, we sketch the main ingredients that we use in order to identify such an irrelevant vertex. In a nutshell, our approach is based on introducing a robust combinatorial framework for finding irrelevant vertices. In fact, what we find is *annotation-irrelevant flat territories*, building on our previous recent work [3, 3, 32, 70–73], which is formulated with enough generality so as to allow for the application of powerful tools such as Gaifman's locality theorem [38] or a variant of Courcelle's theorem on boundaried graphs, intuitively saying that the dynamic programming tables constructed by the proof of Courcelle's theorem are also definable in CMSOL (see [5, Lemma 3.2]).

Flat walls. An essential tool of our approach is the notion of *flat wall*, originating in the work of Robertson and Seymour [67]. Informally speaking, a flat wall W is a structure made up of (non-necessarily planar) pieces, called *flaps*, that are glued together in a bidimensional grid-like way defining the so-called *bricks* of the wall. While such a structure may not be planar, it enjoys topological properties similar to those of planar graphs, in the sense that two paths that are not routed entirely inside a flap cannot "cross", except at a constant-sized vertex set A whose vertices are called *apices*. Hence, flat walls are only "locally non-planar", and after removing apices we can apply useful locality arguments, in the sense that two vertices that are in "distant" flaps should also be "distant" in the whole graph without the apices. One of the most celebrated results in the theory of Graph Minors by Robertson and Seymour [67, 68], known as the Flat Wall theorem (see also [53, 71] for recently proved variants), informally states that graphs of large treewidth contain either a large clique minor or a large flat wall. In this article we use the framework recently introduced in [71] that provides a more accurate view of some previously defined notions concerning flat walls, particularly in [53]. Precise definitions of the concepts of *flatness pair*, *homogeneity*, *regularity*, *tilt*, and *influence* can be found in the full version of this article [31] and we stress that they are *not* critical in order to

understand the main technical contributions of the current article (however, they are critical for their formal correctness). In what follows, when considering a flat wall W with an apex set A in a graph G, for simplicity we refer to W by using indistinguishably the terms "wall" and "compass of a wall", which can be roughly described as the component containing W in the graph obtained from G by removing A and the "boundary" of W.

Working with an annotated version of the problem. We start by defining a convenient equivalent version of the problem, by replacing our sentence $\theta \in \Theta$ with an equivalent *enhanced* sentence $\theta_{\mathsf{R},\mathbf{c}}$. This is done in two steps, as we explain in the following two paragraphs.

Assuming the existence of a flat wall and an apex set in our input graph G, we first transform the question θ on G to a question on a structure obtained from G by "neutralizing" the apex set (see the full version of the paper [31]). The goal of this step is to ask the final FOL-sentences σ of our sentence θ in a "flattened" structure, where apices can no longer "bring close" any distant parts of the wall. This transformation of the problem, which we call *apex-projection*, will allow for the application of the locality-based strategy discussed in the definition of the *in-signature* of a wall in Subsection 2.2. To do this, we introduce some additional constant symbols $\mathbf{c}$ to our vocabulary that will be interpreted as the apex vertices.

The second step consists in defining an equivalent *annotated* version of the problem in order to deal with the FOL-sentences of θ, inspired by the approach of [32]. To do so, we introduce a vertex set $R \subseteq V(G)$, and require, for each FOL-sentence σ of θ, that the vertices interpreting the variables of (the equivalent Gaifman sentence of) σ belong to the annotated set R. We prove that the initial sentence θ and the obtained sentence, denoted by $\theta_{\mathsf{R},\mathbf{c}}$ and called an *enhanced sentence*, are equivalent for *any* choice of the apex set interpreting $\mathbf{c}$ and when R is interpreted as the whole vertex set of the graph. This independence of the choice of the apex set is strongly used in the proofs since, as discussed below, we will consider a number of different flat walls, each of which associated with a different apex set.

Our algorithms will work with the enhanced sentence $\theta_{\mathsf{R},\mathbf{c}}$. Starting with the input graph G with $V(G)$ as the annotated set R, we will create successive equivalent annotated instances, in which vertices from G are removed and such that the annotated set R is only reduced.

Zooming inside a flat wall. Our next step is to find, in G, a large flat wall W_0 to work with. The definition of our logic Θ implies that models of θ exclude a fixed complete graph K_c as a minor, where c depends only on θ. Therefore, we can apply the algorithmic version of the Flat Wall Theorem [72, Proposition 10] (see also [53, 67, 71]) to the input graph G and, assuming that the treewidth of G is large enough, we can find in linear time a flat wall W_0 and an apex set A in G such that the height of W_0 is a sufficiently large function of θ. Moreover, another crucial property guaranteed by this algorithm is that the treewidth of W_0 is bounded from above by a function of θ. This will be exploited in Subsection 2.2 in order to compute the so-called θ-*characteristic* of a wall. We will now apply a series of "zooming" arguments to the wall W_0, which are illustrated in Figure 2.

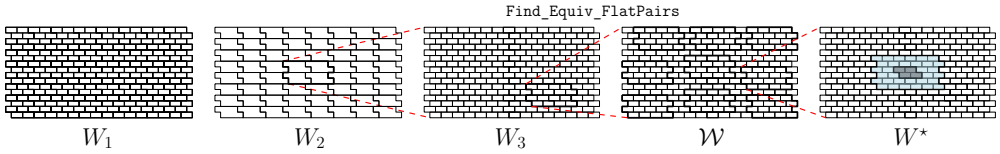


Figure 2 Sequence of walls in the general scheme of our algorithm. The first wall is obtained by applying the algorithm of [72, Proposition 14] for the wall W_0 in the input graph G.

Starting from W_0 and its associated apex set A, we apply the algorithm of [72, Proposition 14] and find, in linear time, a large (again, as a function of θ) subwall W_1 that is λ-*homogeneous*, where λ depends only on θ. The definition of a homogenous flat wall can be found in the full version of the paper [31], and roughly means that each of its bricks can route the same set of partial minors of the graphs corresponding to the minor-exclusion part of the sentence θ. We now apply the algorithm of [73, Lemma 16] to W_1, and obtain in linear time a large subwall W_2 that is *irrelevant* with respect to the minor-exclusion part of θ after the removal of a vertex set $X \subseteq V(G)$ of small enough *bidimensionality* (see Subsection 2.2). Intuitively, working "inside" W_2 allows us to "forget" the minor-exclusion part of θ in what follows. As our next step, we obtain in linear time a still large subwall W_3 of W_2 such that its associated apex set A_3 is "tightly tied" to W_3, in the sense that the neighbors in W_3 of every vertex in A_3 are spread in a "bidimensional" way.

Finding an irrelevant subwall. So far, we have found a large wall W_3 that satisfies the conditions of the above paragraph. Now, in order to identify an irrelevant vertex inside W_3, we find, inside the wall W_3, a collection $\mathcal{W}$ of pairwise disjoint subwalls, and to associate each of these subwalls with an appropriately defined θ-*characteristic* that captures its behavior with respect to the partial satisfaction of the sentence θ. Then the idea is that, if there are sufficiently many subwalls in $\mathcal{W}$ with the same θ-characteristic (called θ-*equivalent*), then some subwall in the interior of one of them can be declared *annotation-irrelevant* and this implies some progress in simplifying the current problem instance.

The above strategy allows to identify a subwall $W^\star$ inside $\mathcal{W}$ such that its central part can be removed from the annotated set R, and such that a smaller central part can be removed from G (the blue and grey subwalls in the rightmost wall of Figure 2, respectively). This is done by an algorithm, called `Find_Equiv_FlatPairs`, that is based on an appropriate definition of the θ-*characteristic* of a wall. In what follows we sketch the main ingredients and key ideas.

2.2 A simplified and illustrative setting

In order to provide some intuition, in this subsection we focus on formulas $\theta \in \Theta$ of a particular form, i.e., belonging to $\bar{\Theta}_1$, a set of formulas which we proceed to define informally in a semantical level: Given a general graph G as input, we seek for a vertex set $X \subseteq V(G)$, called *modulator*, such that, using the notation defined in the introduction, $\mathsf{stell}(G, X)$ satisfies the so-called *modulator sentence* β, and either every connected component C of $G \setminus X$, or the whole graph $G \setminus X$, satisfies the so-called *target sentence* γ, where $\gamma = \sigma \wedge \mu$ with σ being an arbitrary FOL-sentence and μ expressing the property of belonging to a proper minor-closed graph class.

Note that when $\theta \in \bar{\Theta}_1$, the target sentence γ needs to be satisfied either by each of the resulting connected components separately, or jointly by their union. We deal with this easily, by introducing a $\circ/\bullet$ -flag into the corresponding sentences that distinguishes both cases. The latter case is simpler, but in this description, in order to better illustrate our techniques, we assume the former.

Identifying the privileged component. A very useful tool in our algorithms is to identify, for every given X, a *unique* connected component among those of $G \setminus X$, which we call the *privileged component*, that contains "most" of the wall W_3. Let us formalize a bit this idea. For a positive integer q, a *pseudogrid* $\mathbf{W}_q$, is a collection of q "vertical" and q "horizontal" paths that intersect in a "grid-like" way. Note that the considered wall W_3 naturally defines a (large, as a function of θ) pseudogrid. A connected component C of a graph G is *privileged*

with respect to a set $X \subseteq V(G)$ and a pseudogrid $\mathbf{W}_q$ if C is a connected component of $G \setminus X$ that contains entirely at least one vertical and one horizontal path of $\mathbf{W}_q$. It is easy to see that such a privileged component, if it exists, is unique.

Moreover, when X is a modulator, the fact that $\mathsf{torso}(G, X)$ has bounded treewidth implies that every connected component of $G \setminus X$ has a "small interface" to X and thus the flat wall W_0 (and any large subwall of it) is not significantly "damaged" by X, which we formalize via the notion of having small *bidimensionality*. Intuitively for the definition), this means that X intersects a small number of so-called "bags" of the wall. Informally, the *bags* of a wall W in a graph G with apex set A define a partition of $G \setminus A$ into connected sets, such that each bag, except the external one, contains the part of the wall W between two neighboring degree-3 vertices of the wall. This property is used extensively in the proofs and, in particular, it defines, assuming the existence of a large flat wall W_0 and a modulator X, a *unique* privileged component C in $G \setminus X$ (regardless of the $\circ/\bullet$-flag). In our sentences, in order to identify such a component, we need to integrate the "recognition" of a pseudogrid $\mathbf{W}_q$ and its associated privileged component with respect to a modulator X: it is easy to see that these properties can be defined in CMSOL.

Splitting the sentence $\theta_{\mathsf{R},\mathbf{c}}$. The existence of a privileged component C allows us to see the sentence $\theta_{\mathsf{R},\mathbf{c}}$ as a conjunction of two subsentences: one that concerns the privileged component C (where we will find the irrelevant vertex) and another one concerning the modulator X and the other (non-privileged) components of $G \setminus X$. Namely, we define a sentence $\tilde{\theta}_q$, called the *split version* of $\theta_{\mathsf{R},\mathbf{c}}$, that allows us to "break" θ into two questions: one denoted by θ_q^{out} that is the conjunction of the modulator sentence β and the target sentence γ in the non-privileged components of $G \setminus X$ and another one that concerns the target sentence γ in the privileged component C. This latter question is composed of two subsentences, namely one about the satisfaction of the FOL-sentence σ and another one about the minor-exclusion given by μ. Given this decomposition of θ into three questions (one "external" and two "internal" ones), our "irrelevancy" arguments also decompose into three parts. Concerning the "irrelevancy" for minor-exclusion, as discussed above, the fact that the whole wall W_2 is irrelevant with respect to μ allows us to focus on the other two questions. For this, we need to define the *characteristic* of a wall with respect to θ, denoted by θ-char. This characteristic is composed of two parts: the *out-signature* corresponding to the satisfiability of the sentence θ_q^{out}, and the *in-signature* corresponding to the FOL-sentence σ. Let us now explain how we define the out-signature and the in-signature, and sketch why we can eventually declare a subwall irrelevant.

Defining the out-signature of a wall. Dealing with the irrelevancy with respect to the "external" sentence θ_q^{out} turns out to be the most interesting part of the proof and we introduce several ideas which are, in our opinion, one of the main conceptual contributions of this article. The goal is, for each wall W in the collection $\mathcal{W}$, to encode all the necessary information that concerns the satisfiability of θ_q^{out} in the "non-privileged" part of the graph and the modulator X. To do this, for each $W \in \mathcal{W}$ with apex set A, we define a set of *ℓ-boundaried graphs* (i.e., graphs in which ℓ "boundary" vertices are equipped with labels), constructed as we describe below, and where ℓ depends only on θ. The boundary corresponds to where the sentence has been "split" and we need to "guess" how to complement this boundary by the part of the modulator that is not inside the wall. Note that, since θ_q^{out} is a CMSOL-sentence, by a variant of Courcelle's theorem for boundaried graphs [15], there exists a *finite* collection $\mathsf{rep}^{(\ell)}(\theta_q^{\mathsf{out}})$ of sentences on ℓ-boundaried graphs that are "representatives" of the sentence θ_q^{out} and that can be effectively constructed. We next described how these ℓ-boundaried graphs are constructed.

We observe that, using the bounded-treewidth property of the modulator sentence β, there exists a "buffer" I in W, consisting of a set of consecutive layers of the wall, which is disjoint from a hypothetical modulator X. We guess with an integer d where this "buffer" I is placed in the wall and we denote its inner part by $I^{(d)}$. This naturally induces a partition of X into X_{in} and X_{out}, with X_{in} being the part of X that is inside $I^{(d)}$. We also guess which subset of the apex set A will belong to the modulator X and we denote it by $V_L(\mathbf{a})$, where L is the set containing the indices of the corresponding apex vertices. Since parts of the "non-privileged" vertex set of the graph may lie outside the considered wall, we need to guess the part of the modulator (namely, its boundary towards the component) that lies outside the wall. More precisely, we need to guess as well which subset F' of X_{out}, other than $V_L(\mathbf{a})$, will belong to the neighborhood of the privileged component. This is achieved by guessing all ways an (abstract) graph F' with a bounded number of vertices can extend the boundary. We let F be the graph obtained from the union of $V_L(\mathbf{a})$ and F'. Finally, we also need to consider a set Z that corresponds to X_{in} together with the part inside $I^{(d)}$ that has been "chopped off" by the modulator X, that is, the part of W inside $I^{(d)}$ that will not belong to the privileged component after the removal of the modulator X. We denote by $\partial(Z)$ the set of vertices in Z that have a neighbor in $I^{(d)}$. Altogether, these guesses result in the ℓ-boundaried graph $K^{(d,Z,L,F)}$ obtained from the graph induced by $I^{(d)}$ and the set F, whose boundary is the set $\partial(Z) \cup F$.

With each such a guess (R, d, L, Z) we associate the out-signature defined as follows and denoted by $\mathsf{out\text{-}sig}$. Its elements are pairs $(\mathbf{H}, \bar{\theta})$, where $\mathbf{H}$ encodes how the set $V_L(\mathbf{a})$ in the boundary has been extended by the "abstract" graph F', and $\bar{\theta} \in \mathsf{rep}^{(\ell)}(\theta_q^{\mathsf{out}})$ prescribes the equivalence class, within the set of Courcelle's representatives mentioned above, of the considered ℓ-boundaried graph. This concludes the description of the out-signature.

While this out-signature indeed encodes the behavior of the considered wall with respect to the "external" sentence θ_q^{out}, a crucial issue has been overlooked so far: in order to be able to identify an irrelevant subwall inside the collection $\mathcal{W}$ within the claimed running time, we need to be able to *compute* the (in- and out-)signature of a wall in linear time. To do this using Courcelle's theorem, we need to consider a graph that has treewidth bounded by a function of θ. Recall that θ_q^{out} is the conjunction of the modulator sentence β (which is evaluated in the graph $\mathsf{stell}(G, X)$) and the target sentence γ in the "non-privileged" components of $G \setminus X$. It follows that the treewidth of W is bounded by a function of θ, hence the treewidth of the ℓ-boundaried "subwall" $K^{(d,Z,L,F)}$, for which we want to compute the out-signature, is also bounded by a function of θ. However, the graph $K^{(d,Z,L,F)} \setminus V(F)$ "lives" inside the *whole* privileged component C, and we cannot guarantee that the treewidth of C is bounded by a function of θ. We overcome this problem with the following trick. We observe that the satisfaction of θ_q^{out} is preserved if, instead of the whole privileged component C, we consider the graph $K^{(d,Z,L,F)}$, which is obtained by "shrinking" C to the subwall $I^{(d)}$, and which has bounded treewidth as we need. Indeed, this modification does not change any of the non-privileged components in which the target sentence γ is evaluated and, by adding edges from the "guessed extended boundary" F' to $I^{(d)}$ in order to preserve connectivity, the resulting graph $\mathsf{stell}(G, X)$ remains unchanged with this transformation, and therefore the satisfaction of the modulator sentence β is also preserved.

Defining the in-signature of a wall. To deal with the irrelevancy with respect to the FOL-sentence σ, we use arguments strongly inspired by those of [32]. The core tool here is Gaifman's locality theorem, which states that every FOL-sentence σ is a Boolean combination of basic local sentences $\sigma_1, \ldots, \sigma_p$, in the sense that the satisfaction of each σ_i depends only

on the satisfaction of a set of sentences $\psi_1, \ldots, \psi_{\ell_i}$ evaluated on single vertices that can be assumed to be pairwise far apart. As discussed before, taking care of the domain of these vertices is the main reason why we consider a annotated version of the problem, corresponding to the enhanced sentence $\theta_{\mathsf{R},\mathbf{c}}$. Extending the approach of [32] (which does not deal with apices), the in-signature of a wall, denoted by $\mathsf{in\text{-}sig}$, encodes all (partial) sets of variables, one set for each basic local sentence of the so-called Gaifman sentence $\breve{\sigma}$, such that these variables lie inside an "inner part" of the wall, they are scattered in the "apex-projection" of this inner part, and they satisfy the local sentences ψ_i.

Declaring a subwall irrelevant. We now sketch the remaining of the proof for sentences in $\bar{\Theta}_1$. As mentioned above, suppose that we have already found, inside the collection $\mathcal{W}$, a large (as a function of θ) subcollection $\mathcal{W}' \subseteq \mathcal{W}$ of walls all having the same θ-characteristic. We pick one of these walls, say $W^\star \in \mathcal{W}'$, and we declare its central part irrelevant (see Figure 2). We need to prove that, if the input graph G satisfies θ, then the graph G' obtained from G by removing the central part of $W^\star$, also satisfies θ. That is, given a modulator X in the original instance G, we need to construct another set $X' \subseteq V(G)$ that is disjoint from $W^\star$ and that is a modulator in G'. For this, we proceed as follows.

The cardinality of $\mathcal{W}'$ and the fact that X intersects few bags of the wall W_3 imply that there is a large (again, as a function of θ) subcollection $\mathcal{W}'' \subseteq \mathcal{W}'$ of walls that are disjoint from X. We take such a wall $\hat{W} \in \mathcal{W}''$ and, using the fact that $W^\star$ and $\hat{W}$ have the same θ-characteristic, we show that we can "replace" the part of the modulator X that intersects $W^\star$ with another part in $\hat{W}$, together with an alternative assignment of variables that satisfies the corresponding sentences. This results in another set X' that is a modulator in G', hence yielding the annotation-irrelevancy of (the central part of) $W^\star$.

Showing these facts is far from being easy and we need a number of technical details dealing with the irrelevancy with respect to θ_q^{out} (which incorporates β), σ, and μ. In particular, an important idea is that, changing from X to X', we obtain a new boundaried graph, which is in fact the same graph but with a new boundary. The replacement arguments for the in-signature work because of the aforementioned distance-preservation property of the apex-projection. See the full version of the paper [31] for more details.

3 From FOL to FOL+DP: the compound logic Θ^{DP}

In the definition of Θ_0, the base case of Θ, we consider compound sentences $\sigma \wedge \mu$, where $\sigma \in \mathsf{FOL}$ and μ expresses minor-exclusion. However, one can consider extensions of FOL in the compound sentences. A possible candidate is first-order logic with disjoint-paths predicates defined in [75] (see the paragraph below for a formal definition). This way we can define a more general logic Θ^{DP} and prove an algorithmic meta-theorem that encompasses also the results in [42, 43]. To ease reading, in this subsection we deal only with graphs and not with general structures. However, our results can be straightforwardly be extended to general structures. All proofs of the results of this section can be found in Section 3.

The disjoint-paths logic. We define the $2k$-ary predicate $\mathsf{dp}_k(\mathsf{x}_1, \mathsf{y}_1, \ldots, \mathsf{x}_k, \mathsf{y}_k)$, which evaluates true in a graph G if and only if there are paths $P_1, \ldots, P_k$ of G of length at least two between (the interpretations of) x_i and y_i for all $i \in [k]$ such that for every $i, j \in [k]$, $i \neq j$, $V(P_i) \cap V(P_j) = \emptyset$. We let $\mathsf{FOL+DP}$ be the logic obtained from FOL after allowing $\mathsf{dp}_k(\mathsf{x}_1, \mathsf{y}_1, \ldots, \mathsf{x}_k, \mathsf{y}_k), k \geq 1$ as atomic predicates.

The compound logic Θ^{DP}. We define an extension Θ^{DP} of Θ by considering, as the base case, instead of Θ_0, the logic $\Theta_0^{\mathsf{DP}} = \{\sigma \wedge \mu \mid \sigma \in \mathsf{FOL{+}DP} \text{ and } \mu \in \mathsf{EM}[\{\mathsf{E}\}]\}$.

▶ **Theorem 8.** *For every $\theta \in \Theta^{\mathsf{DP}}$, there exists an algorithm that, given a graph G, outputs whether $G \models \theta$ in time $\mathcal{O}_{|\theta|}(n^2)$.*

As we define the alternative $\tilde{\Theta}$ of Θ, we can also define $\tilde{\Theta}^{\mathsf{DP}}$ by taking $\tilde{\Theta}_0^{\mathsf{DP}} = \mathsf{FOL{+}DP}$ as the base case, i.e., by discarding the minor-exclusion from the definition of Θ_0^{DP}. Notice that $\tilde{\Theta}^{\mathsf{DP}}$ contains $\mathsf{FOL{+}DP}$ and can be seen as a natural extension of it. As a corollary of Theorem 8, we get the following analogue of Theorem 6.

▶ **Theorem 9.** *For every $\tilde{\theta} \in \tilde{\Theta}^{\mathsf{DP}}$, there exists an algorithm that, given a graph G, outputs whether $G \models \theta$ in time $\mathcal{O}_{|\theta|,\mathbf{hw}(G)}(n^2)$.*

Theorem 9 contains all results and applications of [42, 43] as a (very) special case. For a visualization of the current meta-algorithmic landscape on subgraph-closed classes, see Figure 3.

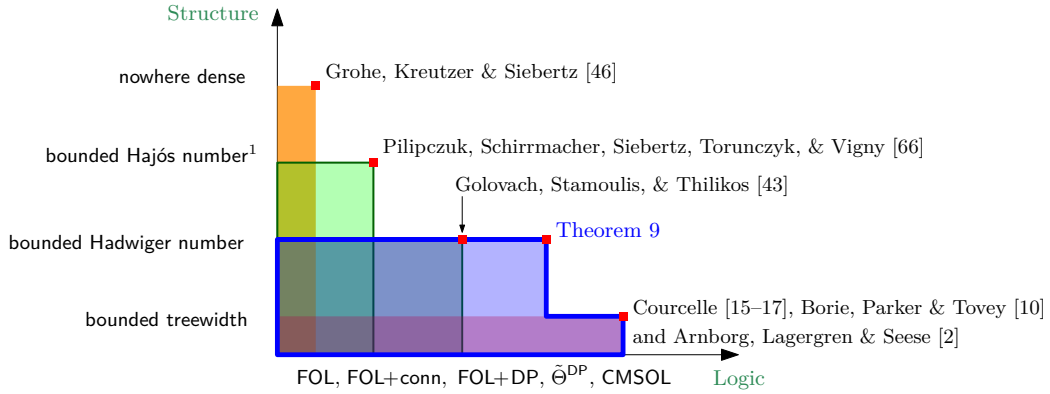


Figure 3 The current meta-algorithmic landscape on subgraph-closed classes and the position of Theorem 9 in it.

4 Further research

The minor-exclusion framework. The graph-structural horizon in both Theorem 5 and Theorem 6 is delimited by minor-exclusion. In the case of Theorem 5, this restriction is applied to the target property defined by μ in the logic Θ, while in Theorem 6 this is the promise combinatorial restriction that yields efficient model-checking for $\tilde{\Theta}$. This restriction is hard-wired in our proof in the way it combines the Flat Wall theorem with Gaifman's theorem. Recently, several efficient algorithms appeared for modification problems targeting or assuming topological minor-freeness (see [1, 36, 49] and the meta-algorithmic results in [66, 74]). For such classes, to achieve efficient model-checking for Θ, or some fragment of it, is an interesting open challenge.

Quadratic time. The proof of Theorem 5 can be seen as a possible "meta-algorithmization" of the *irrelevant vertex technique* introduced by Robertson and Seymour [67], going further than the two known recent attempts in this direction [32, 43]. The main routine of the

[1] The *Hajós number* of a graph G is the maximum k for which G contains K_k as a topological minor.

algorithm transforms the input of the problem to a simpler graph by detecting territories in it that can be safely discarded, therefore producing a simpler instance. This routine is applied repetitively until the graph has "small" treewidth, so that the problem can be solved in linear time by using Courcelle's theorem. This approach gives an algorithm running in quadratic time. Any improvement of this quadratic running time should rely on techniques escaping the above scheme of gradual simplification. The only results in this direction are the cases of making a graph planar by deleting at most k vertices (resp. edges) in [50] (resp. [52]) that run in time $\mathcal{O}_k(n)$.

Further than connectivity closure. One of the key operations defining Θ is the connectivity extension operation, that is, given a sentence φ, to consider the (conjunctive) sentence $\varphi^{(\mathsf{c})}$. We incorporated this operation to our logic in order to express elimination distance modifications (such as those of tree-depth [12] and bridge-depth [11]) where, at each step, we remove some tree-like structure and then we apply the current target sentence to the connected components of the remaining graph. In [24], the notion of *block elimination distance* has been introduced, where the target property is applied to the biconnected components of the remaining graph (instead of the connected components). We are confident that our results can be adapted so to include the biconnectivity extension – or even the 3-connectivity extension, as defined by Tutte's decomposition. However, we prefer to avoid this here as it would add undesirable burden to the statement of our results (and to the proofs as well). Another direction is to consider different versions of $\varphi^{(\mathsf{c})}$. One of them might be a *disjunctive* version, namely $\varphi^{\vee(\mathsf{c})}$, where $G \models \varphi^{\vee(\mathsf{c})}$ if *at least one* of the connected components of G is a model of φ. Another one is a *selective* version, namely $\varphi^{\exists(\mathsf{c})}$, where $G \models \varphi^{\wedge(\mathsf{c})}$ if there is some subset of the connected components of G whose union is a model of φ. Our proof fails if we wish to incorporate any of these two variants of $\varphi^{(\mathsf{c})}$ in Θ. However, it can be easily adapted so to incorporate $\varphi^{\vee(\mathsf{c})}$ in $\tilde{\Theta}$.

Descriptive complexity and the Θ-hierarchy. Recall that $\Theta = \bigcup_{i\in\mathbb{N}} \Theta_i$, where each level of the sentence set Θ_i is defined by adding an extra modulator sentence, followed by some positive Boolean combination of the connectivity closure of the lower level. We extended our result from Θ_1 to every Θ_i because Θ is quite versatile and makes it easier to express more complex hierarchical modification problems. However, it is an open problem whether this hierarchy is proper with respect to the descriptive complexity of the problems that it defines in each of its levels. In simple cases where the modulator sentence asks for a set of bounded size, and under the absence of positive Boolean combinations, it is possible to express any Θ-definable problem using Θ_1. For instance, elimination ordering to some Θ_0-definable class can be straightforwardly expressed in Θ, however with a more technical proof one can also express it in Θ_1 (see [34]). Is this collapse maintained when we consider the full expressive power of Θ? We conjecture a negative answer to this question for both Θ and Θ^{DP}.

References

1 Akanksha Agrawal, Lawqueen Kanesh, Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Deleting, Eliminating and Decomposing to Hereditary Classes Are All FPT-Equivalent. In *Proc. of the 32st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1976–2004, 2022. doi:10.1137/1.9781611977073.79.

2 Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12:308–340, 1991. doi:10.1016/0196-6774(91)90006-K.

3 Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos. A complexity dichotomy for hitting connected minors on bounded treewidth graphs: the chair and the banner draw the boundary. In *Proc. of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 951–970, 2020. doi:10.1137/1.9781611975994.57.

4 Hans L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In *Proc. of the 25th Annual ACM Symposium on Theory of Computing (STOC)*, pages 226–234, 1993. doi:10.1145/167088.167161.

5 Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (Meta) Kernelization. *Journal of the ACM*, 63(5):44:1–44:69, 2016. doi:10.1145/2973749.

6 Mikołaj Bojańczyk. Separator logic and star-free expressions for graphs, 2021. arXiv:2107.13953.

7 Édouard Bonnet, Jan Dreier, Jakub Gajarský, Stephan Kreutzer, Nikolas Mählmann, Pierre Simon, and Szymon Torunczyk. Model checking on interpretations of classes of bounded local cliquewidth. In *Proc. of the 37th Annual ACM/IEEE Symposium on Logic in Computer (LICS)*, pages 54:1–54:13. ACM, 2022. doi:10.1145/3531130.3533367.

8 Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, Pierre Simon, Stéphan Thomassé, and Szymon Torunczyk. Twin-width IV: ordered graphs and matrices. In *Proc. of the 54th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 2022. doi:10.1145/3519935.3520037.

9 Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width I: tractable FO model checking. In *Proc. of the 61st IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 601–612, 2020. doi:10.1109/FOCS46700.2020.00062.

10 Richard B. Borie, R. Gary Parker, and Craig A. Tovey. Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. *Algorithmica*, 7(5-6):555–581, 1992. doi:10.1007/BF01758777.

11 Marin Bougeret, Bart M. P. Jansen, and Ignasi Sau. Bridge-depth characterizes which structural parameterizations of vertex cover admit a polynomial kernel. In *Proc. of the 47th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 168 of *LIPIcs*, pages 16:1–16:19, 2020. doi:10.4230/LIPIcs.ICALP.2020.16.

12 Jannis Bulian and Anuj Dawar. Fixed-parameter tractable distances to sparse graph classes. *Algorithmica*, 79(1):139–158, 2017. doi:10.1007/s00453-016-0235-7.

13 Jianer Chen, Fedor V. Fomin, Yang Liu, Songjian Lu, and Yngve Villanger. Improved algorithms for feedback vertex set problems. *Journal of Computer and System Sciences*, 74(7):1188–1198, 2008. doi:10.1016/j.jcss.2008.05.002.

14 Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved parameterized upper bounds for vertex cover. In *Proc. of the 31st International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 4162 of *LNCS*, pages 238–249, 2006. doi:10.1007/11821069_21.

15 Bruno Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

16 Bruno Courcelle. The monadic second-order logic of graphs III: tree-decompositions, minor and complexity issues. *RAIRO - Theoretical Informatics and Applications*, 26:257–286, 1992. doi:10.1051/ita/1992260302571.

17 Bruno Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*, pages 313–400. World Scientific, 1997.

18 Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*, volume 138 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2012.

19 Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000. doi:10.1007/s002249910009.

20 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

21 Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Minimum bisection is fixed-parameter tractable. *SIAM Journal on Computing*, 48(2):417–450, 2019. doi:10.1137/140988553.

22 Marek Cygan, Dániel Marx, Marcin Pilipczuk, and Michal Pilipczuk. Hitting forbidden subgraphs in graphs of bounded treewidth. *Information and Computation*, 256:62–82, 2017. doi:10.1016/j.ic.2017.04.009.

23 Anuj Dawar, Martin Grohe, and Stephan Kreutzer. Locally excluding a minor. In *Proc. of the 21st IEEE Symposium on Logic in Computer Science (LICS)*, pages 270–279, 2007. doi:10.1109/LICS.2007.31.

24 Oznur Y. Diner, Archontia C. Giannopoulou, Giannos Stamoulis, and Dimitrios M. Thilikos. Block elimination distance. In *Proc. of the 50th International Workshop on Graph Theoretic Concepts in Computer Science (WG)*, volume 12911 of *LNCS*, 2002. doi:10.1007/978-3-030-86838-3_3.

25 Jan Dreier and Peter Rossmanith. Approximate evaluation of first-order counting queries. In Dániel Marx, editor, *Proc. of the 32nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1720–1739. SIAM, 2021. doi:10.1137/1.9781611976465.104.

26 Zdeněk Dvořák, Daniel Král, and Robin Thomas. Testing first-order properties for subclasses of sparse graphs. *Journal of the ACM*, 60(5):36:1–36:24, 2013. doi:10.1145/2499483.

27 Eduard Eiben, Robert Ganian, Thekla Hamm, and O-joung Kwon. Measuring what matters: A hybrid approach to dynamic programming with treewidth. *Journal of Computer and System Sciences*, 121:57–75, 2021. doi:10.1016/j.jcss.2021.04.005.

28 Michael R. Fellows and Michael A. Langston. Nonconstructive tools for proving polynomial-time decidability. *Journal of the ACM*, 35(3):727–739, 1988. doi:10.1145/44483.44491.

29 Samuel Fiorini, Nadia Hardy, Bruce A. Reed, and Adrian Vetta. Planar graph bipartization in linear time. *Discrete Applied Mathematics*, 156(7):1175–1180, 2008. doi:10.1016/j.dam.2007.08.013.

30 Jörg Flum and Martin Grohe. Fixed-parameter tractability, definability, and model-checking. *SIAM Journal on Computing*, 31(1):113–145, 2001. doi:10.1137/S0097539799360768.

31 Fedor V. Fomin, Petr A. Golovach, Ignasi Sau, Giannos Stamoulis, and Dimitrios M. Thilikos. Compound logics for modification problems, 2021. arXiv:2111.02755.

32 Fedor V. Fomin, Petr A. Golovach, Giannos Stamoulis, and Dimitrios M. Thilikos. An algorithmic meta-theorem for graph modification to planarity and FOL. In *Proc. of the 28th Annual European Symposium on Algorithms (ESA)*, volume 173 of *LIPIcs*, pages 51:1–51:17, 2020. doi:10.4230/LIPIcs.ESA.2020.51.

33 Fedor V. Fomin, Petr A. Golovach, and Dimitrios M. Thilikos. On the parameterized complexity of graph modification to first-order logic properties. *Theory of Computing Systems*, 64(2):251–271, 2020. doi:10.1007/s00224-019-09938-8.

34 Fedor V. Fomin, Petr A. Golovach, and Dimitrios M. Thilikos. Parameterized complexity of elimination distance to first-order logic properties. *ACM Transactions on Computational Logic*, 23(3):17:1–17:35, 2022. doi:10.1145/3517129.

35 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar $\mathcal{F}$-Deletion: Approximation, Kernelization and Optimal FPT Algorithms. In *Proc. of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 470–479, 2012. doi:10.1109/FOCS.2012.62.

36 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Hitting topological minors is FPT. In *Proc. of the 52nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 1317–1326, 2020. doi:10.1145/3357713.3384318.

37 Markus Frick and Martin Grohe. Deciding first-order properties of locally tree-decomposable structures. *Journal of the ACM*, 48(6):1184–1206, 2001. doi:10.1145/504794.504798.

38 Haim Gaifman. On local and non-local properties. In *Proc. of the Herbrand Symposium*, volume 107 of *Studies in Logic and the Foundations of Mathematics*, pages 105–135. Elsevier, 1982. doi:10.1016/S0049-237X(08)71879-2.

39 Jakub Gajarský, Stephan Kreutzer, Jaroslav Nesetril, Patrice Ossona de Mendez, Michal Pilipczuk, Sebastian Siebertz, and Szymon Torunczyk. First-order interpretations of bounded expansion classes. *ACM Transactions on Computational Logic*, 21(4):29:1–29:41, 2020. doi:10.1145/3382093.

40 Archontia C. Giannopoulou, Michal Pilipczuk, Jean-Florent Raymond, Dimitrios M. Thilikos, and Marcin Wrochna. Linear kernels for edge deletion problems to immersion-closed graph classes. *SIAM Journal on Discrete Mathematics*, 35(1):105–151, 2021. doi:10.1137/18M1228839.

41 Petr A. Golovach, Dieter Kratsch, and Daniël Paulusma. Detecting induced minors in AT-free graphs. *Theoretical Computer Science*, 482:20–32, 2013. doi:10.1016/j.tcs.2013.02.029.

42 Petr A. Golovach, Giannos Stamoulis, and Dimitrios M. Thilikos. Model-checking for first-order logic with disjoint paths predicates in proper minor-closed graph classes, 2022. arXiv:2211.01723.

43 Petr A. Golovach, Giannos Stamoulis, and Dimitrios M. Thilikos. Model-checking for first-order logic with disjoint paths predicates in proper minor-closed graph classes. In *Proc. of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3684–3699, 2023. doi:10.1137/1.9781611977554.ch141.

44 Martin Grohe. Logic, graphs, and algorithms. In *Logic and Automata: History and Perspectives, in Honor of Wolfgang Thomas*, volume 2 of *Texts in Logic and Games*, pages 357–422. Amsterdam University Press, 2008. URL: https://eccc.weizmann.ac.il/report/2007/091/.

45 Martin Grohe and Stephan Kreutzer. Methods for algorithmic meta theorems. In *Model Theoretic Methods in Finite Combinatorics - AMS-ASL Joint Special Session*, volume 558, pages 181–206. AMS, 2009. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.395.8282&rep=rep1&type=pdf.

46 Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding first-order properties of nowhere dense graphs. *Journal of the ACM*, 64(3):17:1–17:32, 2017. doi:10.1145/3051095.

47 Martin Grohe and Nicole Schweikardt. First-order query evaluation with cardinality conditions. In *Proc. of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*, pages 253–266. ACM, 2018. doi:10.1145/3196959.3196970.

48 Jiong Guo, Falk Hüffner, and Rolf Niedermeier. A structural view on parameterizing problems: Distance from triviality. In *Proc. of the 1st International Workshop on Parameterized and Exact Computation (IWPEC)*, volume 3162 of *LNCS*, pages 162–173, 2004. doi:10.1007/978-3-540-28639-4_15.

49 Bart M. P. Jansen, Jari J. H. de Kroon, and Michal Włodarczyk. Vertex deletion parameterized by elimination distance and even less. In *Proc. of the 53rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 1757–1769, 2021. doi:10.1145/3406325.3451068.

50 Bart M. P. Jansen, Daniel Lokshtanov, and Saket Saurabh. A near-optimal planarization algorithm. In *Proc. of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1802–1811, 2014. doi:10.1137/1.9781611973402.130.

51 Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce A. Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424–435, 2012. doi:10.1016/j.jctb.2011.07.004.

52 Ken-ichi Kawarabayashi and Bruce A. Reed. Computing crossing number in linear time. In *Proc. of the 39th annual ACM symposium on Theory of computing (STOC)*, pages 382–390, 2007. doi:10.1145/1250790.1250848.

53 Ken-ichi Kawarabayashi, Robin Thomas, and Paul Wollan. A new proof of the flat wall theorem. *Journal of Combinatorial Theory, Series B*, 129:204–238, 2018. doi:10.1016/j.jctb.2017.09.006.

54 Eun Jung Kim, Alexander Langer, Christophe Paul, Felix Reidl, Peter Rossmanith, Ignasi Sau, and Somnath Sikdar. Linear kernels and single-exponential algorithms via protrusion decompositions. *ACM Transactions on Algorithms*, 12(2):21:1–21:41, 2016. doi:10.1145/2797140.

55 Tomasz Kociumaka and Marcin Pilipczuk. Deleting vertices to graphs of bounded genus. *Algorithmica*, 81(9):3655–3691, 2019. doi:10.1007/s00453-019-00592-7.

56 Tuukka Korhonen. A single-exponential time 2-approximation algorithm for treewidth. In *Proc. of the 62nd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 184–192, 2021. doi:10.1109/FOCS52979.2021.00026.

57 Stephan Kreutzer. On the parameterised intractability of monadic second-order logic. In *Proc. of the 18th EACSL Annual Conference on Computer Science Logic (CSL)*, pages 348–363, 2009. doi:10.1007/978-3-642-04027-6_26.

58 Stephan Kreutzer. Algorithmic meta-theorems. In *Finite and Algorithmic Model Theory*, volume 379 of *London Mathematical Society Lecture Note Series*, pages 177–270. Cambridge University Press, 2011. URL: http://www.cs.ox.ac.uk/people/stephan.kreutzer/Publications/amt-survey.pdf.

59 Dietrich Kuske and Nicole Schweikardt. First-order logic with counting. In *Proc. of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005133.

60 John M. Lewis and Mihalis Yannakakis. The Node-Deletion Problem for Hereditary Properties is NP-Complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980. doi:10.1016/0022-0000(80)90060-4.

61 Dániel Marx and Igor Razgon. Fixed-Parameter Tractability of Multicut Parameterized by the Size of the Cutset. *SIAM Journal on Computing*, 43(2):355–388, 2014. doi:10.1137/110855247.

62 Dániel Marx and Ildikó Schlotter. Obtaining a planar graph by vertex deletion. *Algorithmica*, 62(3-4):807–822, 2012. doi:10.1007/s00453-010-9484-z.

63 Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-27875-4.

64 Jaroslav Nesetril, Patrice Ossona de Mendez, Michal Pilipczuk, Roman Rabinovich, and Sebastian Siebertz. Rankwidth meets stability. In *Proc. of the 32nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2014–2033. SIAM, 2021. doi:10.1137/1.9781611976465.120.

65 Jaroslav Nesetril, Patrice Ossona de Mendez, and Sebastian Siebertz. Structural properties of the first-order transduction quasiorder. In *Proc. of the 30th EACSL Annual Conference on Computer Science Logic (CSL)*, volume 216 of *LIPIcs*, pages 31:1–31:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CSL.2022.31.

66 Michal Pilipczuk, Nicole Schirrmacher, Sebastian Siebertz, Szymon Torunczyk, and Alexandre Vigny. Algorithms and data structures for first-order logic with connectivity under vertex failures. In *Proc. of the 49th International Colloquium on Automata, Languages, and Programming, (ICALP)*, volume 229 of *LIPIcs*, pages 102:1–102:18, 2022. doi:10.4230/LIPIcs.ICALP.2022.102.

67 Neil Robertson and Paul D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

68 Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004. doi:10.1016/j.jctb.2004.08.001.

69 Ignasi Sau and Uéverton dos Santos Souza. Hitting forbidden induced subgraphs on bounded treewidth graphs. *Information and Computation*, 281:104812, 2021. doi:10.1016/j.ic.2021.104812.

70 Ignasi Sau, Giannos Stamoulis, and Dimitrios M. Thilikos. An FPT-Algorithm for Recognizing k-Apices of Minor-Closed Graph Classes. In *Proc. of the 47th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 168 of *LIPIcs*, pages 95:1–95:20, 2020. doi:10.4230/LIPIcs.ICALP.2020.95.

71 Ignasi Sau, Giannos Stamoulis, and Dimitrios M. Thilikos. A more accurate view of the Flat Wall Theorem, 2021. arXiv:2102.06463.

72 Ignasi Sau, Giannos Stamoulis, and Dimitrios M. Thilikos. k-apices of minor-closed graph classes. II. Parameterized algorithms. *ACM Transactions on Algorithms*, 18(3), 2022. doi:10.1145/3519028.

73 Ignasi Sau, Giannos Stamoulis, and Dimitrios M. Thilikos. k-apices of minor-closed graph classes. I. Bounding the obstructions. *Journal of Combinatorial Theory, Series B*, 161:180–227, 2023. doi:10.1016/j.jctb.2023.02.012.

74 Nicole Schirrmacher, Sebastian Siebertz, Giannos Stamoulis, Dimitrios M. Thilikos, and Alexandre Vigny. Model checking disjoint-paths logic on topological-minor-free graph classes, 2023. arXiv:2302.07033.

75 Nicole Schirrmacher, Sebastian Siebertz, and Alexandre Vigny. First-Order Logic with Connectivity Operators. In *Proc. of the 30th EACSL Annual Conference on Computer Science Logic (CSL)*, volume 216 of *LIPIcs*, pages 34:1–34:17, 2022. doi:10.4230/LIPIcs.CSL.2022.34.

76 Detlef Seese. Linear time computable problems and first-order descriptions. *Mathematical Structures in Computer Science*, 6(6):505–526, 1996. doi:10.1017/S0960129500070079.

Cliques in High-Dimensional Geometric Inhomogeneous Random Graphs

Tobias Friedrich
Hasso Plattner Institute, Universität Potsdam, Germany

Andreas Göbel
Hasso Plattner Institute, Universität Potsdam, Germany

Maximilian Katzmann
Karlsruhe Institute of Technology, Germany

Leon Schiller
Hasso Plattner Institute, Universität Potsdam, Germany

Abstract

A recent trend in the context of graph theory is to bring theoretical analyses closer to empirical observations, by focusing the studies on random graph models that are used to represent practical instances. There, it was observed that geometric inhomogeneous random graphs (GIRGs) yield good representations of complex real-world networks, by expressing edge probabilities as a function that depends on (heterogeneous) vertex weights and distances in some underlying geometric space that the vertices are distributed in. While most of the parameters of the model are understood well, it was unclear how the dimensionality of the ground space affects the structure of the graphs.

In this paper, we complement existing research into the dimension of geometric random graph models and the ongoing study of determining the dimensionality of real-world networks, by studying how the structure of GIRGs changes as the number of dimensions increases. We prove that, in the limit, GIRGs approach non-geometric inhomogeneous random graphs and present insights on how quickly the decay of the geometry impacts important graph structures. In particular, we study the expected number of cliques of a given size as well as the clique number and characterize phase transitions at which their behavior changes fundamentally. Finally, our insights help in better understanding previous results about the impact of the dimensionality on geometric random graphs.

2012 ACM Subject Classification Mathematics of computing → Random graphs; Theory of computation → Computational geometry

Keywords and phrases random graphs, geometry, dimensionality, cliques, clique number, scale-free networks

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.62

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2302.04113 [22]

Funding *Andreas Göbel*: is funded by the project PAGES (project No. 467516565) of the German Research Foundation (DFG).

1 Introduction

Networks are a powerful tool to model all kinds of processes that we interact with in our day-to-day lives. From connections between people in social networks, to the exchange of information on the internet, and on to how our brains are wired, networks are everywhere. Consequently, they have been in the focus of computer science for decades. There, one of the most fundamental techniques used to model and study networks are *random graph models*. Such a model defines a probability distribution over graphs, which is typically done by

specifying a random experiment on how to construct the graph. By analyzing the rules of the experiment, we can then derive structural and algorithmic properties of the resulting graphs. If the results match what we observe on real-world networks, i.e., if the model represents the graphs we encounter in practice well, then we can use it to make further predictions that help us understand real graphs and utilize them more efficiently.

The quest of finding a good model starts several decades ago, with the famous Erdős-Rényi (ER) random graphs [19, 24]. There, all edges in the graph exist independently with the same probability. Due to its simplicity, this model has been studied extensively. However, because the degree distribution of the resulting graphs is rather homogeneous and they lack clustering (due to the independence of the edges), the model is not considered to yield good representations of real graphs. In fact, many networks we encounter in practice feature a degree distribution that resembles a power-law [3, 38, 39] and the clustering coefficient (the probability for two neighbors of a vertex to be adjacent) is rather high [35, 40]. To overcome these drawbacks, the initial random graph model has been adjusted in several ways.

In *inhomogeneous random graphs (IRGs)*, often referred to as *Chung-Lu random graphs*, each vertex is assigned a weight and the probability for two vertices to be connected by an edge is proportional to the product of the weights [1, 11, 12]. As a result, the expected degrees of the vertices in the resulting graphs match their weight. While assigning weights that follow a power-law distribution yields graphs that are closer to the complex real-world networks, the edges are still drawn independently, leading to vanishing clustering coefficients.

A very natural approach to facilitate clustering in a graph model is to introduce an underlying geometry. This was done first in *random geometric graphs (RGGs)*, where vertices are distributed uniformly at random in the Euclidean unit square and any two are connected by an edge if their distance lies below a certain threshold, i.e., the neighborhood of a vertex lives in a disk centered at that vertex [36]. Intuitively, two vertices that connect to a common neighbor cannot be too far away from each other, increasing the probability that they are connected by an edge themselves. In fact, random geometric graphs feature a non-vanishing clustering coefficient [13]. However, since all neighborhood disks have the same size, they all have roughly the same expected degree, again, leading to a homogeneous degree distribution.

To get a random graph model that features a heterogeneous degree distribution *and* clustering, the two mentioned adjustments were recently combined to obtain *geometric inhomogeneous random graphs (GIRGs)* [28]. There, vertices are assigned a weight *and* a position in some underlying geometric space and the probability for two vertices to connected increases with the product of the weights but decreases with increasing geometric distance between them. As a result, the generated graphs have a non-vanishing clustering coefficient and, with the appropriate choice of the weight sequence, they feature a power-law degree distribution. Additionally, recent empirical observations indicate that GIRGs represent real-world networks well with respect to certain structural and algorithmic properties [5].

We note that GIRGs are not the first model that exhibits a heterogeneous degree distribution and clustering. In fact, *hyperbolic random graphs (HRGs)* [30] feature these properties as well and have been studied extensively before (see, e.g., [7, 20, 21, 23, 26]). However, in the pursuit of finding good models to represent real-world networks, GIRGs introduce a parameter that sets them apart from prior models: the choice of the underlying geometric space and, more importantly, the dimensionality of that space.

Unfortunately, this additional parameter that sets GIRGs apart from previous models, has not gained much attention at all. In fact, it comes as a surprise that, while the underlying dimensionality of real-world networks is actively researched [2, 8, 15, 25, 31] and there is a large body of research examining the impact of the dimensionality on different homogeneous graph

models [13, 17, 18] with some advancements being made on hyperbolic random graphs [41], the effects of the dimension on the structure of GIRGs have only been studied sparsely. For example, while it is known that GIRGs exhibit a clustering coefficient of $\Theta(1)$ for any fixed dimension [28], it is not known how the hidden constants scale with the dimension.

In this paper, we initiate the study of the impact of the dimensionality on GIRGs. In particular, we investigate the influence of the underlying geometry as the dimensionality increases, proving that GIRGs converge to their non-geometric counterpart (IRGs) in the limit. With our results we are able to explain seemingly disagreeing insights from prior research on the impact of dimensionality on geometric graph models. Moreover, by studying the clique structure of GIRGs and its dependence on the dimension d, we are able to quantify how quickly the underlying geometry vanishes. In the following, we discuss our results in greater detail. We note that, while we give general proof sketches for our results, the complete proofs are deferred to the full version [22].

2 (Geometric) Inhomogeneous Random Graphs

Before stating our results in greater detail, let us recall the definitions of the two graph models we mainly work with throughout the paper.

Inhomogeneous Random Graphs (IRGs). The model of inhomogeneous random graphs was introduced by Chung and Lu [1, 11, 12] and is a natural generalization of the Erdős-Rényi model. Starting with a vertex set V of n vertices, each $v \in V$ is assigned a weight w_v. Each edge $\{u, v\} \in \binom{V}{2}$ is then independently present with probability

$$\Pr[u \sim v] = \min\left\{1, \frac{\lambda w_u w_v}{n}\right\},$$

for some constant $\lambda > 0$ controlling the average degree of the resulting graph. Note that assigning the same weight to all vertices yields the same connection probability as in Erdős-Rényi random graphs. For the sake of simplicity, we define $\kappa_{uv} = \min\{\lambda w_u w_v, n\}$ such that $\Pr[u \sim v] = \kappa_{uv}/n$. Additionally, for a set of vertices $U_k = \{v_1, \dots, v_k\}$ with weights $w_1, \dots, w_k$, we introduce the shorthand notation $\kappa_{ij} = \kappa_{v_i v_j}$ and write $\{\kappa\}^{(k)} = \{\kappa_{ij} \mid 1 \leq i < j \leq k\}$.

Throughout the paper, we mainly focus on inhomogeneous random graphs that feature a power-law degree distribution in expectation, which is obtained by sampling the weights accordingly. More precisely, for each $v \in V$, we sample a weight w_v from the Pareto distribution $\mathcal{P}$ with parameters $1 - \beta, w_0$ and distribution function

$$\Pr[w_v \leq x] = 1 - \left(\frac{x}{w_0}\right)^{1-\beta}.$$

Then the density of w_v is $\rho_{w_v}(x) = \frac{\beta-1}{w_0^{1-\beta}} x^{-\beta}$. Here, $w_0 > 0$ is a constant that represents a lower bound on the weights in the graph and β denotes the power-law exponent of the resulting degree distribution. Throughout the paper, we assume $\beta > 2$ such that a single weight has finite expectation (and thus the average degree in the graph is constant), but possibly infinite variance. We denote a graph obtained by utilizing the above weight distribution and connection probabilities with $\mathrm{IRG}(n, \beta, w_0)$. For a fixed weight sequence $\{w\}_1^n$, we denote the corresponding graph by $\mathrm{IRG}(\{w\}_1^n)$.

Geometric Inhomogeneous Random Graphs (GIRGs). Geometric inhomogeneous random graphs are an extension of IRGs, where in addition to the weight, each vertex v is also equipped with a position $\mathbf{x}_v$ in some geometric space and the probability for edges to form depends on their weights and the distance in the underlying space [28]. While, in its raw form, the GIRG framework is rather general, we align our paper with existing analysis on GIRGs [6, 29, 34] and consider the d-dimensional torus $\mathbb{T}^d$ equipped with L_∞-norm as the geometric ground space. More precisely, in what we call the *standard* GIRG model, the positions $\mathbf{x}$ of the vertices are drawn independently and uniformly at random from $\mathbb{T}^d$, according to the standard Lebesgue measure. We denote the i-th component of $\mathbf{x}_v$ by $\mathbf{x}_{vi}$. Additionally, the geometric distance between two points $\mathbf{x}_u$ and $\mathbf{x}_v$, is given by

$$d(\mathbf{x}_u, \mathbf{x}_v) = \|\mathbf{x}_u - \mathbf{x}_v\|_\infty = \max_{1\le i\le d}\{|\mathbf{x}_{ui} - \mathbf{x}_{vi}|_C\},$$

where $|\cdot|_C$ denotes the distance on the circle, i.e,

$$|\mathbf{x}_{ui} - \mathbf{x}_{vi}|_C = \min\{|\mathbf{x}_{ui} - \mathbf{x}_{vi}|, 1 - |\mathbf{x}_{ui} - \mathbf{x}_{vi}|\}.$$

In a standard GIRG, two vertices $u \neq v$ are adjacent if and only if their distance $d(\mathbf{x}_u, \mathbf{x}_v)$ in the torus is less than or equal to a *connection threshold* t_{uv}, which is given by

$$t_{uv} = \frac{1}{2}\left(\frac{\lambda w_u w_v}{n}\right)^{1/d} = \left(\frac{w_u w_v}{\tau n}\right)^{1/d},$$

where $\tau = 2^d/\lambda$. Using L_∞ is motivated by the fact that it is the most widely used metric in the literature because it is arguably the most natural metric on the torus. In particular, it has the "nice" property that the ball of radius r is a cube and "fits" entirely into $\mathbb{T}^d$ for all $0 \le r \le 1$.

Note that, as a consequence of the above choice, the marginal connection probability $\Pr[u \sim v]$ is the same as in the IRG model, i.e., $\Pr[u \sim v] = \kappa_{uv}/n$. However, while the probability that any given edge is present is the same as in the IRG model, the edges in the GIRG model are *not* drawn independently. We denote a graph obtained by the procedure described above with $\mathrm{GIRG}(n, \beta, w_0, d)$. As for IRGs, we write $\mathrm{GIRG}(\{w\}_1^n, d)$ when considering standard GIRGs with a fixed weight sequence $\{w\}_1^n$.

As mentioned above, the standard GIRG model is a commonly used instance of the more general GIRG framework [28]. There, different geometries and distance functions may be used. For example, instead of L_∞-norm, any L_p-norm for $1 \le p < \infty$ may be used. Then, the distance between two vertices u, v is measured as

$$\|\mathbf{x}_u - \mathbf{x}_v\|_p := \begin{cases} \left(\sum_{i=1}^d |\mathbf{x}_{ui} - \mathbf{x}_{vi}|^p\right)^{1/p} & \text{if } p < \infty \\ \max_{1\le i\le d}\{|\mathbf{x}_{ui} - \mathbf{x}_{vi}|\} & \text{otherwise.} \end{cases}$$

With this choice, the volume (Lebesgue measure) of the ball $B_p(r)$ of radius r under L_p-norm is equal to the probability that a vertex u falls within distance at most r of v (if $r = o(1)$). We denote this volume by $\nu(r)$. We call the corresponding graphs *standard GIRGs with any L_p-norm* and note that some of our results extend to this more general model. Finally, whenever our insights consider an even broader variant of the model (e.g., variable ground spaces, distances functions, weight distributions), we say that they hold for *any GIRG* and mention the constraints explicitly.

3 Asymptotic Equivalence

Our first main observations is that large values of d diminish the influence of the underlying geometry until, at some point, our model becomes strongly equivalent to its non-geometric counterpart, where edges are sampled independently of each other. We prove that the *total variation distance* between the distribution over all graphs of the two models tends to zero as n is kept fixed and $d \to \infty$. We define the total variation distance of two probability measures P and Q on the measurable space $(\Omega, \mathcal{F})$ as

$$\|P, Q\|_{\mathrm{TV}} = \sup_{A \in \mathcal{F}} |P(A) - P(B)| = \frac{1}{2} \sum_{\omega \in \Omega} |P(\omega) - Q(\omega)|,$$

where the second equality holds if Ω is countable. In our case, Ω is the set $\mathcal{G}(n)$ of all possible graphs on n vertices, and P, Q are distributions over these graphs. If G_1, G_2 are two random variables mapping to Ω, we refer to $\|G_1, G_2\|_{\mathrm{TV}}$ as the total variation distance of the induced probability measures by G_1 and G_2, respectively. Informally, this measures the maximum difference in the probability that any graph G is sampled by G_1 and G_2.

▶ **Theorem 1.** *Let $\mathcal{G}(n)$ be the set of all graphs with n vertices, let $\{w\}_1^n$ be a weight sequence, and consider $G_{IRG} = \mathrm{IRG}(\{w\}_1^n) \in \mathcal{G}(n)$ and a standard GIRG $G_{GIRG} = \mathrm{GIRG}(\{w\}_1^n, d) \in \mathcal{G}(n)$ with any L_p-norm. Then,*

$$\lim_{d \to \infty} \|G_{GIRG}, G_{IRG}\|_{TV} = 0.$$

We note that this theorem holds for arbitrary weight sequences that do not necessarily follow a power law and for arbitrary L_p-norms used to define distances in the ground space. For $p \in [1, \infty)$, the proof is based on the application of a multivariate central limit theorem [37], in a similar way as used to prove a related statement for *spherical random geometric graphs (SRGGs)*, i.e., random geometric graphs with a hypersphere as ground space [17]. Our proof generalizes this argument to arbitrary L_p-norms and arbitrary weight sequences. For the case of L_∞-norm, we present a proof based on the inclusion-exclusion principle and the bounds we develop in the full version [22, Section 4].

Remarkably, while a similar behavior was previously established for SRGGs, there exist works indicating that RGGs on the hypercube do not converge to their non-geometric counterpart [13, 18] as $d \to \infty$. We show that this apparent disagreement is due to the fact that the torus is a homogeneous space while the hypercube is not. In fact, our proof shows that GIRGs on the hypercube *do* converge to a non-geometric model in which edges are, however, not sampled independently. This lack of independence is because, on the hypercube, there is a positive correlation between the distances from two vertices to a given vertex, leading to a higher tendency to form clusters, as was observed experimentally [18]. Due to the homogeneous nature of the torus, the same is not true for GIRGs and the model converges to the plain IRG model with independent edges.

4 Clique Structure

To quantify for which dimensions d the graphs in the GIRG model start to behave similar to IRGs, we investigate the number and size of cliques. Previous results on SRGGs indicate that the dimension of the underlying space heavily influences the clique structure of the model [4, 17]. However, it was not known how the size and the number of cliques depends on d if we use the torus as our ground space, and how the clique structure in high-dimensions behaves for inhomogeneous weights.

Table 1 Asymptotic behavior of the expected number of k-cliques. The behavior in the first column is the same as in hyperbolic random graphs [7], and the behavior in the third column is the same as in the IRG model [14]. Results marked with * were previously known for constant k [34].

	$\mathbb{E}[K_k]$ for $k \geq 4$		
	$d = \Theta(1)$	$d = o(\log(n))$	$d = \omega(\log(n))$
$2 < \beta < 3, k > \frac{2}{3-\beta}$	$n^{\frac{k}{2}(3-\beta)}\Theta(k)^{-k}*$	$n^{\frac{k}{2}(3-\beta)}\Theta(k)^{-k}$	$n^{\frac{k}{2}(3-\beta)}\Theta(k)^{-k}$
$2 < \beta < 3, k < \frac{2}{3-\beta}$	$n\Theta(k)^{-k}*$	$ne^{-\Theta(1)dk}\Theta(k)^{-k}$	$n^{\frac{k}{2}(3-\beta)}\Theta(k)^{-k}$
$\beta > 3$	$n\Theta(k)^{-k}$	$ne^{-\Theta(1)dk}\Theta(k)^{-k}$	$o(1)$

We give explicit bounds on the expected number of cliques of a given size k, which we afterwards turn into bounds on the *clique number* $\omega(G)$, i.e., the size of the largest clique in the graph G. While the expected number of cliques in the GIRG model was previously studied by Michielan and Stegehuis [34] when the power-law exponent of the degree distribution satisfies $\beta \in (2,3)$, to the best of our knowledge, the clique number of GIRGs remains unstudied even in the case of constant (but arbitrary) dimensionality. We close this gap, reproduce the existing results, and extend them to the case $\beta \geq 3$ and the case where d can grow as a function of the number of vertices n in the graph. Furthermore, our bounds for the case $\beta \in (2,3)$ are more explicit and complement the work of Michielan and Stegehuis, who expressed the (rescaled) asymptotic number of cliques as converging to a non-analytically solvable integral. Furthermore, we show that the clique structure in our model eventually behaves asymptotically like that of an IRG if the dimension is sufficiently large. In summary, our main contributions are outlined in Tables 1, 2, and Table 3.

We observe that the structure of the cliques undergoes three phase transitions in the size of the cliques k, the dimension d, and the power-law exponent β.

Transition in k. When $\beta \in (2,3)$ and $d \in o(\log(n))$, the first transition is at $k = \frac{2}{3-\beta}$, as was previously observed for hyperbolic random graphs [7] and for GIRGs of constant dimensionality [34]. The latter work explains this behavior by showing that for $k < \frac{2}{3-\beta}$, the number of cliques is strongly dominated by "geometric" cliques forming among vertices whose distance is of order $n^{-1/d}$ regardless of their weight. For $k > \frac{2}{3-\beta}$, on the other hand, the number of cliques is dominated by "non-geometric" cliques forming among vertices with weights in the order of $\sqrt{n}$. This behavior is in contrast to the behavior of cliques in the IRG model, where this phase transition does not exist and where the expected number of k cliques is $\Theta\left(n^{\frac{k}{2}(3-\beta)}\right)$ for all $k \geq 3$ (if $\beta \in (2,3)$) [14].

Transition in d. Still assuming $\beta \in (2,3)$, the second phase transition occurs as d becomes superlogarithmic. More precisely, we show that in the high-dimensional regime, where $d = \omega(\log(n))$, the phase transition in k vanishes, as the expected number of cliques of size $k \geq 4$ behaves asymptotically like its counterpart in the IRG model. Nevertheless, we can still differentiate the two models as long as $d = o(\log^{3/2}(n))$, by counting triangles among low degree vertices as can be seen in Table 2.

The reason for this behavior is that the expected number of cliques in the case $d = \omega(\log(n))$ is already dominated by cliques forming among vertices of weight close to $\sqrt{n}$. For those, the probability that a clique is formed already behaves like in an IRG although, for vertices of small weight, said probability it is still larger.

Table 2 Asymptotic behavior of the expected number of triangles. The case $\beta = \infty$ refers to the case of constant weights. While in the case $\beta < 3$, the number of triangles already behaves like that of the IRG model if $d = \omega(\log(n))$, in the case $\beta > 3$, the number of triangles remains superconstant as long as $d = o\left(\log^{3/2}(n)\right)$.

Expected number of triangles $\mathbb{E}[K_3]$			
	$d = o(\log(n))$	$d = \omega(\log(n))$	$d = \omega(\log^2(n))$
$2 < \beta < \frac{7}{3}$	$n^{\frac{3}{2}(3-\beta)}\Theta(1)$	$n^{\frac{3}{2}(3-\beta)}\Theta(1)$	$n^{\frac{3}{2}(3-\beta)}\Theta(1)$
$\frac{7}{3} < \beta < 3$	$ne^{-\Theta(1)d}\Theta(1)$	$n^{\frac{3}{2}(3-\beta)}\Theta(1)$	$n^{\frac{3}{2}(3-\beta)}\Theta(1)$
$\beta > 3$	$ne^{-\Theta(1)d}\Theta(1)$	$\Omega\left(\exp\left(\frac{\ln^3(n)}{d^2}\right)\right)$	$\Theta(1)$
$\beta = \infty$	$ne^{-\Theta(1)d}\Theta(1)$	$\Theta\left(\exp\left(\frac{\ln^3(n)}{d^2}\right)\right)$	$\Theta(1)$

Regarding the clique number, in the case $\beta > 3$, we observe a similar phase transition in d. For constant d, the clique number of a GIRG is $\Theta(\log(n)/\log\log(n)) = \omega(1)$. We find that this asymptotic behavior remains unchanged if $d = \mathcal{O}(\log\log(n))$. However, if $d = \omega(\log\log(n))$ but $d = o(\log(n))$, the clique number scales as $\Theta(\log(n)/d)$, which is still superconstant. Additionally if $d = \omega(\log(n))$, we see that, again, GIRGs show the same behavior as IRGs. That is, there are asymptotically no cliques of size larger than 3.

Transition in β. The third phase transition occurs at $\beta = 3$ in the high-dimensional case, which is in line with the fact that networks with a power-law exponent $\beta \in (2, 3)$ contain with high probability (w.h.p., meaning with probability $1 - O(1/n)$) a densely connected "heavy core" of $\Theta\left(n^{\frac{1}{2}(3-\beta)}\right)$ vertices with weight $\sqrt{n}$ or above, which vanishes if β is larger than 3. This heavy core strongly dominates the number of cliques of sufficient size and explains why the clique number is $\Theta\left(n^{\frac{1}{2}(3-\beta)}\right)$ regardless of d if $\beta \in (2, 3)$. As β grows beyond 3, the core disappears and leaves only very small cliques. Accordingly for $\beta > 3$ IRGs contain asymptotically almost surely (a.a.s., meaning with probability $1 - o(1)$) no cliques of size greater than 3. In contrast to that, for GIRGs of dimension $d = o(\log(n))$ (and HRGs), the clique number remains superconstant and so does the number of k-cliques for any constant $k \geq 3$. If $d = \omega(\log(n))$, there are no cliques of size greater than 3 like in an IRG. However, as noted before, GIRGs feature many more triangles than IRGs as long as $d = o(\log^{3/2}(n))$.

Beyond the three mentioned phase transitions, we conclude that, for constant d, the main difference between GIRGs and IRGs is that the former contain a significant number of cliques that form among vertices of low weight, whereas, in the latter model only high-weight vertices contribute significantly to the total number of cliques. In fact, here, the expected number of k cliques in the heavy core is already of the same order as the total expectation of K_k in the whole graph. Similarly, in the GIRG model, the expected number of cliques forming in the low-weight subgraph $G_{\leq w}$ for some constant w, is already of the same order as the total number of cliques if $k < \frac{2}{3-\beta}$ or $\beta \geq 3$ (otherwise, this number is, again, dominated by cliques from the heavy core).

The proofs of our results (i.e., the ones in the above tables) are mainly based on bounds on the probability that a set of k randomly chosen vertices forms a clique. To obtain concentration bounds on the number of cliques as needed for deriving bounds on the clique number, we use the second moment method and Chernoff bounds.

For the case of $d = \omega(\log(n))$, many of our results are derived from the following general insight. We show that for and all $\beta > 2$, the probability that a set of vertices forms a clique already behaves similar as in the IRG model if the weights of the involved nodes are sufficiently large. For $d = \omega(\log(n)^2)$, this holds in the entire graph, that is, regardless of the weights of the involved vertices. In fact our statement holds even more generally. That is, the described behavior not only applies to the probability that a clique is formed but also to the probability that any set of edges (or a superset thereof) is created.

▶ **Theorem 2.** *Let G be a standard GIRG and let $k \geq 3$ be a constant. Furthermore, let $U_k = \{v_1, \dots, v_k\}$ be a set of vertices chosen uniformly at random and let $\{\kappa\}^{(k)} = \{\kappa_{ij} \mid 1 \leq i, j \leq k\}$ describe the pairwise product of weights of the vertices in U_k. Let $E(U_k)$ denote the (random) set of edges formed among the vertices in U_k. Then, for $d = \omega\left(\log^2(n)\right)$ and any set of edges $\mathcal{A} \subseteq \binom{U_k}{2}$,*

$$\Pr\left[E(U_k) \supseteq \mathcal{A} \mid \{\kappa\}^{(k)}\right] = (1 \pm o(1)) \prod_{\{i,j\} \in \mathcal{A}} \frac{\kappa_{ij}}{n}.$$

If $d = \omega\left(\log(n)\right)$,

$$\Pr\left[E(U_k) \supseteq \mathcal{A} \mid \{\kappa\}^{(k)}\right] = (1 \pm o(1)) \prod_{\{i,j\} \in \mathcal{A}} \left(\frac{\kappa_{ij}}{n}\right)^{1 \mp \mathcal{O}\left(\frac{\log(n)}{d}\right)}.$$

For the proof we derive elementary bounds on the probability of the described events and use series expansions to investigate their asymptotic behavior. Remarkably, in contrast to our bounds for the case $d = o(\log(n))$, the high-dimensional case requires us to pay closer attention to the topology of the torus.

We leverage the above theorem to prove that GIRGs eventually become equivalent to IRGs with respect to the total variation distance. Theorem 2 already implies that the expected number of cliques in a GIRG is asymptotically the same as in an IRG for all $k \geq 3$ and all $\beta > 2$ if $d = \omega(\log^2(n))$. However, we are able to show that the expected number of cliques for $\beta \in (2, 3)$ actually already behaves like that of an IRG if $d = \omega(\log(n))$. The reason for this is that the clique probability among high-weight vertices starts to behave like that of an IRG earlier than it is the case for low-weight vertices and cliques forming among these high-weight vertices already dominate the number of cliques. Moreover, the clique number behaves like that of an IRG if $d = \omega(\log(n))$ for all $\beta > 2$. However, the number of triangles among vertices of constant weight asymptotically exceeds that of an IRG as long as $d = o(\log^{3/2}(n))$, which we prove by deriving even sharper bounds on the expected number of triangles. Accordingly, convergence with respect to the total variation distance cannot occur before this point (this holds for all $\beta > 2$).

In contrast to this, for the low-dimensional case (where $d = o(\log(n))$), the underlying geometry still induces strongly notable effects regarding the number of sufficiently small cliques for all $\beta > 2$. However, even here, the expected number of such cliques decays exponentially in dk. The main difficulty in showing this is that we have to handle the case of inhomogeneous weights, which significantly influence the probability that a set of k vertices chosen uniformly at random forms a clique. To this end, we prove the following theorem that bounds the probability that a clique among k vertices is formed if the ratio of the maximal and minimal weight is at most c^d. Note that the vertices forming a star is necessary for a clique to form. For this reason we consider the event $\mathbf{E}^c_{\text{star}}$ of the vertices forming a star centered at the lowest weight vertex. The theorem generalizes a result of Decreusefond et al. [16].

Table 3 Asymptotic behavior of the clique number of G for different values of d in the GIRG model. The behavior of the first column is the same as in hyperbolic random graphs established in [7], and the behavior in the third column is the same as that of IRG graphs established in [27]. All results hold a.a.s. and under L_∞-norm.

	$\omega(G)$		
	$d = \mathcal{O}(\log\log(n))$	$d = o(\log(n))$	$d = \omega(\log(n))$
$\beta < 3$	$\Theta\left(n^{(3-\beta)/2}\right)$	$\Theta\left(n^{(3-\beta)/2}\right)$	$\Theta\left(n^{(3-\beta)/2}\right)$
$\beta = 3$	$\Theta\left(\frac{\log(n)}{\log\log(n)}\right)$	$\Omega\left(\frac{\log(n)}{d}\right)$	$\mathcal{O}(1)$
$\beta > 3$	$\Theta\left(\frac{\log(n)}{\log\log(n)}\right)$	$\Theta\left(\frac{\log(n)}{d}\right)$	≤ 3
	equivalent to HRGs [7]		equivalent to IRGs [27]

▶ **Theorem 3.** *Let G be a standard GIRG and consider $k \geq 3$. Furthermore, let $U_k = \{v_1, v_2, \dots, v_k\}$ be a set of vertices chosen uniformly at random and assume without loss of generality that $w_1 \leq \dots \leq w_k$. Let $\mathbf{E}^c_{star}$ be the event that v_1 connects to all vertices in $U_k \setminus \{v_1\}$ and that $w_k \leq c^d w_1$ for some constant $c \geq 1$ with $c^2 \left(w_1^2/(\tau n)\right)^{1/d} \leq 1/4$. Then, the probability that U_k is a clique conditioned on $\mathbf{E}^c_{star}$ fulfills*

$$\left(\frac{1}{2}\right)^{d(k-1)} k^d \leq \Pr\left[U_k \text{ is clique} \mid \mathbf{E}^c_{star}\right] \leq c^{d(k-2)} \left(\frac{1}{2}\right)^{d(k-1)} k^d.$$

Building on the variant by Decreusefond et al. [16], we provide an alternative proof of the original statement, showing that the clique probability conditioned on the event $\mathbf{E}^c_{\text{star}}$ is monotonous in the weight of all other vertices. Remarkably, this only holds if we condition on the event that the center of our star is of minimal weight among the vertices in U_k.

We apply Theorem 3 to bound the clique probability in the whole graph (where the ratio of the maximum and minimum weight of vertices in U_k is not necessarily bounded). Afterwards, we additionally use Chernoff bounds and the second moment method to bound the clique number.

5 Relation to Previous Analyses

In the following, we discuss how our results compare to insights obtained on similar graph models that (apart from not considering weighted vertices) mainly differ in the considered ground space. We not that, in the following, we consider GIRGs with uniform weights in order to obtain a valid comparison.

Random Geometric Graphs on the Sphere. Our results indicate that the GIRG model on the torus behaves similarly to the model of Spherical Random Geometric Graphs (SRGGs) in the high-dimensional case. In this model, vertices are distributed on the surface of a $d-1$ dimensional sphere and an edge is present whenever the Euclidean distance between two points (measured by their inner product) falls below a given threshold. Analogous to the behavior of GIRGs, when keeping n fixed and considering increasing $d \to \infty$, this model converges to its non-geometric counterpart, which in their case is the Erdős–Rényi model [17]. It is further shown that the clique number converges to that of an Erdős–Rényi graph (up to a factor of $1 + o(1)$) if $d = \omega(\log^3(n))$.

Although the overall behavior of SRGGs is similar to that of GIRGs, the magnitude of d in comparison to n at which non-geometric features become dominant seems to differ. In fact, it is shown in [10, proof of Theorem 3] that the expected number of triangles in sparse SRGGs still grows with n as long as $d = o(\log^3(n))$, whereas its expectation is constant in the non-geometric, sparse case (as for Erdős–Rényi graphs). On the other hand, in the GIRG model, we show that the expected number of triangles in the sparse case converges to the same (constant) value as that of the non-geometric model if only $d = \omega(\log^{3/2}(n))$. This indicates that, in the high-dimensional regime, differences in the nature of the underlying geometry result in notably different behavior, whereas in the case of constant dimensionality, the models are often assumed to behave very similarly.

Random Geometric Graphs on the Hypercube. The work of Dall and Christensen [13] and the recent work of Erba et al. [18] show that RGGs on the hypercube do *not* converge to Erdős–Rényi graphs as n is fixed and $d \to \infty$. However, our results imply that this is the case for RGGs on the torus. These apparent disagreements are despite the fact that Erba et al. use a similar central limit theorem for conducting their calculations and simulations [18].

The tools established in our paper yield an explanation for this behavior. Our proof of Theorem 1 relies on the fact that, for independent zero-mean variables $Z_1, \dots, Z_d$, the covariance matrix of the random vector $Z = \sum_{i=1}^{d} Z_i$ is the identity matrix. This, in turn, is based on the fact that the torus is a *homogeneous space*, which implies that the probability measure of a ball of radius r (proportional to its Lebesgue measure or volume, respectively) is the same, regardless of where this ball is centered. It follows that the random variables $Z_{(u,v)}$ and $Z_{(u,s)}$, denoting the normalized distances from u to s and v, respectively, are independent. As a result their covariance is 0 although both "depend" on the position of u.

For the hypercube, this is not the case. Although one may analogously define the distance of two vertices as a sum of independent, zero-mean random vectors over all dimensions just like we do in this paper, the random variables $Z_{(u,v)}$ and $Z_{(u,s)}$ do *not* have a covariance of 0.

6 Conjectures & Future Work

While making the first steps towards understanding GIRGs and sparse RGGs on the torus in high dimensions, we encountered several questions whose investigation does not fit into the scope of this paper. In the following, we give a brief overview of our conjectures and possible starting points for future work.

In addition to investigating how the number and size of cliques depends on d, it remains to analyze among which vertices k-cliques form dominantly. For constant d and $\beta \in (2, 3)$ this was previously done by Michielan and Stegehuis who noted that cliques of size $k > \frac{2}{3-\beta}$ are dominantly formed among vertices of weight in the order of $\sqrt{n}$ like in the IRG model, whereas cliques of size $k < \frac{2}{3-\beta}$ dominantly appear among vertices within distance in the order of $n^{-1/d}$ [34]. This characterizes the geometric and non-geometric nature of cliques of size larger and smaller than $\frac{2}{3-\beta}$, respectively. As our work indicates that this phase transition vanishes as $d = \omega(\log(n))$, we conjecture that in this regime cliques of all sizes are dominantly found among vertices of weight in the order $\sqrt{n}$. For the case $\beta \geq 3$ it remains to analyze the position of cliques of all sizes. It would further be interesting to find out where cliques of superconstant size are dominantly formed as previous work in this regard only holds for constant k.

Additionally, it would be interesting to extend our results to a noisy variant of GIRGs. While the focus in this paper lies on the standard GIRGs, where vertices are connected by an edge if their distance is below a given threshold, there is a *temperate* version of the model, where the threshold is softened using a *temperature* parameter. That is, while the

probability for an edge to exist still decreases with increasing distance, we can now have longer edges and shorter non-edges with certain probabilities. The motivation of this variant of GIRGs is based on the fact that real data is often noisy as well, leading to an even better representation of real-world graphs.

We note that we expect our insights to carry over to the temperate model, as long as we have constant temperature. Beyond that, we note that both temperature and dimensionality affect the influence of the underlying geometry. Therefore, it would be interesting to see whether a sufficiently high temperature has an impact on how quickly GIRGs converge to the IRGs.

Furthermore, it remains to investigate the dense case of our model, where the marginal connection probability of any pair of vertices is constant and does not decrease with n. For dense SRGGs, an analysis of the high-dimensional case has shown that the underlying geometry remains detectable as long as $d = o(n^3)$. As mentioned above, GIRGs and their non-geometric counterpart can be distinguished as long as $d = o(\log^{3/2}(n))$, by considering triangles among low-weight vertices. For dense SRGGs the geometry can be detected by counting so-called *signed triangles* [10]. Although for the sparse case, signed triangles have no advantage over ordinary triangles, they are much more powerful in the dense case and might hence prove useful for our model in the dense case as well.

Another crucial question is under which circumstances the underlying geometry of our model remains detectable by means of statistical testing, and when (i.e. for which values of d) our model converges in total variation distance to its non-geometric counterpart. A large body of work has already been devoted to this question for RGGs on the sphere [17, 10, 9, 33, 32] and recently also for random intersection graphs [9]. While the question when these models lose their geometry in the dense case is already largely answered, it remains open for the sparse case (where the marginal connection probability is proportional to $1/n$) and progress has only been made recently [9, 32]. It would be interesting to tightly characterize when our model loses its geometry both for the case of constant and for the case of inhomogeneous weights. Our bounds show that the number of triangles in our model for the sparse case (constant weights) is in expectation already the same as in a Erdős-Rényi graph if $d = \omega(\log^{3/2}(n))$, while on the sphere this only happens if $d = \omega(\log^3(n))$ [10]. Accordingly, we expect that our model loses its geometry earlier than the spherical model.

References

1. William Aiello, Fan Chung, and Linyuan Lu. A random graph model for power law graphs. *Experimental Mathematics*, 10(1):53–66, 2001. doi:10.1080/10586458.2001.10504428.
2. Pedro Almagro, Marián Boguñá, and M. Ángeles Serrano. Detecting the ultra low dimensionality of real networks. *Nature Communications*, 13(1):6096, 2022. doi:10.1038/s41467-022-33685-z.
3. Igor Artico, Igor E. Smolyarenko, Veronica Vinciotti, and Ernst C. Wit. How rare are power-law networks really? *Proceedings of the Royal Society A*, 476(2241):20190742, 2020. doi:10.1098/rspa.2019.0742.
4. Konstantin E. Avrachenkov and Andrei V. Bobu. Cliques in high-dimensional random geometric graphs. *Appl. Netw. Sci.*, 5:1–24, December 2020. doi:10.1007/s41109-020-00335-6.
5. Thomas Bläsius and Philipp Fischbeck. On the External Validity of Average-Case Analyses of Graph Algorithms. In *30th Annual European Symposium on Algorithms (ESA 2022)*, pages 21:1–21:14, 2022. doi:10.4230/LIPIcs.ESA.2022.21.
6. Thomas Bläsius, Tobias Friedrich, Maximilian Katzmann, Ulrich Meyer, Manuel Penschuck, and Christopher Weyand. Efficiently generating geometric inhomogeneous and hyperbolic random graphs. *Network Science*, 10(4):361–380, 2022. doi:10.1017/nws.2022.32.

7 Thomas Bläsius, Tobias Friedrich, and Anton Krohmer. Cliques in hyperbolic random graphs. *Algorithmica*, 80(8):2324–2344, 2018. doi:10.1007/s00453-017-0323-3.

8 Anthony Bonato, David F. Gleich, Myunghwan Kim, Dieter Mitsche, Paweł Prałat, Yanhua Tian, and Stephen J. Young. Dimensionality of social networks using motifs and eigenvalues. *PLOS ONE*, 9(9):1–7, 2014. doi:10.1371/journal.pone.0106052.

9 Matthew Brennan, Guy Bresler, and Dheeraj Nagaraj. Phase transitions for detecting latent geometry in random graphs. *CoRR*, August 2020. arXiv:1910.14167.

10 Sébastien Bubeck, Jian Ding, Ronen Eldan, and Miklós Rácz. Testing for high-dimensional geometry in random graphs. *CoRR*, November 2015. arXiv:1411.5713.

11 Fan Chung and Linyuan Lu. The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences*, 99(25):15879–15882, 2002. doi:10.1073/pnas.252631999.

12 Fan Chung and Linyuan Lu. Connected Components in Random Graphs with Given Expected Degree Sequences. *Annals of Combinatorics*, 6(25):125–145, 2002. doi:10.1007/PL00012580.

13 Jesper Dall and Michael Christensen. Random geometric graphs. *Physical Review E*, 66:016121, 2002. doi:10.1103/PhysRevE.66.016121.

14 Fraser Daly, Alastair Haig, and Seva Shneer. Asymptotics for cliques in scale-free random graphs. *CoRR*, August 2020. arXiv:2008.11557.

15 Li Daqing, Kosmas Kosmidis, Armin Bunde, and Shlomo Havlin. Dimension of spatially embedded networks. *Nature Physics*, 7(6):481–484, 2011. doi:10.1038/nphys1932.

16 L. Decreusefond, E. Ferraz, H. Randriambololona, and A. Vergne. Simplicial homology of random configurations. *Advances in Applied Probability*, 46(2):325–347, June 2014. doi:10.1239/aap/1401369697.

17 Luc Devroye, András György, Gábor Lugosi, and Frederic Udina. High-dimensional random geometric graphs and their clique number. *Electronic Journal of Probability*, 16(none):2481–2508, January 2011. doi:10.1214/EJP.v16-967.

18 Vittorio Erba, Sebastiano Ariosto, Marco Gherardi, and Pietro Rotondo. Random geometric graphs in high dimension. *Physical Review E*, 102(1):012306, 2020. doi:10.1103/PhysRevE.102.012306.

19 P. Erdös and A. Rényi. On random graphs i. *Publ. Math. Debrecen*, 6, 1959.

20 Nikolaos Fountoulakis and Tobias Müller. Law of Large Numbers for the Largest Component in a Hyperbolic Model of Complex Networks. *The Annals of Applied Probability*, 28(1):607–650, 2018. doi:10.1214/17-AAP1314.

21 Nikolaos Fountoulakis, Pim van der Hoorn, Tobias Müller, and Markus Schepers. Clustering in a Hyperbolic Model of Complex Networks. *Electronic Journal of Probability*, 26(none):1–132, 2021. doi:10.1214/21-EJP583.

22 Tobias Friedrich, Andreas Göbel, Maximilian Katzmann, and Leon Schiller. Cliques in high-dimensional geometric inhomogeneous random graphs. *CoRR*, abs/2302.04113, 2023. doi:10.48550/arXiv.2302.04113.

23 Tobias Friedrich and Anton Krohmer. On the Diameter of Hyperbolic Random Graphs. *SIAM Journal on Discrete Mathematics*, 32(2):1314–1334, 2018. doi:10.1137/17M1123961.

24 E. N. Gilbert. Random graphs. *The Annals of Mathematical Statistics*, 30:1141–1144, 1959.

25 Weiwei Gu, Aditya Tandon, Yong-Yeol Ahn, and Filippo Radicchi. Principled approach to the selection of the embedding dimension of networks. *Nature Communications*, 12(1):3772, 2021. doi:10.1038/s41467-021-23795-5.

26 Luca Gugelmann, Konstantinos Panagiotou, and Ueli Peter. Random hyperbolic graphs: Degree sequence and clustering. In *39th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 573–585, 2012. doi:10.1007/978-3-642-31585-5_51.

27 Svante Janson, Tomasz Łuczak, and Ilkka Norros. Large cliques in a power-law random graph. *Journal of Applied Probability*, 47(4):1124–1135, December 2010. doi:10.1239/jap/1294170524.

28 Ralph Keusch. *Geometric Inhomogeneous Random Graphs and Graph Coloring Games*. Doctoral thesis, ETH Zurich, 2018. doi:10.3929/ethz-b-000269658.

29 Christoph Koch and Johannes Lengler. Bootstrap Percolation on Geometric Inhomogeneous Random Graphs. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, pages 147:1–147:15, 2016. doi:10.4230/LIPIcs.ICALP.2016.147.

30 Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá. Hyperbolic geometry of complex networks. *Physical Review E*, 82:036106, 2010. doi:10.1103/PhysRevE.82.036106.

31 Elizaveta Levina and Peter J. Bickel. Maximum likelihood estimation of intrinsic dimension. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, pages 777–784, 2004.

32 Siqi Liu, Sidhanth Mohanty, Tselil Schramm, and Elizabeth Yang. Testing thresholds for high-dimensional sparse random geometric graphs. *CoRR*, November 2021. arXiv:2111.11316.

33 Suqi Liu and Miklos Z. Racz. Phase transition in noisy high-dimensional random geometric graphs. *CoRR*, March 2021. arXiv:2103.15249.

34 Riccardo Michielan and Clara Stegehuis. Cliques in geometric inhomogeneous random graphs. *Journal of Complex Networks*, 10(1), 2022. doi:10.1093/comnet/cnac002.

35 Mark Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64:025102, 2001. doi:10.1103/PhysRevE.64.025102.

36 Mathew Penrose. *Random Geometric Graphs*. Oxford University Press, 2003.

37 Martin Raič. A multivariate Berry–Esseen theorem with explicit constants. *Bernoulli*, 25(4A), November 2019. doi:10.3150/18-BEJ1072.

38 Matteo Serafino, Giulio Cimini, Amos Maritan, Andrea Rinaldo, Samir Suweis, Jayanth R. Banavar, and Guido Caldarelli. True scale-free networks hidden by finite size effects. *Proceedings of the National Academy of Sciences*, 118(2), 2021. doi:10.1073/pnas.2013825118.

39 Ivan Voitalov, Pim van der Hoorn, Remco van der Hofstad, and Dmitri Krioukov. Scale-free networks well done. *Physical Review Research*, 1:033034, 2019. doi:10.1103/PhysRevResearch.1.033034.

40 Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998. doi:10.1038/30918.

41 Weihua Yang and David Rideout. High dimensional hyperbolic geometry of complex networks. *Mathematics*, 8(11), 2020. doi:10.3390/math8111861.

An $O(\log k)$-Approximation for Directed Steiner Tree in Planar Graphs

Zachary Friggstad ✉
Department of Computing Science, University of Alberta, Canada

Ramin Mousavi ✉
Department of Computing Science, University of Alberta, Canada

Abstract

We present an $O(\log k)$-approximation for both the edge-weighted and node-weighted versions of Directed Steiner Tree in planar graphs where k is the number of terminals. We extend our approach to Multi-Rooted Directed Steiner Tree[1], in which we get a $O(R + \log k)$-approximation for planar graphs for where R is the number of roots.

2012 ACM Subject Classification Theory of computation → Routing and network design problems

Keywords and phrases Directed Steiner tree, Combinatorial optimization, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.63

Category Track A: Algorithms, Complexity and Games

1 Introduction

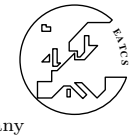
In the Directed Steiner Tree (DST) problem, we are given a directed graph $G = (V, E)$ with edge costs $c_e \geq 0, e \in E$, a root node $r \in V$, and a collection of terminals $X \subseteq V \setminus \{r\}$. The nodes in $V \setminus (X \cup \{r\})$ are called *Steiner* nodes. The goal is to find a minimum cost subset $F \subseteq E$ such that there is an $r - t$ directed path (dipath for short) using only edges in F for every terminal $t \in X$. Note any feasible solution that is inclusion-wise minimal must be an arborescence rooted at r, hence the term "tree". Throughout, we let $n := |V|$ and $k := |X|$.

One key aspect of DST lies in the fact that it generalizes many other important problems, e.g. Set Cover, (non-metric, multilevel) Facility Location, and Group Steiner Tree. Halperin and Krauthgamer [13] show Group Steiner Tree cannot be approximated within $O(\log^{2-\epsilon} n)$ for any $\epsilon > 0$ unless NP $\subseteq$ DTIME $(n^{\text{polylog}\,(n)})$ and therefore the same result holds for DST.

Building on a height-reduction technique of Calinescu and Zelikovsky [6, 21], Charikar et al. give the best approximation for DST which is an $O(k^\epsilon)$-approximation for any constant $\epsilon > 0$ [7] and also an $O(\log^3 k)$-approximation in $O(n^{\text{polylog}(k)})$ time (quasi-polynomial time). This was recently improved by Grandoni, Laekhanukit, and Li [12], who give a quasi-polynomial time $O(\frac{\log^2 k}{\log \log k})$-approximation factor for DST. They also provide a matching lower bound in that no asymptotically-better approximation is possible even for quasi-polynomial time algorithms, unless either the Projection Games Conjecture fails to hold or NP $\subseteq$ ZPTIME(2^{n^δ}) for some $0 < \delta < 1$.

The undirected variant of DST (i.e., Undirected Steiner Tree) is better understood.A series of papers steadily improved over the simple 2-approximation [22, 14, 17, 19] culminating in a $\ln 4 + \epsilon$ for any constant $\epsilon > 0$ [5]. Bern and Plassmann [3] showed that unless P = NP there is no approximation factor better than $\frac{96}{95}$ for Undirected Steiner Tree.

[1] In general graphs Multi-Rooted Directed Steiner Tree and Directed Steiner Tree are easily seen to be equivalent but in planar graphs this is not the case necessarily.



Studying the complexity of network design problems on restricted metrics such as planar graphs and more generally, graphs that exclude a fixed minor has been a fruitful research direction. For example, [4] gives the first *polynomial time approximation scheme* (PTAS) for Undirected Steiner Tree on planar graphs and more generally [2] obtains a PTAS for Steiner Forest on graphs of bounded-genus. Very recently, Cohen-Addad [8] presented a *quasi-polynomial time approximation scheme* (QPTAS) for Steiner tree on minor-free graphs.

A clear distinction in the complexity of Undirected Steiner Tree on planar graphs and general graphs have been established; however, prior to our work we did not know if DST on planar graphs is "easier" to approximate than in general graphs. Demaine, Hajiaghayi, and Klein [9] show that if one takes a standard flow-based relaxation for DST in planar graphs and further constraints the flows to be "non-crossing", then the solution can be rounded to a feasible DST solution while losing only a constant factor in the cost. However, the resulting relaxation is non-convex and, to date, we do not know how to compute a low-cost, non-crossing flow in polynomial time for DST instances on planar graphs. Recently, in [10] a constant factor approximation for planar DST was given for quasi-bipartite instances (i.e. no two Steiner nodes are connected by an edge). Though, we remark that the techniques in that paper are quite different than the techniques we use in this paper; [10] uses a primal-dual algorithm based on a standard LP relaxation whereas the techniques we use in this paper rely on planar separators.

In this paper, we show DST on planar graphs admits a $O(\log k)$-approximation, while DST on general graphs does not have an approximation factor better than $O(\log^{2-\epsilon} n)$ for any $\epsilon > 0$ unless NP $\subseteq$ DTIME $(n^{\text{polylog}\,(n)})$.

Our approach is based on planar separators presented by Thorup [20][2] which states given an undirected graph G with n vertices, one could find a "well-structured" subgraph F such that each connected component of $G \setminus F$ has at most $\frac{n}{2}$ vertices. We show using this separator and an aggressive guessing of optimal value of each subproblems lead to an $O(\log k)$-approximation algorithm in quasi-polynomial time. Then, we show how to modify the guessing part to make the algorithm run in polynomial time. Well-structured separators are useful in enabling divide-and-conquer approach for some problems, such as Maximum Independent Set and Pebbling [16]. Also very recently, Cohen-Addad [8] uses the same separator we consider to design QPTASes for k-MST and Undirected Steiner Tree on planar graphs. He also develops a new separator to deal with these problems in minor-free graphs.

We show the separator theorem of Thorup can be used to obtain a simple logarithmic approximation algorithm for planar DST.

▶ **Theorem 1.** *There is a $O(\log k)$-approximation for planar Directed Steiner Tree, where k is the number of terminals.*

We remark that it is trivial to generalize our algorithm to the node-weighted setting of DST in planar graphs. That is, to instances where Steiner nodes $v \in V \setminus (X \cup \{r\})$ have costs $c_v \geq 0$ and the goal is to find the cheapest S of Steiner Nodes such that the graph $G[\{r\} \cup X \cup S]$ contains an $r - t$ dipath for each $t \in X$. Clearly node-weighted DST generalizes edge-weighted DST even in planar graphs settings since we can subdivide an edge with cost c_e and include this cost on the new node. In general graphs, edge-weighted DST generalizes node-weighted DST because a node v with cost c_v can be turned into two nodes v^+, v^- connected by an edge (v^+, v^-) with cost c_v; edges entering v now enter v^+ and edges exiting v now exit v^-. But this operation does not preserve planarity, it is easy to find examples where this results in a non-planar graph.

[2] As stated in [20] this separator theorem was implicitly proved in [15].

We also extend our result to multi-rooted case. In Multi-Rooted Directed Steiner Tree (MR-DST), instead of one root, we are given multiple roots $r_1, \ldots, r_R$ and the set of terminals $X \subseteq V \setminus \{r_1, \ldots, r_R\}$. The goal here is to find a minimum cost subgraph such that every terminal is reachable from one of the roots.

Note that MR-DST on general graphs is equivalent to DST by adding an auxiliary root node r and adding edges (r, r_i) for $1 \leq i \leq R$ with zero cost. However, this reduction also does not preserve planarity. We prove our result for MR-DST by constructing a "well-structured" separator for the multi-rooted case.

▶ **Theorem 2.** *There is a $O(R + \log k)$-approximation for planar Multi-Rooted Directed Steiner Tree, where R is the number of roots and k is the number of terminals.*

2 Preliminaries

For convenience, we allow our input graphs to contain multiple directed edges between two nodes. All directed paths (dipath for short) in this paper are simple. Fix a digraph $G = (V, E)$ with edge costs $c_e \geq 0$ for all $e \in E$. We identify a dipath P by its corresponding sequence vertices, i.e., $P = v_1, \ldots, v_a$ and we say P is a $v_1 - v_a$-dipath. The *start* and *end* vertices of P are v_1 and v_a, respectively. For a subgraph H of G, we define the cost of a subgraph H by $\text{cost}_c(H) := \sum_{e \in E(H)} c_e$

We say a vertex v is *reachable* from u if there is a dipath from u to v. We denote by $d_c(u, v)$ the cost of a shortest dipath from u to v, in particular, $d_c(u, u) = 0$. The *diameter* of a digraph is defined as the maximum $d_c(u, v)$ for all $u \neq v$ where v is reachable from u. For both $d_c(.)$ and $\text{cost}_c(.)$ we drop the subscript c if the edge costs is clear from the context. For a subset $S \subseteq V$ and a vertex u, we define $d(S, v) := \min_{u \in S}\{d(u, v)\}$. Denote by $G[S]$ the *induced subgraph* of G on the subset of vertices S, i.e., $G[S] = (S, E[S])$ where $E[S]$ is the set of edges of G with both endpoints in S. A *weakly connected component* of G is a connected component of the undirected graph obtained from G by ignoring the orientation of the edges. The *indegree* of a vertex v with respect to $F \subseteq E$ is the number of edges in F oriented towards v.

A *partial arborescence* $T = (V_T, E_T)$ rooted at r in G, is a (not necessarily spanning) subgraph of G such that $r \in V_T$ and T is a directed tree oriented away from r. An arborescence is a partial arborescence that spans all the vertices. A *breadth first search* (BFS) arborescence B_G rooted at r is a (perhaps partial) arborescence including all nodes reachable from r where the dipath from r to any vertex v on B_G is a shortest dipath from r to v.

For two disjoint subsets of vertices $S, T \subseteq V$ denote by $\delta(S, T)$ the set of edges with one endpoint in S and the other endpoint in T (regardless of the orientation).

Given a subgraph H of G, for notational simplicity we write G/H the resulting graph from contracting all the edges in H. Also we denote by $G \setminus H$ the resulting graph by removing H from G, i.e., removing all the vertices of H and the edges incident to these vertices.

Our algorithm is based on planar separators described by Thorup [20].

▶ **Theorem 3** (Lemma 2.3 in [20]). *Let $G = (V, E)$ be a connected and undirected planar graph with non-negative vertex weights, and let T be a spanning tree rooted at a vertex $r \in V$. In linear time, one can find three vertices v_1, v_2, and v_3 such that the union of vertices on paths P_i between r and v_i in $V(T)$ for $i = 1, 2, 3$ forms a separator of G, i.e., every connected component of $G \setminus (P_1 \cup P_2 \cup P_3)$ has at most half the weight of G.*

An immediate consequence of the above result is that given a directed graph and a BFS arborescence rooted at r instead of a spanning tree, one can obtain a separator consisting three shortest dipaths each starting at r.

▶ **Corollary 4** (Directed separator). *Let $G = (V, E)$ be a planar digraph with edge costs $c_e \geq 0$ for all $e \in E$, and non-negative vertex weights such that every vertex $v \in V$ is reachable from r. Given a vertex $r \in V$, in polynomial time, we can find three shortest dipaths P_1, P_2, and P_3 each starting at r such that every weakly connected component of $G \setminus (P_1 \cup P_2 \cup P_3)$ has at most half the weight of G.*

Throughout this paper, we create subinstances from I by contracting a subset of edges F in G. Whenever, we create a subinstance I' we let the edge cost for the subinstance to be the natural restriction of c to G/F, i.e., if e is in both $E(G)$ and $E(G/F)$ then e has cost c_e in I' and if e is in $E(G/F)$ but not in $E(G)$, then its cost in I' is set to be the cost of the corresponding edge in $E(G)$.

Let $I = \big(G = (V, E), c, \{r_1, \ldots, r_R\}, X\big)$ be an instance of MR-DST on planar graphs where G is a planar digraph, $c_e \geq 0$ for all $e \in E$ is the edge costs, $\{r_1, \ldots, r_R\}$ are the roots, and $X \subseteq V \setminus \{r_1, \ldots, r_R\}$ is the set of terminals. By losing a small factor in the approximation guarantee, one can assume in an instance of MR-DST that all the costs are positive integers and $d\big(\{r_1, \ldots, r_R\}, v\big)$ is polynomially bounded by n for all $v \in V$. The very standard proof appears in Appendix 6.

▶ **Lemma 5** (Polynomial bounded distances). *For any constant $\epsilon > 0$, if there is an α-approximation for MR-DST instances in planar graphs where all edges e have positive integer costs $c_e \geq 1$ and $d_c(r, v) \leq \frac{|X| \cdot |V|}{\epsilon} + |V|$ for each $v \in V$, then there is an $(\alpha \cdot (1 + \epsilon))$-approximation for general instances of MR-DST in planar graphs.*

3 Planar DST

In this section we prove Theorem 1. Fix an instance $I = \big(G = (V, E), c, r, X\big)$ of DST on planar graphs that satisfies the assumptions in Lemma 5 for, say, $\epsilon = 1/2$. Let $n := |V|$ and $k := |X|$. Furthermore, fix an optimal solution OPT for this instance and let opt denote its cost. So the distance of every vertex from r is at most $O(n \cdot k)$.

Our algorithm recursively constructs smaller subinstances based on a partial arborescence (as a separator) and disjoint subsets of vertices (as the weakly connected components after removing the separator). The following is a more formal definition of these subinstances.

▶ **Definition 6** (Induced subinstances). *Let $I = (G = (V, E), c, r, X)$ be an instance of DST on planar graphs. Let T be a partial arborescence rooted at r, and let $C_1, \ldots, C_h$ be the weakly connected components of $G \setminus T$. The subinstances of DST induced by tuple $(G, T, C_1, \ldots, C_h)$ are defined as follows: let G_{contract} be the graph obtained from G by contracting T into r. For each C_i where $1 \leq i \leq h$ we construct instance $I_{C_i} := \big(G_{C_i}, c, r, C_i \cap X\big)$ where $G_{C_i} := G_{\text{contract}}[C_i \cup \{r\}]$. See Figure 1.*

Given solutions $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_h$ for the subinstances induced by $(G, T, C_1, \ldots, C_h)$, one can naturally consider the corresponding subset of edges of $E(T) \cup \mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots \cup \mathcal{F}_h$ in G and it is easy to see this forms a feasible solution for instance I. We formalize this in the next lemma.

▶ **Lemma 7** (Merged solution). *Consider the subinstances I_{C_i} for $1 \leq i \leq h$ as defined in Definition 6. Let $\mathcal{F}_{C_i}$ be a solution for I_{C_i}. Let $\mathcal{F} \subseteq E(G)$ be the corresponding edges of $E(T) \cup (\bigcup_{i=1}^{h} \mathcal{F}_{C_i})$ in G. Then, $\mathcal{F}$ is a feasible solution for instance I and furthermore $\text{cost}(\mathcal{F}) = \text{cost}(T) + \sum_{i=1}^{h} \text{cost}(\mathcal{F}_{C_i})$. See Figure 1.*

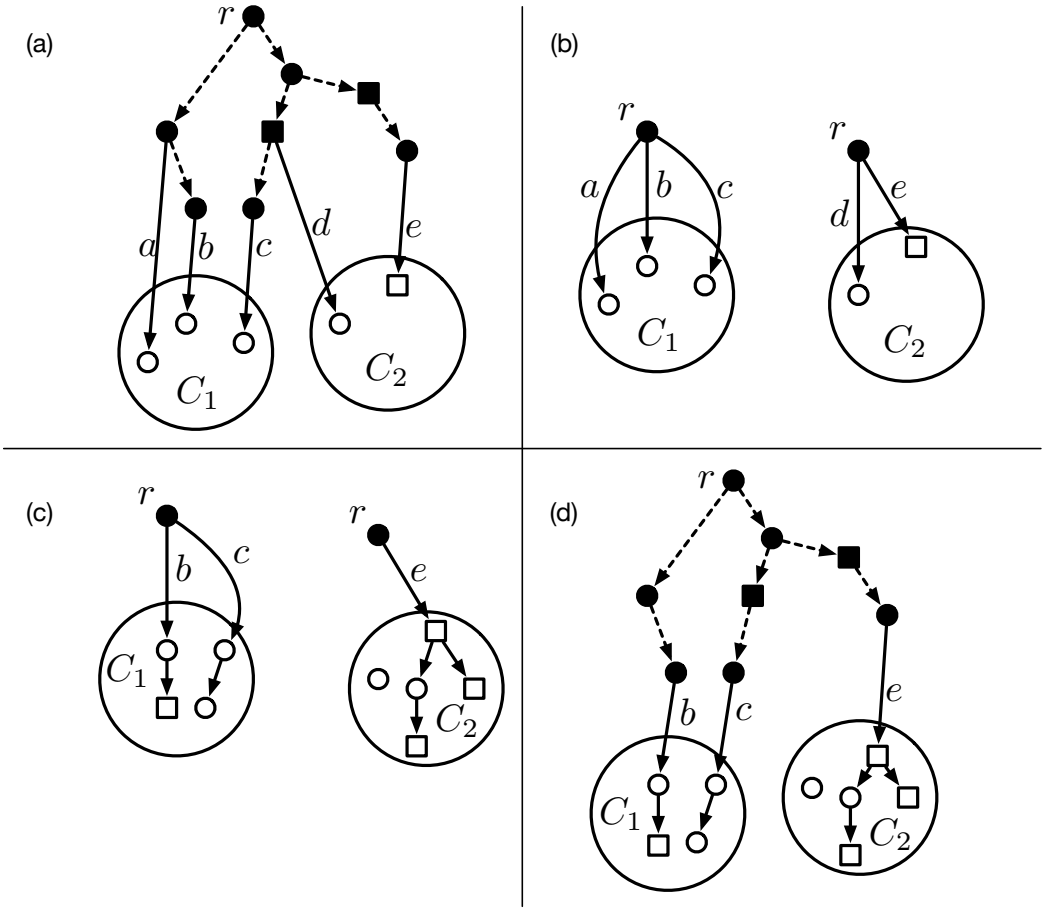


Figure 1 Throughout, squares are terminals and circles are Steiner nodes or the root node r. In (a) the separator is shown with dashed edges and solid vertices. The weakly connected components of $G \setminus T$ are shown as circles denoted by C_1 and C_2. Note that we did not show any edge directed from C_1 or C_2 into the separator because we can safely remove these edges. In (b) the subinstances I_{C_1} and I_{C_2} induced by (G, T, C_1, C_2) are depicted. In (c), the solutions for each subinstances are shown. Finally, (d) shows how to merge the solutions in (c) to get a solution for the original instance. Note that leaf nodes are not necessarily terminals. One could prune them as a post-processing step, but that is not required by our algorithm.

Proof. The furthermore part is obvious so we prove that $\mathcal{F}$ is feasible for I. Consider a terminal node $t \in C_i$. Since $\mathcal{F}_i$ is feasible for I_{C_i}, then there is a dipath P from r to t. Let (r, v) be the first edge on P and let (u, v) be the corresponding edge to (r, v) in $E(G)$. Then, we must have $u \in V(T)$ as $\delta(C_i, C_j) = \emptyset$ for all $1 \leq i \neq j \leq h$. So we can go from r to u in T, then take the edge (u, v) and then go from v to t in $\mathcal{F}_{C_i}$. Since all these edges are present in $\mathcal{F}$ and t is an arbitrary terminal, $\mathcal{F}$ is a feasible solution for I. ◀

We first present a high-level idea of a simple $O(\log k)$-approximation that runs in quasi-polynomial time and then with a little extra work, we can make it run in polynomial time with small loss in the approximation guarantee.

3.1 Warm-up: An overview of a quasi-polynomial time approximation

The algorithm is simple. Fix an optimal solution OPT with cost opt. First we guess opt. Note by Lemma 5, opt is polynomial in n and integral so there are polynomial guesses. Then, we remove all vertices such that their distance from r is more than our guessed value (this is

the preprocessing step). For the purpose of separating into subinstances with balanced weight, we let the weight of each terminal to be 1 and the rest of vertices have zero weight. Apply Corollary 4 and let P_1, P_2, and P_3 be the resulting shortest dipaths each starting at r. Note that $\text{cost}(P_i) \leq \text{opt}$ for $i = 1, 2, 3$ because of the preprocessing step. Let $T := P_1 \cup P_2 \cup P_3$, then T is a branching rooted at r. Let C_i for $1 \leq i \leq h$ be the weakly connected components of $G \setminus T$. Then, we recursively solve the subinstances induced by $(G, T, C_1, \ldots, C_h)$ (see Definition 6), and finally return the corresponding solution of $E(T) \cup \bigcup_{i=1}^{h} \mathcal{F}_{C_i}$ in G. When the number of terminals in a subinstance becomes one, we can solve the problem exactly by finding the shortest dipath between the root and the only terminal.

Note that each recursive call reduces the number of terminals by half. The guess work for each instance is polynomial in n. So it is easy to see the total number of recursive calls is bounded by $n^{O(\log k)}$. Since each time we apply the separator result on an instance I, we buy a branching (union of up to three dipaths) of cost at most $3 \cdot \text{opt}$, and since the *total* cost of optimal solutions across all of the resulting subinstances I_{C_i} is at most opt, a simple induction on the number of terminals shows the final cost is within $(3 \cdot \log k + 1) \cdot \text{opt}$. A slight improvement to the running time could be made by guessing OPT within a constant factor (thus only making $O(\log n)$ guesses since all distances are integers bounded by a polynomial in n), but the size of the recursion tree would still be $O(\log n)^{O(\log k)}$ which is still not quite polynomial.

In the next section, we show how to avoid the above aggressive guessing which gives us the polynomial running time. We remark there are some similarities between our algorithm with the one presented in [11] for quasi-polynomial time algorithm for Submodular Tree Orienteering in the sense that both need to guess some value (in our case opt and in their case the budget) for the subproblems and performing this guess naively is too slow. However, the approaches to overcoming this barrier are different.

3.2 The polynomial-time algorithm

The idea here is similar to the quasi-polynomial time algorithm; however, instead of guessing the diameter of an optimal arborescence for each instance, we keep an estimate of it. Our recursive algorithm tries two different recursive calls: (1) divide the current estimate by half and recurse, or (2) buy a separator and divide the instance into smaller instances and recurs on these instances using the current estimate as the current estimate passed to each smaller instance.

The rationale behind this idea is that if the estimate is close to the optimal value, then our separator is "cheap" compared to optimal value so (2) is a "good progress" otherwise we make the estimate smaller so (1) is a "good progress". The key idea here that leads to polynomial time is that we do not "reset" our guess for the optimal solution cost in each recursive call since we know that if our guess is correct for the current instance, then it is an upper bound for the optimal solution cost in each subinstance.

As we mentioned at the beginning, the algorithm is recursive. The input to the algorithm is a tuple $(I, \widetilde{\text{opt}})$ where $\widetilde{\text{opt}}$ is an estimate of opt. The algorithm computes two solutions and take the better of the two. One solution is by a recursive call to $(I, \frac{\widetilde{\text{opt}}}{2})$ and the other one is obtained by applying Corollary 4 to get smaller subinstances and solve each subinstance recursively and merge the solutions as described in Lemma 7. See Algorithm 1 for the pseudocode.

By Lemma 5, we can assume the edge costs are positive integers and hence $\text{opt} \geq 1$. So if $\widetilde{\text{opt}} < 1$, then the output of $\text{DST}(I, \widetilde{\text{opt}})$ is infeasible. The algorithm will terminate since each recursive call either halves $\widetilde{\text{opt}}$ or halves the number of terminals.

Algorithm 1 $\mathrm{DST}(I, \widetilde{\mathrm{opt}})$.

Input: $I := \big(G = (V, E), c, r, X\big)$ and an estimate $\widetilde{\mathrm{opt}}$.
Output: A feasible solution for instance I or output infeasible.

if $\widetilde{\mathrm{opt}} < 1$ or $d(r, t) > \widetilde{\mathrm{opt}}$ for some terminal $t \in X$ **then**
  return `infeasible`
else if $|X_I| = 1$ **then**
  Let $\mathcal{F}$ be the shortest dipath from r to the only terminal in X_I.
else
  $\mathcal{F}_1 \leftarrow \mathrm{DST}(I, \frac{\widetilde{\mathrm{opt}}}{2})$, if $\mathcal{F}_1$ is `infeasible` solution then set $\mathrm{cost}(F_1) \leftarrow \infty$.
  Remove all vertices v with $d(r, v) > \widetilde{\mathrm{opt}}$. {This is the preprocessing step.}
  Apply Corollary 4 to obtain a partial arborescence T consists of up to 3 shortest dipaths starting at r. Let $C_1, \ldots, C_h$ be the weakly connected components of $G \setminus T$. Let I_{C_i} be the i-th subinstance induced by $(G, T, C_1, \ldots, C_h)$ for $i = 1, \ldots, h$.
  for $i = 1, \ldots, h$ **do**
    $\mathcal{F}'_i \leftarrow \mathrm{DST}(I_{C_i}, \widetilde{\mathrm{opt}})$
  $\mathcal{F}_2 \leftarrow E(T) \cup (\bigcup_{i=1}^{h} \mathcal{F}'_i)$, if any $\mathcal{F}'_i$ is `infeasible` then set $\mathrm{cost}(\mathcal{F}_2) \leftarrow \infty$.
  if both $\mathrm{cost}(\mathcal{F}_1)$ and $\mathrm{cost}(\mathcal{F}_2)$ are ∞ **then**
    return `infeasible`
  $\mathcal{F} \leftarrow \arg\min\{\mathrm{cost}(\mathcal{F}_1), \mathrm{cost}(\mathcal{F}_2)\}$
return $\mathcal{F}$.

3.3 Analysis

In this section, we analyze the cost and the running time of Algorithm 1.

▶ **Lemma 8** (Cost and running time). *Consider an instance $I = \big(G = (V, E), c, r, X\big)$ and a pair $(I, \widetilde{\mathrm{opt}})$. Let ℓ and $\mathcal{O}$ be non-negative integers such that $|X| \le 2^\ell$ and $\widetilde{\mathrm{opt}} \le 2^{\mathcal{O}}$. If $\widetilde{\mathrm{opt}} \ge \mathrm{opt}$ where opt is the optimal value of I, then $\mathrm{DST}(I, \widetilde{\mathrm{opt}})$ returns a solution with cost at most $(6 \cdot \ell + 1) \cdot \mathrm{opt}$. Furthermore, the total number of recursive calls made by $\mathrm{DST}(I, \widetilde{\mathrm{opt}})$ and its subsequent recursive calls is at most $|X| \cdot 2^{2 \cdot \ell + \mathcal{O}}$.*

Proof. First we analyze the cost of the output solution. If $\ell = 0$ then we solve I exactly so the statement holds. So for the rest of the proof we assume $\ell \ge 1$. We proceed by induction on $\ell + \mathcal{O} \ge 1$.

We assume $\widetilde{\mathrm{opt}} \le 2 \cdot \mathrm{opt}$, otherwise we have $\mathrm{DST}(I, \widetilde{\mathrm{opt}}) \le \mathrm{DST}(I, \frac{\widetilde{\mathrm{opt}}}{2}) \le (6 \cdot \ell + 1) \cdot \mathrm{opt}$ by induction where the last inequality holds because $\log \frac{\widetilde{\mathrm{opt}}}{2} \le \log(\widetilde{\mathrm{opt}}) - 1$.

Let $\mathcal{F}$ be the solution returned by $\mathrm{DST}(I, \widetilde{\mathrm{opt}})$. Since $\mathrm{cost}(\mathcal{F}) \le \mathrm{cost}(\mathcal{F}_2)$, it suffices to prove $\mathrm{cost}(\mathcal{F}_2) \le (6 \cdot \ell + 1) \cdot \mathrm{opt}$. Let $\mathcal{F}'_i = \mathrm{DST}(I_{C_i}, \widetilde{\mathrm{opt}})$ for $i = 1, \ldots, h$ be the solutions constructed recursively for the subinstances. Note that each I_{C_i} for $i = 1 \ldots, h$ has at most $2^{\ell - 1}$ terminals and $\widetilde{\mathrm{opt}} \ge \mathrm{opt}_{I_{C_i}}$ where $\mathrm{opt}_{I_{C_i}}$ is the optimal value of I_{C_i}. By the induction hypothesis, we conclude

$$\mathrm{cost}(\mathcal{F}'_i) \le (6 \cdot (\ell - 1) + 1) \cdot \mathrm{opt}_{I_{C_i}} \le 6 \cdot \ell \cdot \mathrm{opt}_{I_{C_i}}, \ \textit{for } i = 1, \ldots, h \tag{1}$$

Note that T is the union of up to three shortest dipaths and because of the preprocessing step, each shortest dipath starting at r has cost at most $\widetilde{\mathrm{opt}} \le 2 \cdot \mathrm{opt}$. So the following holds:

$$\mathrm{cost}(T) \le 3 \cdot \widetilde{\mathrm{opt}} \le 6 \cdot \mathrm{opt}\,. \tag{2}$$

Combining (1) and (2) we get:

$$\begin{aligned}
\mathrm{cost}(\mathcal{F}) &= \mathrm{cost}(T) + \sum_{i=1}^{h} \mathrm{cost}(\mathcal{F}'_i) \\
&\leq \mathrm{cost}(T) + \sum_{i=1}^{h} 6 \cdot \ell \cdot \mathrm{opt}_{I_{C_i}} \\
&\leq 6 \cdot \mathrm{opt} + 6 \cdot \ell \cdot \sum_{i=1}^{h} \mathrm{opt}_{I_{C_i}} \\
&\leq 6 \cdot \mathrm{opt} + 6 \cdot \ell \cdot \mathrm{opt} \\
&= (6 \cdot \ell + 1) \cdot \mathrm{opt},
\end{aligned}$$

where the first equality follows from Lemma 7, the first and the second inequalities follow from (1) and (2), respectively, and finally the last inequality follows from the fact that $\sum_{i=1}^{h} \mathrm{opt}_{I_{C_i}} \leq \mathrm{opt}$ as the restriction of OPT on each G_{C_i} is a feasible solution for I_{C_i} and G_{C_i}'s are edge-disjoint.

Next, we analyze the number of recursive calls $R(\ell, \wp)$ in $\mathrm{DST}(I, \widetilde{\mathrm{opt}})$. We prove by induction on $\ell + \wp$ that $R(\ell, \wp) \leq |X| \cdot 2^{2 \cdot \ell + \wp}$. If $\ell = 0$, then there is no recursive call. So suppose $\ell \geq 1$. Let $X_i := |X \cap C_i| \leq \frac{|X|}{2}$ be the number of terminals in subinstance I_{C_i} and let ℓ_i be the smallest integer where $|X_i| \leq 2^{\ell_i}$. Since the number of terminals in the subinstances are halved, we have $\ell_i \leq \ell - 1$ for all $1 \leq i \leq h$. So we can write

$$\begin{aligned}
R(\ell, \wp) &= 1 + R(\ell, \wp - 1) + \sum_{i=1}^{h} R(\ell_i, \wp) \\
&\leq 1 + |X| \cdot 2^{2 \cdot \ell + \wp - 1} + \sum_{i=1}^{h} |X_i| \cdot 2^{2 \cdot \ell_i + \wp} \\
&\leq 1 + |X| \cdot 2^{2 \cdot \ell + \wp - 1} + 2^{2(\ell - 1) + \wp} \cdot \sum_{i=1}^{h} |X_i| \\
&\leq 1 + |X| \cdot 2^{2 \cdot \ell + \wp - 1} + 2^{2 \cdot \ell + \wp - 2} \cdot |X| \\
&= 1 + |X| \cdot 2^{2 \cdot \ell + \wp - 1} + (2^{2 \cdot \ell + \wp - 1} - 2^{2 \cdot \ell + \wp - 2}) \cdot |X| \\
&= 1 + |X| \cdot 2^{2 \cdot \ell + \wp} - |X| \cdot 2^{2 \cdot \ell + \wp - 2} \\
&\leq |X| \cdot 2^{2 \cdot \ell + \wp},
\end{aligned}$$

where the first inequality follows from the induction hypothesis, the second inequality comes from the fact that $\ell_i \leq \ell - 1$, the third inequality holds because $\sum_{i=1}^{h} |X_i| \leq |X|$, and the last inequality follows from the fact that $|X| \geq 1$ and $\ell \geq 1$. ◀

Proof of Theorem 1. For any $\epsilon > 0$, we can assume all the shortest dipaths starting at the root are bounded by $\mathrm{poly}(n, \epsilon)$ by losing a $(1 + \epsilon)$ multiplicative factor in the approximation guarantee, see Lemma 5. So we assume properties of Lemma 5 holds for the rest of the proof.

Let Δ be the maximum distance from the root to any terminal. Let $\widetilde{\mathrm{opt}} := k \cdot \Delta \leq \mathrm{poly}(n)$. We find a solution by calling $\mathrm{DST}(I, \widetilde{\mathrm{opt}})$. Applying Lemma 8 with $\widetilde{\mathrm{opt}} := k \cdot \Delta$, $\ell := \lceil \log k \rceil \leq \log k + 1$ and $\wp := \lceil \log \widetilde{\mathrm{opt}} \rceil$ guarantees the solution has cost at most $(6 \cdot (\log k + 1) + 1) \cdot \mathrm{opt}$.

For running time of Algorithm 1, we have by Lemma 8 that the number of recursive calls is at most $k \cdot 2^{2 \cdot \ell + \wp} = O(k^4 \cdot \Delta)$. So the total number of recursive calls is $\mathrm{poly}(n)$ (recall $k \cdot \Delta = \mathrm{poly}(n)$). The running time within each recursive call is also bounded by $\mathrm{poly}(n)$ so the algorithm runs in polynomial time. ◀

4 Multi-rooted planar DST

The algorithm for the multi-rooted case is similar to Algorithm 1. We need analogous versions of the separator, how we define the subinstances, and how we merge the solutions of smaller subinstances to get a solution for the original instance for the multi-rooted case.

We start by a generalization of partial arborescence in the single rooted case to multiple roots.

▶ **Definition 9** (Multi-rooted partial arborescence). *Given a digraph $G = (V, E)$, R vertices $r_1, \ldots, r_R$ designated as roots. We say a subgraph T of G is a multi-rooted partial arborescence if it satisfies the following properties:*

1. *There are vertex-disjoint partial arborescences $T_{i_1}, \ldots, T_{i_q}$ rooted at $r_{i_1}, \ldots, r_{i_q}$, respectively, and a subset of edges $F \subseteq E \setminus \left(\bigcup_{j=1}^{q} E(T_{i_j})\right)$, where the endpoints of each edge in F belong to $\bigcup_{j=1}^{q} V(T_{i_j})$, such that $T = F \cup (\bigcup_{j=1}^{q} T_{i_q})$.*
2. *T is weakly connected and has no cycle (in the undirected sense).*

If a multi-rooted partial arborescence T covers all the vertices in G, then we say T is a *multi-rooted arborescence* for G. See Figure 2 for an example.

Fix an instance $I = (G, c, \{r_1, \ldots, r_R\}, X)$ of R-rooted DST on planar graphs. Next, we present subinstnaces induced by a partial multi-rooted arborescence and bunch of disjoint subsets analogous to Definition 6.

▶ **Definition 10** (Induced subinstances, multi-rooted). *Let $I = (G, c, \{r_1, \ldots, r_R\}, X)$ of R-rooted DST on planar graphs. Let $T = F \cup (\bigcup_{j=1}^{q} T_{p_j})$ be a multi-rooted partial arborescence where T_{p_j} is a partial arborescence rooted at r_{p_j} for $1 \le j \le q$. In addition, let $C_1, \ldots, C_h$ be the weakly connected components of $G \setminus T$. The subinstances of multi-rooted DST induced by tuple $(G, T, C_1, \ldots, C_h)$ are defined as follows: let G_{contract} be the graph obtained from G by contracting T into a singleton vertex called r_T. For each C_i where $1 \le i \le h$ we construct instance $I_{C_i} := \left(G_{C_i}, c, \{r_T\} \cup \left(C_i \cap \left(\{r_1, \ldots, r_R\} \setminus \{r_{p_1}, \ldots, r_{p_q}\}\right)\right), C_i \cap X\right)$ where $G_{C_i} := G_{\text{contract}}[C_i \cup \{r_T\}]$.*

The following is analogous to Lemma 7 for merging solution in the multi-rooted case.

▶ **Lemma 11** (Merged solutions, multi-rooted). *Let $T = F \cup (\bigcup_{j=1}^{q} T_{p_j})$ be a partial multi-rooted arborescence in G. Consider the subinstances I_{C_i} for $1 \le i \le h$ as defined in Definition 10 and let $\mathcal{F}_{C_i}$ be a solution for I_{C_i}. Let $\mathcal{F} \subseteq E(G)$ be the corresponding edges in $(E(T) \setminus F) \cup (\bigcup_{i=1}^{h} \mathcal{F}_{C_i})$. Then, $\mathcal{F}$ is a feasible solution for instance I and furthermore* $\text{cost}(\mathcal{F}) = \text{cost}(T \setminus F) + \sum_{i=1}^{h} \text{cost}(\mathcal{F}_{C_i})$.

Proof. The furthermore part follows directly from the definition of $\mathcal{F}$. We prove $\mathcal{F}$ is feasible for I.

Consider a terminal t. If $t \in V(T)$, then $t \in V(T_{p_j})$ for some $1 \le j \le q$ (recall the vertices in T is the union of the vertices in all the partial arborescences T_{p_j}'s) so t is reachable from r_{p_j}, the root of T_{p_j}, in $\mathcal{F}$. Suppose $t \in C_i$ for some $1 \le i \le h$. If t is reachable from a root other than r_T in $\mathcal{F}_{C_i}$ then we are done because the same dipath exists in $\mathcal{F}$. So we suppose not and let P be the dipath in $\mathcal{F}_{C_i}$ from r_T to t. Let (u, v) be the corresponding edge to (r_T, v) in G. Note that $u \in V(T_{p_j})$ for some $1 \le j \le q$ because $\delta(C_s, C_{s'}) = \emptyset$ for $1 \le s \ne s' \le h$. Hence, t is reachable from r_{p_j}, the root of T_{p_j}, in $\mathcal{F}$ as $E(T_{p_j}) \subseteq \mathcal{F}$. ◀

Given an instance I with roots $r_1, \ldots, r_R$, temporarily add an auxiliary node r and add edges (r, r_i) for all $1 \le i \le R$ with zero cost (it might destroy the planarity). Run the BFS algorithm as usual rooted at r. Then, remove r and all the edges incident to r. The result is a vertex-disjoint BFS arborescences $A_1, A_2, \ldots, A_R$ rooted at $r_1, \ldots, r_R$. Note that for every $v \in V(A_i)$, v is closest to r_i than any other roots, i.e., the dipath from r_i to v has cost $d\big(\{r_1, \ldots, r_R\}, v\big)$.

Finally, we present the separator result for the multi-rooted case.

▶ **Lemma 12** (A structured separator, multi-rooted). *Let $I = (G = (V, E), c, \{r_1, \ldots, r_R\}, X)$ be an instance of multi-rooted DST on planar graphs, and let $A_1, \ldots, A_R$ be the vertex-disjoint BFS arborescence rooted at $r_1, \ldots, r_R$. There is a multi-rooted partial arborescence $T = F \cup (\bigcup_{j=1}^{R} T_{i_j})$, where T_{i_j} could possibly be empty (i.e., with no vertices) such that the following hold:*

- **(a)** *T_j is either empty or is a subtree of A_j rooted at r_j that consists of the union of up to four shortest dipaths each starting at r_j.*
- **(b)** *Let $C_1, \cdots, C_h$ be the weakly connected components of $G \setminus T$. Then, each subinstance I_{C_i} induced by $(G, T, C_1, \ldots, C_h)$ has at most $\frac{|X|}{2}$ terminals for $1 \le i \le h$.*
- **(c)** *Let $\mathcal{F}_i$ be a solution to subinstance I_{C_i} for $1 \le i \le h$. Then, the corresponding solution $(E(T) \setminus F) \cup (\bigcup_{i=1}^{h} \mathcal{F}_i)$ in G is feasible for I with cost exactly $\mathrm{cost}(T \setminus F) + \sum_{i=1}^{h} \mathrm{cost}(\mathcal{F}_i)$.*

Proof. Figure 2 helps to visualize this proof.

Since G is weakly connected, there is a subset of edges F' in G such that $T' := F' \cup (\bigcup_{i=1}^{R} A_i)$ is a multi-rooted arborescence of G (spanning all the vertices) and the endpoints of edges in F are in $\bigcup_{i=1}^{R} V(A_i)$. Make T' rooted at an arbitrarily chosen root, say r_1. Apply Theorem 3 with terminal vertices having weight 1 and the rest of vertices having weight 0, and T' as the spanning tree (in the undirected sense). This gives three paths P_1, P_2, and P_3 in T' each with starting vertex r_1 such that every weakly connected component C_i of $G \setminus (P_1 \cup P_2 \cup P_3)$ has at most $\frac{|X|}{2}$ terminals for $1 \le i \le h$. Note, these three paths do not necessarily follow the directions of the edges.

Fix A_i for some $1 \le i \le R$ and a path $P_j := (r_1 = v_1), v_2, \ldots, v_N$ for $1 \le j \le 3$. Let a and b (possibly $a = b$) be the smallest and the largest indices, respectively, such that v_a and v_b are in $V(A_i)$. We claim the subpath $P_{[a,b]} := v_a, v_{a+1}, \ldots, v_b$ is a subgraph of A_i. Suppose not, so there must be two indices $a \le a' < b' \le b$ such that $v_{a'}, v_{b'} \in V(A_i)$ and $v_{a'+1}, v_{a'+2}, \ldots, v_{b'-1} \notin V(A_i)$. Let $P_{A_i}^{a'}$ and $P_{A_i}^{b'}$ be the paths from r_i to a' and b' in $V(A_i)$, respectively. So $P_{A_i}^{a'} \cup P_{A_i}^{b'} \cup P_{[a',b']}$ forms a cycle in T', a contradiction. Furthermore, for $j = 1, 2, 3$ let v_j be the closest vertex to r_1 on P_j (in terms of edge hops) that is in A_i as well (if exists). Then, $v_1 = v_2 = v_3$ as otherwise we have a cycle in T' because all P_j's start at r_1.

For each $1 \le i \le R$ and $1 \le j \le 3$, we **mark** the nodes with smallest and largest indices in P_j that are in A_i. We proved above, that the number of these marked vertices in each A_i is at most 4. Furthermore, $(P_1 \cup P_2 \cup P_3) \cap A_i$ is a subgraph of the union of dipaths from r_i to each marked vertices in A_i for all $1 \le i \le h$.

We construct our partial multi-rooted arborescence T as follows: let T_i be the union of (up to four) shortest dipaths from r_i to the marked vertices in A_i. Let $F := E\big(P_1 \cup P_2 \cup P_3\big) \setminus (\bigcup_{i=1}^{R} E(T_i))$ which is the subset of edges whose endpoints are in different $V(A_i)$'s, i.e., $F \subseteq F'$. Let $T := F \cup (\bigcup_{i=1}^{R} T_i)$. Note that for A_i's with no marked vertices, T_i is empty

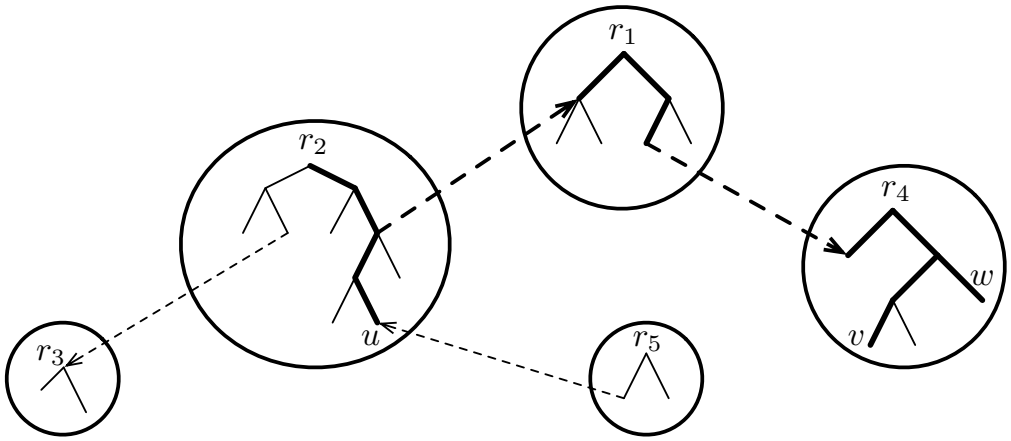


Figure 2 A depiction of the multirooted separator in an instance with $R = 5$ roots. The solid edges (thick and thin) are the shortest-path arborescences A_i for $i = 1, \ldots, R$. The dashed edges are F', they exist solely to allow us to apply Theorem 3 starting from a spanning tree of the underlying undirected graph and to witness the contraction of all vertices on the thick edges results in a planar graph. After applying Theorem 3, we get three vertices depicted as u, v, w. The vertices touching the thick and solid edges then form the multirooted separator: these include all vertices lying on paths from r to u, v, or w (as in Theorem 3). Additionally, for each $i = 1, \ldots, R$ that includes at least one node from some $r_1 - a$ path for some $a \in \{u, v, w\}$, the multirooted separator includes vertices on the unique path connecting r_i to the $r - a$ path (eg. the path from r_2 to the $r_1 - u$ path). In the algorithm, the solution will purchase the thick solid edges, but not the thick dashed edges. However, we do contract all thick edges (dashed and solid) to generate the subproblems: the number of roots also drops by 2 since the separator touches 3 shortest-path arborescences. Any solution that is connected from the new contracted root will be connected from either r_1, r_2 or r_4 using the thick and solid edges after uncontracting.

(with no vertices not even r_i). Since T is a partial multi-rooted arborescence that contains $P_1 \cup P_2 \cup P_3$ as a subgraph, every weakly connected components of $G \setminus T$ has at most $\frac{|X|}{2}$ terminals. This finishes the proof of parts (a) and (b).

Property (c) follows from Lemma 11 and the fact that the conditions in Lemma 11 are satisfied. ◀

The algorithm for the multi-rooted version is the same as Algorithm 1 with the following two tweaks: (1) in the preprocessing step we remove vertices v where $d\big(\{r_1, \ldots, r_R\}, v\big) > \widetilde{\text{opt}}$, and (2) instead of Corollary 4 we apply Lemma 12 to obtain the subinstances.

Next, we analyze the cost and the running time of this algorithm.

▶ **Lemma 13** (Cost and running time, multi-rooted). *Consider an instance* $I = \big(G = (V, E), w, \{r_1, \ldots, r_R\}, X\big)$ *and a pair* $(I, \widetilde{\text{opt}})$. *Let* ℓ *and* $\mathcal{O}$ *be non-negative integers such that* $|X| \leq 2^\ell$ *and* $\widetilde{\text{opt}} \leq 2^{\mathcal{O}}$. *If* $\widetilde{\text{opt}} \geq \text{opt}$ *where* opt *is the optimal value of* I, *then* $\text{DST}(I, \widetilde{\text{opt}}) \leq \big(8 \cdot (R + \ell) + 1\big) \cdot \text{opt}$ *and the number of recursive calls is at most* $|X| \cdot 2^{2 \cdot \ell + \mathcal{O}}$.

Proof. The proof of the number of recursive calls is exactly the same as in the proof of Lemma 8. So we turn to proving the bound on the returned solution's cost.

The proof is by induction on $R + \ell + \mathcal{O}$. As in the proof of Lemma 8, we only need to focus on the case that $\widetilde{\text{opt}} \leq 2 \cdot \text{opt}$ and show that $\text{cost}(\mathcal{F}_2) \leq \big(8 \cdot (R + \ell) + 1\big) \cdot \text{opt}$.

Let $T = F \cup (\bigcup_{i=1}^{R} T_i)$ be the partial multi-rooted arborescence obtained from Lemma 12. Suppose T contains R' many of the roots. Then, exactly R' many of T_i's are non-empty. By Lemma 12 (a) we have that each non-empty T_i is consists of up to four shortest dipaths rooted at r_i so $\text{cost}(T_i) \leq 4 \cdot \widetilde{\text{opt}}$ because of the preprocessing step plus the fact that $\widetilde{\text{opt}} \leq 2 \cdot \text{opt}$, we conclude

$$\text{cost}(T \setminus F) \leq 8 \cdot R' \cdot \text{opt}\,. \tag{3}$$

Since T contains R' many roots, each subinstance I_{C_i} induced by $(G, T, C_1, \ldots, C_h)$ has at most $R - R' + 1$ many roots for $1 \leq i \leq h$. Furthermore, by Lemma 12 (b) each I_{C_i}'s has at most $\frac{|X|}{2} \leq 2^{\ell-1}$ many terminals. So by induction hypothesis, for $i = 1, \ldots, h$ we have

$$\text{cost}(\mathcal{F}_{C_i}) \leq \Big(8 \cdot \big((R - R' + 1) + \ell - 1\big) + 1\Big) \cdot \text{opt}_{I_{C_i}} \leq \big(8 \cdot (R - R' + \ell) + 1\big) \cdot \text{opt}_{I_{C_i}}\,. \tag{4}$$

Using Lemma 12 (c), the bounds in (3) and (4) we have

$$\begin{aligned}
\text{cost}(\mathcal{F}) &\leq \text{cost}(T \setminus F) + \sum_{i=1}^{h} \text{cost}(\mathcal{F}_{I_{C_i}}) \\
&\leq 8 \cdot R' \cdot \text{opt} + \big(8 \cdot (R - R' + \ell) + 1\big) \cdot \sum_{i=1}^{h} \text{opt}_{I_{C_i}} \\
&\leq 8 \cdot R' \cdot \text{opt} + \big(8 \cdot (R - R' + \ell) + 1\big) \cdot \text{opt} \\
&= \big(8 \cdot (R + \ell) + 1\big) \cdot \text{opt},
\end{aligned}$$

where the third inequality follows from the fact that $\sum_{i=1}^{h} \text{opt}_{I_{C_i}} \leq \text{opt}$ as the restriction of OPT on each G_{C_i} is a feasible solution for I_{C_i} and G_{C_i}'s are edge-disjoint.. ◀

Proof of Theorem 2. Note both of the tweaks in Algorithm 1 are implementable in polynomial time. The proof has exactly the same structure as in the proof of Theorem 1 with the difference that we use Lemma 13 here instead of Lemma 8. ◀

5 Concluding Remarks

One possible direction is to extend our result to minor-free families of graphs. However, as pointed out in [1, 8], minor-free (undirected) graphs do not have shortest-path separators. In [8], Cohen-Addad bypassed this difficulty by designing a new separator called a *mixed separator* for undirected minor-free graphs. It is not clear that analogous separators exist in directed graphs. For example, the mixed separators in [8] are obtained, in part, by contracting certain paths. These paths are obtained using structural results in minor-free graphs [18] and it is not clear how to find analogous paths in the directed case. Obtaining an $O(\log k)$-approximation for DST in minor-free graphs remains an interesting open problem.

References

1 Ittai Abraham and Cyril Gavoille. Object location using path separators. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 188–197, 2006.

2 MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Dániel Marx. Approximation schemes for steiner forest on planar graphs and graphs of bounded treewidth. *Journal of the ACM (JACM)*, 58(5):1–37, 2011.

3 Marshall Bern and Paul Plassmann. The steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32(4):171–176, 1989.

4 Glencora Borradaile, Philip Klein, and Claire Mathieu. An $O(n \log n)$ approximation scheme for steiner tree in planar graphs. *ACM Transactions on Algorithms (TALG)*, 5(3):1–31, 2009.

5 Jarosław Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *Journal of the ACM (JACM)*, 60(1):1–33, 2013.

6 Gruia Calinescu and Alexander Zelikovsky. The polymatroid steiner problems. *J. Combonatorial Optimization*, 33(3):281–294, 2005.

7 Moses Charikar, Chandra Chekuri, To-Yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation algorithms for directed steiner problems. *Journal of Algorithms*, 33(1):73–91, 1999.

8 Vincent Cohen-Addad. Bypassing the surface embedding: approximation schemes for network design in minor-free graphs. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 343–356, 2022.

9 Erik D Demaine, MohammadTaghi Hajiaghayi, and Philip N Klein. Node-weighted steiner tree and group steiner tree in planar graphs. *ACM Transactions on Algorithms (TALG)*, 10(3):1–20, 2014.

10 Zachary Friggstad and Ramin Mousavi. A constant-factor approximation for quasi-bipartite directed steiner tree on minor-free graphs. *arXiv preprint*, 2021. `arXiv:2111.02572`.

11 Rohan Ghuge and Viswanath Nagarajan. Quasi-polynomial algorithms for submodular tree orienteering and other directed network design problems. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1039–1048. SIAM, 2020.

12 Fabrizio Grandoni, Bundit Laekhanukit, and Shi Li. $O(\log 2k/\log\log k)$-approximation algorithm for directed steiner tree: a tight quasi-polynomial-time algorithm. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 253–264, 2019.

13 Eran Halperin and Robert Krauthgamer. Polylogarithmic inapproximability. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 585–594, 2003.

14 Marek Karpinski and Alexander Zelikovsky. New approximation algorithms for the steiner tree problems. *Journal of Combinatorial Optimization*, 1(1):47–65, 1997.

15 Richard J Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

16 Richard J Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM journal on computing*, 9(3):615–627, 1980.

17 Hans Jürgen Prömel and Angelika Steger. A new approximation algorithm for the steiner tree problem with performance ratio 5/3. *Journal of Algorithms*, 36(1):89–101, 2000.

18 Neil Robertson and Paul D Seymour. Graph minors. xvi. excluding a non-planar graph. *Journal of Combinatorial Theory, Series B*, 89(1):43–76, 2003.

19 Gabriel Robins and Alexander Zelikovsky. Tighter bounds for graph steiner tree approximation. *SIAM Journal on Discrete Mathematics*, 19(1):122–134, 2005.

20 Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of the ACM (JACM)*, 51(6):993–1024, 2004.

21 Alexander Zelikovsky. A series of approximation algorithms for the acyclic directed steiner tree problem. *Algorithmica*, 18(1):99–110, 1997.

22 Alexander Z Zelikovsky. An 11/6-approximation algorithm for the network steiner problem. *Algorithmica*, 9(5):463–470, 1993.

6 Proof of Lemma 5

Proof. Let $\Delta := \max_{t \in X} \Big\{ d\big(\{r_1, \ldots, r_R\}, t\big) \Big\}$, i.e., Δ is the maximum distance from any root to a terminal. Let opt_I be the optimal value of instance I. Then, $\Delta \leq \text{opt}_I \leq k \cdot \Delta$.

If $\Delta = 0$, then $\text{opt}_I = 0$ and the collection of all shortest dipaths from the roots to the terminals is a solution of cost 0. So we assume $\Delta > 0$.

We can safely remove any edge e having $c_e > k \cdot \Delta$ and any Steiner node v (along with its incident edges) having $d(\{r_1, \ldots, r_R\}, v) > k \cdot \Delta$ since no optimal solution of I uses e or v. Since we have only deleted elements of G, it remains planar.

Define a new edge costs $c'_e := \lceil c_e \cdot \frac{n}{\epsilon \cdot \Delta} \rceil$ and form the instance $I' = (G, c', \{r_1, \ldots, r_R\}, X)$. Note for any shortest dipath P starting at root r_i, we have

$$\text{cost}_{c'}(P) \leq \sum_{e \in P} c'_e \leq \sum_{e \in P} (c_e \cdot \frac{n}{\epsilon \cdot \Delta} + 1) \leq \text{cost}_c(P) \cdot \frac{n}{\epsilon \cdot \Delta} + n \leq \frac{n \cdot k}{\epsilon} + n,$$

where the last inequality follows because all the distances from the root has length at most $k \cdot \Delta$. So all the shortest dipaths starting at r in I' are bounded by $O(\frac{n^2}{\epsilon})$.

Let $\text{opt}_{I'}$ be the optimal value of instance I'. Similar calculation as before shows $\text{opt}_{I'} \leq \frac{n}{\epsilon \cdot \Delta} \cdot \text{opt}_I + n$.

Let F be an α-approximate solution for I'. Then, we have

$$\begin{aligned}
\text{cost}_c(F) &\leq \frac{\epsilon \cdot \Delta}{n} \cdot \text{cost}_{c'}(F) \\
&\leq \frac{\epsilon \cdot \Delta}{n} \cdot \alpha \cdot \text{opt}_{I'} \\
&\leq \frac{\epsilon \cdot \Delta}{n} \cdot \alpha \cdot (\frac{n}{\epsilon \cdot \Delta} \cdot \text{opt}_I + n) \\
&\leq \alpha \cdot \text{opt}_I + \alpha \cdot \epsilon \cdot \Delta \\
&\leq \alpha \cdot (1 + \epsilon) \cdot \text{opt}_I,
\end{aligned}$$

where the first inequality follows because $c'_e \geq c_e \cdot \frac{n}{\epsilon \cdot \Delta}$ and the last because $\text{opt}_I \geq \Delta$. ◀

Parallel Self-Testing of EPR Pairs Under Computational Assumptions

Honghao Fu
CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA

Daochen Wang
QuICS, University of Maryland, College Park, MD, USA

Qi Zhao
QuICS, University of Maryland, College Park, MD, USA
QICI, The University of Hong Kong, China

Abstract

Self-testing is a fundamental feature of quantum mechanics that allows a classical verifier to force untrusted quantum devices to prepare certain states and perform certain measurements on them. The standard approach assumes at least two spatially separated devices. Recently, Metger and Vidick [39] showed that a single EPR pair of a single quantum device can be self-tested under *computational assumptions*. In this work, we generalize their results to give the first parallel self-test of N EPR pairs and measurements on them in the single-device setting under the same computational assumptions. We show that our protocol can be passed with probability negligibly close to 1 by an honest quantum device using $\mathrm{poly}(N)$ resources. Moreover, we show that any quantum device that fails our protocol with probability at most ϵ must be $\mathrm{poly}(N, \epsilon)$-close to being honest in the appropriate sense. In particular, our protocol can test any distribution over tensor products of computational or Hadamard basis measurements, making it suitable for applications such as device-independent quantum key distribution [38] under computational assumptions. Moreover, a simplified version of our protocol is the first that can efficiently certify an arbitrary number of qubits of a single cloud quantum computer using only classical communication.

2012 ACM Subject Classification Theory of computation → Interactive proof systems

Keywords and phrases Quantum complexity theory, self-testing, LWE

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.64

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2201.13430` [21]

Funding *Honghao Fu*: NSF QLCI program (grant OMA-2016245).
Daochen Wang: Army Research Office (grant W911NF-20-1-0015); the Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Accelerated Research in Quantum Computing program; and the National Science Foundation (grant DMR-1747426).
Qi Zhao: Department of Defense through the QuICS Hartree Postdoctoral Fellowship.

Acknowledgements We especially thank Carl Miller, Tony Metger, and Thomas Vidick for many helpful discussions and correspondence. We also thank Nai-Hui Chia, Shih-Han Hung, Yi Lee, Atul Mantri, and Jiayu Zhang for helpful discussions.

1 Introduction

Self-testing is a fundamental feature of quantum mechanics that allows a classical verifier to force a quantum device (sometimes called prover) to prepare certain states and measure them in certain bases up to local isometries [4, 47, 50, 43, 7, 35, 48, 18, 25, 8, 36, 37, 41, 42, 13, 20, 46, 45, 19, 28]. In the standard *nonlocal setting*, the key assumption is that there are two

or more spatially separated devices. However, it is difficult to certify spatial separation in practice, especially if the devices fall outside our physical control. Therefore, it is interesting to ask whether we can replace this assumption by another one so that we can self-test a *single* quantum device. We illustrate the nonlocal and single-device settings in Fig. 1.

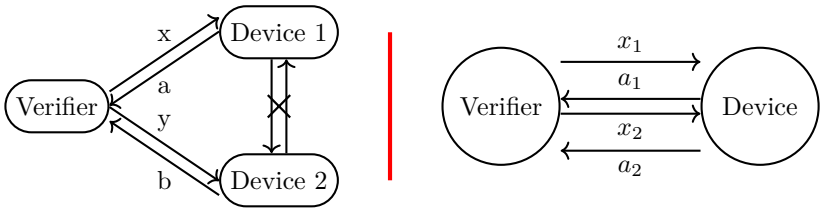


Figure 1 Self-testing in the nonlocal setting (left) involves (at least) two spatially separated devices that cannot communicate. In the single-device setting (right), there is only one device.

Computational self-testing. Recently, beginning with seminal work by Mahadev [33] on the classical verification of quantum computations, a series of works, e.g., [24, 6, 14, 1, 52, 53, 29, 5, 27, 32, 55, 39, 38, 40], have explored how computational assumptions can be leveraged by a classical verifier to control a single quantum device in certain ways. Typically, the assumption used is that the Learning-With-Errors (LWE) [44] problem is hard to solve efficiently, even for quantum computers, which is a standard assumption. However, except for [24, 53, 39, 38, 40], the level of control established in these works is much weaker than in nonlocal self-testing. For example, if a device passes Mahadev's verification protocol [33], it only means that, to quote [39], "*there exists* a quantum state such that the distribution over the prover's answers *could have been* produced by performing the requested measurements on this state". We do not know whether the prover *actually* prepared that state and performed the requested measurements on it.

Metger and Vidick [39] are the first to explicitly propose the self-testing of a single device under computational assumptions. The main limitation of [39] and follow-up work [40] is that they only self-test two and three qubits, respectively. In this work, we introduce a self-test that certifies the preparation and measurement of N EPR pairs in the computational (or single-device) setting. Our work differs from the concurrent work [23] in that [23] certifies the preparation (but not measurement) of BB84 states.

Main results. We give a self-test that certifies the EPR pairs:

$$\left\{ |\tau^{\diamond,v}\rangle \coloneqq \frac{1}{\sqrt{2^N}} \bigotimes_{i=1}^{N} (\sigma^X)^{v_i} \otimes (\sigma^X)^{v_{N+i}} (|0\rangle_i |+\rangle_{N+i} + |1\rangle_i |-\rangle_{N+i}) \;\middle|\; v \in \{0,1\}^{2N} \right\},$$

and states $\{|\tau^{\theta,v}\rangle \mid \theta \in \{0,1\ldots,2N\}, v \in \{0,1\}^{2N}\}$, which is a subset of BB84 states specified in Section 3.

Moreover, our self-test certifies any distribution over tensor products of computational (Pauli-Z) or Hadamard (Pauli-X) basis measurements on $2N$ qubits:

$$\left\{ \left\{ \Pi_q^u \coloneqq |B_{q_1}^{u_1}\rangle\langle B_{q_1}^{u_1}| \otimes \ldots \otimes |B_{q_{2N}}^{u_{2N}}\rangle\langle B_{q_{2N}}^{u_{2N}}| \;\middle|\; u \in \{0,1\}^{2N} \right\} \;\middle|\; q \in \{0,1\}^{2N} \right\}, \tag{1}$$

where $|B_0^0\rangle \coloneqq |0\rangle, |B_0^1\rangle \coloneqq |1\rangle, |B_1^0\rangle \coloneqq |+\rangle$, and $|B_1^1\rangle \coloneqq |-\rangle$.

Our self-test generalizes protocols in [24, 39] and uses the Extended Noisy Trapdoor Claw-Free function Families (ENTCFs) introduced by Mahadev in [33]. An ENTCF consists of two function-pair families, a claw-free family $\mathcal{F}$ and an injective family $\mathcal{G}$, that have certain cryptographic properties under the LWE hardness assumption.

In our self-test, the classical verifier first samples $\theta \in \{0, 1, \ldots, 2N\} \cup \{\diamond\}$ uniformly at random. Then it generates the public keys and trapdoors of $2N$ function pairs from $\mathcal{F} \cup \mathcal{G}$ according to θ as follows.

1. $\theta = 0$: all pairs are from $\mathcal{G}$.
2. $\theta \in \{1, \ldots, 2N\}$: the θ^{th} pair is from $\mathcal{F}$ and the remaining $2N - 1$ pairs are from $\mathcal{G}$.
3. $\theta = \diamond$: all pairs are from $\mathcal{F}$.

The verifier sends the public keys to the device. The device then sends back $2N$ images, $y_1, \ldots, y_{2N}$, of these function pairs – these play the role of a commitment. In the second round, the verifier either (i) checks the commitment by asking for preimages of the y_is and accepts or rejects accordingly, or (ii) asks for an equation involving the preimages of the y_is. In case (ii), there is a final round where the verifier sends with probability $1/2$ a uniformly random $q \in \{0^{2N}, 1^{2N}, 0^N1^N, 1^N0^N\}$ and with probability $1/2$ a random $q \in \{0,1\}^{2N}$ according to some distribution μ of its choosing. The device sends back the result $u \in \{0,1\}^{2N}$ of performing some measurement $\{P_q^u\}_u$. The verifier lastly checks that u is consistent with measuring $|\tau^{\theta,v}\rangle$ using $\{\Pi_q^u\}_u$, where $v \in \{0,1\}^{2N}$ is some bitstring that the verifier can compute efficiently using the trapdoors, and accepts or rejects accordingly. We allow our verifier to pick any distribution μ on $q \in \{0,1\}^{2N}$ so that our protocol can be composed with other protocols. For example, in our applications, the distribution on $q \in \{0,1\}^{2N}$ is non-uniform.

▶ **Theorem 1** (Informal). *Let $\lambda \in \mathbb{N}$ be a security parameter and let $N = \text{poly}(\lambda)$ be a fixed polynomially-bounded function of λ. Assuming the LWE problem of size λ cannot be solved in $\text{poly}(\lambda)$ time, our self-test satisfies the following properties.*

Completeness. *Using $\text{poly}(\lambda)$ qubits and quantum gates, a quantum device can prepare one of the $2N$-qubit states in $\{|\tau^{\theta,v}\rangle \mid \theta \in \{0, 1, \ldots, 2N\} \cup \{\diamond\}, v \in \{0,1\}^{2N}\}$ and measure it using $\{\Pi_q^u \mid u \in \{0,1\}^{2N}\}$ upon question $q \in \{0,1\}^{2N}$ to pass our self-test with probability $\geq 1 - \text{negl}(\lambda)$. Moreover, the verifier can be classical and run in $\text{poly}(\lambda)$ time.*

Soundness. *If a quantum device passes our self-test in $\text{poly}(\lambda)$ time with probability $\geq 1 - \epsilon$, then the device must have prepared a (sub-normalized) state $\sigma^{\theta,v}$, measured it using $\{P_q^u\}_u$, and received outcome u, such that*

$$\sum_{v \in \{0,1\}^{2N}} \|V\sigma^{\theta,v}V^\dagger - |\tau^{\theta,v}\rangle\langle\tau^{\theta,v}| \otimes \alpha^{\theta,v}\|_1 \leq O(N^{7/4}\epsilon^{1/32}) \quad \textit{and} \tag{2}$$

$$\mathbb{E}_{q \leftarrow \mu}\Big[\sum_{u,v \in \{0,1\}^{2N}} \|VP_q^u\,\sigma^{\theta,v}\,P_q^uV^\dagger - \Pi_q^u\,|\tau^{\theta,v}\rangle\langle\tau^{\theta,v}|\,\Pi_q^u \otimes \alpha^{\theta,v}\|_1\Big] \leq O(N^2\epsilon^{1/32}), \tag{3}$$

where $\theta \in \{0, 1, \ldots, 2N\} \cup \{\diamond\}$, μ is the distribution on $\{0,1\}^{2N}$ chosen by the verifier in our self-test, $u, v \in \{0,1\}^{2N}$ are known to the verifier, V is an efficient isometry independent of $\{\theta, \mu, u, v\}$, and the $\alpha^{\theta,v}$s are some auxiliary states that are computationally indistinguishable from some fixed state α.

Note that $\theta = \diamond$ corresponds to self-testing EPR pairs. We also highlight the $\text{poly}(N, \epsilon)$ soundness error (or robustness) that we achieve. Good robustness is critical if we want to use our self-test in practice because real quantum devices are imperfect. The more imperfect a device is, the more robust a self-test needs to be to control it.

Techniques. The main challenge is to prove soundness. We give a high-level overview here and provide more details in Section 4. We start by defining $4N$ observables of the device $\{X_i, Z_i \mid i \in [2N]\}$ using its measurement operators. The strategy is to characterize these observables as the standard σ_i^X and σ_i^Z Pauli observables on $2N$ qubits where i indexes

those qubits. Then, we characterize the device's states by their invariance under products of projectors corresponding to these observables and the device's measurements as products of these projectors. To characterize X_i and Z_j, we first generalize techniques in [39] to show that X_i and Z_j obey certain state-dependent commutation and anti-commutation relations (Proposition 10). To carry out the generalization, it is important for the verifier to select θ from the set $[2N] \cup \{0, \diamond\}$ for two reasons. The first is that they allow us to bound the failure probability associated with *each* σ^θ by $2N + 2$ (the number of possible θs) times the *average* failure probability over all θs. The second is that this restricted set of θs suffices for us to characterize X_i and Z_i as σ_i^X and σ_i^Z. Intuitively, $\theta = 0$ is used to characterize $\{Z_1, \ldots, Z_{2N}\}$, $\theta \in [2N]$ is used to characterize X_θ, and $\theta = \diamond$ is used to characterize EPR pairs. We give a more precise correspondence in Table 1.

Then, we introduce new techniques to handle products of projectors corresponding to the X_i, Z_i observables. These techniques differ significantly from [39] because their techniques are not susceptible to generalization to arbitrary N (as we discuss after Proposition 15). These techniques also differ significantly from those used in nonlocal self-testing because we lack the perfect state-*independent* commutation relations between observables on two spatially-separated devices. More specifically, we introduce a "operator-state commutation" relation (Proposition 11) that, together with the computational indistinguishability of the σ^θs (which follows from the LWE hardness assumption), gives us the ability to "commute an observable past a state". We then use this ability to handle products of projectors. The usefulness of the ability to commute can be seen in the following simple example. Observe that $X_1 Z_2 X_3 \psi = Z_2 X_1 X_3 \psi$ (1) does not follow from the commutation relation $X_1 Z_2 \psi = Z_2 X_1 \psi$, where ψ is some density operator. However, (1) would follow if we could commute X_3 past ψ first because X_1 and Z_2 would then be directly next to ψ. Having all (1)-like relations involving products of up to N X_i and Z_i implies that these observables can be characterized as σ_i^X and σ_i^Z respectively, which follows from results in approximate representation theory [51, 26]. We remark that the preceding discussion is for intuition only: in fact, our proof directly shows that an explicit "swap" isometry (defined in Definition 12) approximately maps X_i and Z_i to σ_i^X and σ_i^Z respectively.

Applications. We present two applications of our result, the first is for device-independent (DI) quantum key distribution (QKD), and the second is for dimension testing. We stress that for both applications, we crucially rely on the characterization of *measurements* in Equation (3) of Theorem 1.

DIQKD. A DI protocol is one where the parties involved do not need to trust the inner working of the devices they use to be sure that the devices have successfully implemented the protocol. A QKD protocol is one for establishing information-theoretically secure keys between two parties. Previous DIQKD protocols rely on the nonlocal assumption. This assumption is usually justified experimentally by spatially separating two devices by a large distance, which is difficult to implement. Recently, Metger et. al. [38] proposed a different setting for DIQKD: they replace the nonlocal assumption with the assumption that the two devices are computationally bounded. However, since their protocol sequentially repeats the self-test in [39], their soundness proof relies on the IID assumption that the device behaves identically and independently at each repetition to argue that it has prepared and measured many EPR pairs.

Our DIQKD protocol consists of a random number of "test rounds" followed by a final "generation round", where both round types are based on our self-test. The N EPR pairs certified in the generation round are used to generate $\Omega(N)$ shared keys. Because of the

parallel nature of our self-test, our DIQKD protocol does not require the IID assumption. We sketch a soundness proof that uses a "cut-and-choose" argument from [23, Theorem 4.33] to upper bound the failure probability of the device in the generation round, conditioned on the protocol not aborting in the test rounds. This argument does not require an IID assumption *between* rounds. Then, we use Equation (3) of Theorem 1 to lower bound the key rate, which does not require an IID assumption *within* any round. Hence we remove the IID assumption overall. The application of our self-test to remove the IID assumption from DIQKD in the computational setting can be viewed as analogous to the application of the nonlocal self-test to remove the IID assumption from DIQKD in the usual nonlocal setting [45].

Dimension testing. Our dimension-test is a simplified version of our self-test and is inspired by the non-local dimension test in [11] and its exposition in [51, Section 2.5.2]. The protocol in [11] works as follows. The verifier chooses a random bit $\theta \in \{0,1\}$ and random bitstring $x \in \{0,1\}^n$ and sends n qubits to the device such that the qubits encode x in the computational basis ($\theta = 0$) or in the Hadamard basis ($\theta = 1$). After the device has received all n qubits, the verifier sends θ to the device and asks it to return a bitstring $x' \in \{0,1\}^n$. If $x' = x$, the verifier certifies that the device has a large quantum dimension. Our protocol can be viewed as a version of this protocol, where the verifier classically delegates the preparation of the appropriate n-qubit states to the prover in a secure manner. Although our protocol is inspired by [11], our security proof uses Theorem 1 and differs significantly from that in [11].

We prove that, under the same computational assumptions as in Theorem 1, if a quantum device runs in $\text{poly}(\lambda)$ time and passes our dimension-test with probability $\geq 1 - \epsilon$, then its quantum dimension is at least $(1 - O(N^2\epsilon^{1/32}))2^N$ $(*)$. To obtain a non-trivial bound, it suffices to estimate ϵ to precision $1/\text{poly}(N)$, which can be done by repeating the dimension-test $\text{poly}(N)$ times. Since a single run of the dimension test also only takes $\text{poly}(N)$ time, the total time taken is $\text{poly}(N)$. Intuitively, we prove $(*)$ by using Equation (3) of Theorem 1 to argue that the Hilbert space $\mathcal{H}$ of the device must be able to accommodate all possible post-measurement states that could result from performing a Hadamard basis measurement of N qubits in a computational basis state. Since there are 2^N such post-measurement states, and they are all orthogonal, we deduce a quantum dimension lower bound of 2^N. A formal proof is more challenging because Equation (3) of Theorem 1 gives an approximation and we need to prove that the rank of a quantum state is robust against the approximation error.

Compared to nonlocal dimension-tests [9, 10, 17], the advantage of ours is that we do not need to assume spatial separation between multiple devices. Compared to prepare-and-measure dimension-tests [22, 12, 13, 11], the advantage of ours is that the verifier does not need to be quantum – all computations and communications are classical. To the best of our knowledge, our dimension-test is the first[1] that can test for an arbitrary quantum dimension in the computational setting. In fact, whether this is possible was recently raised as an open question by Vidick in [51, pg. 84].

Discussion. One interesting direction is to further improve the efficiency and robustness of our protocol. When $N = \lambda$, one bottleneck in improving the efficiency is that sending (the public key of) one function pair already requires $\text{poly}(\lambda) = \text{poly}(N)$ bits of communication. In recent work, it has been shown that, instead of sending the public keys, the verifier can apply a *succinct batch key generation algorithm* to reduce the cost of sending public keys [3]. We expect that techniques in [3] can be used to shorten other messages of our protocol as

[1] More recently, [34] also claims a dimension test using completely different methods.

well. Turning to robustness, we note that there exists a nonlocal self-test [41] which uses poly(N) bits of communication and achieves robustness poly(ϵ). It might be possible to combine our techniques with those in [41] to achieve similar robustness in the computational setting. Another interesting question to ask is what MIP* protocols can be compiled into computation delegation protocols under computational assumptions. For comparison, it has been shown that classical MIP protocols sound against non-signalling provers can be turned into computation delegation protocols [49, 31]. It would also be interesting to see if a systematic way exists to translate nonlocal self-tests into computational ones. We note that [29] suggests that the two settings might not be too different at a conceptual level by presenting a test of quantumness in the computational setting that closely resembles the nonlocal CHSH test [16]. Recently, Kalai et. al. proposed a way to construct a proof-of-quantumness protocol from any nonlocal game with a classical and quantum separation using quantum homomorphic encryption [30]. However, it is unknown if the aforementioned protocols are quantumly sound. Going beyond quantum dimension testing, it would be interesting to see if our protocol can be combined with those that test quantum circuit depth [15, 2] to give a protocol that tests the quantum volume of a quantum computer.

2 Preliminaries

Notation. $\mathbb{N}$ is the set of positive integers. For $k \in \mathbb{N}$, we write $[k] \coloneqq \{1, 2, \ldots, k\}$. For a probability distribution μ on X, we use the notation $x \leftarrow_\mu X$ to mean that x is sampled from X according to μ. $\mathcal{H}$ denotes a finite-dimensional Hilbert space, $\mathcal{L}(\mathcal{H})$ denotes the set of linear operators on $\mathcal{H}$, and $\mathrm{Pos}(\mathcal{H})$ denotes the set of positive semi-definite operators on $\mathcal{H}$. We sometimes refer to operators in $\mathrm{Pos}(\mathcal{H})$ or vectors in $\mathcal{H}$, not necessarily normalized, as (quantum) states. For an operator $X \in \mathcal{L}(\mathcal{H})$, we write $\|X\|_p \coloneqq \mathrm{Tr}[|X|^p]^{1/p}$, where $|X| \coloneqq \sqrt{X^\dagger X}$, for the Schatten p-norm. For $\phi, \psi \in \mathcal{L}(\mathcal{H})$, we write $\phi \approx_\epsilon \psi$ to mean $\|\phi - \psi\|_1^2 \leq O(\epsilon)$. For $A, B \in \mathcal{L}(\mathcal{H})$ and $\psi \in \mathrm{Pos}(\mathcal{H})$, we write $\|A\|_\psi^2 \coloneqq \mathrm{Tr}[A^\dagger A \psi] = \|A\sqrt{\psi}\|_2^2$ and $A \approx_{\epsilon,\psi} B \iff \|A - B\|_\psi^2 \leq O(\epsilon)$. The single-qubit Z and X Pauli operators are denoted $\sigma_Z \coloneqq \left(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right)$ and $\sigma_X \coloneqq \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$ which have eigenstates $\{|0\rangle \coloneqq \left(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right), |1\rangle \coloneqq \left(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right)\}$ and $\{|(-)^0\rangle \coloneqq \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), |(-)^1\rangle \coloneqq \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\}$, respectively.

We write $\lambda \in \mathbb{N}$ for the security parameter. Most quantities in this work are dependent on λ. Therefore, for convenience, we often make the dependence implicit. A function $f : \mathbb{N} \to \mathbb{R}$ is said to be negligible if for any polynomial $p \in \mathbb{R}[x]$, $\lim_{\lambda \to \infty} f(\lambda)p(\lambda) = 0$. We denote such functions by $\mathrm{negl}(\lambda)$.

ENTCFs. We informally summarize the properties that we employ of Extended Noisy Trapdoor Claw-free function Families (ENTCFs). For full details about the properties of ENTCFs, see the arXiv version of [33].

Let $\lambda \in \mathbb{N}$ be a security parameter. Let $\mathcal{X} \subseteq \{0,1\}^w$ and $\mathcal{Y}$ be finite sets that depend on λ, where $w = w(\lambda)$ is some integer that is a polynomially-bounded function of λ. An ENTCF consists of two families of function pairs, $\mathcal{F}$ and $\mathcal{G}$. Function pairs from these two families are labeled by public keys. The set of public keys for $\mathcal{F}$ is denoted by $\mathcal{K}_\mathcal{F}$, and the set of public keys for $\mathcal{G}$ is denoted by $\mathcal{K}_\mathcal{G}$. For $k \in \mathcal{K}_\mathcal{F}$, a function pair $(f_{k,0}, f_{k,1})$ from $\mathcal{F}$ is called a *claw-free* pair. For $k \in \mathcal{K}_\mathcal{G}$, a function pair $(f_{k,0}, f_{k,1})$ from $\mathcal{G}$ is called an *injective* pair. For any $k \in \mathcal{K}_\mathcal{F} \cup \mathcal{K}_\mathcal{G}$, the functions[2] $f_{k,0}$, $f_{k,1} \colon \mathcal{X} \to \mathcal{Y}$. Note that the keys and function pairs of an ENTCF are functions of λ. We use the terms "efficient" and "negligible" to refer to poly(λ)-time and negl(λ) respectively. We need the following properties of ENTCFs:

[2] This is a convenient simplification. These functions actually map to probability distributions on $\mathcal{Y}$. See Section 2.2 of the full version for details.

1. *Efficient function generation property* [33, Definitions 4.1 (1), 4.2 (1)]. There exist efficient classical probabilistic algorithms $\mathrm{Gen}_{\mathcal{F}}$ and $\mathrm{Gen}_{\mathcal{G}}$ for $\mathcal{F}$ and $\mathcal{G}$ respectively with $\mathrm{Gen}_{\mathcal{F}}(1^\lambda) \to (k \in \mathcal{K}_{\mathcal{F}}, t_k)$ and $\mathrm{Gen}_{\mathcal{G}}(1^\lambda) \to (k \in \mathcal{K}_{\mathcal{G}}, t_k)$, where t_k is known as a trapdoor.
2. *(Disjoint) injective pair property* [33, Definitions 4.1 (2), 4.2 (2)]. For all $k \in \mathcal{K}_{\mathcal{F}} \cup \mathcal{K}_{\mathcal{G}}$, $x, x' \in \mathcal{X}$ with $x \neq x'$, and $b \in \{0,1\}$, $f_{k,b}(x) \neq f_{k,b}(x')$. For all $k \in \mathcal{K}_{\mathcal{F}}$ and $x \in \mathcal{X}$, there exists an $x' \neq x$ such that $f_{k,0}(x) = f_{k,1}(x')$. We call any such pair of (x, x') a *claw*.
3. *Efficient range superposition property* [33, Definitions 4.1 (3.c), 4.2 (3.b), 4.3 (1)]. Given $k \in \mathcal{K}_{\mathcal{F}} \cup \mathcal{K}_{\mathcal{G}}$, there exists an efficient quantum algorithm that prepares a state that is negligibly close to $|\psi\rangle := \frac{1}{\sqrt{2\cdot|\mathcal{X}|}} \sum_{b\in\{0,1\}} \sum_{x\in\mathcal{X}} |b\rangle\,|x\rangle\,|f_{k,b}(x)\rangle$, in trace distance.
4. *Efficient decoding property* [33, Definitions 4.1 (2, 3.a, 3.b), 4.2 (2, 3.a), 4.3 (1)]. We define the following "decoding maps" that decode the output of functions from an ENTCF. For $k \in (\mathcal{K}_{\mathcal{F}} \cup \mathcal{K}_{\mathcal{G}})^m$ with $m = \mathrm{poly}(\lambda)$, $k_{\mathcal{G}} \in \mathcal{K}_{\mathcal{G}}$, $k_0 \in \mathcal{K}_{\mathcal{F}} \cup \mathcal{K}_{\mathcal{G}}$, and $k_{\mathcal{F}} \in \mathcal{K}_{\mathcal{F}}$

   $$\mathrm{CHK}(k, y, b, x) = 0 \text{ if } y_i = f_{k_i,b_i}(x_i) \text{ for all } i \in [m], \text{ else } = 1$$
   $$\hat{b}(k_{\mathcal{G}}, y) = b \text{ if } y \in \mathrm{Im}(f_{k_{\mathcal{G}},b}), \text{ else } = \perp$$
   $$\hat{x}(b, k_0, y) := x \text{ if } f_{k_0,b}(x) = y, \text{ else } = \perp$$
   $$\hat{h}(k_{\mathcal{F}}, y, d) := d \cdot (\hat{x}(0, k_{\mathcal{F}}, y) \oplus \hat{x}(1, k_{\mathcal{F}}, y)) \text{ if } y \in \mathrm{Im}(f_{k_{\mathcal{F}},0}) \text{ and } d \neq 0^w, \text{else } = \perp .$$

   The efficient decoding property states that $\hat{b}$, $\hat{x}$, and $\hat{h}$ can be computed efficiently given a trapdoor t_k for k by a classical deterministic algorithm and that CHK can be computed efficiently even without a trapdoor by a classical deterministic algorithm.
5. *Adaptive hardcore bit property* [33, Definition 4.1 (4)]. There does not exist an efficient quantum algorithm that, given $k \leftarrow \mathrm{Gen}_{\mathcal{F}}(1^\lambda)_{\mathrm{key}}$, can compute $b \in \{0,1\}$ and $x_b \in \mathcal{X}$ for some $b \in \{0,1\}$, $d \in \{0,1\}^w \backslash \{0^w\}$, and, with non-negligible advantage, a bit $d \cdot (x_0 \oplus x_1) \in \{0,1\}$ such that (x_0, x_1) is a claw.
6. *Injective invariance property* [33, Definition 4.3 (2)]. There does not exist an efficient quantum algorithm that can distinguish between the marginal key distributions of $\mathrm{Gen}_{\mathcal{F}}(1^\lambda)$ and of $\mathrm{Gen}_{\mathcal{G}}(1^\lambda)$ with non-negligible advantage.

3 Completeness of self-testing protocol

In this section, we present our self-testing protocol in Fig. 2. We sketch a proof of its completeness (Theorem 2), partly to establish some notation. For details, see Section 3 of the full version.

▶ **Theorem 2.** *There exists an efficient quantum device that is accepted by our self-testing protocol with probability* $\geq 1 - \mathrm{negl}(\lambda)$. *Moreover, the classical verifier is efficient.*

Proof sketch. In the first round, for each $i \in [2N]$, by the efficient range superposition property of ENTCFs (Item 3), the device uses k_i to efficiently prepare a state that is negligibly close to

$$|\psi_i\rangle := \frac{1}{\sqrt{2\cdot|\mathcal{X}|}} \sum_{b\in\{0,1\}} \sum_{x\in\mathcal{X}} |b\rangle\,|x\rangle\,|f_{k_i,b}(x)\rangle\,.$$

Then, the device measures the (image) y register of $|\psi_i\rangle$ and sends the outcome to the verifier. By the (disjoint) injective pair property of ENTCFs (Item 2), after the y measurement, the state $|\psi_i\rangle$ collapses to $|\phi_i\rangle\,|y_i\rangle$, where

$$|\phi_i\rangle := \begin{cases} |\hat{b}(k_i, y_i)\rangle\,|\hat{x}(k_i, y_i)\rangle & \text{if } k_i \in \mathcal{K}_{\mathcal{G}}, \\ \frac{1}{\sqrt{2}}(|0\rangle\,|\hat{x}_0(k_i, y_i)\rangle + |1\rangle\,|\hat{x}_1(k_i, y_i)\rangle) & \text{if } k_i \in \mathcal{K}_{\mathcal{F}}. \end{cases}$$

1. Input: $\lambda \in \mathbb{N}$. Set $N = \mathrm{poly}(\lambda)$. Given a distribution μ on $\{0,1\}^{2N}$. Sample $\theta \leftarrow_U [2N] \cup \{0, \diamond\}$ uniformly at random. Sample $2N$ key-trapdoor pairs $(k_1, t_{k_1}), \ldots, (k_{2N}, t_{k_{2N}})$ from an ENTCF according to θ as follows:

   $\theta \in [2N]$: the θ-th key-trapdoor pair is sampled from $\mathrm{Gen}_{\mathcal{F}}(1^\lambda)$ and the remaining $2N-1$ pairs are all sampled from $\mathrm{Gen}_{\mathcal{G}}(1^\lambda)$.

   $\theta = 0$: all the key-trapdoor pairs are sampled from $\mathrm{Gen}_{\mathcal{G}}(1^\lambda)$.

   $\theta = \diamond$: all the key-trapdoor pairs are sampled from $\mathrm{Gen}_{\mathcal{F}}(1^\lambda)$.

Send the keys $k = (k_1, \ldots, k_{2N})$ to the device.

2. Receive $y = (y_1, \ldots, y_{2N}) \in \mathcal{Y}^{2N}$ from the device.

3. Sample round type "preimage" or "Hadamard" uniformly at random and send to the device.

Case "preimage": receive

$$(b, x) = (b_1, \ldots, b_{2N}, x_1, \ldots, x_{2N})$$

from the device, where $b \in \{0,1\}^{2N}$ and $x \in \{0,1\}^{2Nw}$.
If $\mathrm{CHK}(k_i, y_i, b_i, x_i) = 0$ for all $i \in [2N]$, **accept**, else **reject**.

Case "Hadamard": receive

$$d = (d_1, \ldots, d_{2N}) \in \{0,1\}^{2Nw}$$

from the device.

4. With probability $1/2$, sample $q \leftarrow_U \{0^{2N}, 1^{2N}, 0^N 1^N, 1^N 0^N\}$ uniformly at random, and with probability $1/2$ sample $q \leftarrow_\mu \{0,1\}^{2N}$ according to the distribution μ. Send q to the device.

Receive $u \in \{0,1\}^{2N}$ from the device.

case A $\theta = 0$ and
if $q_i = 0$ and $\hat{b}(k_i, y_i) \neq u_i$ for some $i \in [2N]$, **reject**,
else **accept**.

case B $\theta \in [2N]$ and
if $q_i = 0$ and $\hat{b}(k_i, y_i) \neq u_i$ for some $i \neq \theta$, **reject**,
if $q_\theta = 1$ and $\hat{h}(k_\theta, y_\theta, d_\theta) \oplus \hat{b}(k_{\theta+N}, y_{\theta+N}) \neq u_\theta$, **reject**,
else **accept**.

case C $\theta = \diamond$ and
if $q_i = 0$, $q_{N+i} = 1$ and $u_i \oplus u_{N+i} \neq \hat{h}(k_{N+i}, y_{N+i}, d_{N+i})$ for some $i \in [N]$, **reject**,
if $q_i = 1$, $q_{N+i} = 0$ and $u_i \oplus u_{N+i} \neq \hat{h}(k_i, y_i, d_i)$ for some $i \in [N]$, **reject**,
else **accept**.

Figure 2 A protocol that self-tests EPRs of a computationally efficient device.

In the following, we use the shorthand $\hat{b}_i \coloneqq \hat{b}(k_i, y_i) \in \{0,1\}$ and, for $a \in \{0,1\}$, $\hat{x}_{a,i} \coloneqq \hat{x}(a, k_i, y_i) \in \mathcal{X}$.

In the second round, there are two cases, "preimage" or "Hadamard". In the "preimage" case, the device measures the b and x registers of each $|\phi_i\rangle$ in the computational basis and sends the outcome to the device. This will always be accepted by the device using the definition of CHK.

In the "Hadamard" case, the device measures the x register of each $|\phi_i\rangle$ in the Hadamard basis and sends the outcome $d = (d_1, d_2, \ldots, d_{2N})$ to the verifier. After this measurement, $|\phi_i\rangle$ collapses to $|\alpha_i\rangle\,|d_i\rangle$, where, if $\theta \in [2N]$, then

$$|\alpha_i\rangle = \begin{cases} |\hat{b}_i\rangle & \text{if } i \neq \theta, \\ (|0\rangle + (-1)^{d_\theta \cdot (\hat{x}_{0,\theta} \oplus \hat{x}_{1,\theta})}\,|1\rangle)/\sqrt{2} & \text{if } i = \theta; \end{cases}$$

if $\theta = 0$, then $|\alpha_i\rangle = |\hat{b}_i\rangle$; and if $\theta = \diamond$, then $|\alpha_i\rangle = (|0\rangle + (-1)^{d_i \cdot (\hat{x}_{0,i} \oplus \hat{x}_{1,i})}\,|1\rangle)/\sqrt{2}$.

In the following, we use the shorthand $\hat{h}_i \coloneqq d_i \cdot (\hat{x}_{0,i} \oplus \hat{x}_{1,i}) \in \{0,1\}$ and $\hat{h}' \coloneqq (\hat{h}_{N+1}, \ldots, \hat{h}_{2N}, \hat{h}_1, \hat{h}_2, \ldots, \hat{h}_N) \in \{0,1\}^{2N}$.

For $v \in \{0,1\}^{2N}$, we also define the state

$$|\psi^v\rangle := \frac{1}{\sqrt{2^N}} \bigotimes_{i=1}^{N} (\sigma^X)^{v_i} \otimes (\sigma^X)^{v_{N+i}} (|0\rangle_i |+\rangle_{N+i} + |1\rangle_i |-\rangle_{N+i}), \tag{4}$$

which consists of N (locally-rotated) EPR pairs.

Then, the device applies N controlled-σ^Z gates between the i-th and $(N+i)$-th qubits of $\bigotimes_{i=1}^{2N} |\alpha_i\rangle$ for all $i \in [N]$ (note that the controlled-σ^Z gate is independent of which qubit is the control and which qubit is the target). The device has now prepared the $2N$-qubit state

$$|\alpha\rangle := \begin{cases} |\hat{b}_1, \ldots, \hat{b}_{\theta-1}\rangle \, |(-)^{\hat{b}_{\theta+N} \oplus \hat{h}_\theta}\rangle \, |\hat{b}_{\theta+1}, \ldots, \hat{b}_{2N}\rangle & \text{if } \theta \in [2N],\ \theta \leq N, \\ |\hat{b}_1, \ldots, \hat{b}_{\theta-1}\rangle \, |(-)^{\hat{b}_{\theta-N} \oplus \hat{h}_\theta}\rangle \, |\hat{b}_{\theta+1}, \ldots, \hat{b}_{2N}\rangle & \text{if } \theta \in [2N],\ \theta > N, \\ |\hat{b}_1, \ldots, \hat{b}_{2N}\rangle & \text{if } \theta = 0, \\ |\psi^{\hat{h}'}\rangle & \text{if } \theta = \diamond. \end{cases} \tag{5}$$

In the "Hadamard" case, there is a third and final round where the verifier sends a bitstring $q \in \{0,1\}^{2N}$ to the device. The device performs the following q-dependent measurements. For $i \in [2N]$, if $q_i = 0$, measure the ith qubit of $|\alpha\rangle$ in the computational basis, otherwise, measure the ith qubit of $|\alpha\rangle$ in the Hadamard basis. The device finally sends the outcome $u \in \{0,1\}^{2N}$ of these measurements to the verifier. The right-hand side of Equation (5) implies that the device passes the last checks made by the verifier.

The "moreover" part of the theorem follows directly from the efficient function generation and the efficient decoding properties of ENTCFs (Items 1 and 4). ◀

4 Soundness of self-testing protocol

In this section, we show that our self-testing protocol achieves $\mathrm{poly}(N, \epsilon)$ soundness error. Unlike the proof of completeness in Section 3, we use the adaptive hardcore bit and injective invariance properties of ENTCFs to prove soundness in this section. Therefore, it is necessary for us to make the LWE hardness assumption throughout this section. All proofs can be found in Section 4 of the full version.

We start with a mathematical model of quantum devices.

▶ **Definition 3.** *A device $D = (S, M, \Pi, P)$ is specified by Hilbert spaces named $\mathcal{H}_D$, $\mathcal{H}_Y$, and $\mathcal{H}_R$, with $\dim(\mathcal{H}_Y) = |\mathcal{Y}|^{2N}$ and $\dim(\mathcal{H}_R) = 2^{2Nw}$, and the following.*

1. *A set $S := \{\psi^\theta \mid \theta \in [2N] \cup \{0, \diamond\}\} \subset \mathcal{D}(\mathcal{H}_D \otimes \mathcal{H}_Y)$ of states where each state ψ^θ is classical on $\mathcal{H}_Y$:*

$$\psi^\theta := \sum_{y \in \mathcal{Y}^{2N}} \psi_y^\theta \otimes |y\rangle\langle y|.$$

The state ψ_y^θ models the device's state immediately after returning $y \in \mathcal{Y}^{2N}$ to the verifier if the verifier initially sampled $\theta \in [2N] \cup \{0, \diamond\}$. More precisely, ψ_y^θ (and hence ψ^θ) is a function of the public keys $k \in (\mathcal{K}_\mathcal{F} \cup \mathcal{K}_\mathcal{G})^{2N}$ that the verifier sampled according to θ, as described in the protocol. We choose to make the k-dependence implicit for notational convenience.

2. *A projective measurement Π for the preimage test on $\mathcal{H}_D \otimes \mathcal{H}_Y$:*

$$\Pi := \left\{ \Pi^{b,x} := \sum_{y \in \mathcal{Y}^{2N}} \Pi_y^{b,x} \otimes |y\rangle\langle y| \;\middle|\; b \in \{0,1\}^{2N}, x \in \mathcal{X}^{2N} \right\}.$$

The measurement outcome b, x is the device's answer for the preimage test.

3. *A projective measurement M on $\mathcal{H}_D \otimes \mathcal{H}_Y$ for the device's first answer in the Hadamard test:*

$$M \coloneqq \left\{ M^d \coloneqq \sum_{y \in \mathcal{Y}^{2N}} M_y^d \otimes |y\rangle\langle y| \;\middle|\; d \in \{0,1\}^{2Nw} \right\}. \tag{6}$$

We write $\sigma^\theta(D)$ for the classical-quantum state that results from measuring M on ψ^θ followed by writing measurement outcome d into another classical register whose Hilbert space is denoted by $\mathcal{H}_R$. That is,

$$\sigma^\theta(D) \coloneqq \sum_{y \in \mathcal{Y}^{2N},\, d \in \{0,1\}^{2Nw}} \sigma^\theta_{y,d}(D) \otimes |y,d\rangle\langle y,d| \in \mathcal{H}_D \otimes \mathcal{H}_Y \otimes \mathcal{H}_R, \tag{7}$$

where $\sigma^\theta_{y,d}(D) \coloneqq M_y^d \psi_y^\theta M_y^d$.

4. *Projective measurements P_q on $\mathcal{H}_D \otimes \mathcal{H}_Y \otimes \mathcal{H}_R$ for the device's second answer in the Hadamard test when asked questions $q \in \{0,1\}^{2N}$:*

$$P_q \coloneqq \left\{ P_q^u = \sum_{y \in \mathcal{Y}^{2N}, d \in \{0,1\}^{2Nw}} P^u_{q,y,d} \otimes |y,d\rangle\langle y,d| \;\middle|\; u \in \{0,1\}^{2N} \right\}. \tag{8}$$

The measurement outcome v is the device's answer for the question q.

▶ **Definition 4.** *A device $D = (S, \Pi, M, P)$ is efficient if all the states in S can be efficiently prepared and all the measurements $\Pi, M,$ and P are efficient.*

We use P to define observables of the quantum device that we call X_i and Z_i, which should act as Pauli X and Z operators on the ith qubit respectively.

▶ **Definition 5** (Marginal observables)**.** *Let $D = (S, \Pi, M, P)$ be a device. For $i \in [2N]$ and $q \in \{0,1\}^{2N}$, we define the binary observables*

$$Z_{q,i}(D) \coloneqq \sum_{v \in \{0,1\}^{2N}} (-1)^{v_i} P_q^v \quad \text{if } q_i = 0 \quad \text{and} \quad X_{q,i}(D) \coloneqq \sum_{v \in \{0,1\}^{2N}} (-1)^{v_i} P_q^v \quad \text{if } q_i = 1.$$

Note that $Z_{q,j}(D)$ commutes with $X_{q,k}(D)$ for $j \neq k$ according to these definitions.

In the rest of the paper, we use the abbreviations $Z_i(D) \coloneqq Z_{0^{2N},i}(D)$ and $X_i(D) \coloneqq X_{1^{2N},i}(D)$ for all $i \in [2N]$; $\widetilde{Z}_i(D) \coloneqq Z_{0^N 1^N,i}(D)$ if $i \leq N$; $\widetilde{Z}_i(D) \coloneqq Z_{0^N 1^N,i}(D)$ if $i > N$; $\widetilde{X}_i(D) \coloneqq X_{1^N 0^N,i}(D)$ if $i \leq N$; and $\widetilde{X}_i(D) \coloneqq X_{0^N 1^N,i}(D)$ if $i > N$.

For different choices of θ, our goal is to characterize the actions of the observables X_i and Z_i on the state σ^θ, which is the post-M-measurement state defined below.

▶ **Definition 6** ($\sigma^{\theta,v}$)**.** *Let D be a device. For $\theta \in [2N] \cup \{0, \diamond\}$ and $v \in \{0,1\}^{2N}$, we define the state*

$$\sigma^{\theta,v}(D) \coloneqq \sum_{(y,d) \in \Sigma(\theta,v)} \sigma^\theta_{y,d}(D) \otimes |y,d\rangle\, \langle y,d| \in \mathcal{H}_D \otimes \mathcal{H}_Y \otimes \mathcal{H}_R, \tag{9}$$

where, $\Sigma(\theta, v)$ is set to

$$\begin{cases} \left\{(y,d) \;\middle|\; \hat{b}(k_i, y_i) = v_i \text{ for all } i \neq \theta \text{ and } \hat{h}(k_\theta, y_\theta, d_\theta) = v_\theta \oplus v_{\mathrm{mod}(\theta+N,2N)}\right\} & \text{if } \theta \in [2N], \\ \left\{(y,d) \;\middle|\; \hat{b}(k_i, y_i) = v_i \text{ for all } i\right\} & \text{if } \theta = 0, \\ \left\{(y,d) \;\middle|\; \hat{h}(k_i, y_i, d_i) = v_{\mathrm{mod}(i+N,2N)} \text{ for all } i\right\} & \text{if } \theta = \diamond. \end{cases}$$

In all cases, (y,d) ranges over $\mathcal{Y}^{2N} \times \{0,1\}^{2Nw}$, i ranges over $[2N]$, and the state $\sigma^{\theta,v}(D)$ implicitly depends on keys $k \in (\mathcal{K}_\mathcal{F} \cup \mathcal{K}_\mathcal{G})^{2N}$ chosen according to θ as described in the protocol.

Unlike the nonlocal self-testing case, where there is only one state, e.g. EPR pairs, to characterize, we have multiple states and multiple observables to characterize. Hence, we first decompose $\sigma^\theta \approx \sum_{v\in\{0,1\}^{2N}} \sigma^{\theta,v}$, where $\sigma^{\theta,v}$ are defined above. We then characterize the behavior of different observables on different $\sigma^{\theta,v}$ using the failure probabilities of different test cases:

▶ **Definition 7** (Failure probabilities). *Let D be a device. For $q \in \{0,1\}^{2N}$, we define $\epsilon_P(D)$ to be the probability that D fails the preimage test, $\epsilon_{H,q}(D)$ to be the probability that D fails question q of the Hadamard test, and $\epsilon_H(D)$ to be the maximum of $\epsilon_{H,q}(D)$ over $q \in \{0^{2N}, 1^{2N}, 0^N1^N, 1^N0^N\}$. Then, the average failure probability is*

$$\epsilon(D) \coloneqq \epsilon_P(D)/2 + \Big(\sum_{q\in\{0^{2N},1^{2N},0^N1^N,1^N0^N\}} \frac{1}{4}\epsilon_{H,q}(D) + \sum_{q\in\{0,1\}^{2N}} \mu(q)\epsilon_{H,q}(D)\Big)/4.$$

Henceforth, when D is clear from the context, we mostly omit the D dependence.

The probability that this device can pass the tests of our protocol allows us to say that the operator acts in the same way as the ideal operator acts on the ideal state. Therefore, we will use ϵ_P and $\epsilon_{H,q}$ to bound how far away the $Z_{q,i}, X_{q,i}$ observables and $\sigma^{\theta,v}$ states are from the ideal observables and states. How we characterize the states and observables using the passing probabilities of the four key questions: $q = 0^2N$, 1^{2N}, 0^N1^N and 1^N0^N is summarized in Table 1. Note that $q \in \{0^N1^N, 1^N0^N\}$ are for testing EPR pairs.

Table 1 Correspondence between the (θ, q) used in our protocol and the observables tested.

(q_i, q_{i+N}) with $i \le N$	$\theta = 0$	$\theta \in [2N]$	$\theta = \diamond$
$(0,0)$	$Z_{q,i}$ and $Z_{q,i+N}$	-	-
$(1,1)$	-	$X_{q,\theta}$ if $\theta \in \{i, i+N\}$	-
$(0,1)$	-	-	$Z_{q,i} \cdot X_{q,i+N}$
$(1,0)$	-	-	$X_{q,i} \cdot Z_{q,i+N}$

Our use of only $2N+2$ distinct θs allows us to bound the failure probability associated with each σ^θ by $O(N\epsilon)$. If we had naively used $\theta \in \{0,1\}^{2N}$, the robustness of our self-test would be $2^{\Omega(N)}\epsilon$. The fact that using only $2N+2$ distinct θs is *sufficient* for self-testing crucially relies on the following proposition, which can be proven using the injective invariance property of ENTCFs (Item 6).

▶ **Proposition 8.** *Any pair of states in $\{\sigma^\theta \mid \theta \in [2N] \cup \{0, \diamond\}\}$ of an efficient device D are computationally indistinguishable.*

In the next step, we use the computational indistinguishability of the σ^θs to argue that for all (q,i), the observables $Z_{q,i}$ and $X_{q,i}$ act like Z_i and X_i on any σ^θ.

▶ **Proposition 9.** *For all $q \in \{0,1\}^{2N}$, $\theta \in [2N] \cup \{0,\diamond\}$, and $i \in [2N]$, we have*

$$Z_{q,i} \approx_{N(\epsilon_{H,q}+\epsilon_H+\epsilon_P),\sigma^\theta} Z_i \text{ if } q_i = 0, \quad \text{and} \quad X_{q,i} \approx_{N(\epsilon_{H,q}+\epsilon_H+\epsilon_P),\sigma^\theta} X_i \text{ if } q_i = 1.$$

For self-testing, we not only need to characterize the action of a single operator on σ^θ as sketched above, we also need to characterize the actions of products of the operators. Next, we establish the commutation and anti-commutation relations of the observables with respect to σ^θ. Proving commutation is straightforward, while proving anti-commutation relies on the adaptive hardcore bit property. Our proof generalizes and refines techniques in [39, 24]: one difference is that we associate error parameters to each $\sigma^{\theta,v}$, where $v \in \{0,1\}^{2N}$, and use them collectively to bound the overall approximation error associated with σ^θ.

▶ **Proposition 10.** *Let D be an efficient perfect device. For all $i, j, \theta \in [2N]$, we have*

Commutation. $[Z_i, Z_j] = 0,\ [X_i, X_j] = 0,$ *and* $[Z_i, X_j] \approx_{N\epsilon_H + \mathrm{negl}(\lambda), \sigma^\theta} 0$ *if* $i \neq j$.
Anti-commutation. $\{Z_i, X_i\} \approx_{\sqrt{N\epsilon_H} + \mathrm{negl}(\lambda), \sigma^\theta} 0$.

The above relations allow us to handle products of two operators from $\{Z_i, X_i\}_{i \in [2N]}$. However, as mentioned in Section 1, we also want to show relations such as $Z_1 X_3 Z_2 \sigma^3 \approx X_3 Z_1 Z_2 \sigma^3$ $(\star)$, which does *not* directly follow because Z_1 and X_3 are not directly next to the state σ^3. We want to establish relations like $(\star)$ involving products of multiple observables Z_i and X_i in order to characterize Z_i and X_i as Pauli operators σ_i^Z and σ_i^X under the swap isometry defined later.

Our solution to this problem is the next proposition which shows observable-state commutation relations for certain pairs of observables and states. For example, we can now easily prove $(\star)$ by first using the proposition to commute Z_2 past σ^3. We view our use of observable-state commutation relations, which has no analog in prior work, as one of the main technical contributions of this work. These techniques should be useful in any future work that aims to efficiently self-test more than one qubit.

▶ **Proposition 11** (Operator-state commutation)**.** *Let D be an efficient perfect device. For all $i, \theta \in [2N]$ with $i \neq \theta$ and $q \in \{0,1\}^{2N}$, we have*

$$Z_{q,i}\, \sigma^\theta \approx_{N(\epsilon_H + \epsilon_{H,q}) + \mathrm{negl}(\lambda)} \sigma^\theta\, Z_{q,i} \text{ if } q_i = 0 \text{ and } X_{q,\theta}\, \sigma^\theta \approx_{N(\epsilon_H + \epsilon_{H,q}) + \mathrm{negl}(\lambda)} \sigma^\theta\, X_{q,\theta} \text{ if } q_\theta = 1.$$

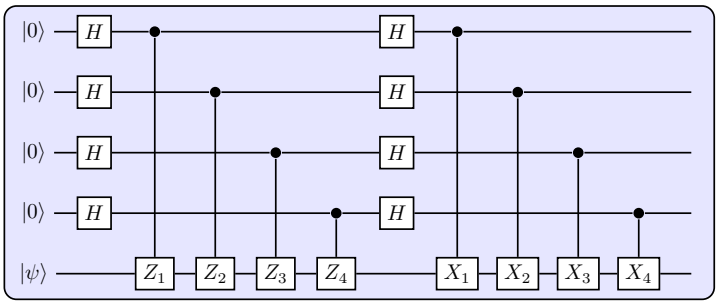
Observe that the proposition above does *not* say $Z_{q,i}$ and $X_{q,i}$ commute with σ^θ for all pairs (i, θ) as we would have desired to prove all $(\star)$-like relations. To get around this problem, we make use of the computational indistinguishability of the σ^θs to argue that efficient observables must act similarly on different σ^θs. For example, consider the following relation that looks similar to $(\star)$: $Z_1 X_3 Z_2 \sigma^2 \approx X_3 Z_1 Z_2 \sigma^2$ $(\star')$. In this case, we cannot directly apply Proposition 11, since Z_2 does not commute with σ^2. Nevertheless, by using the computational indistinguishability of σ^2 and σ^3, we can derive an "operational version" of $(\star')$ from $(\star)$. The operational version allows us to interchange the left-hand and right-hand sides of $(\star')$ when they appear inside traces (i.e., Tr). We can only derive such an operational version because the computational indistinguishability of σ^2 and σ^3 only allows us to interchange σ^2 and σ^3 inside traces; see the lifting lemmas in the full version. For an example of our using this technique, see the long aligned equation in the proof of Lemma 4.33 in the full version.

Next, we define our swap isometry $\mathcal{V}$. We will show that $\mathcal{V}$ maps the states, observables, and measurements of the device to their ideal counterparts. This swap isometry can be viewed as a special case of the swap isometry proposed in [54, Figure 2] in the nonlocal setting. It is not the obvious generalization of the swap isometry used in [39, Proof of Lemma 4.28] as that is more difficult to analyze.

▶ **Definition 12.** *Let D be a device and let $\mathcal{H} := \mathcal{H}_D \otimes \mathcal{H}_Y \otimes \mathcal{H}_R$. The swap isometry is the map $\mathcal{V} : \mathcal{H} \to \mathbb{C}^{2^{2N}} \otimes \mathcal{H}$ defined by*

$$\mathcal{V} = \sum_{u \in \{0,1\}^{2N}} |u\rangle \otimes \prod_{i \in [2N]} X_i^{u_i} \prod_{j \in [2N]} Z_j^{(u_j)}.$$

We illustrate $\mathcal{V}$ when $2N = 4$ below.



We proceed to analyze the effect of the swap isometry on the observables and states of the device. More specifically, in Proposition 13, we show that $\mathcal{V}$ maps the X_i and Z_i observables approximately to σ^X and σ^Z operators acting on the ith qubit of an auxiliary system.

▶ **Proposition 13.** *Let D be an efficient perfect device. For all $k \in [2N]$, $\theta \in [2N] \cup \{0, \diamond\}$, and $q \in \{0,1\}^{2N}$, we have*

$$\mathcal{V}^\dagger(\sigma_k^Z \otimes \mathbb{1})\mathcal{V} \approx_{N(\epsilon_H+\epsilon_{H,q})+\mathrm{negl}(\lambda),\sigma^\theta} Z_{q,k} \text{ if } q_k = 0, \text{ and}$$
$$\mathcal{V}^\dagger(\sigma_k^X \otimes \mathbb{1})\mathcal{V} \approx_{N^{3/2}\sqrt{\epsilon_H}+N\epsilon_{H,q},\sigma^\theta} X_{q,k} \text{ if } q_k = 1.$$

Moreover, for $k \in [N]$ and $\theta \in [2N] \cup \{0, \diamond\}$,

$$\mathcal{V}^\dagger(\sigma_k^X \otimes \sigma_{N+k}^Z \otimes \mathbb{1})\mathcal{V} \approx_{N^{3/8}\epsilon_H^{1/8},\sigma^\theta} \widetilde{X}_k\widetilde{Z}_{N+k} \quad \text{and} \quad \mathcal{V}^\dagger(\sigma_k^Z \otimes \sigma_{N+k}^X \otimes \mathbb{1})\mathcal{V} \approx_{N^{3/8}\epsilon_H^{1/8},\sigma^\theta} \widetilde{Z}_k\widetilde{X}_{N+k}.$$

In Proposition 15, we show that $\mathcal{V}$ maps the states of the device to states of the form $\tau^{\theta,v} \otimes \alpha^{\theta,v}$, where $\tau^{\theta,v}$ is the ideal state defined below and $\alpha^{\theta,v}$ is some junk state that is computationally indistinguishable to a fixed state α for all θ and v.

▶ **Definition 14** (density operators $\tau^{\theta,v}$)**.** *Let $v \in \{0,1\}^{2N}$. For $\theta \in [2N] \cup \{0, \diamond\}$, we define the $2N$-qubit density operator $\tau^{\theta,v} := |\tau^{\theta,v}\rangle\langle\tau^{\theta,v}|$, according to the following three cases.*

$$|\tau^{\theta,v}\rangle := \begin{cases} |v_1\rangle \otimes \cdots \otimes |v_{\theta-1}\rangle \otimes |(-)^{v_\theta}\rangle \otimes |v_{\theta+1}\rangle \otimes \cdots \otimes |v_{2N}\rangle & \text{if } \theta \in [2N], \\ |v\rangle := |v_1\rangle \otimes \cdots \otimes |v_{2N}\rangle & \text{if } \theta = 0, \\ |\psi^v\rangle & \text{if } \theta = \diamond, \end{cases} \tag{10}$$

where $|\psi^v\rangle$ is as defined in Equation (4).

▶ **Proposition 15.** *Let D be an efficient perfect device. For all $\theta \in [2N] \cup \{0, \diamond\}$ and $v \in \{0,1\}^{2N}$, there exists a state $\alpha^{\theta,v} \in \mathrm{Pos}(\mathcal{H})$ such that*

$$\sum_{v\in\{0,1\}^{2N}} \|\mathcal{V}\sigma^{\theta,v}\mathcal{V}^\dagger - \tau^{\theta,v} \otimes \alpha^{\theta,v}\|_1 \le O(N^{7/4}\epsilon_H^{1/4}), \text{ for } \theta \neq \diamond, \text{ and}$$
$$\sum_{v\in\{0,1\}^{2N}} \|\mathcal{V}\sigma^{\diamond,v}\mathcal{V}^\dagger - \tau^{\diamond,v} \otimes \alpha^{\diamond,v}\|_1 \le O(N^{35/32}\epsilon_H^{1/32}).$$

Moreover, there exists a state $\alpha \in \mathrm{Pos}(\mathcal{H})$ and numbers $\{\delta(v) \ge 0 \mid v \in \{0,1\}^{2N}\}$ such that any efficient device can distinguish between $\alpha^{\theta,v}$ and $\alpha/2^{2N}$ with advantage at most $O(\delta(v))$ for all $v \in \{0,1\}^{2N}$, with $\sum_{v\in\{0,1\}^{2N}} \delta(v) \le O(N^{49/32}\epsilon_H^{1/32})$.

The proofs of the two propositions above rely heavily on Propositions 10 and 11 which crucially allows us to bound the soundness error in Theorem 16 by $O(\mathrm{poly}(N, \epsilon))$. If we directly generalize the soundness analysis of [39], we would obtain an $O(2^N\epsilon^{1/2^N})$ bound on the soundness error which is extremely loose.[3]

[3] [23, End of Section 1.3] explains why the technique in [39] would lead to such a loose bound.

Lastly, we put everything together to give our main soundness result, Theorem 16. The main task is to characterize the measurement operator P^u_q, which is approximately a product of $2N$ binary projectors of the form $Z^{(u_i)}_{q,i}$ and $X^{(u_i)}_{q,i}$. We use the operator-state commutation relation to sequentially replace each projector in the product by its ideal counterpart.

▶ **Theorem 16.** *Let D be an efficient device. Let $\mathcal{H} \coloneqq \mathcal{H}_D \otimes \mathcal{H}_Y \otimes \mathcal{H}_R$ be the Hilbert space of D. Let $\mathcal{V} : \mathcal{H} \to \mathbb{C}^{2^{2N}} \otimes \mathcal{H}$ be the swap isometry defined in Definition 12. For $\theta \in [2N] \cup \{0, \diamond\}$ and $v \in \{0,1\}^{2N}$, let $\sigma^{\theta,v} \in \mathrm{Pos}(\mathcal{H})$ be the states that D prepares after returning the first answer in the Hadamard round, as defined in Definition 6. Let $\{\{P^u_q\}_{u\in\{0,1\}^{2N}} \mid q \in \{0,1\}^{2N}\}$ be the measurements defined in Equation (8) of Definition 3.*

Suppose that D fails the protocol in Fig. 2 (with an input distribution μ on $\{0,1\}^{2N}$ and $N = \mathrm{poly}(\lambda)$) with probability at most ϵ. Then, there exist states $\{\alpha^{\theta,v} \mid \theta \in [2N] \cup \{0,\diamond\}, v \in \{0,1\}^{2N}\}$, that are computationally indistinguishable from a single state $\alpha \in \mathrm{Pos}(\mathcal{H})$ in the way specified in Proposition 15, such that

$$\sum_{v\in\{0,1\}^{2N}} \|\mathcal{V}\sigma^{\theta,v}\mathcal{V}^\dagger - \tau^{\theta,v} \otimes \alpha^{\theta,v}\|_1 \leq O(N^{7/4}\epsilon^{1/32}),$$

$$\mathop{\mathbb{E}}_{q\leftarrow\mu}\left[\sum_{u,v\in\{0,1\}^{2N}} \|\mathcal{V}P^u_q\sigma^{\theta,v}P^u_q\mathcal{V}^\dagger - \langle B^u_q|\,\tau^{\theta,v}\,|B^u_q\rangle\,|B^u_q\rangle\langle B^u_q| \otimes \alpha^{\theta,v}\|_1\right] \leq O(N^2\epsilon^{1/32}),$$

and, for all $q \in \{0^{2N}, 1^{2N}, 0^N1^N, 1^N0^N\}$,

$$\sum_{u,v\in\{0,1\}^{2N}} \|\mathcal{V}P^u_q\sigma^{\theta,v}P^u_q\mathcal{V}^\dagger - \langle B^u_q|\,\tau^{\theta,v}\,|B^u_q\rangle\,|B^u_q\rangle\langle B^u_q| \otimes \alpha^{\theta,v}\|_1 \leq O(N^2\epsilon^{1/32}).$$

5 Applications

In this section, we briefly describe two applications of our self-test: DIQKD and dimension-testing. For details, see Section 5 of the full version.

DIQKD. We describe how to adapt the protocol for DIQKD under computational assumptions in [38] to use our self-testing protocol as its main component. The resulting DIQKD protocol operates under the same setting and assumptions as in [38] except we remove the IID assumption. In particular, we highlight the fact that we retain the advantage of the generated key being information-theoretically secure.

Recall that in our self-testing protocol, there is a single verifier interacting with a single device. On the other hand, in DIQKD, there are two verifiers, Alice and Bob, that each interact with their own (untrusted) device. In DIQKD *under computational assumptions*, the two devices are not assumed to be non-communicating and are modeled as a single device with two *components*, one on Alice's side, and one on Bob's. At a high level, to resolve the difference in the number of verifiers, we will let Alice play the role of the single verifier in our self-testing protocol while Bob will play a relaying role.

In Fig. 3, we describe a single test round of our DIQKD protocol. In Fig. 4, we describe how to modify the test round to give a single generation round of our DIQKD protocol. We construct our overall DIQKD protocol by using multiple test rounds followed by a single generation round. After the generation round, Alice and Bob proceed to key extraction, which is essentially the same as that in [38, Protocol 3].

1. Alice samples $\theta \leftarrow_U [2N] \cup \{0, \diamond\}$ uniformly at randomly, generates $2N$ key-trapdoor pairs $(k_1, t_1), \ldots, (k_{2N}, t_{2N})$ according to θ, and sends $k_{N+1}, \ldots, k_{2N}$ to Bob. Note that Alice has all the trapdoors $\{t_i\}_{i=1}^{2N}$. Then Alice sends $k_1, \ldots, k_N$ to her component. Bob sends $k_{N+1}, \ldots, k_{2N}$ to his component.
2. Alice receives back $(y_1, \ldots, y_N) \in \mathcal{Y}^N$ and Bob receives back images $(y_{N+1}, \ldots, y_{2N}) \in \mathcal{Y}^N$.
3. Alice samples $c \leftarrow_U \{\text{preimage}, \text{Hadamard}\}$ uniformly at random, sends it to Bob, and they both send c to their components.

   Case c = preimage. Alice receives $(b_1, \ldots b_N, x_1, \ldots, x_N) \in \{0,1\}^{N+Nw}$ from her component and Bob receives $(b_{N+1}, \ldots, b_{2N}, x_{N+1}, \ldots, x_{2N}) \in \{0,1\}^{N+Nw}$ from his component and sends it to Alice. Alice verifies $(b_1, \ldots, b_{2N}, x_1, \ldots, x_{2N})$ according to our self-testing protocol.

   Case c = Hadamard.

   a. Alice receives $(d_1, \ldots, d_N) \in \{0,1\}^{Nw}$ from her component and Bob receives $(d_{N+1}, \ldots, d_{2N}) \in \{0,1\}^{Nw}$ from his component.
   b. Alice samples $a \leftarrow_U \{0,1\}$ uniformly at random.
      - If $a = 0$, Alice samples $q \leftarrow_U \{0^{2N}, 1^{2N}, 0^N1^N, 1^N0^N\}$ uniformly at random.
      - If $a = 1$, Alice sets $q = 1^N0^N$.

      Note that the resulting distribution on $(q_1, \ldots, q_{2N}) \in \{0,1\}^{2N}$ is the same as in Step 4 of our self-testing protocol (Fig. 2) with μ chosen as the distribution that always outputs 1^N0^N.
      Alice sends q_{N+1} to Bob. Alice sends $q_1, \ldots, q_N$ to her component. Bob sends $q_{N+1}, \ldots, q_{N+1}$ $(= q_{N+1}, \ldots, q_{2N})$ to his component.
   c. Alice receives $(u_1, \ldots, u_N) \in \{0,1\}^N$ from her component and Bob receives $(u_{N+1}, \ldots, u_{2N}) \in \{0,1\}^N$ from his component. Alice sends "Test" to Bob. Bob sends $\{(y_i, d_i, u_i)\}_{i=N+1}^{2N}$ to Alice. Alice verifies $\{(y_i, d_i, u_i)\}_{i=1}^{2N}$ according to our self-testing protocol using the trapdoors that she holds, $(t_1, \ldots, t_{2N})$.

Figure 3 Test round for device-independent quantum key distribution (DIQKD) protocol.

Same as the test round (see Fig. 3) except with the following modifications.

- At Step 1, Alice chooses $\theta = \diamond$.
- At the start of Step 3, Alice chooses c = Hadamard.
- At the start of Step 3(b), instead of sampling q, Alice sets $q = 1^N0^N$.
- Replace Step 3 (c) by the following. Alice receives $(u_1, \ldots, u_N) \in \{0,1\}^N$ from her component and Bob receives $(u_{N+1}, \ldots, u_{2N}) \in \{0,1\}^N$ from his component. Alice sends "Generation" to Bob.

Figure 4 Generation round for device-independent quantum key distribution (DIQKD) protocol.

The completeness of this DIQKD protocol essentially follows from the completeness of our self-testing protocol. The soundness follows from the soundness of our self-testing protocol combined with the key rate analysis used to prove [38, Theorem 1] and the "cut-and-choose" argument used to prove [23, Theorem 4.33].

Dimension-testing. We simplify our self-testing protocol to give a protocol that tests if a quantum device can store N qubits. The simplifications are: 1. θ is sampled from $\{0, 1, \ldots, N\}$, 2. in the Hadamard case, there are only two questions $q = 0^N$ and $q = 1^N$. Details of this protocol can be found in Section 5.2 of the full version. The honest prover's behavior is similar to that of our self-test.

The intuition behind the soundness of this protocol is that, when it is passed with high probability, Theorem 16 guarantees the existence of a quantum state $\rho^\star$ on the quantum part of the device's memory that is close to the maximally mixed state up to some isometry. More specifically, $\rho^\star$ comes from using Theorem 16 to force the device to perform a Hadamard basis measurement on N qubits that are in the computational basis and discarding the measurement results. Then, the main proposition of this section, Proposition 17, shows that the guarantee on $\rho^\star$ is strong enough for us to lower bound the rank of $\rho^\star$, which is also a lower bound on the quantum dimension of the device's memory.

▶ **Proposition 17.** *Let $\rho, \alpha \in D(\mathcal{H})$ be density operators. If there exists a unitary $U \in \mathcal{L}(\mathbb{C}^{2^n} \otimes \mathcal{H})$ such that $\|U(|0\rangle\langle 0|^{\otimes n} \otimes \rho)U^\dagger - 2^{-n}\mathbb{1} \otimes \alpha\|_1 \leq \epsilon$, then $\mathrm{Rank}(\rho) \geq (1-\epsilon)2^n$.*

We now use Proposition 17 to prove the main theorem of this section. Much of the proof is devoted to bookkeeping to ensure that the (normalized) density operator condition in Proposition 17 is satisfied and that we are bounding the *quantum* dimension.

▶ **Theorem 18.** *Let D be an efficient device with Hilbert space $\mathcal{H} = \mathcal{H}_D \otimes \mathcal{H}_Y \otimes \mathcal{H}_R$. Let the classical-quantum decomposition of $\mathcal{H}$ be $\mathcal{H}_C \otimes \mathcal{H}_Q$, so that all states and observables of D on $\mathcal{H}$ are classical on $\mathcal{H}_C$, i.e., block-diagonal in a fixed basis $\{|c\rangle \mid c \in [\dim(\mathcal{H}_C)]\}$ of $\mathcal{H}_C$. If D can pass the dimension test protocol with probability $\geq 1-\epsilon$, then the quantum dimension of D, $\dim(H_Q)$, is at least $(1 - O(N^2\epsilon^{1/32}))2^N$.*

References

1. Gorjan Alagic, Andrew M. Childs, Alex B. Grilo, and Shih-Han Hung. Non-interactive Classical Verification of Quantum Computation. In *Theory of Cryptography*, pages 153–180. Springer International Publishing, 2020. doi:10.1007/978-3-030-64381-2_6.
2. Atul Singh Arora, Andrea Coladangelo, Matthew Coudron, Alexandru Gheorghiu, Uttam Singh, and Hendrik Waldner. Quantum depth in the Random Oracle Model, 2022. arXiv:2210.06454
3. James Bartusek, Yael Tauman Kalai, Alex Lombardi, Fermi Ma, Giulio Malavolta, Vinod Vaikuntanathan, Thomas Vidick, and Lisa Yang. Succinct Classical Verification of Quantum Computation, 2022. arXiv:2206.14929
4. J. S. Bell. On the Einstein Podolsky Rosen paradox. *Physics Physique Fizika*, 1:195–200, 1964. doi:10.1103/PhysicsPhysiqueFizika.1.195.
5. Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh Vazirani, and Thomas Vidick. A Cryptographic Test of Quantumness and Certifiable Randomness from a Single Quantum Device. *Journal of the ACM*, 68(5), 2021. doi:10.1145/3441309.
6. Zvika Brakerski, Venkata Koppula, Umesh V. Vazirani, and Thomas Vidick. Simpler Proofs of Quantumness. In *Proceedings of the 15th Conference on the Theory of Quantum Computation, Communication, and Cryptography (TQC)*, 2020. doi:10.4230/LIPIcs.TQC.2020.8.
7. Samuel L. Braunstein, A. Mann, and M. Revzen. Maximal violation of Bell inequalities for mixed states. *Physical Review Letters*, 68:3259–3261, 1992. doi:10.1103/PhysRevLett.68.3259.
8. Spencer Breiner, Amir Kalev, and Carl A. Miller. Parallel Self-Testing of the GHZ State with a Proof by Diagrams. *Electronic Proceedings in Theoretical Computer Science*, 287:43–66, 2019. doi:10.4204/eptcs.287.3.
9. Nicolas Brunner, Stefano Pironio, Antonio Acin, Nicolas Gisin, André Allan Méthot, and Valerio Scarani. Testing the Dimension of Hilbert spaces. *Physical Review Letters*, 100(21):210503, 2008. doi:10.1103/PhysRevLett.100.210503.
10. Yu Cai, Jean-Daniel Bancal, Jacquiline Romero, and Valerio Scarani. A new device-independent dimension witness and its experimental implementation. *Journal of Physics A: Mathematical and Theoretical*, 49(30):305301, 2016. doi:10.1088/1751-8113/49/30/305301.
11. Rui Chao and Ben W. Reichardt. Quantum dimension test using the uncertainty principle, 2020. arXiv:2002.12432
12. Rui Chao, Ben W. Reichardt, Chris Sutherland, and Thomas Vidick. Overlapping Qubits. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 48:1–48:21, 2017. doi:10.4230/LIPIcs.ITCS.2017.48.
13. Rui Chao, Ben W. Reichardt, Chris Sutherland, and Thomas Vidick. Test for a large amount of entanglement, using few measurements. *Quantum*, 2:92, 2018. doi:10.22331/q-2018-09-03-92.

14 Nai-Hui Chia, Kai-Min Chung, and Takashi Yamakawa. Classical Verification of Quantum Computations with Efficient Verifier. In *Theory of Cryptography*, pages 181–206. Springer International Publishing, 2020. doi:10.1007/978-3-030-64381-2_7.

15 Nai-Hui Chia and Shih-Han Hung. Classical verification of quantum depth, 2022. arXiv:2205.04656

16 John F Clauser, Michael A Horne, Abner Shimony, and Richard A Holt. Proposed experiment to test local hidden-variable theories. *Physical Review Letters*, 23(15):880, 1969. doi:10.1103/PhysRevLett.23.880.

17 Andrea Coladangelo. A two-player dimension witness based on embezzlement, and an elementary proof of the non-closure of the set of quantum correlations. *Quantum*, 4:282, 2020. doi:10.22331/q-2020-06-18-282.

18 Andrea Coladangelo, Koon Tong Goh, and Valerio Scarani. All pure bipartite entangled states can be self-tested. *Nature Communications*, 8(1):15485, 2017. doi:10.1038/ncomms15485.

19 Andrea Coladangelo, Alex B. Grilo, Stacey Jeffery, and Thomas Vidick. Verifier-on-a-Leash: New Schemes for Verifiable Delegated Quantum Computation, with Quasilinear Resources. In *Advances in Cryptology – EUROCRYPT 2019*, pages 247–277, 2019. doi:10.1007/978-3-030-17659-4_9.

20 Honghao Fu. Constant-sized correlations are sufficient to self-test maximally entangled states with unbounded dimension. *Quantum*, 6:614, 2022. doi:10.22331/q-2022-01-03-614.

21 Honghao Fu, Daochen Wang, and Qi Zhao. Parallel self-testing of EPR pairs under computational assumptions. *arXiv preprint arXiv:2201.13430*, 2022.

22 Rodrigo Gallego, Nicolas Brunner, Christopher Hadley, and Antonio Acín. Device-independent tests of classical and quantum dimensions. *Physical Review Letters*, 105(23):230501, 2010. doi:10.1103/PhysRevLett.105.230501.

23 Alexandru Gheorghiu, Tony Metger, and Alexander Poremba. Quantum cryptography with classical communication: parallel remote state preparation for copy-protection, verification, and more, 2022. arXiv:2201.13445

24 Alexandru Gheorghiu and Thomas Vidick. Computationally-Secure and Composable Remote State Preparation. In *Proceedings of the 60th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1024–1033, 2019. doi:10.1109/FOCS.2019.00066.

25 Koon Tong Goh, Jedrzej Kaniewski, Elie Wolfe, Tamás Vértesi, Xingyao Wu, Yu Cai, Yeong-Cherng Liang, and Valerio Scarani. Geometry of the set of quantum correlations. *Physical Review A*, 97:022104, 2018. doi:10.1103/PhysRevA.97.022104.

26 W T Gowers and O Hatami. Inverse and stability theorems for approximate representations of finite groups. *Sbornik: Mathematics*, 208(12):1784–1817, 2017. doi:10.1070/sm8872.

27 Shuichi Hirahara and François Le Gall. Test of Quantumness with Small-Depth Quantum Circuits. In *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 59:1–59:15, 2021. doi:10.4230/LIPIcs.MFCS.2021.59.

28 Zhengfeng Ji, Anand Natarajan, Thomas Vidick, John Wright, and Henry Yuen. MIP*=RE, 2020. arXiv:2001.04383.

29 Gregory D. Kahanamoku-Meyer, Soonwon Choi, Umesh V. Vazirani, and Norman Y. Yao. Classically verifiable quantum advantage from a computational Bell test. *Nature Physics*, 18(8):918–924, 2022. doi:10.1038/s41567-022-01643-7.

30 Yael Kalai, Alex Lombardi, Vinod Vaikuntanathan, and Lisa Yang. Quantum Advantage from Any Non-Local Game, 2022. arXiv:2203.15877

31 Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to Delegate Computations: The Power of No-Signaling Proofs. In *Proceedings of the 46th ACM Symposium on the Theory of Computing (STOC)*, pages 485–494, 2014. doi:10.1145/2591796.2591809.

32 Zhenning Liu and Alexandru Gheorghiu. Depth-efficient proofs of quantumness. *Quantum*, 6:807, 2022. doi:10.22331/q-2022-09-19-807.

33 U. Mahadev. Classical Verification of Quantum Computations. In *Proceedings of the 59th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 259–267, 2018. doi:10.1109/FOCS.2018.00033.

34 Urmila Mahadev, Umesh Vazirani, and Thomas Vidick. Efficient Certifiable Randomness from a Single Quantum Device, 2022. arXiv:2204.11353

35 Dominic Mayers and Andrew Yao. Self Testing Quantum Apparatus. *Quantum Info. Comput.*, 4(4):273–286, 2004. doi:10.26421/QIC4.4-3.

36 M McKague, T H Yang, and V Scarani. Robust self-testing of the singlet. *Journal of Physics A: Mathematical and Theoretical*, 45(45):455304, 2012. doi:10.1088/1751-8113/45/45/455304.

37 Matthew McKague. Self-testing in parallel with CHSH. *Quantum*, 1:1, 2017. doi:10.22331/q-2017-04-25-1.

38 Tony Metger, Yfke Dulek, Andrea Wei Coladangelo, and Rotem Arnon-Friedman. Device-independent quantum key distribution from computational assumptions. *New Journal of Physics*, 2021. doi:10.1088/1367-2630/ac304b.

39 Tony Metger and Thomas Vidick. Self-testing of a single quantum device under computational assumptions. *Quantum*, 5:544, 2021. doi:10.22331/q-2021-09-16-544.

40 Akihiro Mizutani, Yuki Takeuchi, Ryo Hiromasa, Yusuke Aikawa, and Seiichiro Tani. Computational self-testing for entangled magic states. *Physical Review A*, 106:L010601, 2022. doi:10.1103/PhysRevA.106.L010601.

41 Anand Natarajan and Thomas Vidick. A Quantum Linearity Test for Robustly Verifying Entanglement. In *Proceedings of the 49th ACM Symposium on the Theory of Computing (STOC)*, pages 1003–1015, 2017. doi:10.1145/3055399.3055468.

42 Anand Natarajan and Thomas Vidick. Low-Degree Testing for Quantum States, and a Quantum Entangled Games PCP for QMA. In *Proceedings of the 59th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 731–742, 2018. doi:10.1109/FOCS.2018.00075.

43 Sandu Popescu and Daniel Rohrlich. Which states violate Bell's inequality maximally? *Physics Letters A*, 169(6):411–414, 1992. doi:10.1016/0375-9601(92)90819-8.

44 Oded Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *Journal of the ACM*, 56(6), 2009. doi:10.1145/1568318.1568324.

45 Ben W. Reichardt, Falk Unger, and Umesh Vazirani. Classical command of quantum systems. *Nature*, 496(7446):456–460, 2013. doi:10.1038/nature12035.

46 Pavel Sekatski, Jean-Daniel Bancal, Sebastian Wagner, and Nicolas Sangouard. Certifying the Building Blocks of Quantum Computers from Bell's Theorem. *Physical Review Letters*, 121:180505, 2018. doi:10.1103/PhysRevLett.121.180505.

47 Stephen J. Summers and Reinhard Werner. Maximal violation of Bell's inequalities is generic in quantum field theory. *Communications in Mathematical Physics*, 110(2):247–259, 1987. doi:10.1007/BF01207366.

48 Ivan Šupić and Joseph Bowles. Self-testing of quantum systems: a review. *Quantum*, 4:337, 2020. doi:10.22331/q-2020-09-30-337.

49 Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. Delegation for bounded space. In *Proceedings of the 45th ACM Symposium on the Theory of Computing (STOC)*, pages 565–574, 2013. doi:10.1145/2488608.2488679.

50 B. S. Tsirel'son. Quantum analogues of the Bell inequalities. The case of two spatially separated domains. *Journal of Soviet Mathematics*, 36(4):557–570, 1987. doi:10.1007/BF01663472.

51 Thomas Vidick. Course FSMP, Fall 2020: Interactions with Quantum Devices, 2020. Lecture notes available at: http://users.cms.caltech.edu/~vidick/teaching/fsmp/fsmp.pdf. Date accessed: 29th March 2023.

52 Thomas Vidick and Tina Zhang. Classical zero-knowledge arguments for quantum computations. *Quantum*, 4:266, 2020. doi:10.22331/q-2020-05-14-266.

53 Thomas Vidick and Tina Zhang. Classical Proofs of Quantum Knowledge. In *Advances in Cryptology – EUROCRYPT 2021*, pages 630–660, 2021. doi:10.1007/978-3-030-77886-6_22.

54 Tzyh Haur Yang and Miguel Navascués. Robust self-testing of unknown quantum systems into any entangled two-qubit states. *Physical Review A*, 87:050102, 2013. doi:10.1103/PhysRevA.87.050102.

55 Daiwei Zhu, Gregory D. Kahanamoku-Meyer, Laura Lewis, Crystal Noel, Or Katz, Bahaa Harraz, Qingfeng Wang, Andrew Risinger, Lei Feng, Debopriyo Biswas, Laird Egan, Alexandru Gheorghiu, Yunseong Nam, Thomas Vidick, Umesh Vazirani, Norman Y. Yao, Marko Cetina, and Christopher Monroe. Interactive Protocols for Classically-Verifiable Quantum Advantage, 2021. arXiv:2112.05156

Matching Augmentation via Simultaneous Contractions

Mohit Garg[1]
Department of Computer Science and Automation, Indian Institute of Science, Bengaluru, India

Felix Hommelsheim
Faculty of Mathematics and Computer Science, Universität Bremen, Germany

Nicole Megow
Faculty of Mathematics and Computer Science, Universität Bremen, Germany

Abstract

We consider the matching augmentation problem (MAP), where a matching of a graph needs to be extended into a 2-edge-connected spanning subgraph by adding the minimum number of edges to it. We present a polynomial-time algorithm with an approximation ratio of $13/8 = 1.625$ improving upon an earlier 5/3-approximation. The improvement builds on a new α-approximation preserving reduction for any $\alpha \geq 3/2$ from arbitrary MAP instances to well-structured instances that do not contain certain forbidden structures like parallel edges, small separators, and contractible subgraphs. We further introduce, as key ingredients, the technique of repeated simultaneous contractions and provide improved lower bounds for instances that cannot be contracted.

2012 ACM Subject Classification Theory of computation → Routing and network design problems

Keywords and phrases matching augmentation, approximation algorithms, 2-edge-connectivity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.65

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2211.01912` [14]

Funding *Mohit Garg*: Supported by SERB Core Research Grant (CRG/2022/001176) on "Optimization under Intractability and Uncertainty".

1 Introduction

In the `Matching Augmentation Problem (MAP)`, we are given an undirected graph G, where each edge $e \in E(G)$ has a weight in $\{0, 1\}$, and all the zero-weight edges form a matching. The task is to compute a minimum weight 2-edge-connected spanning subgraph (2-ECSS) of G, which is a connected graph $(V(G), F)$ with $F \subseteq E(G)$ that remains connected on deleting an arbitrary edge.

`MAP` is a fundamental problem in the field of *survivable network design* and is known to be MAX-SNP-hard with several better-than-2 approximation algorithms [3, 5, 6]. Prior to this work, the best-known approximation ratio for `MAP` was $\frac{5}{3}$, achieved by Cheriyan et al. [6].

Both [5, 6] provide combinatorial algorithms for `MAP`, where the approximation ratios are achieved by comparing the outputs against the minimum-cardinality 2-edge-cover (D_2). A 2-*edge-cover* of an undirected graph is a spanning subgraph in which each node has a degree of at least 2, but it may not be connected. Thus, computing a D_2 is a relaxation

[1] A part of this work was done while the author was affiliated with the University of Bremen and the University of Hamburg.

of MAP. In contrast to solving MAP, a D_2 can be computed exactly in polynomial time by extending Edmonds' matching algorithm [10]. A weaker approximation result for MAP by Bamas et al. [3] follows a very different approach: the output is compared against another lower bound on an optimal MAP solution, obtained by solving a linear programming relaxation, the so-called Cut LP. The integrality gap of the Cut LP is at least $\frac{4}{3}$ [3].

Our result. We present a polynomial-time algorithm for MAP with an approximation ratio of $\frac{13}{8} = 1.625$, improving the previous best ratio of 5/3.

▶ **Theorem 1.** *There is a polynomial-time $\frac{13}{8}$-approximation algorithm for MAP.*

This improvement builds on a new α-approximation preserving reduction for any $\alpha \geq 3/2$ from arbitrary MAP instances to well-structured instances that do not contain certain forbidden structures like parallel edges, small separators, and contractible subgraphs. We further introduce, as key ingredients, the technique of repeated simultaneous contractions and provide improved lower bounds for instances that cannot be contracted.

Further related work. MAP sits between the minimum unweighted 2-ECSS and the minimum weighted 2-ECSS problems. For the minimum weighted 2-ECSS problem, improving known 2-approximations [2, 18, 19, 27] is a major open problem. Whereas for the unweighted case, in a recent breakthrough, Garg et al. [13] provided a 1.326-approximation, improving the earlier $\frac{4}{3}$-approximations [17, 24].

Research on the minimum weighted 2-ECSS problem assuming that the set of zero-weight edges in the input graph has a certain structure such as forest, spanning tree, matching, or disjoint paths has received a lot of attention recently. A general variant is the Forest Augmentation Problem (FAP), where the edges in the input graph have 0/1 edge weights. For FAP, only recently, Grandoni et al. [15] obtained a 1.9973-approximation, breaking the approximation barrier of 2. A famous special case of FAP is the unweighted Tree Augmentation Problem (TAP) where the zero-weight edges in the input graph form a single connected component. In a long line of research, several better-than-2 approximations have been achieved for TAP [1, 4, 7–9, 11, 12, 16, 20–23, 25, 26], culminating in a 1.393-approximation by Cecchetto et al. [4].

Notice that MAP is somewhat orthogonal to TAP in terms of connectivity as it has many small connected components as input instead of a single big one. Understanding both the extreme cases well, TAP and MAP, seems promising for making further progress for FAP.

Organization of the rest of the paper. Section 2 contains preliminaries and a high-level overview of our work along with some important definitions. Section 3 and Section 4 consist of the description of our reduction and algorithm, respectively, along with the corresponding theorem and lemma statements which we prove in the appendix of the full version of this paper [14]. Using these theorems and lemmas, in Section 5 we prove Theorem 1. In Section 6, we conclude with final remarks, pointing out the bottleneck for improving our algorithm.

In the full version [14] we have included detailed proofs of various lemmas in Appendices A–F, which makes our write-up lengthy. A lot of material, especially in Appendices A, C, and F are standard, but formally necessary; the new innovations are mainly contained in Appendices B, D, and E. While some proofs admit a case analysis, no single proof has too many cases. We have tried to keep the exposition reader-friendly and make the proofs easily verifiable at the expense of making the write-up a bit lengthy; a terser style might have saved some pages.

2 Technical overview

2.1 Preliminaries

We use standard notation for graphs. We consider weighted undirected graphs where each edge has a weight of either 0 or 1. A *MAP instance* consists of a graph G such that the zero-weight edges of G form a matching, and the task is to compute a minimum weight 2-edge-connected spanning subgraph (2-ECSS) of G, which is a connected subgraph $(V(G), F)$ which remains connected on deletion of an arbitrary edge. Without loss of generality, we may assume that the input graph G is 2-edge-connected; this can be checked in polynomial time by testing for each edge whether its deletion results in a disconnected graph.

Given a set of edges $F \subseteq E(G)$, $||F||$ denotes the weight of F, i.e., the number of unit edges in F. With slight abuse of notation, we denote the weight of a subgraph H of G by $||H||$. Thus, $||H|| = ||E(H)||$. Given a MAP instance G, let $\mathsf{OPT}(G)$ represent a 2-edge-connected spanning subgraph of G of minimum total weight $\mathsf{opt}(G) := ||\mathsf{OPT}(G)||$. When G is clear from the context we sometimes omit G.

Whenever we speak of components of a graph we refer to its *connected* components.

2.2 Algorithmic template and the previous $\frac{5}{3}$-approximation

The algorithm and analysis of Cheriyan et al. [6], for obtaining a $\frac{5}{3}$-approximate solution for MAP, exemplifies the general template for obtaining combinatorial algorithms for 2-edge-connected spanning subgraphs used in several works [5, 11, 17, 20]. We first explain this template, by giving an overview of the algorithm of Cheriyan et al. [6], and then in the next subsection highlight our approach where we alter this template to achieve our improvement.

The algorithm consists of two parts. The first part is a preprocessing step which constitutes a $\frac{5}{3}$-approximation preserving reduction from arbitrary MAP instances to well-structured instances that do not contain certain forbidden structures. This is a key element of their work, which helps them to improve upon an earlier $\frac{7}{4}$-approximation by getting rid of certain hard instances.

In the second part, they handle instances that do not contain any of the forbidden structures through a *discharging scheme*. Their algorithm starts by computing a minimum 2-edge-cover, D_2 (in polynomial time). Additionally, all the zero-edges are added to the D_2, so that the edges not in the D_2 are all unit-edges. Now, since a 2-ECSS is a 2-edge-cover, $||D_2||$ lower bounds opt, the weight of the minimum weight 2-ECSS. To output a $(1+c)$-approximate solution (for $c = \frac{2}{3}$), one has $(1+c)||D_2||$ *charge* to work with. This charge is used to buy the edges of the D_2 and a charge of c is distributed to each of the unit edges of the D_2. Now, they incrementally transform this D_2 into a 2-ECSS by adding edges to it. For each edge that is added, a charge of 1 is used up from the available charge (which is taken from nearby edges), i.e., their D_2 incrementally evolves into a 2-ECSS at the expense of *discharging*. This is an oversimplified view of their actual algorithm. In reality, sometimes they even delete edges in the process which results in gaining charge.

We briefly describe the two steps involved in transforming the D_2 into a 2-ECSS, namely *bridge covering* and *gluing*. Note that a D_2 can have several connected components. Some of these components can be 2-edge-connected, whereas some might have bridges (i.e., deleting those edges will result in increasing the number of components). The first step is to *cover* all the bridges one by one. Given a bridge, they add edges so that the bridge becomes part of a cycle; as a side effect, multiple components might merge into one. At the end of the bridge-covering step, their graph has only 2-edge-connected components. They ensure that

after using up the charge for buying the edges in the process, each component with at least 3 unit-edges has at least a charge of 2 leftover. Components with exactly 2 unit-edges (cycles of lengths 3 and 4) keep the initial charge of $2c = \frac{4}{3}$.

Next, in the gluing step, the components are merged into a single component using the leftover charge in the components resulting in a feasible solution. To see how this might be done, momentarily assume that all components have at least 3 unit-edges, i.e., having a leftover charge of at least 2. Here, one can simply contract each component into a single node, find a cycle in the contracted graph, and buy all the edges in that cycle. This will result in merging all the components corresponding to the nodes in the cycle into a single component. To be able to repeat such merges, we need to ensure that we have a leftover charge of at least 2 in this newly formed component. For a cycle of length $k \geq 2$, initially there was a charge of at least $2k$ in the corresponding components, and we need to buy exactly k edges. Thus, after the merge, the leftover charge in the new component is at least $2k - k = k \geq 2$, maintaining the charge invariant. Repeating this process eventually leads to a feasible solution. Unfortunately, this idea is not guaranteed to work if there are components with 2 unit-edges; a cycle with 2 nodes corresponding to such components will have a total charge of $2 \times \frac{4}{3} = \frac{8}{3}$, and after buying the 2 edges in the cycle, we will be left with a charge of only $\frac{2}{3}$. On repeating such merges, the graph will run out of charge before the gluing finishes. To handle such small components one needs to delete edges to gain charge.

2.3 Highlights of our approach and innovations for the $\frac{13}{8}$-approximation

Our approach follows the same broad framework explained above with some key innovations. Formal definitions will be given in later sections.

Preprocessing

For all $\alpha \geq \frac{3}{2}$, we provide an α-approximation preserving reduction from arbitrary MAP instances to *structured graphs*. Our list of forbidden structures subsumes the one by Cheriyan et al. [6] and consists of parallel edges, cut vertices, small separators, and *contractible subgraphs*.

Contractible subgraphs are a general form of contractible cycles as considered in [17]. A 2-edge-connected subgraph H of a graph G is contractible if each 2-ECSS(G) includes at least $\frac{1}{\alpha}||H||$ unit-edges from $G[V(H)]$. Since we are interested in only an α-approximate solution, we may contract $V(H)$ into a single node, solve the problem on the contracted graph, and add the edges of H to the solution without any loss in the approximation ratio. As an example, suppose a 6-cycle in G of weight 6 has 2 antipodal vertices that have degree 2 in G. Then, $\mathsf{OPT}(G)$ must include the 4 edges incident on these two vertices. As the cycle costs 6, and $\mathsf{OPT}(G)$ is guaranteed to pick at least weight 4 from the subgraph induced on the vertices of the cycle, for a $\frac{3}{2}$-approximation, it suffices to buy all edges of this cycle, contract it and solve the reduced problem. We can detect all contractible subgraphs with constant-size vertex sets in polynomial time and remove them during preprocessing. Interestingly, some intricate structures considered in [6] are simply contractible subgraphs.

We further exclude several small separators, which is crucial for our bridge covering and gluing steps as we have less charge at our disposal. Given a separator, we split the graph into two or three parts, recursively solve the problem on the smaller parts, and then combine the solutions arguing that the approximation ratio is preserved. If each of these parts has at least 5 vertices, this step is relatively straightforward. But for parts containing at most 4 vertices, the argument becomes significantly more challenging, in particular, since we are

aiming for a better guarantee than previous work. Handling the small separators forms a substantial part of our work consisting of several innovations. In particular, given a separator that splits the graph into two parts, the structure of the interaction of the separator with the parts is exploited in carefully constructing the subproblems. Here, we sometimes introduce *pseudo-edges*, representing possible connections via the other part, and suitably remove them during the combining step. Our reduction, which works for any $\alpha \geq \frac{3}{2}$, might be useful for future works. We only need $\alpha = \frac{13}{8}$ for our main result.

Bridge covering

Empowered by a stronger preprocessing, we can rule out more structures in the input graph, which enables us to obtain a bridgeless 2-edge-cover of G, even for an approximation ratio of $(1 + c)$, for $c = \frac{5}{8}$. In fact, our bridge-covering works even for $c = \frac{3}{5}$, so it might also be useful for future works. At the end of the bridge-covering step, we have the following charges left in the 2-edge-connected components: 2 in the *large* components (containing 4 or more unit-edges), $3c = \frac{15}{8} < 2$ in the *medium* components (containing 3 unit-edges) and $2c = \frac{5}{4} \ll 2$ in the *small* components (containing 2 unit-edges).

Gluing

In the gluing step, we are able to merge all the medium components into large components even though medium components have strictly less than 2 charge. We are also able to handle some small components that have a particular configuration by deleting edges and gaining charge. Unfortunately, we were unable to handle all the small components as they have a minuscule charge. In the end, we are left with a *special* configuration that has only large (with charge ≥ 2) and small components (with charge $\frac{5}{4}$) which cannot be merged.

Two-edge-connecting special configurations

The small components of the special configuration originate in the initial D_2 and could not be merged. So we ask the following question. How *close* are these small components to OPT restricted to the vertices of the small components? Intuitively, if they are close, we should be able to do something algorithmically as we are roughly doing what OPT is doing on this part of the graph. Otherwise, if they are not close, we should be able to argue that OPT does much worse than what the D_2 does on this part of the graph, giving us an improved lower bound. Our main conceptual innovation is in articulating a notion of closeness and making this intuition work.

Method of simultaneous contractions

We now describe our measure of closeness. Let G be our input structured graph and let $H_1, \cdots, H_s$ be the small components of the special configuration obtained. We count the total number of unit-edges bought by OPT from the following subgraphs $G[V(H_1)], \cdots, G[V(H_s)]$. If this number is more than $\frac{8}{13}$ times the number of unit-edges in the small components of our special configuration, which is precisely $2s$, we say that the small components are close to OPT. Otherwise, they are not close.

Observe when the small components are indeed close, on average, each H_i is contractible, preserving an approximation ratio of $\frac{13}{8}$. Thus, algorithmically, we can simultaneously contract each $V(H_i)$ into a distinct single node, solve the problem on the reduced instance (which can be done recursively, as contracting vertices into nodes decreases the size of the graph), and add the edges of the small components to the solution, without incurring a loss in approximation.

When the small components are not close to OPT, i.e., the H_i's are not *simultaneously contractible*, we rely on the gluing step of Cheriyan et al. [6] using a charge of $\frac{4}{3}$ per small component instead of our original charge of $\frac{5}{4}$, increasing our cost. Our improvement, in this case, comes from improving the lower bound.

Note that it is not possible for us to check in polynomial time whether the small components are simultaneously contractible or not; so what should we do – contract, or use the gluing algorithm of Cheriyan et al. [6]? We do both and return the solution with a smaller weight and argue that in either scenario the algorithm performs well.

Improved lower bound

In the case when the small components are not simultaneously contractible, OPT picks at most $\frac{8}{13} \cdot 2s$ unit-edges from the $G[V(H_i)]$'s put together. Thus, at least $2s - \frac{16}{13}s = \frac{10}{13}s$ unit-edges are not picked from within the small components. We show that for each unit-edge not picked by OPT from this part, OPT buys on average at least $1 + \frac{1}{12}$ edges that go between different small components. To argue this, we crucially use the fact that the special configurations have a restricted structure, as certain merges are not possible in it. Thus, we show that in total the number of unit-edges used by OPT on the vertices of the small components is at least $\frac{8}{13} \cdot 2s + (1 + \frac{1}{12})\frac{10}{13}s = \frac{161}{78}s$, which is strictly more than $2s$, which is the number of unit-edges used by the D_2 on this part. Finally, through an elegant argument, we are able to use this improved lower bound on OPT restricted only to the vertices of the small components to show that it compensates for the increased cost incurred during gluing the small components.

2.4 Important definitions

We give some definitions that we need for the presentation of our algorithm.

▶ **Definition 2** ($f(\cdot)$). *Given a MAP instance G, let*

$$f(G) = \max\{\frac{13}{8} \cdot \mathsf{opt}(G) - 2, \mathsf{opt}(G)\}.$$

For a MAP instance G, we will compute a 2-ECSS of G with weight at most $f(G)$. Observe that the "-2" term gives us a slightly better bound than claimed, which we crucially exploit in our preprocessing.

▶ **Definition 3** (size of a graph $s(\cdot)$). *Given a graph G, its* **size** *is $s(G) = 10 \cdot |V(G)|^2 + |E(G)|$.*

We will show that the running time of our algorithm is upper bounded by a polynomial in the size of the input graph.

▶ **Definition 4** (notation graph contraction). *Given a graph G and a set of vertices $T \subseteq V(G)$, G/T denotes the graph obtained from G after contracting all the vertices in T into a single vertex. More generally, given disjoint vertex sets $T_1, \cdots T_k \subseteq V(G)$, $G/\{T_1, \cdots, T_k\}$ denotes the graph obtained from G after contracting vertices of each T_i into single vertices.*

Note that edges in G and $G/\{T_1, \cdots, T_k\}$ are in one-to-one correspondence. Given a subgraph H of the contracted graph, we use $\hat{H}$ to refer to the subgraph of G containing precisely those edges that correspond to the edges of H.

▶ **Definition 5** (contractible subgraphs). *Let $\alpha \geq 1$ and $t \geq 2$ be fixed constants. Given a 2-edge-connected graph G, a collection of vertex-disjoint 2-edge-connected subgraphs $H_1, H_2, ..., H_k$ of G is called* (α, t, k)-contractible *if $2 \leq |V(H_i)| \leq t$ for every $i \in [k]$ and every 2-ECSS of G contains at least $\frac{1}{\alpha}||\bigcup_{i\in[k]} E(H_i)||$ unit-edges from $\bigcup_{i\in[k]} E(G[V(H_i)])$.*

In our preprocessing, we will remove all $(\frac{13}{8}, 12, 1)$-contractible subgraphs, which we simply refer to as contractible subgraphs. Later, when considering a special configuration with n_s small components, we will work with a $(\frac{13}{8}, 4, n_s)$-contractible collection of small components; we will refer to the special configuration simply as $\frac{13}{8}$-simultaneously contractible.

2.5 Algorithm overview

Here, we give a brief overview of our main algorithm.

Step 1: Preprocessing: We apply our reduction to obtain a collection of subproblems of MAP on structured graphs (Section 3). We then assume that we are given some structured graph G.

Step 2: Bridge covering: We compute a D_2 in polynomial-time and apply bridge covering to obtain an *economical* bridgeless 2-edge-cover H – a bridgeless 2-edge-cover of low cost (Section 4.1).

Step 3: Special configuration: Given H, we compute a special configuration S of G (Section 4.2).

Step 4: Contract vs. glue: We compute two feasible solutions S_1 and S_2: S_2 is obtained by applying the algorithm of [6] to G and S (Section 4.3); S_1 is obtained by calling Step 1 for G_S, which arises from G by contracting each small component of S to a single vertex. Finally, we output $\arg\min\{||S_1||, ||S_2||\}$.

3 Preprocessing

We show that, for purposes of approximating MAP with any approximation ratio at least $\frac{3}{2}$, it suffices to consider MAP instances that do not contain certain forbidden configurations. These configurations are cut vertex, parallel edge, contractible subgraph, S_0, S_1, S_2, $S_{\{3,4\}}$, S_3, S_4, S_5, S_6, S'_3, S'_4, S'_5, and S'_6. The formal definitions of these structures are provided in Appendix B of the full version [14]. Each of these configurations is referred to as a **type** and is of constant size. A MAP instance with at least 20 vertices that does not contain any of these forbidden configurations is termed as **structured**.

We briefly describe some of the types that we forbid in a structured graph. Apart from cut vertex, parallel edge, and contractible subgraph, the other forbidden structures we consider can be broadly divided into two categories: (a) "Path-like"-separators and (b) "Component-like"-separators. Path-like separators are certain paths which when removed from the input graph disconnects it. Forbidding these structures in the structured graph is mostly used in the bridge-covering step. Roughly speaking, the absence of these structures helps us in finding sufficient credit while covering some path (consisting of bridges) between 2-edge connected blocks of the 2-edge-cover. Component-like structures, on the other hand, are certain 2-edge-connected subgraphs that when removed from the input graph disconnects it. Their absence from structured graphs is mainly exploited in the gluing step; it allows us to find certain cycles through some small components which help us gain credit that is needed for the gluing.

The reduction from MAP instances to structured graphs is given by the following algorithm where we assume ALG is an algorithm that works on structured graphs. Our reduction is essentially a divide-and-conquer algorithm. It searches for a forbidden configuration and if it detects one, it divides the problem into a few subproblems (at most 3) of smaller sizes, solves them recursively, and then combines the returned solutions into a solution for the original instance. In case there are no forbidden configurations in the input (the input is structured), it calls ALG to solve the problem.

Algorithm 1 Preprocessing.

function REDUCE(G)
  if G is simple and $|V(G)| \leq 20$ **then return** $\mathsf{opt}(G)$. ▷ by brute force

  Look for a forbidden configuration in G in the following type order:
  cut vertex, parallel edge, contractible subgraph, S_0, S_1, S_2, $S_{\{3,4\}}$, S_k, S'_k for $k \in \{3,4,5,6\}$.
  Stop immediately on detecting a forbidden configuration.

  if a forbidden configuration is detected **then**
    Call it F and let T be the type of F.
    $(H_1, H_2, H_3) = \mathsf{Divide}_T(G, F)$. ▷ H_2 and/or H_3 are always empty for certain types
    $H_i^* = \mathsf{Reduce}(H_i)$ for all $i \in \{1,2,3\}$.
    return $\mathsf{Combine}_T(G, H_1^*, H_2^*, H_3^*)$.
                                                ▷ G is now structured
  return $\mathsf{ALG}(G)$.

In the above algorithm, Divide_T and $\mathsf{Combine}_T$ are subroutines that are defined in Appendix B of the full version [14], which also contains proofs of the following lemmas.

▶ **Lemma 6.** *For all types T, Divide_T and $\mathsf{Combine}_T$ are polynomial time algorithms. Furthermore, given a MAP instance G and a type T, one can check in polynomial time whether G contains a forbidden configuration of type T.*

▶ **Lemma 7.** *Given a MAP instance G and a forbidden configuration F that appears in G of type T from the list L =(cut vertex, parallel edge, contractible subgraph, S_0, S_1, S_2, S_k, S'_k, $k \in \{3,4,5,6\}$) such that G does not contain any forbidden configuration of a type that precedes T in the list L and $\mathsf{Divide}_T(G, F) = (H_1, H_2, H_3)$, then the following statements hold:*

- **(i)** *for each $i \in \{1,2,3\}$ H_i is a MAP instance,*
- **(ii)** *$s(H_1) + s(H_2) + s(H_3) < s(G)$, and*
- **(iii)** *if for each $i \in \{1,2,3\}$, H_i^* is a 2-ECSS of H_i such that $||H_i^*|| \leq f(H_i)$, then $\mathsf{Combine}_T(G, H_1^*, H_2^*, H_3^*)$ is a 2-ECSS of G such that $||\mathsf{Combine}_T(G, H_1^*, H_2^*, H_3^*)|| \leq f(G)$.*

Using the above lemma, we can establish the following result: If for all structured graphs G, $\mathsf{ALG}(G)$ is a 2-ECSS of G such that $||\mathsf{ALG}(G)|| \leq f(G)$, then for all MAP instances G, $\mathsf{Reduce}(G)$ is a 2-ECSS of G such that $||\mathsf{Reduce}(G)|| \leq f(G)$.

We will produce an *admissible* ALG that calls Reduce on a smaller instance. Since Reduce and ALG call each other, we need a slightly stronger result, which is obtained using an induction argument. We first define an admissible algorithm.

▶ **Definition 8** (admissible)**.** *An algorithm* ALG *is* **admissible** *if the following holds. If for all MAP instances G with $s(G) \leq t$, $\mathsf{Reduce}(G)$ is a 2-ECSS of G such that $||\mathsf{Reduce}(G)|| \leq f(G)$, then for all structured graphs G such that $s(G) = t + 1$, $\mathsf{ALG}(G)$ is a 2-ECSS of G such that $||\mathsf{ALG}(G)|| \leq f(G)$. Furthermore, if $T(s)$ denotes the running time of* ALG *for structured graphs of size s, and $T'(s)$ denotes the running time of* Reduce *on MAP instances of size s', then $T(s) \leq T'(s-1) + poly(s)$.*

Our main results in this section are the following.

▶ **Theorem 9.** *If* ALG *is an admissible algorithm, then for all MAP instances* G, Reduce(G) *is a 2-ECSS of* G *such that* $||\text{Reduce}(G)|| \leq f(G)$.

▶ **Theorem 10.** *If* ALG *is an admissible algorithm, then* Reduce *runs in polynomial time.*

The proofs of the above two results follow from a straightforward application of Lemmas 6 and 7 and are included in Appendix A of the full version [14]. Now, if we can find an admissible ALG, Theorem 9 and Theorem 10 immediately imply Theorem 1. In the next subsections, we exhibit an admissible ALG.

4 Algorithm for structured graphs

We exhibit an admissible algorithm ALG that takes as input a structured graph G and outputs a 2-ECSS of G with weight at most $f(G)$. Our algorithm has three main steps. First, we compute an "economical" bridgeless 2-edge-cover of G. Then, with the aid of this 2-edge-cover, we compute a "special" configuration of G. We two-edge-connect the special configuration in two ways and return the solution with minimum weight. We define the relevant terms and explain these steps below.

Algorithm 2 Main algorithm for structured graphs.

function ALG(G) ▷ G is structured
  $H =$ economical bridgeless 2-edge-cover(G)
  $S =$ special configuration(G, H)
  $R =$ Contract-vs-Glue(G, S)
  return R

4.1 Computing an economical bridgeless 2-edge-cover

Given a structured graph G, we first compute an economical bridgeless 2-edge-cover of it. Before we define an economical bridgeless 2-edge-cover, we need to first define **small, medium,** and **large**, which is used to categorize a 2-edge-connected subgraph of G based on its weight.

▶ **Definition 11** (small, medium, large)**.** *For a weighted graph* G*, we call a 2-edge-connected subgraph* H *of* G **small** *if* $||H|| \leq 2$, **medium** *if* $||H|| = 3$, *and* **large** *if* $||H|| \geq 4$.

Note that for structured graphs, the only possible small components are cycles of length 3 or 4 with exactly 2 unit-edges, and medium components are cycles of length 3, 4, 5, or 6 with exactly 3 unit-edges.

▶ **Definition 12.** *A bridgeless 2-edge-cover* H *of a graph* G *is* **economical** *if all the zero-edges of* G *are in* H, $||H|| \leq \frac{13}{8} \cdot ||D_2(G)|| - 2n_\ell - \frac{15}{8}n_m - \frac{5}{4}n_s$, *where* n_ℓ, n_m, *and* n_s *are the number of large, medium, and small components of* H, *respectively. Furthermore, there exists a* D_2 *of* G *such that each small component of* H *is a small component of the* D_2.

Our main result for this subsection is as follows.

▶ **Theorem 13.** *Given a structured graph* G*, we can compute an economical bridgeless 2-edge-cover of* G *in polynomial time.*

To compute an economical bridgeless 2-edge-cover of G, we first find a D_2 of G and include all the zero-edges in it. Next, we transform this D_2 into a "canonical" D_2, which is defined in Appendix C of the full version [14]. Then, we cover the bridges of the canonical D_2 to get an "economical" bridgeless 2-edge-cover. All of this can be done in polynomial time. The details with the proof of Theorem 13 are in Appendix C of the full version [14].

4.2 Computing a special configuration

Next, given an economical bridgeless 2-edge cover H of a structured graph G, we compute a "special" configuration of G. A special configuration is a bridgeless 2-edge-cover that satisfies certain additional properties. In particular, it does not contain any medium components.

▶ **Definition 14** (Special configuration). *Given a structured graph G, we say H is a special configuration of G if*

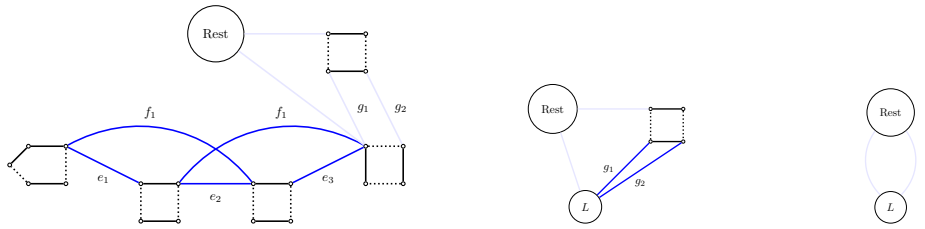
- **(i)** *H is an economical bridgeless 2-edge-cover of G,*
- **(ii)** *H does not contain medium-size components,*
- **(iii)** *G/H does not contain good cycles*
- **(iv)** *G/H does not contain open 3-augmenting paths, and*
- **(v)** *H does not contain a small to medium merge or small to large merge.*

The terms and notation used in conditions (iii)-(v) are formally defined in Appendix D of the full version [14]. Without going into details, we briefly describe the structures defined in (iii)-(v). A good cycle is a simple cycle C in G/H that contains either a) two large components, b) one large component and one small component containing a shortcut, or c) two small components each containing a shortcut. Here, we say a small component S is shortcut w.r.t. C if in $G[V(S)]$ there exists a Hamiltonian path from u to v of weight 1, where u and v are the vertices incident to C when S is expanded. Hence, there is a unit-edge in the small component S that is redundant for the 2-edge-connectivity of H after we add the edges of C to it. An open 3-augmenting path is a simple path P in G/H through 4 small components such that for each of the two interior small components there is a shortcut w.r.t. P. Finally, a small to medium merge (or small to large merge) is a set of 3 small components $S_1, S_2, S_3 \in H$ such that in $G' = G[V(S_1 \cup S_2 \cup S_3)]$ there exists a set of edges that form two medium components (or one large component) of weight precisely 6 spanning $V(G')$.

Essentially, these conditions restrict the structure of special configurations. For example, in the graph G/H (the graph obtained by contracting the various components of H into single nodes), there is no cycle that has 2 or more nodes corresponding to large components. In particular, a special configuration contains at least one small component or is already feasible. The restricted structure of special configurations will be crucially exploited while proving an improved lower bound on $\mathsf{OPT}(G)$.

From an economical bridgeless 2-edge-cover H, we obtain a special configuration by repeatedly searching for the four forbidden structures (properties (ii)-(v) above) and buying and selling certain edges such that we turn H into an economical bridgeless 2-edge cover H' with fewer components. One can show that searching for such structures can be done in polynomial time.

We briefly explain this process by an example: Figure 1a. The black edges correspond to the economical bridgeless 2-edge-cover H, where the dotted edges are of weight 0. The (bold and faint) blue edges are edges of G that are not in H. The 3 blue edges e_1, e_2, and e_3 form an open 3-augmenting path in G/H (as it can "shortcut" the two black unit edges adjacent to e_2). By the properties of structured graphs, one can show that the blue edges f_1 and f_2 must exist, and hence four components of H can be merged into one large component by



(a) open 3-augmenting path. **(b)** good cycle. **(c)** after merges.

Figure 1 An example of obtaining a special configuration.

buying all the 5 bold blue edges and selling the 2 black unit edges adjacent to e_2, which are "shortcut" by the open 3-augmenting path, to obtain Figure 1b; as initially the 4 components incident to the blue edges have a credit of $9 \times \frac{5}{8} \geq 5$, we buy 5 blue and sell 2 black edges to have a final credit of at least 2. One can show that this step is tight for our analysis with an approximation ratio of 13/8. Now, in Figure 1b the blue edges g_1 and g_2 form a good cycle in G/H (as it can be merged into a single large component). Initially, the credits in the large and small components that are part of this good cycle have a credit of $2 + 2 \times \frac{5}{8} \geq 3$. We buy the edges g_1 and g_2 and sell the unit edge adjacent to both g_1 and g_2 to form a single 2-edge-connected component having a credit of at least $3 - 2 + 1 = 2$, and thus the good cycle merges into a large component as shown in Figure 1c. In general, we show the following theorem, which is proved in Appendix D of the full version [14].

▶ **Theorem 15.** *Given a structured graph G and an economical bridgeless 2-edge-cover of it, we can compute a special configuration of G in polynomial time.*

4.3 Two-edge-connecting special configurations

Finally, we present the last part of our algorithm, which we call "Contract-vs-Glue", that converts a special configuration into a 2-edge-connected graph. Recall, a special configuration is an economical bridgeless 2-edge-cover of a structured graph that contains only small and large components and satisfies certain additional properties. Our algorithm computes two solutions and returns the one with a lower weight. The first solution is obtained by contracting the small components into single nodes and recursively computing the solution on the contracted graph (this is done by calling Reduce on the contracted graph) and then adding the edges in the small components to the solution after expanding it back. The second solution is obtained by following the "Gluing Algorithm" of Cheriyan et al. [6], which we call "Glue", and reproduce it below for completeness' sake.

Algorithm 3 Contract-vs-Glue.

function CONTRACT-VS-GLUE(G, S) ▷ G is structured, S is a special configuration of G
  if S is a 2-ECSS of G **then return** S
  Let $H_1, \cdots, H_k$ be the small components of S. ▷ now S must have small components
  $G_1^* = \mathsf{Reduce}(G/\{H_1, \cdots, H_k\})$
  $S_1 = (V(G), E(\hat{G}_1^*) \cup \bigcup_{i \in [k]} E(H_i))$
  $S_2 = \mathsf{Glue}(G, S)$
  return $\arg\min\{||S_1||, ||S_2||\}$

We will be using the following lemmas to prove our main result.

▶ **Lemma 16.** *Let G be a structured graph, S be a special configuration of G with small components $H_1, \cdots, H_k$, and let $G_1^* = \mathsf{Reduce}(G/\{H_1, \cdots, H_k\})$. If G_1^* is a 2-edge-connected spanning subgraph of $G/\{H_1, \cdots, H_k\}$, then $(V(G), E(\hat{G}_1^*) \cup \bigcup_{i\in[k]} E(H_i))$ is a 2-edge-connected spanning subgraph of G. Furthermore, if $H_1, \cdots, H_k$ is $(\frac{13}{8}, 4, k)$-contractible in G and $||G_1^*|| \leq f(G/\{H_1, \cdots, H_k\})$, then $||(V(G), E(\hat{G}_1^*) \cup \bigcup_{i\in[k]} E(H_i))|| \leq f(G)$.*

The first statement in the above lemma is straightforward to see. The second part is obtained by specializing Lemma 33 of Appendix B of the full version [14] to our parameters.

The following lemma is proved implicitly in [6]. For completeness, we give a full proof in Appendix F of the full version [14].

▶ **Lemma 17.** *Let G be a structured graph and S be a special configuration of G with n_ℓ large and n_s small components. Then, $\mathsf{Glue}(G, S)$ is a 2-edge-connected spanning subgraph of G with $||\mathsf{Glue}(G, S)|| \leq ||S|| + 2n_\ell + \frac{4}{3}n_s - 2$.*

Also, we prove the following lower bound result. Informally, if a 2-ECSS includes t fewer edges from within the small components of a special configuration, it must include $t(1 + \frac{1}{12})$ edges going between different small components. The proof is given in Appendix E of the full version [14]..

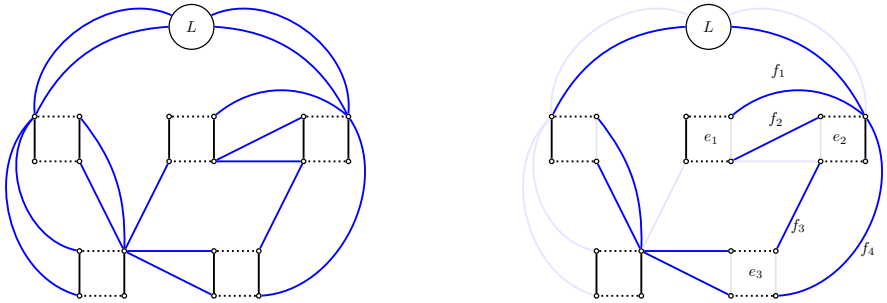
▶ **Lemma 18.** *Let G be a structured graph and S be a special configuration of G with k small components: $H_1, \cdots, H_k$. Let R be any 2-edge-connected spanning subgraph of G such that $2k - \sum_{i\in[k]} ||R[V(H_i)]|| = t$, then $\sum_{i<j\leq k} e_R(V(H_i), V(H_j)) \geq (1 + \frac{1}{12})t$, where $e_R(A, B)$ represents the number of unit-edges going between vertex sets A and B in R.*

Here, we give an intuition on how to prove Lemma 18. Fix a structured graph G together with a special configuration S and a 2-edge-connected spanning subgraph R as specified in Lemma 18. In order to simplify things, here we assume that each small component H_i of S is a cycle of length 4 such that G does not contain any of the diagonals of H_i. Furthermore, let $H_1, H_2, \ldots, H_\ell$ be the set of small components in S.

An edge between two small components is called **crossing**, whereas an edge inside a small component is called **inside**. Informally speaking, Lemma 18 states that, on average, for each inside edge $e \in E(S)$ of some small component of S that is not present in R, $E(R)$ has to contain at least $1 + \frac{1}{12}$ crossing edges. First, one can show that the vertices incident to an inside edge e that is not present in $E(R)$ cannot be adjacent to vertices of a large component of S, as otherwise this implies that S contains a good cycle, contradicting that S is special. Hence, each vertex incident to an inside edge e that is not present in $E(R)$ must be incident to at least one crossing edge in R.

In order to show that R contains sufficiently many crossing edges, we define an assignment ξ that distributes for each inside or crossing edge of R a total charge of one to the small components $H_1, \ldots, H_\ell$. The sum over all charges of edges incident to some component H_i then defines the *load* of the component H_i. Note that, by construction, the total load over all small components is equal to the number of inside and crossing edges in R.

Each inside edge contributes one to the charge of its component, while each crossing edge distributes a charge of one to the components incident to it: if only one of the two unit edges of $E(S)$ adjacent to a crossing edge is shortcut (absent) in R, then the component with the shortcut edge receives a charge of one from that crossing edge. Otherwise, both incident components receive a charge of $\frac{1}{2}$. Consider for example Figure 2b, where the bold edges represent R. By the above assignment, the components incident to f_3 receive a charge of $\frac{1}{2}$ each from f_3, while only the component containing e_3 receives a charge of 1 from f_4. The total load of the components (from left to right, top to bottom) then is $3, \frac{5}{2}, 2, 2$, and $\frac{7}{2}$, respectively.



(a) Special Configuration. **(b)** Optimal Solution.

Figure 2 Example: special configuration, optimal solution, and lower bound.

From our assignment, one can easily argue that the load of each small component is ≥ 2. Furthermore, if there are no two shortcut edges that are adjacent to a crossing edge, then it clearly follows that Lemma 18 holds. In fact, in this case, we could replace the $1 + \frac{1}{12}$ by 2 in the lemma – a much stronger result.

Hence, we may assume that there are some shortcuts that share crossing edges, e.g. edges e_1, e_2, and e_3 in Figure 2b. However, in this case, the edges f_2 and f_3 (which form an *open 2-augmenting path*) cannot be extended to an open 3-augmenting path (since S is special); the edges f_1 and f_4 have to go back to the component containing e_2. One can show that in this case (since there are also no good cycles or local merges), the average load of the components containing e_1, e_2, and e_3 is at least $\frac{5}{2}$. In the remaining case when there are no open 2-augmenting paths in R, using a similar argument we can also show that the average load of a component is at least $2 + \frac{1}{6}$. This load assignment then implies the statement of Lemma 18.

5 ALG is admissible

As noted earlier, from Theorems 9 and 10, it follows that if we can show ALG is admissible, Theorem 1 follows. Thus, we will now focus on proving that ALG is admissible.

▶ **Lemma 19.** ALG *is admissible.*

Before we proceed with the proof, we develop some key definitions and propositions that will be used in the proof. Throughout this subsection, G is a structured graph and S is a special configuration of G with small components $H_1, \cdots, H_{n_s}$.

▶ **Definition 20** (simultaneously-contractible)**.** *We say S is $\frac{13}{8}$-simultaneously contractible if the small components of S are $(\frac{13}{8}, 4, n_s)$-contractible in G.*

▶ **Definition 21** ($\mathsf{OPT}^L, \mathsf{OPT}^R, D_2^L, D_2^R$)**.** *We partition the vertex set of G in two sets: $V(G) = L \cup R$, where L consists of the vertices in the large components of the special configuration S and R is the set of remaining vertices, i.e., the set of vertices in the small components of S. Let OPT^L be the edges of $\mathsf{OPT}(G)$ that have at least one endpoint incident on a vertex in L, and OPT^R be the remaining edges of $\mathsf{OPT}(G)$, i.e., the edges whose both endpoints are in R. D_2^L and D_2^R are defined analogously: D_2^L is the set of edges of $D_2(G)$ that are incident on at least one vertex of L and $D_2^R = E(D_2(G)) \setminus D_2^L$. $\mathsf{opt}^L, \mathsf{opt}^R, d_2^L$, and d_2^R are defined to be $||\mathsf{OPT}^L||, ||\mathsf{OPT}^R||, ||D_2^L||$, and $||D_2^R||$, respectively.*

The following relationships are immediate.

▶ **Proposition 22.**

$||\mathsf{OPT}(G)|| := \mathsf{opt} = \mathsf{opt}^L + \mathsf{opt}^R$.

$||D_2(G)|| := d_2 = d_2^L + d_2^R$.

The following proposition is key to proving our bound.

▶ **Proposition 23.**

$\mathsf{opt}^L \geq d_2^L$

Proof. Assume for contradiction $\mathsf{opt}^L < d_2^L$. Observe $\mathsf{OPT}^L \cup D_2^R$ forms a 2-edge-cover of G, since each vertex of L has at least 2 edges incident on it from OPT^L (as OPT is a feasible 2-ECSS of G) and each vertex of D_2^R has 2 edges incident on it from D_2^R (as D_2^R are the edges of D_2 restricted to the small components of S, which were originally small in D_2). But

$$||\mathsf{OPT}^L \cup D_2^R|| = \mathsf{opt}^L + d_2^R < d_2^L + d_2^R = d_2,$$

which contradicts the fact that D_2 is a minimum 2-edge-cover of G. ◀

Now, we are ready to prove that ALG is admissible.

Proof of Lemma 19. Fix a structured graph G. To show ALG is admissible, we need to show two properties: (i) $\mathsf{ALG}(G)$ is a 2-edge-connected spanning subgraph of G with $||\mathsf{ALG}(G)|| \leq f(G)$ under the assumption that $\mathsf{Reduce}(G')$ is a 2-edge-connected spanning subgraph of G' with $||\mathsf{Reduce}(G')|| \leq f(G')$ for all MAP instances G' of size strictly smaller than the size of G, and (ii) $T(s(G)) \leq T'(s(G)-1) + poly(s)$, where T is the running time of ALG and T' is the running time of Reduce. Note that (ii) follows from the fact that each of the three steps in ALG takes polynomial time and in the final step, namely Contract-vs-Glue, ALG calls the subroutine Reduce only once on a smaller graph. Thus, we will focus on proving (i) below.

Note that ALG on input G first computes a special configuration S and then applies the algorithm Contract-vs-Glue on (G, S). If S is 2-ECSS, Contract-vs-Glue returns S, and $||S|| \leq \frac{13}{8}d_2 - 2n_\ell - \frac{5}{4}n_s$, where $n_\ell = 1$ is the number of large components and $n_s = 0$ is the number of small components in S (since S is an economical bridgeless 2-edge-cover of G). Thus, $||\mathsf{ALG}|| \leq f(G)$. Otherwise, S must contain at least one small component as observed in Section 4.2.

Let $H_1, \ldots, H_{n_s}$ be the small components of S. Contract-vs-Glue on (G, S) computes two solutions S_1 and S_2 and returns the one with lower weight. Recall S_1 is obtained by contracting the small components of S, calling Reduce on it, and then expanding the contracted nodes and adding back the edges of the small components. S_2 is computed by calling the $\mathsf{Glue}(G, S)$ subroutine. In either case, the output is guaranteed to be a 2-edge-connected spanning subgraph of G from Lemmas 16 and 17.

Now, to show $||\mathsf{ALG}(G)|| \leq f(G)$, we have two cases based on whether the special configuration S is $\frac{13}{8}$-simultaneously contractible in G. If S is a $\frac{13}{8}$-simultaneously contractible in G, then by invoking Lemma 16 (whose precondition holds since the contracted graph has size strictly smaller than G and then we have the guarantee that $||\mathsf{Reduce}(G')|| \leq f(G')$ for all MAP instances G'), we have $||\mathsf{ALG}(G)|| \leq ||S_1|| \leq f(G)$ and we are done.

In the case S is not $\frac{13}{8}$-simultaneously contractible in G, we will first lower bound opt and then upper bound $||S_2||$ to show $||\mathsf{ALG}(G)|| \leq f(G)$.

Lower bound on opt

From Propositions 22 and 23 we have

$$\mathsf{opt} = \mathsf{opt}^L + \mathsf{opt}^R \geq d_2^L + \mathsf{opt}^R.$$

We now focus on lower bounding opt^R. Recall OPT^R consists of edges whose both endpoints are contained in $\bigcup_{i\in[n_s]} V(H_i)$. We categorize the edges of OPT^R into two types.

- An edge of OPT^R is **inside** if both its endpoints belong to the same $V(H_i)$ for some i.
- An edge of OPT^R is **crossing** if its endpoints lie in distinct $V(H_i)$ and $V(H_j)$ for some $i \neq j$.

Since S is not $\frac{13}{8}$-simultaneously contractible in G, the number of unit-edges that are inside is at most $\frac{8}{13} \cdot 2n_s$. Let us say the number of inside edges is exactly t less than the number of unit-edges in the small components of S, i.e., $2n_s - t$, and this number is at most $\frac{8}{13} \cdot 2n_s$.

Now, to lower bound the number of unit-edges that are crossing, we invoke Lemma 18, which states that the number of unit-edges going between $V(H_i)$ and $V(H_j)$ for all $i \neq j$ is at least $(1 + \frac{1}{12})$t.

Thus, we have the following lower bound for opt.

$$\begin{aligned}\mathsf{opt} = \mathsf{opt}^L + \mathsf{opt}^R &\geq d_2^L + \mathsf{opt}^R = d_2^L + ||\text{inside}|| + ||\text{crossing}|| \\ &\geq d_2^L + (2n_s - t) + \left(1 + \frac{1}{12}\right) t,\end{aligned}$$

where $2n_s - t \leq \frac{8}{13} \cdot 2n_s$. The lower bound is minimized when t is kept as small as possible, i.e., when $2n_s - t = \frac{8}{13} \cdot 2n_s$, i.e., for $t = \frac{5}{13} \cdot 2n_s$. Thus,

$$\mathsf{opt} \geq d_2^L + \frac{8}{13} \cdot 2n_s + \left(1 + \frac{1}{12}\right) \frac{5}{13} \cdot 2n_s = d_2^L + \left(\frac{16}{13} + \frac{10}{12}\right) n_s = d_2^L + \frac{161}{78} \cdot n_s.$$

Upper bound on $||\mathsf{ALG}(G)||$

Since S is an economical bridgeless 2-edge-cover of G, we have

$$||S|| \leq \frac{13}{8} \cdot d_2 - 2n_\ell - \frac{5}{4} \cdot n_s,$$

where n_ℓ and n_s denote the number of large and small components of S, respectively. Also, from Lemma 17, we have

$$||S_2|| = ||\mathsf{Glue}(G, S)|| \leq ||S|| + 2n_\ell + \frac{4}{3} \cdot n_s - 2.$$

Combing the two bounds we obtain

$$||S_2|| \leq \frac{13}{8} \cdot d_2 + \frac{1}{12} \cdot n_s - 2.$$

Now we can split d_2 as $d_2^L + d_2^R$, and use the fact that $d_2^R = 2n_s$ to obtain our bound.

$$\begin{aligned}||\mathsf{ALG}(G)|| \leq ||S_2|| &\leq \frac{13}{8} \cdot d_2 + \frac{1}{12} \cdot n_s - 2 = \left(\frac{13}{8} \cdot d_2^L + \frac{13}{8} \cdot 2n_s\right) + \frac{1}{12} \cdot n_s - 2 \\ &= \frac{13}{8} \cdot d_2^L + \frac{10}{3} \cdot n_s - 2 \leq \frac{13}{8}\left(d_2^L + \frac{160}{78} \cdot n_s\right) - 2 \leq \frac{13}{8} \cdot \mathsf{opt} - 2 \leq f(G),\end{aligned}$$

where the second last inequality follows from the lower bound on opt obtained above. ◀

6 Conclusion

In this work, we presented a $\frac{13}{8}$-approximation for MAP, which is a fundamental problem in network design. While several of our steps also work for smaller approximation ratios, two of our steps are tight for $\frac{13}{8}$: First, constructing a special configuration is tight for $\frac{13}{8}$. In particular, the merge involving 3-augmenting paths, in the worst case, uses all the available credits. On the other hand, such a merge could not be avoided, as their absence from special configurations helps us improve the lower bound later. Furthermore, the lower bound is tight. Hence, simply obtaining a better construction of a special configuration is not enough as one has to improve upon the lower bound as well. Finally, even if one can resolve these two issues, our approximation ratio would still be tight for 1.6 at two places: First, constructing a bridgeless 2-edge-cover is tight for 1.6, even though we believe that this result can be strengthened. Second, in the construction of special configuration, handling the medium components is also tight for precisely 1.6. Hence, also here new ideas are needed in order to obtain an approximation ratio below 1.6.

Our result builds on a new $\frac{3}{2}$-approximation preserving reduction to instances not containing certain structures including small separators and contractible subgraphs. Furthermore, we introduced the method of simultaneous contractions and improved lower bounds to achieve our main result. These techniques seem general and applicable to other problems in network design.

References

1 David Adjiashvili. Beating approximation factor two for weighted tree augmentation with bounded costs. *ACM Trans. Algorithms*, 15(2):19:1–19:26, 2019.

2 Ajit Agrawal, Philip N. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM J. Comput.*, 24(3):440–456, 1995.

3 Étienne Bamas, Marina Drygala, and Ola Svensson. A simple lp-based approximation algorithm for the matching augmentation problem. In *IPCO*, volume 13265 of *Lecture Notes in Computer Science*, pages 57–69. Springer, 2022.

4 Federica Cecchetto, Vera Traub, and Rico Zenklusen. Bridging the gap between tree and connectivity augmentation: unified and stronger approaches. In *STOC*, pages 370–383. ACM, 2021.

5 Joe Cheriyan, Jack Dippel, Fabrizio Grandoni, Arindam Khan, and Vishnu V. Narayan. The matching augmentation problem: a $\frac{7}{4}$-approximation algorithm. *Math. Program.*, 182(1):315–354, 2020.

6 Joseph Cheriyan, Robert Cummings, Jack Dippel, and Jasper Zhu. An improved approximation algorithm for the matching augmentation problem. In *ISAAC*, volume 212 of *LIPIcs*, pages 38:1–38:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

7 Joseph Cheriyan and Zhihan Gao. Approximating (unweighted) tree augmentation via lift-and-project, part I: stemless TAP. *Algorithmica*, 80(2):530–559, 2018.

8 Joseph Cheriyan and Zhihan Gao. Approximating (unweighted) tree augmentation via lift-and-project, part II. *Algorithmica*, 80(2):608–651, 2018.

9 Nachshon Cohen and Zeev Nutov. A (1+ln2)(1+ln2)-approximation algorithm for minimum-cost 2-edge-connectivity augmentation of trees with constant radius. *Theor. Comput. Sci.*, 489-490:67–74, 2013.

10 Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17:449–467, 1965.

11 Guy Even, Jon Feldman, Guy Kortsarz, and Zeev Nutov. A 1.8 approximation algorithm for augmenting edge-connectivity of a graph from 1 to 2. *ACM Trans. Algorithms*, 5(2):21:1–21:17, 2009.

12 Samuel Fiorini, Martin Groß, Jochen Könemann, and Laura Sanità. Approximating weighted tree augmentation via chvátal-gomory cuts. In *SODA*, pages 817–831. SIAM, 2018.

13 Mohit Garg, Fabrizio Grandoni, and Afrouz Jabal Ameli. Improved approximation for two-edge-connectivity. In *SODA (to appear)*, 2023. `arXiv:2209.10265`.

14 Mohit Garg, Felix Hommelsheim, and Nicole Megow. Matching augmentation via simultaneous contractions. *arXiv preprint*, 2022. `arXiv:2211.01912`.

15 Fabrizio Grandoni, Afrouz Jabal Ameli, and Vera Traub. Breaching the 2-approximation barrier for the forest augmentation problem. In *STOC*, pages 1598–1611. ACM, 2022.

16 Fabrizio Grandoni, Christos Kalaitzis, and Rico Zenklusen. Improved approximation for tree augmentation: saving by rewiring. In *STOC*, pages 632–645. ACM, 2018.

17 Christoph Hunkenschröder, Santosh S. Vempala, and Adrian Vetta. A 4/3-approximation algorithm for the minimum 2-edge connected subgraph problem. *ACM Trans. Algorithms*, 15(4):55:1–55:28, 2019.

18 Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Comb.*, 21(1):39–60, 2001.

19 Samir Khuller and Uzi Vishkin. Biconnectivity approximations and graph carvings. *J. ACM*, 41(2):214–235, 1994.

20 Guy Kortsarz and Zeev Nutov. A simplified 1.5-approximation algorithm for augmenting edge-connectivity of a graph from 1 to 2. *ACM Trans. Algorithms*, 12(2):23:1–23:20, 2016.

21 Guy Kortsarz and Zeev Nutov. Lp-relaxations for tree augmentation. *Discret. Appl. Math.*, 239:94–105, 2018.

22 Hiroshi Nagamochi. An approximation for finding a smallest 2-edge-connected subgraph containing a specified spanning tree. *Discret. Appl. Math.*, 126(1):83–113, 2003.

23 Zeev Nutov. On the tree augmentation problem. *Algorithmica*, 83(2):553–575, 2021.

24 András Sebő and Jens Vygen. Shorter tours by nicer ears: 7/5-approximation for the graph-tsp, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. *Comb.*, 34(5):597–629, 2014.

25 Vera Traub and Rico Zenklusen. A better-than-2 approximation for weighted tree augmentation. In *FOCS*, pages 1–12. IEEE, 2021.

26 Vera Traub and Rico Zenklusen. Local search for weighted tree augmentation and steiner tree. In *SODA*, pages 3253–3272. SIAM, 2022.

27 David P. Williamson, Michel X. Goemans, Milena Mihail, and Vijay V. Vazirani. A primal-dual approximation algorithm for generalized steiner network problems. *Comb.*, 15(3):435–454, 1995.

On Differentially Private Counting on Trees

Badih Ghazi
Google, Mountain View, CA, US

Pritish Kamath
Google, Mountain View, CA, US

Ravi Kumar
Google, Mountain View, CA, US

Pasin Manurangsi
Google, Bangkok, Thailand

Kewen Wu
University of California at Berkeley, CA, US

Abstract

We study the problem of performing counting queries at different levels in hierarchical structures while preserving individuals' privacy. Motivated by applications, we propose a new error measure for this problem by considering a combination of multiplicative and additive approximation to the query results. We examine known mechanisms in differential privacy (DP) and prove their optimality, under this measure, in the pure-DP setting. In the approximate-DP setting, we design new algorithms achieving significant improvements over known ones.

2012 ACM Subject Classification Theory of computation → Theory of database privacy and security

Keywords and phrases Differential Privacy, Algorithms, Trees, Hierarchies

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.66

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2212.11967`

Funding *Kewen Wu*: Most of this work was done while at Google.

Acknowledgements KW wants to thank Xin Lyu for helpful references on the sparse vector technique. We thank anonymous ITCS'23 and ICALP'23 reviewers for helpful feedback.

1 Introduction

With the increasing need to preserve the privacy of users, differential privacy (DP) [18, 17] has emerged as a widely popular notion that provides strong guarantees on user privacy and satisfies compelling mathematical properties. There have been many deployments of DP in the field of data analytics both in industry [4, 14] and by government agencies [3].

We start by recalling the formal definition of DP, tailored to our setting.

▶ **Definition 1** (Differential Privacy). *Let $\mathcal{A}$ be a randomized algorithm taking an integer vector as input. We say $\mathcal{A}$ is* (ε, δ)-differentially private *(i.e., (ε, δ)-DP) if*

$$\mathbf{Pr}\left[\mathcal{A}(\boldsymbol{x}) \in S\right] \leq e^{\varepsilon} \cdot \mathbf{Pr}\left[\mathcal{A}(\boldsymbol{x}') \in S\right] + \delta,$$

holds for any measurable subset S of $\mathcal{A}$'s range and any two neighboring inputs $\boldsymbol{x}, \boldsymbol{x}'$, where $\boldsymbol{x}, \boldsymbol{x}'$ are considered neighbors *iff $\|\boldsymbol{x} - \boldsymbol{x}'\|_1 = 1$.*

When $\delta = 0$, we say $\mathcal{A}$ is ε-DP *(aka* pure-DP*); the case $\delta > 0$ is* approximate-DP.

Estimating Counts in Trees

A fundamental task in data analytics is to aggregate counts over hierarchical subsets (specifically, trees) of the input points. For example, the government might be interested in the number of households, aggregated at the state, country, and city levels. As another example, online advertisers might be interested in the number of user clicks on product ads, when there is a category hierarchy on the products. The tree aggregation problem has been the subject of several previous works in DP including in the context of range queries [13, 42, 21, 43], the continuous release model [20, 10], private machine learning [31, 30], and the US census top-down algorithms [2, 1, 12, 11], to name a few[1]. In this work, we revisit this basic problem and present new perspectives and results.

Let $\mathcal{T}$ be a rooted tree of depth[2] d and arity k; the structure of $\mathcal{T}$ is known a priori. Let $\mathsf{nodes}(\mathcal{T})$ be the set of *nodes* and $\mathsf{leaves}(\mathcal{T})$ be the set of *leaves* in $\mathcal{T}$. The problem of private aggregation in trees can be formalized as follows.

▶ **Problem 2** (Tree Aggregation). *Given a tree $\mathcal{T}$, the input to the problem is a vector $\boldsymbol{x} \in \mathbb{N}^{\mathsf{leaves}(\mathcal{T})}$, where $x_v \in \mathbb{N}$ is a value for $v \in \mathsf{leaves}(\mathcal{T})$. For each node $u \in \mathcal{T}$, define its weight w_u by*

$$w_u = \sum_{v \text{ is a leaf under } u} x_v.$$

The desired output is a DP estimate vector $\widetilde{\boldsymbol{w}} \in \mathbb{R}^{\mathsf{nodes}(\mathcal{T})}$ of $\boldsymbol{w}$.

In the above formulation, the input x_v represents the number of individuals that contribute to the leaf v, and the weight w_u counts all the number of individuals that contribute to any of its descendants (or itself). As before, $\boldsymbol{x}, \boldsymbol{x}'$ are neighbors iff $\|\boldsymbol{x} - \boldsymbol{x}'\|_1 = 1$.

Besides being a natural problem on its own, algorithms for tree aggregation also serve as subroutines for solving other problems such as range queries [13, 42, 21, 43].

Linear Queries and Error Measure

Tree aggregation in fact belongs to a class of problems called *linear queries* – one of the most widely studied problems in DP (see, e.g., [15, 19, 28, 5, 39, 9, 37, 6, 24, 38]). In its most general form, the problem can be stated as follows.

▶ **Problem 3** (Linear Queries). *For a given workload matrix $\boldsymbol{W} \in \mathbb{R}^{m \times n}$, the input to the $\boldsymbol{W}$-*linear query *problem is a vector $\boldsymbol{x} \in \mathbb{N}^n$ and the output is a DP estimate of $\boldsymbol{W}\boldsymbol{x}$.*

It is easy to see that the tree aggregation problem can be viewed as a linear query problem, where the binary workload matrix $\boldsymbol{W}^{\mathcal{T}} \in \{0,1\}^{\mathsf{nodes}(\mathcal{T}) \times \mathsf{leaves}(\mathcal{T})}$ encodes if each leaf (corresponding to a column index) is a descendant of (or itself) each node (corresponding to a row index).

Two error measures have been studied in the literature: the (expected) ℓ_2^2*-error*[3]

$$\ell_2^2\text{-}\mathsf{error}(\mathcal{M}; \boldsymbol{W}) := \max_{\boldsymbol{x} \in \mathbb{N}^n} \sqrt{\frac{1}{m} \mathbb{E}\left[\|\mathcal{M}(\boldsymbol{x}) - \boldsymbol{W}\boldsymbol{x}\|_2^2\right]},$$

[1] We remark that there is a reduction from our problem to that of releasing thresholds, which we discuss in more detail in Section 1.3.

[2] The *depth* is defined to be the maximum number of nodes along a root-to-leaf path of the tree.

[3] In some previous work, $\ell_2^2\text{-}\mathsf{error}(\mathcal{M}; \boldsymbol{W})$ is defined as $\max_{\boldsymbol{x} \in \mathbb{N}^n} \frac{1}{m} \mathbb{E}\left[\|\mathcal{M}(\boldsymbol{x}) - \boldsymbol{W}\boldsymbol{x}\|_2^2\right]$ (without the square root). We use the current version as it is more convenient to deal with in our error analysis. In any case, we can obviously convert a bound in one version to the other.

and the (expected) ℓ_∞-*error*

$$\ell_\infty\text{-error}(\mathcal{M}; \boldsymbol{W}) := \max_{\boldsymbol{x} \in \mathbb{N}^n} \mathbb{E}\left[\|\mathcal{M}(\boldsymbol{x}) - \boldsymbol{W}\boldsymbol{x}\|_\infty\right],$$

where $\mathcal{M}$ is a DP mechanism for the $\boldsymbol{W}$-linear query problem. Indeed, previous works have characterized the best possible errors in the approximate-DP case up to polylogarithmic factors for any given workload $\boldsymbol{W}$. (See the discussion in [24] for more details.)

It is worth noting that these measures focus only on the *additive error* of the query, i.e., $\mathcal{M}(\boldsymbol{x}) - \boldsymbol{W}\boldsymbol{x}$. In many scenarios, however, this is not the only possible measure of error. Specifically, in this work, we seek to expand the error measure by additionally incorporating *multiplicative error*. Intuitively, multiplicative errors are meaningful when the true answer (i.e., $(\boldsymbol{W}\boldsymbol{x})_i$) is quite large; e.g., if the true error is 10^6, then we should not be distinguishing whether the additive error is 10 or 100 as both of them are very small compared to 10^6. In addition to this intuition, multiplicative errors have also been used in other contexts such as in empirical evaluations of range queries (e.g., [13, 40, 43]).

With the above discussion in mind, we now proceed to define the error measure.

▶ **Definition 4** (Multiplicative Root Mean Squared Error). *Given parameter $\alpha > 0$, we define an* α-multiplicative root mean squared error (α-RMSE) *of an estimate $\widetilde{z}$ of the true answer $z \geq 0$ as*

$$\mathsf{RMSE}_\alpha(\widetilde{z}, z) := \sqrt{\mathop{\mathbb{E}}_{\widetilde{z}}\left[\left(\max\left\{|\widetilde{z} - z| - \alpha \cdot z, 0\right\}\right)^2\right]}.$$

For $\boldsymbol{W}$-linear query, we define an α-multiplicative maximum root mean squared error (α-mRMSE) *of a mechanism $\mathcal{M}$ to be*

$$\mathsf{mRMSE}_\alpha(\mathcal{M}; \boldsymbol{W}) := \max_{\boldsymbol{x} \in \mathbb{Z}^n} \max_{i \in [m]} \mathsf{RMSE}_\alpha\left(\mathcal{M}(\boldsymbol{x})_i, (\boldsymbol{W}\boldsymbol{x})_i\right).$$

Note that when $\alpha = 0$ (i.e., the error is only additive), our notion of α-RMSE coincides with that of the standard RMSE. By taking the maximum error across all queries when defining the error for linear queries, we mitigate the weakness of ℓ_2^2-error bound, which allows some queries to incur huge errors, while still avoiding the "union bound issue" faced in the ℓ_∞-error. The latter can be significant as the number of queries here can be exponential in the depth d.

We remark that our algorithms also achieve the usual with high probability guarantees, i.e., with probability at most η, $|\widetilde{z} - z| \leq \alpha \cdot \max\{z, \tau\}$ for some threshold τ. We defer such a statement to later sections for simplicity of comparing the bounds. Furthermore, our error notion implies upper bounds on "smoothed relative errors" used for empirical evaluations in previous works [40, 43]. We provide a formal statement in the full version.

When $\alpha = 0$, we drop the "α-multiplicative" or "α-" prefixes and refer to the errors simply as maximum RMSE or mRMSE. Similarly, we also drop α from the subscript and simply write mRMSE instead of mRMSE_0.

1.1 Our Results

Two known baselines for tree aggregation are the ε-DP Laplace mechanism and (ε, δ)-DP Gaussian mechanism, which achieve mRMSE of $O(d/\varepsilon)$ and $O(\sqrt{d\log(1/\delta)}/\varepsilon)$ respectively. We start by showing that these are already tight for the additive-only errors:

▶ **Theorem 5** (Informal; see Theorem 28). *There is no ε-DP algorithm for tree aggregation with* mRMSE $o(d/\varepsilon)$*, even for binary trees.*

Table 1 Overview of results; entries indicate upper/lower bounds on α-mRMSE. Upper bounds corresponding to Laplace and Gaussian mechanisms are formally stated in Corollary 15. For simplicity we omit the dependence on α in the additive-multiplicative bounds here.

Type of error	ε-DP	(ε, δ)-DP
Additive-only ($\alpha = 0$)	$O(d/\varepsilon)$: Laplace	$O_{\varepsilon,\delta}(\sqrt{d})$: Gaussian
	$\Omega(d/\varepsilon)$: Theorem 28	$\Omega_{\varepsilon,\delta}(\sqrt{d})$: Theorem 29
Additive-Multiplicative ($0 < \alpha < 1$)	$\Omega(d/\varepsilon)$: Theorem 28	$O_{\varepsilon,\delta}(\log d)$: Theorem 21

▶ **Theorem 6** (Informal; see Theorem 29). *There is no (ε, δ)-DP algorithm for tree aggregation with* mRMSE $o_{\varepsilon,\delta}(\sqrt{d})$*, even for binary trees.*

Given the above results, it is therefore natural to ask whether multiplicative errors can help reduce the error bound. For pure-DP, we show that this unfortunately is not the case.

▶ **Theorem 7** (Informal; see Theorem 28). *For any constant $\alpha < 1$, there is no ε-DP algorithm for tree aggregation with α-*mRMSE *$o(d/\varepsilon)$, even for binary trees.*

Our next – and perhaps the most surprising – result is that, unlike in the pure-DP case, allowing multiplicative approximation in approximate-DP allows us to reduce the upper bounds exponentially from $O_{\varepsilon,\delta}(\sqrt{d})$ to $O_{\varepsilon,\delta}(\log d)$:

▶ **Theorem 8** (Informal; see Theorem 21). *For any constant $\alpha > 0$, there is an efficient (ε, δ)-DP algorithm for tree aggregation with α-*mRMSE *$O(\log(d/\delta)/\varepsilon)$.*

We remark that Theorem 8 has worse dependency on δ than the (ε, δ)-DP Gaussian mechanism. Indeed, the former has $\log(1/\delta)$ whereas the latter only has $\sqrt{\log(1/\delta)}$. However this gap is somewhat unavoidable as we will discuss in Remark 33 when δ is small, say, $\delta = 2^{-\Omega(d)}$. Our results are summarized in Table 1.

Our bounds do *not* depend on the arity k of $\mathcal{T}$. This is immediate for the lower bounds (Theorems 5–7) since it suffices to prove it for binary trees $k = 2$. The reason for the upper bounds (Theorem 8) is less clear, relying on the fact that the weights of two nodes are correlated iff they are on the same root-to-leaf path, which is irrelevant of the arity.

1.2 Proof Overview

Probabilistic Utility Guarantee

Recall that we defined the error α-mRMSE as a variant of RMSE but with a multiplicative error subtracted out. While this gives us a nice scalar quantity (once α is fixed) to work with and state the results, it will be useful in the subsequent analyses to define a probabilistic version of the guarantee with additional fixed thresholds.

In this different utility guarantee (formalized in (1)), every node u is given an additional threshold τ_u, and a randomized output $\widetilde{\boldsymbol{w}}$ is accurate if for all $u \in \mathsf{nodes}(\mathcal{T})$ we have with probability at least $1 - \eta$ that

$$|\widetilde{w}_u - w_u| \leq \alpha \cdot \max\{w_u, \tau_u\}.$$

The smaller the thresholds τ_u's are, the better the accuracy will be.

The benefit of having τ_u is that it is independent of w_u and is explicitly available to the algorithms; this formulation may be of independent interest. On the other hand, this probabilistic guarantee is closely related to the original α-mRMSE in Definition 4:

1. Any algorithm with low α-mRMSE has good probabilistic guarantee (Lemma 19).
2. Any algorithm with good probabilistic guarantee can be converted into one with low α-mRMSE (Lemma 20).

Improved Approximate-DP Algorithm

Given Item 2, it suffices to design an algorithm with a good probabilistic guarantee, i.e., works for thresholds τ_u's as small as possible. Our algorithm can be decomposed into two parts: a reduction step and a classification algorithm.

Reduction. Since the error measure is multiplicative when w_u is large, we design a geometric sequence of thresholds and classify each w_u into the correct interval created by the thresholds. To do so, every time we use a classification algorithm to find nodes that are above the current threshold, and in the next round we only focus on the ones below the threshold. Assuming previous classifications are all correct, the weights of the nodes above the current threshold are actually below the previous threshold. Such a sandwiching relation provides a good approximation if the granularity of the thresholds is not too large compared with the ratio α (Lemma 26).

Moreover, we assign privacy and error parameters in the same geometric fashion, thus their telescoping sum (from composition theorems) converges.

Classification. Given any fixed threshold τ, the goal is to correctly classify each w_u to be either *above* or *below* τ. Naively, to ensure every node is correctly classified, we need to apply a union bound over all the nodes. This will incur a $\mathsf{poly}(d)$ overhead if we use Laplace noise or Gaussian noise as in the standard mechanisms (Lemmas 13 and 14) since the tree can have exponential size. To deal with this issue, we use a *truncated Laplace mechanism* where the Laplace noise is truncated to be bounded (Lemma 16); this ensures that the estimation error is always at most the truncation range and thus can obviate a union bound. However, we still need to pay the privacy loss from compositions. If one node is the ancestor of another, then the input leaves they depend on must overlap, which means simply estimating every node's weight will incur d rounds of composition. To improve this, our classification algorithm will find the transition nodes in the tree: a *transition node* is one whose weight exceeds τ but none of its children has weight above τ. Given the locations of the transition nodes, we can easily classify the other nodes: a node is above τ iff it is the ancestor of (or itself) a transition node. Assuming the previous classification with threshold $\tau' > \tau$ succeeds, none of the nodes' weights should exceed τ' now. Since there are at most τ'/τ transition nodes in a tree, we can use the *sparse vector technique* [22] to find them, and incur fewer rounds of composition.

We remark that the idea of using increasing thresholds and sparse vector techniques has been used in [7, 26, 25].

Pure-DP Lower Bounds

Given Item 1, it suffices to rule out DP algorithms with very strong probabilistic guarantees, i.e., works for thresholds τ_u's that are too small.

Our proof uses the *packing argument* [28]: we construct extremal datasets where any two datasets have a large distance. Then if the output has small error, we can correctly identify the input dataset. On the other hand, by the privacy guarantee, the output distribution for different datasets, though having a large distance, should not be too different, contradicting the fact that they decode to different input datasets.

Not surprisingly, our extremal datasets place the maximal value on a leaf and keep other leaves empty. But the key issue is the decoding step. Indeed, previous packing arguments work with ℓ_∞-error, where the error on *all* output coordinates is small with high probability, thus admitting simple decoding algorithms. We, however, can only guarantee the error on any *fixed* node is small with high probability; we also cannot use a union bound since the output size can be exponential.

The way we circumvent this is by designing a novel probabilistic decoding algorithm where we will correctly decode to the input dataset with probability large enough to derive a contradiction. The decoding algorithm itself performs a random walk on the tree where each step favors the larger estimated weight. Then the success probability of decoding can be lower bounded in terms of the number of correctly classified nodes, which in turn can be lower bounded by its expectation and our probabilistic guarantee suffices.

Additive-Only Lower Bounds

The above packing argument only works for pure-DP setting (or (ε, δ)-DP but with exponentially small δ). Indeed, as shown by our improved approximate-DP algorithm (Theorem 8), the bound can be exponentially small if we allow both approximate-DP and $\alpha > 0$. Therefore we now turn to the only remaining case: approximate-DP and $\alpha = 0$. In this case, by Definition 4, α-mRMSE is an additive-only error.

Our proof starts by slightly modifying the error characterization of linear queries from [24]. This shows that mRMSE for $\boldsymbol{W}$-linear query is characterized by a factorization norm of $\boldsymbol{W}$. Thus proving Theorem 6 boils down to showing a lower bound on this factorization norm of the binary tree matrix. Following previous works on range queries (e.g., [36]), we do so by invoking a dual (maximum-based) characterization of the factorization norm from [35, 33] and give an explicit solution to this dual formulation.

1.3 Relation Between Tree Aggregation and Releasing Thresholds

There is a simple reduction from the tree aggregation problem (Problem 18) to the problem of releasing thresholds. Recall that the problem of releasing thresholds is the same as linear queries with the workload matrix $\boldsymbol{W}^{\mathrm{th}} \in \{0,1\}^{m \times m}$ being the matrix with upper-triangular entries (including the diagonal entries) equal to one.

The reduction works as follows. First, we may index the leaves from $1, \ldots, |\mathsf{leaves}(\mathcal{T})|$ based, say, on their order in the DFS traversal of the tree. It is not hard to see that w_u of any node in the tree corresponds to $(\boldsymbol{W}^{\mathrm{th}}\boldsymbol{x})_b - (\boldsymbol{W}^{\mathrm{th}}\boldsymbol{x})_{a-1} = \sum_{v \in [a,b]} x_v$ for some $a, b \in \{1, \ldots, |\mathsf{leaves}(\mathcal{T})|\}$. Therefore, we may run any DP threshold releasing algorithm (with $m = |\mathsf{leaves}(\mathcal{T})|$) and solve the tree aggregation problem.

The above reduction yields an mRMSE error for the tree aggregation problem that is of the same order as that of releasing thresholds (with $m = |\mathsf{leaves}(\mathcal{T})|$). For the latter, it is known that the tight error is $\Theta_{\varepsilon,\delta}(\log m)$ [29][4]. Assuming that each node at depth less than d has at least two children, $|\mathsf{leaves}(\mathcal{T})| \geq 2^d$ and therefore, the error yielded by this reduction is at least $\Omega_{\varepsilon,\delta}(d)$. In other words, this is not even as good as the straightforward Gaussian mechanism for our problem, which yields an error of $O_{\varepsilon,\delta}(\sqrt{d})$ (and we have shown this to be tight in Theorem 29).

We note that there is another line of research that studies privately learning threshold functions. Recent work has shown that learning threshold functions with (ε, δ)-DP in the PAC model only requires $O_{\alpha,\varepsilon,\delta}\left((\log^* m)^{O(1)}\right)$ samples [32]. Due to the connection

[4] In fact, the lower bound in [29] is even stronger as it holds against the ℓ_2^2-error.

between learning thresholds and releasing threshold functions presented in [8], this gives an algorithm for the latter with an error bound of $O_{\varepsilon,\delta}\left((\log^* m)^{O(1)} \cdot (\log n)^{2.5}\right)$, where n denotes $\|\boldsymbol{x}\|_1$ (i.e., the total count across all leaves in our setting). When combining this bound with the above reduction, one gets a tree aggregation algorithm with error $O_{\varepsilon,\delta}\left((\log^* |\mathsf{leaves}(\mathcal{T})|)^{O(1)} \cdot (\log n)^{2.5}\right)$. Although the term $(\log^* |\mathsf{leaves}(\mathcal{T})|)^{O(1)}$ is very small, this error bound is not directly comparable to the lower and upper bounds achieved in our paper because our bounds are *independent* of n whereas there is a dependency of $(\log n)^{2.5}$ in this releasing threshold-based bound. (Note also that the dependency on n cannot be removed while keeping the dependency on m sublogarithmic, as this would contradict with the aforementioned lower bound of [29].)

Finally, we remark that the reduction does *not* work if we allow the threshold releasing algorithm to incur multiplicative errors. This is because we need to subtract two thresholds to get w_u for each node u in the tree, and subtraction does not preserve multiplicative approximation guarantees.

Paper Organization

We formalize the notation in Section 2. Then in Section 3 we introduce the error measure we will actually use in designing algorithms and proving lower bounds; and relate it to the α-mRMSE measure. The improved approximate-DP algorithm is presented in Section 4 and the corresponding lower bounds are in Section 5. The concluding remarks are in Section 6.

2 Preliminaries

We use $\ln(\cdot)$ and $\log(\cdot)$ to denote the logarithm with base e and 2 respectively. For a positive integer n, let $[n]$ denote the set $\{1, \ldots, n\}$. Let $\mathbb{N}$ denote the set of non-negative integers.

2.1 Norms

We use boldface uppercase (e.g., $\boldsymbol{A}$) to denote matrices and boldface lowercase (e.g., $\boldsymbol{x}$) to denote vectors. We use $\mathbf{0}, \mathbf{1}$ to denote the all-zeros and all-ones vectors / matrices.

For $\boldsymbol{x} \in \mathbb{R}^m$, its *$\ell_p$-norm* is defined as $\|\boldsymbol{x}\|_p := \left(\sum_{i\in[m]} |x_i|^p\right)^{1/p}$ for any $1 \le p < \infty$. Its *ℓ_∞-norm* is defined as $\|\boldsymbol{x}\|_\infty := \max_{i\in[m]} |x_i|$.

2.2 Tools from Differential Privacy

Here we note some useful facts regarding differential privacy [18, 17, 41].

▶ **Fact 9** (Post-Processing). *Let $\mathcal{A}_1$ be an (ε, δ)-DP algorithm and $\mathcal{A}_2$ be a (randomized) post-processing algorithm. Then the algorithm $\mathcal{A}(\boldsymbol{x}) = \mathcal{A}_2(\mathcal{A}_1(\boldsymbol{x}))$ is still an (ε, δ)-DP algorithm.*

▶ **Fact 10** (Group Privacy). *Let $\mathcal{A}$ be an (ε, δ)-DP algorithm and $\boldsymbol{x}, \boldsymbol{x}'$ be two arbitrary inputs. Define $k = \|\boldsymbol{x} - \boldsymbol{x}'\|_1$. Then for any measurable subset S of $\mathcal{A}$'s range, we have*

$$\mathbf{Pr}\left[\mathcal{A}(\boldsymbol{x}) \in S\right] \le e^{k\cdot\varepsilon} \cdot \mathbf{Pr}\left[\mathcal{A}(\boldsymbol{x}') \in S\right] + \delta \cdot \frac{e^{k\cdot\varepsilon} - 1}{e^{\varepsilon} - 1}.$$

▶ **Fact 11** (Basic Composition). *Let $\mathcal{A}_1$ be an $(\varepsilon_1, \delta_1)$-DP algorithm and $\mathcal{A}_2$ be an $(\varepsilon_2, \delta_2)$-DP algorithm. Then $\mathcal{A}(\boldsymbol{x}) = (\mathcal{A}_1(\boldsymbol{x}), \mathcal{A}_2(\mathcal{A}_1(\boldsymbol{x}), \boldsymbol{x}))$ is an $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-DP algorithm.*

▶ **Fact 12** (Parallel Composition). *Let $\mathcal{A}_1$ be an $(\varepsilon_1, \delta_1)$-DP algorithm and $\mathcal{A}_2$ be an $(\varepsilon_2, \delta_2)$-DP algorithm. Assume $\mathcal{A}_1$ and $\mathcal{A}_2$ depend on disjoint subsets of input coordinates. Then the algorithm $\mathcal{A}(\boldsymbol{x}) = (\mathcal{A}_1(\boldsymbol{x}), \mathcal{A}_2(\mathcal{A}_1(\boldsymbol{x}), \boldsymbol{x}))$ is a $(\max\{\varepsilon_1, \varepsilon_2\}, \max\{\delta_1, \delta_2\})$-DP algorithm.*

Two of the most ubiquitous mechanisms in DP are the Laplace and Gaussian mechanisms [16, 17, 23]. We use $\mathsf{Lap}(\sigma)$ to denote the *Laplace distribution* with parameter σ, whose density function is $\frac{1}{2\sigma}\exp\left(-\frac{|x|}{\sigma}\right)$. We use $\mathsf{N}(\mu, \sigma^2)$ to denote the *Gaussian distribution* with mean μ and variance σ^2, whose density function is $\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)$.

For matrix $\boldsymbol{A}$ and $p \geq 1$, we use $\|\boldsymbol{A}\|_{\infty,p}$ to denote the maximum ℓ_p-norm among all column vectors of $\boldsymbol{A}$. The Laplace and Gaussian mechanisms for linear queries are stated next.

▶ **Lemma 13** (Laplace Mechanism, [16, 34]). *For the $\boldsymbol{W}$-linear query problem, the algorithm that outputs $\boldsymbol{W}\boldsymbol{x} + \boldsymbol{z}$ is ε-DP, where each entry of $\boldsymbol{z}$ is drawn i.i.d. from $\mathsf{Lap}\left(\|\boldsymbol{W}\|_{\infty,1}/\varepsilon\right)$.*

▶ **Lemma 14** (Gaussian Mechanism, [17, 23]). *Assume $\varepsilon, \delta \in (0,1)$. For the $\boldsymbol{W}$-linear query problem, the algorithm that outputs $\boldsymbol{W}\boldsymbol{x} + \boldsymbol{z}$ is (ε, δ)-DP, where each entry of $\boldsymbol{z}$ is drawn i.i.d. from $\mathsf{N}\left(0, 2\ln(1.25/\delta)\|\boldsymbol{W}\|^2_{\infty,2}/\varepsilon^2\right)$.*

Recall that for a tree $\mathcal{T}$, we let $\boldsymbol{W}^{\mathcal{T}} \in \{0,1\}^{\mathsf{nodes}(\mathcal{T})\times\mathsf{leaves}(\mathcal{T})}$ be the indicator matrix whether a leaf is a descendant of (or itself) a node. This represents the tree aggregation problem as $\boldsymbol{W}^{\mathcal{T}}$-linear queries. Observe also that $\left\|\boldsymbol{W}^{\mathcal{T}}\right\|_{\infty,1} = d$ and $\left\|\boldsymbol{W}^{\mathcal{T}}\right\|_{\infty,2} = \sqrt{d}$. Therefore, we can apply Lemma 13 and Lemma 14 (together with tail bounds for Laplace and Gaussian distributions) to obtain the following baselines for tree aggregation.

▶ **Corollary 15** (Baseline Algorithms). *For the tree aggregation problem, there exists an ε-DP (resp., (ε, δ)-DP) algorithm with mRMSE $O(d/\varepsilon)$ (resp., $O(\sqrt{d \cdot \log(1/\delta)}/\varepsilon)$).*

We will also use the Laplace mechanism with a bounded range. For any $R > 0$, we use $\mathsf{TruncLap}(\sigma, R)$ to denote the *truncated Laplace distribution* with parameter σ and range $[-R, R]$, whose density function is proportional to $\exp(-|x|/\sigma)$ for $x \in [-R, R]$ and is 0 if $|x| > R$. Note that $\mathsf{Lap}(\sigma) = \mathsf{TruncLap}(\sigma, +\infty)$.

▶ **Lemma 16** (Truncated Laplace Mechanism, [27]). *The algorithm that, on input $x \in \mathbb{Z}$, outputs $x + z$ is (ε, δ)-DP, where $z \sim \mathsf{TruncLap}\left(\frac{1}{\varepsilon}, \frac{1}{\varepsilon}\ln\left(1 + \frac{e^{\varepsilon}-1}{2\delta}\right)\right)$.*

Our algorithm will also make use of the celebrated *sparse vector technique* [22]. For convenience, we apply it in a black-box way as the following oracle.

▶ **Lemma 17** (Sparse Vector Technique, [22, 23]). *There exists an ε-DP algorithm $\texttt{Sparse}(\boldsymbol{x}, \{f_i\}; \eta, c, \tau, \varepsilon)$ such that:*

- *Input. A dataset $\boldsymbol{x}$, an adaptively chosen stream $\{f_i\}_{i=1,\ldots,d}$ of sensitivity-1 queries, error probability $\eta > 0$, a cutoff point $c \geq 1$, a threshold τ, and a privacy bound $\varepsilon > 0$.*[5]
- *Output. A stream $\{a_i\}_{i=1,\ldots,d} \in \{\bot, \top\}^*$ of on-the-fly answers.*
- *Accuracy. Let i^* be the index of the cth $\top$ in $\{a_i\}_{i\in[d]}$; if there are less than c $\top$'s, let $i^* = d$. Then, for $\Delta = 8c/\varepsilon \cdot \ln(2d/\eta)$, with probability at least $1-\eta$ the following holds for all $i \leq i^*$: If $a_i = \top$, then $f_i(\boldsymbol{x}) \geq \tau - \Delta$; otherwise (i.e., $a_i = \bot$) $f_i(\boldsymbol{x}) < \tau + \Delta$.*

The $\texttt{Sparse()}$ algorithm is in [23, Algorithm 2] with $\delta = 0$, where its privacy is proved in [23, Theorem 3.25]. The accuracy part is also immediate from the algorithm description. For completeness, we give a proof in the full version.

[5] We say a query f is *sensitivity-1* if $\left|f(\boldsymbol{x}) - f(\boldsymbol{x}')\right| \leq 1$ for any two neighboring inputs $\boldsymbol{x}, \boldsymbol{x}'$.

3 Threshold-Based Utility

As mentioned in Section 1.2, we consider a probabilistic utility guarantee with a threshold value supplied at each node in the tree. Then we relate it to the original α-mRMSE notion.

▶ **Problem 18** (Tree Aggregation with Thresholds). *Consider the tree aggregation problem (i.e., Problem 2), wherein additionally, we have a threshold $\tau_u \geq 0$ corresponding to each node $u \in \mathsf{nodes}(\mathcal{T})$ (both internal nodes and leaves). The desired output for the problem is an estimate $\widetilde{\boldsymbol{w}} \in \mathbb{R}^{\mathsf{nodes}(\mathcal{T})}$ of $\boldsymbol{w}$ as before.*

*For parameters $\eta, \alpha \in [0, 1)$, we say that an algorithm for tree aggregation is (α, η)-*accurate *(w.r.t. the given thresholds) if its output vector $\widetilde{\boldsymbol{w}} \in \mathbb{R}^{\mathsf{nodes}(\mathcal{T})}$ satisfies the following:*

$$\text{For all } u \in \mathsf{nodes}(\mathcal{T}) \quad : \quad \mathbf{Pr}\left[|\widetilde{w}_u - w_u| \leq \alpha \cdot \max\{w_u, \tau_u\}\right] \geq 1 - \eta. \tag{1}$$

Unless otherwise specified, for the rest of the paper, we assume that the input to the tree aggregation problem includes a threshold at each node in the tree. Let $\tau_{\mathsf{min}} := \min_{u \in \mathsf{nodes}(\mathcal{T})} \tau_u$ and $\tau_{\mathsf{max}} := \max_{u \in \mathsf{nodes}(\mathcal{T})} \tau_u$. Now, we show that any algorithm with low α-mRMSE also yields a certain utility guarantee in the sense of (1).

▶ **Lemma 19** (Proof in Full Version). *For any $\alpha' > \alpha \geq 0$, $\eta > 0$, any tree aggregation algorithm with α-mRMSE at most $(\alpha' - \alpha)\sqrt{\eta} \cdot \tau_{min}$ is also (α', η)-accurate.*

In contrast to the above bound, our algorithms will satisfy much stronger exponential tail bounds, which will be clear in the next section. To complement Lemma 19, we show that any algorithm that is (α, η)-accurate, as per (1), can be made having small α-mRMSE.

▶ **Lemma 20** (Proof in Full Version). *If there is an $(\varepsilon/2, \delta/2)$-DP algorithm that is (α, η)-accurate for tree aggregation, then there is an (ε, δ)-DP algorithm for tree aggregation with α-mRMSE at most $O\left(\alpha \cdot \tau_{max} + d\sqrt{\eta} \cdot \left(1 + \frac{1}{\varepsilon}\log\left(\frac{1}{\delta}\right)\right)\right)$.*

With the above two lemmas in mind, it essentially suffices for us to consider the accuracy notion in (1), which will be convenient for the rest of the paper.

4 Upper Bounds

In this section, we present a new algorithm for tree aggregation in the approximate-DP setting, achieving a significant improvement over the baseline algorithm (i.e., Corollary 15).

▶ **Theorem 21.** *Let $\alpha > 0$ be a parameter. For any $\varepsilon > 0$ and $\delta \in (0, 1)$, there is an (ε, δ)-DP algorithm for tree aggregation with α-mRMSE at most $O\left(\frac{1}{\alpha^3} \cdot \left(\frac{1}{\varepsilon}\log\left(\frac{2d}{\delta}\right) + 1\right)\right)$.*

By Lemma 20, it suffices to design an efficient algorithm for Problem 18 with small thresholds for every node in the tree. This will be the focus of the section. To this end, we will reduce the estimation problem to a classification problem, and design algorithms for the classification task. We first present the classification algorithm in Section 4.1, then describe the reduction in Section 4.2, and finally put them together in Section 4.3.

4.1 A Classification Problem

For later reduction, the classification task here needs to have a stronger notion of success: the classification for nodes should be correct *simultaneously* with high probability, whereas Problem 18 only requires the estimation of any fixed node to be correct with high probability.

▶ **Problem 22.** *Let $\mathcal{T}$ be a tree of depth d and arity k. Let $\tau \geq 0$ be the (common) threshold for all nodes. Let $M > 0, \eta \in [0, 1/2), \alpha \in [0, 1)$ be parameters.*

The input to the problem is non-negative integer values x_v for each $v \in \mathsf{leaves}(\mathcal{T})$. For each node u, its weight w_u is $w_u = \sum_{v \text{ is a leaf under } u} x_v$.

The desired output is a vector $\boldsymbol{w}' \in \{\bot, \top\}^{\mathsf{nodes}(\mathcal{T})}$ that with probability at least $1 - \eta$ satisfies the following: when the weight of the root of $\mathcal{T}$ is at most M, for each $u \in \mathsf{nodes}(\mathcal{T})$,

- *if $w_u \geq (1 + \alpha) \cdot \tau$, then $w'_u = \top$;*
- *if $w_u < (1 - \alpha) \cdot \tau$, then $w'_u = \bot$;*
- *otherwise (i.e., $(1 - \alpha) \cdot \tau \leq w_u < (1 + \alpha) \cdot \tau$), w'_u can be arbitrary.*

We now present our algorithm for this classification problem and its guarantees.

▶ **Lemma 23.** *There is an (ε, δ)-DP algorithm $\texttt{Classification}(\mathcal{T}; M, \eta, \alpha, \tau, \varepsilon, \delta)$ such that it solves Problem 22 assuming $\tau \geq \sqrt{\frac{2M}{\alpha\varepsilon}} \cdot \max\left\{\sqrt{48 \ln\left(\frac{2d}{\eta}\right)}, \sqrt{6 \ln\left(1 + \frac{e^{\varepsilon/2} - 1}{\delta}\right)}\right\}$.*

Proof. Without loss of generality we assume $\alpha \leq 1/2$. Note that if $M < \tau$, then we can simply set $w'_u \leftarrow \bot$ for all $u \in \mathsf{nodes}(\mathcal{T})$. Therefore we assume without loss of generality $M \geq \tau$ from now on. For any node u, let $\mathsf{depth}(u)$ denote the number of nodes on the path from the root to u. Algorithm 1 contains the formal description.

We first prove the privacy bound. By Lemma 17, Line 7 is $\varepsilon/2$-DP. On the other hand, by Lemma 16 and Fact 12, Lines 11–18 are $(\varepsilon/(2c), \delta/c)$-DP; and since they are executed at most c times, they are $(\varepsilon/2, \delta)$-DP in total. Therefore by Fact 11, Algorithm 1 is (ε, δ)-DP.

Now we turn to the correctness of Algorithm 1. Define i^* to be the index of the cth $\top$ in $a_d, a_{d-1}, \ldots$. If there are less than c $\top$'s, let $i^* \geq 1$ be the index of the last query. Define $\mathcal{E}$ to be the event that the following holds for any $i \geq i^*$:[6] If $a_i = \top$, then $f_i \geq \tau - \Delta$; and if $a_i = \bot$, then $f_i < \tau + \Delta$. By Lemma 17, $\mathbf{Pr}[\mathcal{E}] \geq 1 - \eta$.

We first show, conditioned on $\mathcal{E}$, there are always less than c $\top$'s. Assume towards contradiction that there are c $\top$'s. Then, conditioned on $\mathcal{E}$, for any $a_i = \top$, there exists some $u \in \mathcal{S}_i$ such that $w_u = f_i \geq \tau - \Delta$. Therefore $\widetilde{w}_u \geq w_u - R \geq \tau - \Delta - R$, which implies it will be assigned to $\top$ on Line 14. By design, all these u's satisfying Line 13 form a subset of $\mathcal{T}$ where none of them is an ancestor of another. Therefore the weight of the root is lower bounded by the total weights of these nodes, which is at least $c \cdot (\tau - \Delta)$ but at most M. On the other hand, by the assumption on M and assuming $\Delta < \alpha \cdot \tau$, we also have $c > \frac{M}{\tau - \Delta}$, which gives a contradiction.

Then for the correctness part, it suffices to show for any fixed node $v \in \mathsf{nodes}(\mathcal{T})$, we have $w'_v = \top$ if $w_v \geq (1 + \alpha) \cdot \tau$, and $w'_v = \bot$ if $w_v < (1 - \alpha) \cdot \tau$:

- Case $w_v \geq (1 + \alpha) \cdot \tau$. Assume towards contradiction that $w'_v = \bot$. Let $i_v = \mathsf{depth}(v)$. Then it means when $i = i_v$ on Line 4, $v \in \mathcal{S}_i$. Thus $f_i \geq w_v \geq (1 + \alpha) \cdot \tau$. On the other hand since $w'_v = \bot$, we must proceed to Line 9. Conditioned on $\mathcal{E}$, this implies $f_i < \tau + \Delta$, which is a contradiction assuming $\Delta \leq \alpha \cdot \tau$.
- Case $w_v < (1 - \alpha) \cdot \tau$. Assume towards contradiction that $w'_v = \top$. Let $r \in \mathsf{nodes}(\mathcal{T})$ be the deepest node in the subtree below v that is assigned $\top$. Let $i_r = \mathsf{depth}(r)$. Then it means when $i = i_r$ we execute Line 14 for r. Thus $w_r + R \geq \widetilde{w}_r \geq \tau - \Delta - R$. Meanwhile, we also have $w_r \leq w_v < (1 - \alpha) \cdot \tau$, which is a contradiction assuming $\Delta + 2R \leq \alpha \cdot \tau$.

[6] Note that i goes from d down to 1 in our algorithm.

Algorithm 1 Classification.

Input: $\mathcal{T}$ and parameters $M, \eta, \alpha, \tau, \varepsilon, \delta$ described in Problem 22
Output: $w'_u \in \{\bot, \top\}$ for all $u \in \mathsf{nodes}(\mathcal{T})$

if $M < \tau$ **then** set $w'_u \leftarrow \bot$ for all $u \in \mathsf{nodes}(\mathcal{T})$ and **return** $\boldsymbol{w}'$
Set $c \leftarrow \frac{M}{(1-\alpha)\tau}$ and

$$\Delta \leftarrow \frac{16c}{\varepsilon} \ln\left(\frac{2d}{\eta}\right), \quad R \leftarrow \frac{2c}{\varepsilon} \ln\left(1 + \frac{c \cdot (e^{\varepsilon/(2c)} - 1)}{\delta}\right)$$

Define $\mathcal{S}_i \leftarrow \{u \in \mathsf{nodes}(\mathcal{T}) \mid \mathsf{depth}(u) = i\}$ for each $i \in [d]$
foreach $i = d$ **to** 1 **do**
  if $\mathcal{S}_i = \emptyset$ **then continue**
  Define query $f_i \leftarrow \max_{u \in \mathcal{S}_i} w_u$
  Get $a_i \leftarrow \mathtt{Sparse}(\boldsymbol{x}, f_i; \eta, c, \tau, \varepsilon/2)$ `/* x is the values of leaves(T) */`
  if $a_i = \bot$ **then**
    Set $w'_u \leftarrow \bot$ for all $u \in \mathcal{S}_i$ and update $\mathcal{S}_i \leftarrow \emptyset$
  else `/*` $a_i = \top$ `*/`
    foreach $u \in \mathcal{S}_i$ **do**
      Compute $\widetilde{w}_u \leftarrow w_u + \mathsf{TruncLap}(2c/\varepsilon, R)$
      if $\widetilde{w}_u \geq \tau - \Delta - R$ **then**
        Set $w'_v \leftarrow \top$ and remove v from $\mathcal{S}_{i_v}$ for each u's ancestor v (including u itself) where $i_v = \mathsf{depth}(v)$.
      else `/*` $\widetilde{w}_u < \tau - \Delta - R$ `*/`
        Set $w'_u \leftarrow \bot$ and remove u from $\mathcal{S}_i$
      end
    end
  end
end
return $\boldsymbol{w}'$

Thus it suffices to make sure $\Delta, R \leq \alpha \cdot \tau/3$. Since $M \geq \tau$, we have $c \geq 1$ and $c \cdot \left(e^{\varepsilon/(2c)} - 1\right) \leq e^{\varepsilon/2} - 1$, which gives the assumption in the statement by rearranging terms and noticing $1 - \alpha \geq 1/2$. ◀

Eventually, we will use $\mathtt{Classification}(\mathcal{F}; M, \eta, \alpha, \tau, \varepsilon, \delta)$ algorithm on a forest $\mathcal{F}$ of disjoint trees with the same set of parameters. There we do not need all nodes in $\mathcal{F}$ to be classified correctly. Instead, it suffices to have all nodes in any $\mathcal{T} \in \mathcal{F}$ classified correctly.

▶ **Problem 24.** *Let $\mathcal{F} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots\}$ be a forest of disjoint trees of depth d and arity k. Let $\tau \geq 0$ be the threshold for every node. Let $M > 0, \eta \in [0, 1/2), \alpha \in [0, 1)$ be parameters.*

The input to the problem is non-negative integer values x_v for each leaf v in $\mathcal{F}$. For each node u, its weight w_u is $w_u = \sum_{v \text{ is a leaf under } u} x_v$.

The desired output is a vector $\boldsymbol{w}' \in \{\bot, \top\}^{\mathsf{nodes}(\mathcal{F})}$ that, for any $\mathcal{T} \in \mathcal{F}$ with probability at least $1 - \eta$, satisfies the following: when the root of $\mathcal{T}$ has weight at most M, for each $u \in \mathsf{nodes}(\mathcal{T})$,

- *if $w_u \geq (1 + \alpha) \cdot \tau$, then $w'_u = \top$;*
- *if $w_u < (1 - \alpha) \cdot \tau$, then $w'_u = \bot$;*
- *otherwise (i.e., $(1 - \alpha) \cdot \tau \leq w_u < (1 + \alpha) \cdot \tau$), w'_u can be arbitrary.*

The algorithm for Problem 24 is simply running $\texttt{Classification}(\mathcal{T}; M, \eta, \alpha, \tau, \varepsilon, \delta)$ for each $\mathcal{T} \in \mathcal{F}$. Since the trees are disjoint, the privacy bound follows from Fact 12. Therefore we omit the proof and summarize the following.

▶ **Corollary 25.** *There is an (ε, δ)-DP algorithm $\texttt{Classification}(\mathcal{F}; M, \eta, \alpha, \tau, \varepsilon, \delta)$ such that it solves Problem 24 assuming $\tau \geq \sqrt{\frac{2M}{\alpha\varepsilon}} \cdot \max\left\{\sqrt{48\ln\left(\frac{2d}{\eta}\right)}, \sqrt{6\ln\left(1+\frac{e^{\varepsilon/2}-1}{\delta}\right)}\right\}$.*

4.2 A Reduction from Estimation to Classification

Now we present the reduction algorithm from the estimation problem (i.e., Problem 18) to the classification problem (i.e., Problem 24). The reduction here is given with large flexibility for choosing parameters. Later we will design geometric convergent sequences for simplicity of calculation and derive the final bounds.

▶ **Lemma 26.** *Let $\ell \geq 1$ be an integer. Let M, M_0, and $(M_i, \eta_i, \alpha_i, \tau_i, \varepsilon_i, \delta_i)_{i\in[\ell]}$ be a sequence of parameters. There is an (ε, δ)-DP algorithm $\texttt{Reduction}(\mathcal{T}; \ell, (M_i, \eta_i, \alpha_i, \tau_i, \varepsilon_i, \delta_i)_{i\in[\ell]})$, where $(\varepsilon, \delta) = \left(\sum_{i=1}^{\ell}\varepsilon_i, \sum_{i=1}^{\ell}\delta_i\right)$ such that it solves Problem 18 by carefully combining results from $\texttt{Classification}(\cdot; M_i, \eta_i, \alpha_i, \tau_i, \varepsilon_i, \delta_i)$'s and assuming the weight of the root of $\mathcal{T}$ is at most M and*

$$\tau_i \;\geq\; \sqrt{\tfrac{2M_i}{\alpha_i\varepsilon_i}} \cdot \max\left\{\sqrt{48\ln\left(\tfrac{2d}{\eta_i}\right)}, \sqrt{6\ln\left(1+\tfrac{e^{\varepsilon_i/2}-1}{\delta_i}\right)}\right\} \quad \forall i \in [\ell], \tag{2}$$

$$\eta \;\geq\; \sum_{i=1}^{\ell}\eta_i, \tag{3}$$

$$M_i \;\geq\; (1+\alpha_{i+1})\cdot\tau_{i+1} \quad \forall i = 0, 1, \ldots, \ell-1 \quad \text{and} \quad M_\ell \;\geq\; M, \tag{4}$$

$$(1-\alpha_i)\cdot\tau_i \;\leq\; M_i \;\leq\; (1+\alpha)(1-\alpha_i)\cdot\tau_i \quad \forall i \in [\ell], \tag{5}$$

$$0 \;\leq\; M_0 \;\leq\; \alpha\cdot\tau_{\mathit{min}}. \tag{6}$$

The reduction algorithm is formalized in Algorithm 2 and analyzed in the full version.

Algorithm 2 Reduction.

Input: $\mathcal{T}$ and parameters $\ell, M_0, (M_i, \eta_i, \alpha_i, \tau_i, \varepsilon_i, \delta_i)_{i\in[\ell]}$ described above
Output: $\widetilde{w}_u \in \mathbb{R}$ for all node $u \in \mathsf{nodes}(\mathcal{T})$

Initialize $\mathcal{F}_\ell \leftarrow \{\mathcal{T}\}$
foreach $i = \ell$ **to** 1 **do**
  Initialize $\mathcal{F}_{i-1} \leftarrow \emptyset$
  Compute $\boldsymbol{w}' \leftarrow \texttt{Classification}(\mathcal{F}_i; M_i, \eta_i, \alpha_i, \tau_i, \varepsilon_i, \delta_i)$
  For each node u in $\mathcal{F}_i$, let $\mathcal{T}_u$ be the subtree of u in $\mathcal{F}_i$
  foreach *node u satisfying $w'_u = \top$ and $w'_v = \bot$ for all $v \in \mathsf{nodes}(\mathcal{T}_u)\setminus\{u\}$* **do**
    Set $\widetilde{w}_v \leftarrow M_i$ for each u's ancestor v (including u itself) in $\mathcal{T}$
    Update $\mathcal{F}_{i-1} \leftarrow \mathcal{F}_{i-1} \cup \{\mathcal{T}_1, \mathcal{T}_2, \ldots\}$ where $\mathcal{T}_1, \mathcal{T}_2, \ldots$ are the disjoint trees of $\mathcal{T}_u \setminus \{u\}$
  end
end
Set $\widetilde{w}_v \leftarrow M_0$ for each node v in $\mathcal{F}_0$

4.3 Putting Everything Together

Now we give the algorithm for Problem 18. To this end, we carefully choose parameters and apply Lemma 26, where the required upper bound M is privately estimated with Lemma 16.

▶ **Corollary 27** (Proof in Full Version). *There is an (ε, δ)-DP algorithm* `Estimation`$(\mathcal{T}, \alpha, \varepsilon, \delta, \eta)$ *such that it solves Problem 18 assuming*

$$\tau_{min} \geq \frac{324 \cdot (1+\alpha)^2}{\alpha^4 \cdot \varepsilon} \cdot \max\left\{8 \ln\left(\frac{4d}{\eta}\right), \ln\left(1 + \frac{2 \cdot \left(e^{\varepsilon/4} - 1\right)}{\delta}\right)\right\}.$$

Now we complete the proof of Theorem 21 using Lemma 20 and Corollary 27.

Proof of Theorem 21. We first note that if $\alpha \geq 1$ then the α-mRMSE is trivially zero by outputting the all-zeros vector. Therefore we assume without loss of generality $\alpha \in (0, 1)$.

Let $C > 0$ be a constant to be optimized later. Fix $\eta = d^{-2}$ and $\tau = \frac{C}{\alpha^4} \cdot \left(\frac{1}{\varepsilon} \log\left(\frac{2d}{\delta}\right) + 1\right)$. Let $\varepsilon' = \varepsilon/2$ and $\delta' = \delta/2$. Since $\alpha \in (0, 1)$, $d \geq 1$, and $\delta \in (0, 1]$, we have

$$\tau^* := \frac{324 \cdot (1+\alpha)^2}{\alpha^4 \cdot \varepsilon'} \cdot \max\left\{8 \ln\left(\frac{4d}{\eta}\right), \ln\left(1 + \frac{2 \cdot \left(e^{\varepsilon'/4} - 1\right)}{\delta'}\right)\right\} \leq O\left(\frac{1}{\alpha^4} \cdot \left(\frac{1}{\varepsilon} \log\left(\frac{2d}{\delta}\right) + 1\right)\right).$$

We set C large enough such that $\tau \geq \tau^*$. By Corollary 27, there is an $(\varepsilon', \delta') = (\varepsilon/2, \delta/2)$-DP algorithm for Problem 18 when $\tau_u \equiv \tau$ for all $u \in \mathsf{nodes}(\mathcal{T})$.

The desired α-mRMSE bound now follows from Lemma 20 and the parameters above. ◀

5 Lower Bounds

In this section we prove lower bounds for DP tree aggregation algorithms. In particular, Theorem 28 proves pure-DP lower bounds for all α-mRMSE whenever $\alpha \in [0, 1)$; and Theorem 29 proves approximate-DP lower bounds for additive-only error, i.e., $(\alpha = 0)$-mRMSE.

▶ **Theorem 28** (Pure-DP Lower Bound). *Let $\alpha \in [0, 1)$ be a parameter. For any $\varepsilon > 0$, any ε-DP algorithm for tree aggregation on the complete depth-d binary tree must incur α-mRMSE at least $\Omega\left((1-\alpha)^2 \cdot d/\varepsilon\right)$.*

▶ **Theorem 29** (Approximate-DP Lower Bound for $\alpha = 0$). *For any $\varepsilon > 0$ and any $\delta > 0$ sufficiently small depending on ε, there is a constant $C_{\varepsilon,\delta} > 0$ that any (ε, δ)-DP algorithm for tree aggregation on the complete depth-d binary tree must incur mRMSE at least $C_{\varepsilon,\delta} \cdot \sqrt{d}$.*

Theorem 28 is proved in Section 5.1. The proof of Theorem 29 relies on results from [24] and the factorization norm of the binary tree matrix, which we defer to the full version.

5.1 Pure-DP Lower Bound

To prove Theorem 28, by Lemma 19 it suffices to rule out DP algorithms for Problem 18 with small thresholds. Since Problem 18 is interesting on its own, we will present its lower bound in the approximate-DP setting for full generality.

▶ **Lemma 30.** *Assume $\mathcal{T}$ in Problem 18 is a complete binary tree of depth d. Let $D = 2 \cdot \lceil \tau_{max}/(1-\alpha) \rceil$. If $\mathcal{A}$ is an (ε, δ)-DP algorithm for Problem 18 and suppose $\eta \leq 1/8$ and*

$$\delta \cdot \frac{e^{\varepsilon \cdot D} - 1}{e^{\varepsilon} - 1} \leq \frac{1}{8} \cdot 2^{-(d-1) \cdot \mathcal{H}(4\eta)}, \tag{7}$$

then

$$\tau_{max} = \Omega\left((1-\alpha)\cdot(d-3-(d-1)\cdot\mathcal{H}(4\eta))/\varepsilon\right),$$

where $\mathcal{H}(x) = x\log(1/x) + (1-x)\log(1/(1-x))$ *is the binary entropy function.*

Proof. We define input datasets $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{2^d}$ where $\boldsymbol{x}^i$ assigns $D/2$ to the ith leaf and 0 to the remaining leaves. Let $\mathcal{P}_i$ be the path from root to the ith leaf. We define a randomized decoding algorithm Dec as follows:

- Dec takes the output $\widetilde{\boldsymbol{w}}$ of $\mathcal{A}$ as input and starts from the root of $\mathcal{T}$.
- Assume Dec is at node $u \in \mathsf{nodes}(\mathcal{T})$.
  - If u is a leaf, then output the index of u among all the leaves.
  - Otherwise let u_0, u_1 be the children of u and we divide into the following cases.
    - If $\widetilde{w}_{u_0} \geq \tau_{\mathsf{max}}$ and $\widetilde{w}_{u_1} \geq \tau_{\mathsf{max}}$, then we move to u_0 or u_1 with equal probability.
    - If $\widetilde{w}_{u_0} < \tau_{\mathsf{max}}$ and $\widetilde{w}_{u_1} < \tau_{\mathsf{max}}$, then we move to u_0 or u_1 with equal probability.
    - Otherwise let $p \in \{0,1\}$ be such that $\widetilde{w}_{u_p} \geq \tau_{\mathsf{max}}$ and $\widetilde{w}_{u_{1-p}} < \tau_{\mathsf{max}}$, then we move to u_p with probability κ and to u_{1-p} with probability $1-\kappa$, where $\kappa = 1-4\eta \in [1/2, 1]$.

Now we fix an index $i \in [2^d]$. Let $\mathcal{P}_i$ be $u_1, \ldots, u_d$. Then for each $j \in [d-1]$, let u_j^0, u_j^1 be the children of u_j and assume without loss of generality $u_j^0 = u_{j+1}$; then we define the following indicators:

- $a_j = \mathbb{I}[(\widetilde{w}_{u_j^0} \geq \tau_{\mathsf{max}} \text{ and } \widetilde{w}_{u_j^1} \geq \tau_{\mathsf{max}}) \text{ or } (\widetilde{w}_{u_j^0} < \tau_{\mathsf{max}} \text{ and } \widetilde{w}_{u_j^1} < \tau_{\mathsf{max}})]$.
- $b_j = \mathbb{I}[\widetilde{w}_{u_j^0} \geq \tau_{\mathsf{max}} \text{ and } \widetilde{w}_{u_j^1} < \tau_{\mathsf{max}}]$.
- $c_j = \mathbb{I}[\widetilde{w}_{u_j^0} < \tau_{\mathsf{max}} \text{ and } \widetilde{w}_{u_j^1} \geq \tau_{\mathsf{max}}]$.

Let $A = \sum_j a_j$, $B = \sum_j b_j$, and $C = \sum_j c_j$. Then it is easy to see $A + B + C = d - 1$ and

$$\mathbf{Pr}\left[\mathsf{Dec}(\mathcal{A}(\boldsymbol{x}^i)) = i\right] = \mathbb{E}\left[2^{-A}\kappa^B(1-\kappa)^C\right] = \kappa^{d-1}\,\mathbb{E}\left[1/(2\kappa)^{d-1-B}(2-2\kappa)^C\right]. \tag{8}$$

Now we further fix the input to be $\boldsymbol{x}^i$ defined above. Then $w_u = D/2$ for $u \in \mathcal{P}_i$ and $w_u = 0$ if otherwise. For $p \in \{0,1\}$, consider the event $\mathcal{E}_j^p$: $\left|\widetilde{w}_{u_j^p} - w_{u_j^p}\right| \leq \alpha \cdot \max\left\{w_{u_j^p}, \tau_{u_j^p}\right\}$. Hence when $\mathcal{E}_j^0$ happens, we have

$$\widetilde{w}_{u_j^0} \geq w_{u_j^0} - \alpha \cdot \max\left\{w_{u_j^0}, \tau_{u_j^0}\right\} \geq D/2 - \alpha \cdot \max\{D/2, \tau_{\mathsf{max}}\} = (1-\alpha)\cdot D/2 \geq \tau_{\mathsf{max}}.$$

Similarly when $\mathcal{E}_j^1$ happens, we have $\widetilde{w}_{u_j^1} \leq w_{u_j^1} + \alpha \cdot \max\left\{w_{u_j^1}, \tau_{u_j^1}\right\} \leq \alpha \cdot \tau_{\mathsf{max}} < \tau_{\mathsf{max}}$. Meanwhile by the definition of Problem 18, we know $\mathbf{Pr}\left[\mathcal{E}_j^p\right] \geq 1 - \eta$. Thus

$$\mathbf{Pr}\left[b_j = 1\right] \geq \mathbf{Pr}\left[\mathcal{E}_j^0 \wedge \mathcal{E}_j^1\right] \geq 1 - 2\eta, \quad \text{and} \quad \mathbf{Pr}\left[c_j = 1\right] \leq \mathbf{Pr}\left[\neg\mathcal{E}_j^0 \wedge \neg\mathcal{E}_j^1\right] \leq \mathbf{Pr}\left[\neg\mathcal{E}_j^0\right] \leq \eta,$$

which implies $\mathbb{E}\left[d - 1 - B\right] \leq 2\eta \cdot (d-1)$ and $\mathbb{E}\left[C\right] \leq \eta \cdot (d-1)$. Note that $d - 1 - B \geq 0$. Define the event $\mathcal{E}$: $d - 1 - B \leq 4\eta \cdot (d-1)$ and $C \leq 4\eta \cdot (d-1)$. Then by Markov's inequality and a union bound, we have $\mathbf{Pr}\left[\mathcal{E}\right] \geq 1 - 1/2 - 1/4 = 1/4$. Plugging into (8), we have

$$\begin{aligned}
\mathbf{Pr}\left[\mathsf{Dec}(\mathcal{A}(\boldsymbol{x}^i)) = i\right] &\geq \kappa^{d-1}/4 \cdot \mathbb{E}\left[1/(2\kappa)^{d-1-B}(2-2\kappa)^C \;\middle|\; \mathcal{E}\right] \\
&\geq \kappa^{d-1}/4 \cdot 1/(2\kappa)^{4\eta\cdot(d-1)} \cdot (2-2\kappa)^{4\eta\cdot(d-1)} && \text{(since } \kappa \in [1/2,1]\text{)} \\
&= \frac{1}{4} \cdot 2^{-(d-1)\cdot\mathcal{H}(4\eta)}. && \text{(setting } \kappa = 1-4\eta \in [1/2,1]\text{)}
\end{aligned}$$

Since $\sum_{i'} \mathbf{Pr}\left[\mathsf{Dec}(\mathcal{A}(\boldsymbol{x}^i)) = i'\right] = 1$, by an averaging argument there exists an i^* such that $\mathbf{Pr}\left[\mathsf{Dec}(\mathcal{A}(\boldsymbol{x}^i)) = i^*\right] \leq 2^{-d}$. Since $\left\|\boldsymbol{x}^i - \boldsymbol{x}^{i^*}\right\|_1 \in \{0, D\}$, by Fact 10 we have

$$\mathbf{Pr}\left[\mathsf{Dec}(\mathcal{A}(\boldsymbol{x}^{i^*})) = i^*\right] \leq \mathbf{Pr}\left[\mathsf{Dec}(\mathcal{A}(\boldsymbol{x}^i)) = i^*\right] \cdot e^{\varepsilon\cdot D} + \delta \cdot \tfrac{e^{\varepsilon\cdot D}-1}{e^\varepsilon - 1} \leq 2^{-d} \cdot e^{\varepsilon\cdot D} + \delta \cdot \tfrac{e^{\varepsilon\cdot D}-1}{e^\varepsilon - 1}.$$

In all, we have

$$\frac{1}{4} \cdot 2^{-(d-1)\cdot\mathcal{H}(4\eta)} \leq \mathbf{Pr}\left[\mathsf{Dec}(\mathcal{A}(\boldsymbol{x}^{i^*})) = i^*\right] \leq 2^{-d} \cdot e^{\varepsilon \cdot D} + \delta \cdot \frac{e^{\varepsilon \cdot D} - 1}{e^{\varepsilon} - 1},$$

which proves the bound after plugging in Assumption (7) and rearranging the terms. ◀

To deal with general η from Problem 18, we simply run independent copies of the algorithm to decrease the error probability.

▶ **Corollary 31** (Proof in Full Version). *Assume $\mathcal{T}$ in Problem 18 is a complete binary tree of depth d. Define $D = 2 \cdot \left\lceil \frac{\tau_{max}}{1-\alpha} \right\rceil$ and $s = \left\lceil \frac{\ln(4/\kappa)}{2\cdot(1/2-\eta)^2} \right\rceil$ for any parameter $\kappa \in (0, 1/2]$. If $\mathcal{A}$ is an (ε, δ)-DP algorithm for Problem 18 and suppose $s \cdot \delta \cdot \frac{e^{s\cdot\varepsilon\cdot D}-1}{e^{s\cdot\varepsilon}-1} \leq \frac{1}{8} \cdot 2^{-(d-1)\cdot\mathcal{H}(\kappa)}$, then*

$$\tau_{max} = \Omega\left(\frac{(1-\alpha)\cdot(1/2-\eta)^2\cdot(d-3-(d-1)\mathcal{H}(\kappa))}{\varepsilon\cdot\ln(4/\kappa)}\right).$$

▶ Remark 32 (General Tree Structures). The proof above works almost identically for binary tree $\mathcal{T}$ that is not necessarily complete, where the bound is simply replacing $d-3$ with $\log(|\mathsf{leaves}(\mathcal{T})|/8)$. To deal with general arity k, we can embed a binary tree $\mathcal{T}'$ into $\mathcal{T}$ where we say $\mathcal{T}$ *embeds* a tree $\mathcal{T}'$ if we can obtain $\mathcal{T}'$ from $\mathcal{T}$ by deleting nodes and edges. Then we can ignore nodes in $\mathsf{nodes}(\mathcal{T}) \setminus \mathsf{nodes}(\mathcal{T}')$ and obtain lower bounds for $\mathcal{T}'$.

Now we are ready to establish Theorem 28 using Lemma 19 and Corollary 31.

Proof of Theorem 28. Let $\mathcal{T}$ be the complete binary tree of depth d. Let C, τ, η be parameters to be optimized later. We consider Problem 18 where $\tau_u \equiv \tau$ for all $u \in \mathsf{nodes}(\mathcal{T})$.

Assume towards contradiction that there is an ε-DP tree aggregation algorithm with α-mRMSE at most $C \cdot (1-\alpha)^2 d/\varepsilon$. Then by Lemma 19, the algorithm is also (α', η)-accurate for Problem 18 if $\alpha' > \alpha$ and $(\alpha' - \alpha)\sqrt{\eta} \cdot \tau \leq C \cdot (1-\alpha)^2 d/\varepsilon$.

Now we set $\eta = 1/4$ and apply Corollary 31 with $\delta = 0, \kappa = 1/4$. This gives a lower bound $\tau = \Omega\left((1-\alpha')\cdot d/\varepsilon\right)$, which means $\frac{C\cdot(1-\alpha)^2\cdot d}{\varepsilon} \geq \Omega\left(\frac{(\alpha'-\alpha)(1-\alpha')\cdot d}{\varepsilon}\right)$. Then we set $\alpha' = (1+\alpha)/2 > \alpha$ and $C = O(1)$ small enough to derive a contradiction. ◀

▶ Remark 33 ($\log(1/\delta)$ Factor in the Approximate-DP Algorithm). As mentioned in Section 1.1, our improved (ε, δ)-DP algorithm (See Theorem 8) has a $\log(1/\delta)$ factor, which is worse than the $\sqrt{\log(1/\delta)}$ factor in the Gaussian mechanism (see Corollary 15). One may wonder if we can further improve the dependency on δ to, say, $\sqrt{\log(1/\delta)}$, without influencing the other parameters. Combining Corollary 31, we show this is in some sense impossible.

Consider the complete binary tree of depth d. Let $\delta = 2^{-\Omega(d)}$. We consider the case where α, η are constants and all the τ_u's are equal to τ. In the approximate-DP setting, we naturally seek bounds better than the ones in the pure-DP setting (recall we can obtain $\tau = O(d/\varepsilon)$ from Corollary 15). Thus we assume $\tau = O(d/\varepsilon)$ in advance.

Then for a suitable choice of $\kappa = \Theta(1)$, the condition in Corollary 31 holds, which gives a lower bound $\tau = \Omega(d/\varepsilon)$ for Problem 18. Then by Lemma 19, this rules out the possibility of improving the dependency on δ in Theorem 8 without worsening the dependency on d.

6 Conclusions

We study the problem of privately estimating counts in hierarchical data, and give several algorithms and lower bounds. We propose a new error measure that takes the multiplicative error into account. The commonly used ℓ_2^2-error measure in evaluating utilities of DP

mechanisms allows some queries to have huge error. On the other hand, the standard measure ℓ_∞-error has a "union bound issue" on particularly long output vector (which is the case in Census and Ads applications).

To mitigate these weaknesses, we propose α-multiplicative root mean squared error (α-mRMSE). Then we examine the standard Laplace mechanism for pure-DP and Gaussian mechanism for approximate-DP, and prove their optimality. Informally, we show Laplace mechanism already achieves optimal bounds in the pure-DP setting for all multiplicative factor α and Gaussian mechanism is optimal in the approximate-DP setting when $\alpha = 0$ (i.e., additive-only error).

For the remaining case where we allow $\alpha > 0$ and an approximate-DP algorithm, we design a new algorithm with exponential improvements over Gaussian mechanism. More precisely, Gaussian mechanism incurs α-mRMSE of $O_{\alpha,\varepsilon,\delta}(\sqrt{d})$ while our algorithm gives improved bounds of $O_{\alpha,\varepsilon,\delta}(\log(d))$ (and $O\left(\frac{1}{\alpha^3} \cdot \left(\frac{1}{\varepsilon}\log\left(\frac{2d}{\delta}\right)+1\right)\right)$ specifically). It remains an interesting question if the dependency on d or α can be improved further. Indeed, current lower bounds do not preclude bounds of the form $O_{\varepsilon,\delta}\left(\frac{1}{\alpha}\cdot\log^*(d)\right)$ or even $O_{\varepsilon,\delta}(1/\alpha)$.

Throughout this work, we assumed that the entries of the input $\boldsymbol{x}$ are non-negative. Another interesting direction is to extend the study to the case where the entries of $\boldsymbol{x}$ can be negative. Here, the multiplicative error would be with respect to the absolute value of the true answer. Our algorithms do not apply here and it is unclear whether allowing a multiplicative error can help reduce the additive error in this setting.

References

1 John Abowd, Daniel Kifer, Brett Moran, Robert Ashmead, Philip Leclerc, William Sexton, Simson Garfinkel, and Ashwin Machanavajjhala. Census topdown: Differentially private data, incremental schemas, and consistency with public knowledge, 2019. Available at `https://github.com/uscensusbureau/census2020-das-e2e/blob/master/doc/20190711_0945_Consistency_for_Large_Scale_Differentially_Private_Histograms.pdf`.

2 John M. Abowd, Robert Ashmead, Ryan Cumings-Menon, Simson L. Garfinkel, Micah Heineck, Christine Heiss, Robert Johns, Daniel Kifer, Philip Leclerc, Ashwin Machanavajjhala, Brett Moran, William Sexton, Matthew Spence, and Pavel Zhuravlev. The 2020 census disclosure avoidance system TopDown algorithm. *Harvard Data Sci. Rev.*, 2022. Special Issue 2.

3 John M Abowd and Ian M Schmutte. An economic analysis of privacy protection and statistical accuracy as social choices. *Amer. Econ. Rev.*, 109(1):171–202, 2019.

4 Apple Differential Privacy Team. Learning with privacy at scale. *Apple ML J.*, 2017.

5 Aditya Bhaskara, Daniel Dadush, Ravishankar Krishnaswamy, and Kunal Talwar. Unconditional differentially private mechanisms for linear queries. In *STOC*, pages 1269–1284, 2012.

6 Jaroslaw Blasiok, Mark Bun, Aleksandar Nikolov, and Thomas Steinke. Towards instance-optimal private query release. In *SODA*, pages 2480–2497, 2019.

7 Jean Bolot, Nadia Fawaz, Shanmugavelayutham Muthukrishnan, Aleksandar Nikolov, and Nina Taft. Private decayed predicate sums on streams. In *ICDT*, pages 284–295, 2013.

8 Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil P. Vadhan. Differentially private release and learning of threshold functions. In *FOCS*, pages 634–649, 2015.

9 Mark Bun, Jonathan R. Ullman, and Salil P. Vadhan. Fingerprinting codes and the price of approximate differential privacy. *SIAM J. Comput.*, 47(5):1888–1938, 2018. `doi:10.1137/15M1033587`.

10 T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3):26:1–26:24, 2011. `doi:10.1145/2043621.2043626`.

11 Aloni Cohen, Moon Duchin, J. N. Matthews, and Bhushan Suwal. Census topdown: The impacts of differential privacy on redistricting. In *FORC*, pages 5:1–5:22, 2021.

12 Aloni Cohen, Moon Duchin, JN Matthews, and Bhushan Suwal. Private Numbers in Public Policy: Census, Differential Privacy, and Redistricting. *Harvard Data Sci. Rev.*, 2022.

13 Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. Differentially private spatial decompositions. In *ICDE*, pages 20–31, 2012.

14 Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *NeurIPS*, pages 3571–3580, 2017.

15 Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210, 2003.

16 Cynthia Dwork. Differential privacy: A survey of results. In *TAMC*, pages 1–19, 2008.

17 Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, pages 486–503, 2006.

18 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. *J. Priv. Confidentiality*, 7(3):17–51, 2016. doi:10.29012/jpc.v7i3.405.

19 Cynthia Dwork, Frank McSherry, and Kunal Talwar. The price of privacy and the limits of LP decoding. In *STOC*, pages 85–94, 2007.

20 Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In *STOC*, pages 715–724, 2010. doi:10.1145/1806689.1806787.

21 Cynthia Dwork, Moni Naor, Omer Reingold, and Guy N. Rothblum. Pure differential privacy for rectangle queries via private partitions. In *ASIACRYPT*, pages 735–751, 2015.

22 Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC*, pages 381–390, 2009.

23 Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014. doi:10.1561/0400000042.

24 Alexander Edmonds, Aleksandar Nikolov, and Jonathan R. Ullman. The power of factorization mechanisms in local and central differential privacy. In *STOC*, pages 425–438, 2020.

25 Alessandro Epasto, Jieming Mao, Andres Munoz Medina, Vahab Mirrokni, Sergei Vassilvitskii, and Peilin Zhong. Differentially private continual releases of streaming frequency moment estimations. In *ITCS*, pages 48:1–48:24, 2023.

26 Hendrik Fichtenberger, Monika Henzinger, and Wolfgang Ost. Differentially private algorithms for graphs under continual observation. In *ESA*, pages 42:1–42:16, 2021.

27 Quan Geng, Wei Ding, Ruiqi Guo, and Sanjiv Kumar. Tight analysis of privacy and utility tradeoff in approximate differential privacy. In *AISTATS*, pages 89–99, 2020.

28 Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In *STOC*, pages 705–714, 2010.

29 Monika Henzinger, Jalaj Upadhyay, and Sarvagya Upadhyay. Almost tight error bounds on differentially private continual counting. In *SODA*, pages 5003–5039, 2023.

30 James Honaker. Efficient use of differentially private binary trees. In *TPDP*, 2015.

31 Peter Kairouz, Brendan McMahan, Shuang Song, Om Thakkar, Abhradeep Thakurta, and Zheng Xu. Practical and private (deep) learning without sampling or shuffling. In *ICML*, pages 5213–5225, 2021.

32 Haim Kaplan, Katrina Ligett, Yishay Mansour, Moni Naor, and Uri Stemmer. Privately learning thresholds: Closing the exponential gap. In *COLT*, pages 2263–2285, 2020.

33 Troy Lee, Adi Shraibman, and Robert Spalek. A direct product theorem for discrepancy. In *CCC*, pages 71–80, 2008.

34 Chao Li, Gerome Miklau, Michael Hay, Andrew McGregor, and Vibhor Rastogi. The matrix mechanism: optimizing linear counting queries under differential privacy. *VLDB J.*, 24(6):757–781, 2015. doi:10.1007/s00778-015-0398-x.

35 Roy Mathias. The hadamard operator norm of a circulant and applications. *SIAM J. Matr. Anal. Appl.*, 14(4):1152–1167, 1993.

36 Jiří Matoušek, Aleksandar Nikolov, and Kunal Talwar. Factorization norms and hereditary discrepancy. *Intl. Math. Res. Not.*, 2020(3):751–780, 2018.

37 Aleksandar Nikolov. An improved private mechanism for small databases. In *ICALP*, pages 1010–1021, 2015.

38 Aleksandar Nikolov. Private query release via the Johnson–Lindenstrauss transform. In *SODA*, pages 4982–5002, 2023.

39 Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The geometry of differential privacy: the sparse and approximate cases. In *STOC*, pages 351–360, 2013.

40 Wahbeh H. Qardaji, Weining Yang, and Ninghui Li. Differentially private grids for geospatial data. In *ICDE*, pages 757–768, 2013.

41 Salil P. Vadhan. The complexity of differential privacy. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 347–450. Springer International Publishing, 2017. doi:10.1007/978-3-319-57048-8_7.

42 Yonghui Xiao, Li Xiong, Liyue Fan, Slawomir Goryczka, and Haoran Li. DPCube: Differentially private histogram release through multidimensional partitioning. *Trans. Data Priv.*, 7(3):195–222, 2014.

43 Jun Zhang, Xiaokui Xiao, and Xing Xie. Privtree: A differentially private algorithm for hierarchical decompositions. In *SIGMOD*, pages 155–170, 2016.

Quantum Cryptography with Classical Communication: Parallel Remote State Preparation for Copy-Protection, Verification, and More

Alexandru Gheorghiu
Department of Computer Science and Engineering, Chalmers University of Technology, Göteborg, Sweden
Institute for Theoretical Studies, ETH Zürich, Switzerland

Tony Metger
Institute for Theoretical Physics, ETH Zürich, Switzerland

Alexander Poremba
Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA, USA

Abstract

Quantum mechanical effects have enabled the construction of cryptographic primitives that are impossible classically. For example, quantum copy-protection allows for a program to be encoded in a quantum state in such a way that the program can be evaluated, but not copied. Many of these cryptographic primitives are two-party protocols, where one party, Bob, has full quantum computational capabilities, and the other party, Alice, is only required to send random BB84 states to Bob. In this work, we show how such protocols can *generically* be converted to ones where Alice is fully classical, assuming that Bob cannot efficiently solve the LWE problem. In particular, this means that all communication between (classical) Alice and (quantum) Bob is classical, yet they can still make use of cryptographic primitives that would be impossible if both parties were classical. We apply this conversion procedure to obtain quantum cryptographic protocols with classical communication for unclonable encryption, copy-protection, computing on encrypted data, and verifiable blind delegated computation.

The key technical ingredient for our result is a protocol for *classically-instructed parallel remote state preparation of BB84 states*. This is a multi-round protocol between (classical) Alice and (quantum polynomial-time) Bob that allows Alice to certify that Bob must have prepared n uniformly random BB84 states (up to a change of basis on his space). While previous approaches could only certify one- or two-qubit states, our protocol allows for the certification of an n-fold tensor product of BB84 states. Furthermore, Alice knows which specific BB84 states Bob has prepared, while Bob himself does not. Hence, the situation at the end of this protocol is (almost) equivalent to one where Alice sent n random BB84 states to Bob. This allows us to replace the step of preparing and sending BB84 states in existing protocols by our remote-state preparation protocol in a generic and modular way.

2012 ACM Subject Classification Theory of computation; Theory of computation → Cryptographic protocols

Keywords and phrases Quantum cryptography, Remote state preparation, Self-testing, Learning with errors, Quantum copy-protection, Unclonable encryption, Quantum verification

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.67

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2201.13445

Funding *Alexandru Gheorghiu*: Acknowledges support from the Knut and Alice Wallenberg Foundation through the Wallenberg Centre for Quantum Technology (WACQT) and Dr. Max Rössler, the Walter Haefner Foundation and the ETH Zürich Foundation, while at ETH.

Tony Metger: Acknowledges support from the QuantERA project "eDict" and the Air Force Office of Scientific Research (AFOSR) Grant No. FA9550-19-1-0202.
Alexander Poremba: Acknowledges partial support from AFOSR YIP award number FA9550-16-1-0495 and the Institute for Quantum Information and Matter (an NSF Physics Frontiers Center; NSF Grant PHY-1733907)

Acknowledgements We thank Honghao Fu, Thomas Vidick, and Daochen Wang for helpful discussions, and Jeffrey Champion and John Wright for allowing us to use the results in Section 4.3 of the full version of the manuscript, which are based on unpublished joint work by them and the second author. We also thank Matty Hoban for pointing out a typo in an earlier draft.

1 Introduction

A central distinction between classical and quantum information is that a classical string can always be copied, but a quantum state cannot: the *no-cloning theorem* states that there cannot exist a procedure that produces the state $\rho \otimes \rho$ when given as input an arbitrary quantum state ρ [51]. The first cryptographic protocols that made use of the no-cloning theorem were Wiesner's proposal to use quantum states as unforgeable banknotes [50] and Bennett and Brassard's protocol for information-theoretically secure quantum key-distribution (the BB84 QKD protocol) [6]. These protocols rely on the idea of a *conjugate coding scheme*: classical information can be encoded into a quantum state in (at least) two incompatible bases, most commonly the standard basis $\{|0\rangle, |1\rangle\}$ and the Hadamard basis $\{|+\rangle, |-\rangle\}$, where $|\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$. These four states are commonly referred to as *BB84 states*. If we encode a bit $b \in \{0, 1\}$ as either $|b\rangle$ or $|(-)^b\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^b|1\rangle)$, then an adversary who does not know which basis we chose for the encoding cannot create a copy of this quantum state. Furthermore, if the adversary tries to measure the state, with probability $1/2$ they will choose the "wrong" measurement basis, which disturbs the state and means that the adversary's tampering can be detected.

There is an important conceptual difference between the BB84 protocol and Wiesner's quantum money scheme. The former addresses the problem of key-distribution, which is a task that can also be achieved classically under computational assumptions using public-key cryptography [18]. In contrast, Wiesner's quantum money scheme achieves a functionality which is entirely impossible classically, even under computational assumptions. Recently there has been renewed interest in this latter kind of application, i.e. to use BB84 states to construct quantum cryptographic primitives that have no classical analogue. Perhaps the most striking example of this is the idea of *quantum copy-protection* [1]. Suppose that a vendor has created a piece of software (viewed as a function that maps some input to some output) and wants to allow a user to run it (i.e. to evaluate the function), while preventing the user from producing additional "pirated" copies of the original software. Clearly, this is impossible classically: any piece of software is specified by a string of symbols, which can easily be copied. Surprisingly, it has been shown that it is possible to encode certain narrow classes of functions in the form of a quantum state in such a way that a user can evaluate the function without being able to copy it [16].

Copy-protection and many related protocols require only limited quantum capabilities from one party, e.g. the *vendor* in the case of copy-protection: they only need to prepare random BB84 states and send them to the other party (e.g. the *user* in copy-protection), who has full quantum computational capabilities. In particular, this requires a quantum channel between the two parties to send the BB84 states. The purpose of this paper is to show that such protocols, where one party's quantum operations are limited to preparing and sending

random BB84 states, can be converted into protocols where that party is *fully classical*. This *dequantises* such protocols in the sense that all communication becomes classical. To achieve this, we need to construct a protocol between a *classical verifier* and a *computationally bounded quantum prover* that achieves the same outcome as if the verifier had prepared and sent random BB84 states to the prover. We call this task *classically-instructed parallel remote state preparation of BB84 states*, or *parallel RSP* for short. Our protocol builds on techniques introduced in [33, 7, 24] that allow the verifier to use post-quantum cryptography to constrain the actions of an untrusted (but computationally bounded) prover and certify the result of a certain computation or the preparation of certain states. In contrast to earlier works on remote state preparation (or *self-testing*) in this setting, which could only certify states comprised of a constant number of qubits, our protocol allows for the certification of an n-fold tensor product of states. We discuss the difference between our approach and previous approaches to RSP (in particular the protocol of [24]) in Section 4. Proving soundness for this parallel RSP protocol is the main technical result of our work. We then use this result to dequantise a number of cryptographic protocols, namely unclonable quantum encryption, quantum copy-protection, quantum computing on encrypted data and blind verification of quantum computation.

2 Main results

We start by first describing the soundness guarantee achieved by our parallel RSP protocol. Intuitively, the goal of our protocol is to guarantee that the prover has prepared a quantum state of the form $H^{\theta_1}|v_1\rangle\langle v_1|H^{\theta_1} \otimes \ldots \otimes H^{\theta_n}|v_n\rangle\langle v_n|H^{\theta_n}$, where $\vec{v}, \vec{\theta} \in \{0,1\}^n$. Additionally, the prover should not have any information about $\vec{v}$ and $\vec{\theta}$ beyond what is contained in its BB84 states, while the verifier should know both $\vec{v}$ and $\vec{\theta}$. Our protocol achieves a guarantee of this kind assuming the quantum-intractability of the *Learning with Errors* (LWE) problem introduced by Regev [43]. Our main result is the following (see the full manuscript for the corresponding formal statement):

▶ **Theorem 1** (Informal). *There exists an interactive protocol between a classical verifier and a computationally bounded quantum prover such that the following holds assuming the quantum-intractability of LWE (with quantum advice). Fix a number n of BB84 states. Consider any efficient prover strategy and let W and P be the verifier's and prover's systems at the end of the protocol, respectively. Then there exists an isometry $V : P \to QP'$ (for $\mathcal{H}_Q \cong (\mathbb{C}^2)^{\otimes n}$ and P' arbitrary) and an additional (subnormalised) state $\alpha_{P'}$ such that for any basis choice $\vec{\theta} \in \{0,1\}^n$, the protocol's final state σ_{WP} conditioned on the prover being accepted satisfies*

$$p_{\text{success}} V \sigma_{WP} V^\dagger \overset{c}{\approx}_{1/\operatorname{poly}(n)} \frac{1}{2^n} \sum_{\vec{v} \in \{0,1\}^n} |v\rangle\langle v|_W \otimes \left(H^{\theta_1}|v_1\rangle\langle v_1|H^{\theta_1} \otimes \ldots \otimes H^{\theta_n}|v_n\rangle\langle v_n|H^{\theta_n}\right) \otimes \alpha_{P'} .$$

Here, p_{success} is the prover's success probability in the protocol and $\overset{c}{\approx}_{1/\operatorname{poly}(n)}$ denotes computational indistinguishability up to inverse polynomial error.

We make two remarks regarding this security guarantee. Firstly, the theorem makes a statement about the joint state of the verifier's system W and the prover's system P after applying an isometry V that *only acts on the prover's space*. This additional isometry is unavoidable: it represents the prover's freedom to use any basis of its choice on its space.

Hence, we cannot guarantee that the prover prepares BB84 states (in the standard basis), only that it prepares BB84 states up to a change of basis. However, crucially this change of basis is *independent of which BB84 state was supposed to be prepared*, i.e., V is independent of $\vec{v}$ and $\vec{\theta}$ (but it can of course depend on the prover's strategy). Put differently, the theorem guarantees that the prover prepares one of 4^n possible states whose relation to each other is the same as the relation between the 4^n BB84 states. This does not affect the utility of the prover's state for applications. In fact, this freedom also exists if the verifier sent n BB84 states to the prover via a quantum channel: the prover could apply an isometry V to these states immediately upon receipt, but the security of any application using the BB84 states is not impacted by this.

Secondly, the theorem holds *for any* basis choice $\vec{\theta}$, but *on average* over the values $\vec{v}$. In other words, in the protocol, the verifier gets to choose the bases at will, but the values will be uniformly random and cannot be chosen by the verifier. Furthermore, the only dependence on $\vec{v}$ and $\vec{\theta}$ in the prover's state is via the BB84 states. This means that the protocol forces the prover to prepare these states "blindly", i.e., the prover does not know which BB84 states were actually prepared. In contrast, the verifier does know, because they chose $\vec{\theta}$ and are in possession of the system W, which contains information about $\vec{v}$. This asymmetry of knowledge about the prover's state is the same as what is achieved by preparing and sending BB84 states through a quantum channel and is crucial for applications.

We also note that a consequence of Theorem 1 is the certification of an n-fold tensor product structure within the prover's system. This can be interpreted as saying that any successful prover must have a quantum memory capable of storing n-qubits. Being able to certify an n-qubit state in the prover's system is the main technical challenge towards proving soundness, as we outline in the next subsection. This notion of a computational proof of quantum space has been formalised in [21], who prove a similar parallel rigidity result to ours, but for a different class of states that does not immediately allow for cryptographic applications.

2.1 Soundness proof for parallel RSP protocol

The full RSP protocol is described as Protocol 3 (though our discussion here is restricted to Protocol 1). Its soundness proof can be found in the full version of the manuscript.

We briefly explain the difference between Protocol 1 and Protocol 3: Protocol 1 is a protocol to *test* the prover, i.e. in this protocol the prover is asked to prepare *and measure* a quantum state, and the verifier runs checks on the prover's answer. The soundness statement for this protocol is a self-testing statement in the sense of [36], which characterises which states and measurements the prover used in the protocol. Although we do not spell this out, it is easy to obtain an explicit self-testing statement from our proof. In contrast, Protocol 3 is a protocol for remote state preparation, so the prover is supposed to prepare, but not yet measure, a particular quantum state. Instead, this quantum state will be used for other applications. This means that we do not want to make a statement about how the prover measured its state, but rather what state remains in its quantum memory. The soundness of Protocol 3 follows from that of Protocol 1 via a statistical argument. In the following, we focus on Protocol 1. We do not explain the protocol and the cryptographic primitives underlying it in detail; instead, we give a very high-level description of the relevant part of the soundness proof of the RSP protocol from [24] and then explain our method for proving a parallel rigidity statement based on that result.

The main cryptographic primitive underlying the RSP protocol is a so-called extended noisy trapdoor claw-free function (ENTCF) family, which can be constructed assuming the quantum hardness of LWE [43, 33]. An ENTCF family is a family of functions indexed by

▶ **Protocol 1. Test round protocol.**

Let $\lambda \in \mathbb{N}$ be the security parameter, $(\mathcal{F}, \mathcal{G})$ an ENTCF family, and $n = \text{poly}(\lambda)$ the number of BB84 states that the verifier wishes to prepare.

1. The verifier selects a uniformly random basis $\theta \xleftarrow{\$} \{0,1\}$, where 0 corresponds to the computational and 1 to the Hadamard basis.
2. The verifier samples keys and trapdoors $(k_1, t_{k_1}; \dots k_n, t_{k_n})$ by computing $(k_i, t_{k_i}) \leftarrow \text{GEN}_{\mathcal{K}_\theta}(1^\lambda)$. The verifier then sends $(k_1, \dots k_n)$ to the prover (but keeps the trapdoors t_{k_i} private).
3. The verifier receives $(y_1, \dots, y_n) \in \mathcal{Y}^{\times n}$ from the prover.
4. The verifier selects a round type $\in$ {preimage round, Hadamard round} uniformly at random and sends the round type to the prover.
   a. For a *preimage round*: The verifier receives $(b_1, x_1; \dots b_n, x_n)$ from the prover, with $b_i \in \{0,1\}$ and $x_i \in \mathcal{X}$. The verifier sets $\texttt{flag} \leftarrow \texttt{fail}_{\texttt{Pre}}$ if $\text{CHK}(k_i, y_i, b_i, x_i) = 0$.
   b. For a *Hadamard round*: The verifier receives $d_1, \dots d_n \in \{0,1\}^w$ from the prover (for some w depending on the security parameter). The verifier sends $q = \theta$ to the prover, and receives answers $v_1, \dots v_n \in \{0,1\}$. The verifier performs the following checks:

Case	Verifier's check
$q = \theta = 0$	Set $\texttt{flag} \leftarrow \texttt{fail}_{\texttt{Had}}$ if $\hat{b}(k_i, y_i) \neq v_i$ for some i.
$q = \theta = 1$	Set $\texttt{flag} \leftarrow \texttt{fail}_{\texttt{Had}}$ if $\hat{u}(k_i, y_i, d_i) \neq v_i$.

Note. We denote the "question" separately by q (even though here we always have $q = \theta$) because when the variant of this protocol in Protocol 2 is used in the context of another cyrptographic task, the verifier can also send questions q which are different from θ.

▶ **Protocol 2. Preparation round protocol.**

Let $\lambda \in \mathbb{N}$ be the security parameter, $(\mathcal{F}, \mathcal{G})$ an ENTCF family, and $n = \text{poly}(\lambda)$ the number of BB84 states that the verifier wishes to prepare.

1. The verifier selects bases $\vec{\theta} \xleftarrow{\$} \{0,1\}^n$, where 0 corresponds to the computational and 1 to the Hadamard basis.
2. The verifier samples keys and trapdoors $(k_1, t_{k_1}; \dots k_n, t_{k_n})$ by computing $(k_i, t_{k_i}) \leftarrow \text{GEN}_{\mathcal{K}_{\theta_i}}(1^\lambda)$. The verifier then sends $(k_1, \dots k_n)$ to the prover (but keeps the trapdoors t_{k_i} private).
3. The verifier receives $y_1, \dots y_n \in \mathcal{Y}$ from the prover.
4. The verifier sends "Hadamard round" to the prover as the round type.
5. The verifier receives $d_1, \dots d_n \in \{0,1\}^w$ from the prover (for some w depending on the security parameter). The verifier computes a string $\vec{v}$ according to

$$v_i = \begin{cases} \hat{b}(k_i, y_i) & \text{if } \theta_i = 0\,, \\ \hat{u}(k_i, y_i, d_i) & \text{if } \theta_i = 1\,. \end{cases}$$

▶ **Protocol 3. Multi-round protocol for preparation of BB84 states.**

Let $\lambda \in \mathbb{N}$ be the security parameter, $(\mathcal{F}, \mathcal{G})$ an ENTCF family, $n = \text{poly}(\lambda)$ the number of BB84 states that the verifier wishes to prepare, $N = M^2$ the maximum number of test rounds (for $M \in \mathbb{N}$), and δ an error tolerance parameter. For $j \in [M]$ we denote by $B_j = \{(j-1)M + 1, \ldots, jM\}$ the j-th "block" of M rounds.

1. The verifier (privately) samples $S \xleftarrow{\$} \{0, \ldots, M-1\}$ (the number of M-round blocks of test rounds that will be performed).
2. The verifier performs SM executions of Protocol 1 with the prover. The verifier aborts if *for any* $j \in [S]$, the fraction of rounds in B_j for which $\texttt{flag} = \texttt{fail}_{\texttt{Pre}}$ or $\texttt{flag} = \texttt{fail}_{\texttt{Had}}$ exceeds δ.
3. The verifier (privately) samples $R \xleftarrow{\$} [M]$ and executes Protocol 1 with the prover $R-1$ times. Then, the verifier executes Protocol 2 with the prover and records the basis choice $\vec{\theta}$ and the string $\vec{v}$ from that execution.

a set of keys $\mathcal{K}_0 \cup \mathcal{K}_1$. $\mathcal{K}_0$ and $\mathcal{K}_1$ are disjoint sets of keys with the property that given a $k \in \mathcal{K}_0 \cup \mathcal{K}_1$, it is computationally intractable to determine which set this key belongs to. See [33, Section 4] for further details on ENTCF families.

In the RSP protocol from [24], for a given *basis choice* $\theta \in \{0, 1\}$ (where "0" corresponds to the computational and "1" to the Hadamard basis), the verifier samples a key $k \in \mathcal{K}_\theta$, alongside some trapdoor information t. The verifier sends k to the prover and keeps t private. The verifier and prover then interact classically; for us, the main point of interest is the last round of the protocol, i.e. the last message from the verifier to the prover and back. Let us denote the protocol's transcript up to the last round by ts. Before the last round, the remaining quantum state of an *honest* prover is the single-qubit state $H^\theta|v\rangle\langle v|H^\theta$ for $v \in \{0, 1\}$. From the transcript and the trapdoor information, the verifier can compute v; in contrast, the prover, who does not know the trapdoor, cannot efficiently compute θ or v. In the last round, the verifier sends θ to the prover, who returns $v' \in \{0, 1\}$; the verifier then checks whether $v' = v$. The honest prover would generate v' by measuring its remaining qubit $H^\theta|v\rangle\langle v|H^\theta$ in the basis θ and therefore always pass the verifier's check.

We can model this last round of the protocol (with a potentially dishonest prover) as follows: at the start, the prover has a state $\sigma^{(\theta,v)}$, which it produced as a result of the previous rounds of the protocol. For an honest prover, $\sigma^{(\theta,v)} = H^\theta|v\rangle\langle v|H^\theta$. Of course, this state can depend on all of ts, but we only make the dependence on θ and v explicit. Upon receiving $\theta \in \{0, 1\}$ the prover measures a binary observable Z (if $\theta = 0$) or X (if $\theta = 1$) and returns the outcome v'. An honest prover would simply use the Pauli observables $Z = \sigma_Z$ and $X = \sigma_X$. The key step in the proof of [24] is to show that, due to the properties of ENTCF families, for any (potentially dishonest) prover that is accepted with high probability, the observables X and Z must anti-commute when acting on the prover's state. Then, Theorem 1 (for $n = 1$) follows from known results [34, 39, 26].

For our parallel RSP protocol we run n independent copies of the protocol from [24] in parallel, except that the basis choice θ_i is the same for each copy.[1] The prover's state before the last round of each copy of the RSP protocol is now denoted by $\sigma^{(\theta,\vec{v})}$, where $\vec{v} \in \{0, 1\}^n$

[1] The advantage of this is that a prover that succeeds with high probability on average over θ must also succeed with high probability for each θ individually. If we were to sample θ independently for each of the parallel copies we could not conclude that a prover succeeds with high probability for any particular choice of $\theta_1, \ldots, \theta_n$ as there are exponentially many such choices.

can be calculated by the verifier from the transcript ts by repeating the same calculation as above for each parallel copy. Generalising from the single-qubit case, given $\theta \in \{0,1\}$ the prover performs a measurement to generate $\vec{v} \in \{0,1\}^n$, which we can describe by binary observables Z_i, X_i (for $\theta = 0, 1$ respectively) that correspond to the observable used to produce the i-th entry of $\vec{v}$. (For an honest prover, $\sigma^{(\theta,\vec{v})} = H^\theta|v_1\rangle\langle v_1|H^\theta \otimes \ldots \otimes H^\theta|v_n\rangle\langle v_n|H^\theta$ and Z_i is a Pauli-Z measurement on the i-th qubit.)

The main challenge in the proof is to establish that the prover must treat all of the parallel copies of the RSP protocol independently, i.e. to show that its (a priori uncharacterised) Hilbert space can be partitioned into n identical subspaces, one for each copy of the protocol. At first sight, it might look as though for this it suffices to show that X_i and Z_j (approximately) commute for all $i \neq j$. However, this is not the case because any such commutation statement can only be shown in a special *state-dependent distance* [47], which does not allow us to combine individual commutation statements into the *global* statement that the Hilbert space factorises into n subspaces. Instead, we need to consider the family $\{Z(\vec{a})X(\vec{b})\}_{\vec{a},\vec{b}\in\{0,1\}^n}$ of 4^n binary observables, where $Z(\vec{a}) = Z_1^{a_1} \cdots Z_n^{a_n}$. We then have to show that $\{Z(\vec{a})X(\vec{b})\}$ form an approximate representation of the Pauli group [26, 48].[2] This means that when acting on the prover's (unknown) state $\sigma^{(\theta)}$ (where $\sigma^{(\theta)}$ is like $\sigma^{(\theta,\vec{v})}$, but averaged over all $\vec{v}$), the operators $\{Z(\vec{a})X(\vec{b})\}$ behave essentially like Pauli operators. Formally, this means showing that on average over $\vec{a}, \vec{b} \in \{0,1\}^n$,

$$\mathrm{Tr}\left[Z(\vec{a})X(\vec{b})Z(\vec{a})X(\vec{b})\sigma^{(\theta)}\right] \approx (-1)^{\vec{a}\cdot\vec{b}}. \tag{2.1}$$

This is the appropriate generalisation of the statement that Z and X anti-commute in the single-qubit case. It is easy to check that Equation (2.1) holds when Z_i and X_i are the Pauli observables.

Our proof of Equation (2.1) has five main steps, which we briefly sketch here with references to the corresponding parts of the formal proof.

(1) Instead of working with the observables X_i, we define "inefficient observables" $\tilde{X}_i = (-1)^{v_i}X_i$, where v_i is the i-th bit of the verifier's string $\vec{v}$. $\tilde{X}_i$ is not an observable that an efficient prover can implement because it depends on v_i, which requires the trapdoor information to be computed efficiently. Intuitively, while X_i describes the prover's answer, $\tilde{X}_i$ describes whether that answer is accepted by the verifier. This has the advantage that the state $\sigma^{(\theta=1)}$ (averaged over $\vec{v}$) of a successful prover is an approximate $+1$-eigenstate of $\tilde{X}_i$, but not of X_i.

(2) We extend the family of states $\{\sigma^{(\theta)}\}_{\theta\in\{0,1\}}$ to a larger family of "counterfactual states" $\{\sigma^{(\vec{\theta})}\}_{\vec{\theta}\in\{0,1\}^n}$, which are defined as the states the prover would have prepared if the verifier had sent keys $k_i \in \mathcal{K}_{\theta_i}$. In Protocol 1 the basis choice is the same for all i, i.e. $\vec{\theta} = \vec{0}$ or $\theta = \vec{1}$, so for other choices of $\vec{\theta}$ these states are never actually prepared. However, they are still well-defined because for any prover in the actual protocol, we can fix that prover's operations (as a quantum circuit acting on a given input) and then consider what state those operations would produce if given keys with an arbitrary basis choice $\vec{\theta}$. The reason these counterfactual states are useful is that we can show that, as a consequence of the properties of ENTCF families, the states $\{\sigma^{(\vec{\theta})}\}_{\vec{\theta}}$ are computationally indistinguishable.

[2] When we say "Pauli group" we always mean the Pauli group modulo complex conjugation, which is also sometimes called the Heisenberg-Weyl group.

(3) We now want to show various commutation and anti-commutation relations for the observables $Z(\vec{a})$ and $\tilde{X}(\vec{b})$. For example, we want to show that Z_i and $\tilde{X}_i$ anti-commute, but Z_i and $\tilde{X}_j$ commute (for $i \neq j$). To show these relations, we make use of the counterfactual states $\sigma^{(\vec{\theta})}$ in the following way: for any particular relation, we can pick a $\vec{\theta}$ that makes showing this relation especially convenient. For example, to show that Z_i and $\tilde{X}_j$ commute, we would choose a $\vec{\theta}$ with $\theta_i = 0$ and $\theta_j = 1$ since the verifier can check the outcomes of "Z-type observables" for $\theta = 0$ and "X-type observables" for $\theta = 1$. Using the properties of ENTCF families, we can argue that the prover's measurements on these counterfactual states still yield outcomes that would pass the verifier's checks for each choice of θ_i. Based on this, we can show the desired relations for a "convenient" choice of counterfactual state $\sigma^{(\vec{\theta})}$. Then, we can relate these statements back to the prover's actual states $\sigma^{(\theta)}$ using the computational indistinguishability of $\{\sigma^{(\vec{\theta})}\}$. This is somewhat delicate because $\tilde{X}_i$ are inefficient.

(4) We can combine the various commutation and anti-commutation statements from the previous step to show that the observables $\{Z(\vec{a})\tilde{X}(\vec{b})\}$ behave like Pauli observables on $\sigma^{(\theta=1)}$, i.e. we show Equation (2.1) but with $\tilde{X}$ instead of X. This step relies on the fact that $\sigma^{(\theta=1)}$ is an approximate $+1$-eigenstate of $\tilde{X}(\vec{b})$ for all $\vec{b}$.

(5) Since we now know that $\{Z(\vec{a})\tilde{X}(\vec{b})\}$ behave essentially like Pauli observables, we can define an explicit isometry $\tilde{V}$ which can be shown to map $\{Z(\vec{a})\tilde{X}(\vec{b})\}$ to the corresponding Pauli observables. This means that we have good control over these inefficient observables, and we know how the inefficient and efficient observables are related. We can use this to define a modified isometry V that maps the efficient observables $\{Z(\vec{a})X(\vec{b})\}$ to the corresponding Pauli observables. This is a stronger version of Equation (2.1) and, combined again with the verifier's checks in the protocol and properties of ENTCF families, can be used to show that the prover must have prepared BB84 states.

We briefly comment on the relation between our soundness proof and that in [36]. At a high level, the soundness proof in [36] also shows a kind of "parallel rigidity" of two executions of a remote state preparation protocol. However, their proof proceeds quite differently from ours: they first show that observables "on the first qubit" anti-commute, which allows them to make a partial statement about the prover's state. This in turn can be used to extend the statement about the prover's observables to two-qubit observables, which is finally used to prove a statement about the prover's two-qubit state. This qubit-by-qubit approach is extremely costly in terms of parameters due to switching back and forth between making partial statements about the observables and state, and cannot reasonably be extended to n qubits. In contrast, we can make a global statement about the prover's 4^n possible observables without first characterising parts of the prover's state. This allows us to prove a parallel rigidity statement for n qubits without an exponential degradation of parameters.

3 Applications

Having introduced our parallel RSP theorem, we can turn to its cryptographic applications. We consider various cryptographic primitives that have previously been defined and constructed in a setting where one party sends random BB84 states to the other. For each primitive, we give a formal definition of the "classical-client version" and show that this definition can be satisfied using our parallel RSP protocol as a building block. Since our parallel RSP protocols relies on the LWE assumption, so do the dequantised protocols we present here. Furthermore, Theorem 1 only guarantees the preparation of BB84 states up to an inverse polynomial error, so as a result, the dequantised protocols only have inverse polynomial security (see Section 5

for a discussion of this point). Some of these primitives have previously been dequantised using an application-specific approach (and similarly relying on computational assumptions) [24, 13, 42, 28, 32]; in contrast, our approach is *generic* and simply uses RSP to replace the sending of BB84 states. We give a short overview of the different applications and refer to the full manuscript for details.

Unclonable quantum encryption. As a first application of our parallel RSP protocol, we consider the notion of *unclonable quantum encryption*. This cryptographic functionality was coined by Gottesman [25] and then formalised by Broadbent and Lord [11]. In a private-key unclonable quantum encryption scheme, a classical message is encrypted into a quantum state (the *quantum ciphertext*) with the following property: given only a single quantum ciphertext, it is impossible to create two states that can later both be decrypted with access to the private key. We consider an unclonable conjugate coding *hybrid encryption* scheme which is inspired by the work of Broadbent and Lord: a plaintext $\vec{m} \in \{0,1\}^n$ is encrypted with a randomly chosen secret key $k = (\vec{s}, \vec{\theta}) \xleftarrow{\$} \{0,1\}^n \times \{0,1\}^n$ and randomness $\vec{v} \xleftarrow{\$} \{0,1\}^n$ into the quantum ciphertext given by $\mathsf{Enc}_k(\vec{m}) = \bigotimes_{i=1}^n H^{\theta_i}|v_i\rangle\langle v_i|H^{\theta_i} \otimes |\vec{v} \oplus \vec{s} \oplus \vec{m}\rangle\langle \vec{v} \oplus \vec{s} \oplus \vec{m}|$. To decrypt using the secret key $k = (\vec{s}, \vec{\theta})$, one applies $H^{\theta_1} \otimes \cdots \otimes H^{\theta_n}$ to the first half of the ciphertext, measures in the computational basis with outcome $\vec{x}$, and then uncomputes the one-time pad in the second half using $\vec{x}$ and $\vec{s}$. The fact that this scheme is unclonable is a consequence of the *monogamy of entanglement* [11, 46].

To dequantise this protocol, we consider a scenario in which a classical client $\mathcal{C}$ wishes to delegate an unclonable ciphertext to a quantum receiver $\mathcal{R}$. As a first step, $\mathcal{C}$ and $\mathcal{R}$ run our parallel RSP protocol to delegate a collection of random BB84 states of the form $H^{\theta_1}|v_1\rangle \otimes \cdots \otimes H^{\theta_n}|v_n\rangle$, where $\vec{v}, \vec{\theta} \in \{0,1\}^n$ are random strings known only to $\mathcal{C}$. Then, $\mathcal{C}$ can choose $\vec{s} \in \{0,1\}^n$ and output the string $\vec{v} \oplus \vec{s} \oplus \vec{m}$ and set $\vec{k} = (\vec{s}, \vec{\theta})$ as the secret key. With this choice of key, the delegated parallel BB84 states are exactly the ciphertext $\mathsf{Enc}_k(\vec{m})$. Because the final output state of the protocol is computationally indistinguishable from a tensor product of BB84 states (known to the client), we can follow a similar proof as in [11] to obtain a classical-client unclonable encryption scheme with inverse-polynomial security.

Quantum copy-protection. In quantum copy-protection (QCP), a vendor wishes to encode a program into a quantum state in a way that enables a recipient to run the program, but not to create functionally equivalent "pirated" copies. The notion of QCP was introduced by Aaronson [1], who gave the first construction for unlearnable and efficiently computable functions in a strong quantum oracle model, which has since been improved to only requiring classical oracles [2]. Recent work [16] has also provided the first construction of QCP for compute-and-compare programs in the quantum random oracle model (QROM) as well as a scheme for multi-bit point functions in the QROM based on unclonable encryption with wrong-key detection (WKD) – a property which enables the decryption procedure to recognise incorrect keys.

Our QCP scheme for multi-bit point functions combines our unclonable hybrid encryption scheme with the generic WKD transformation in the QROM proposed by Coladangelo et al. [16]. The basic idea behind our QCP scheme is as follows. To encode a point function $P_{\vec{y},\vec{m}}$ (which is defined as returning $\vec{m}$ on input $\vec{y}$ and 0^n, otherwise) we simply output $\mathsf{Enc}_{\vec{y}}(\vec{m})$ together with $h(\vec{y})$, where h is a suitable hash function which we model as a truly random function (in the QROM). To evaluate the program on an input $\vec{x} \in \{0,1\}^{2n}$, we first check whether $\vec{x}$ hashes to $h(\vec{y})$ under h. If true, we decrypt as in the aforementioned hybrid encryption scheme and recover $\vec{m}$. Otherwise, we output 0^n.

We then show how to obtain a QCP scheme with a classical client through the use of our parallel RSP protocol for preparing random BB84 states, similar to our aforementioned classical-client unclonable encryption scheme. Our scheme enables a classical client to delegate a correct copy-protected program from the class of multi-bit point functions consisting of uniformly random marked inputs $\vec{y}$ and output strings $\vec{m}$ with inverse-polynomial security.

Quantum computing on encrypted data. Suppose a client wishes to perform some quantum computation, represented as the action of a quantum circuit C on an input state $|x\rangle$, with $x \in \{0,1\}^n$. For simplicity, we will assume the desired output is classical and corresponds to a computational basis measurement of $C|x\rangle$. The client only has limited quantum capability and therefore wishes to delegate the computation to a quantum server while ensuring the privacy of the input $|x\rangle$ and the output resulting from the measurement of $C|x\rangle$. Essentially, the client would like to send the server an encryption of the input and, after performing an interactive protocol, obtain an encryption of the output (which the client can decrypt, but the server cannot)[3]. This primitive is called quantum computing on encrypted data (QCED).

Many protocols for QCED with differing quantum requirements on the client have been developed (see [20] for a survey). Here we will focus on the protocol of Broadbent [8] which achieves QCED with a client that is only required to prepare BB84 states and send them to the server. This makes the protocol well-suited for dequantisation via our parallel RSP protocol. Before explaining this dequantisation, we (informally) define what a QCED protocol *with a classical client* should achieve. As before, the client's input is the string $x \in \{0,1\}^n$ and the goal is to obtain the outcome of measuring $C|x\rangle$ in the computational basis. In contrast to before, this must be achieved using only *classical* interaction with the quantum server. The requirement that the client's input must stay private is captured by the condition that after interacting with the client, it must be computationally intractable for the server to decide which one of two distinct inputs the client used.

Our QCED protocol with a classical client works as follows. The client first performs the parallel RSP protocol with the server, resulting in the preparation of BB84 states (or the client aborting). Provided the protocol succeeded, the client proceeds to run Broadbent's protocol [8] as if the server had received those BB84 states via a quantum channel. The security proof is straightforward. First, we know that after performing RSP the server's state is computationally indistinguishable from a tensor product of BB84 states (known to the client). Furthermore, the interaction in [8] preserves this computational indistinguishability. Hence, the server's state at the end of the protocol is indistinguishable from the state the server would have obtained by executing the protocol with random BB84 states and the security of our protocol follows from [8].

Verifiable delegated blind quantum computation. The final application we consider is verifiable delegated blind quantum computation (VDBQC). VDBQC is an interactive protocol between two parties, in this case denoted as the verifier and the prover. The verifier delegates a computation to the prover and, in addition to ensuring input-output privacy as in QCED, the protocol also ensures that the probability for the verifier to accept an incorrect output is small. In other words, if the prover deviates from the protocol and does not perform the verifier's instructed computation, the verifier should be able to detect this and abort with high probability. As with QCED, a number of such protocols have been developed and we refer the reader to [23] for a survey.

[3] This also allows the client to hide the computation itself from the server by suitably encoding it as part of the input x and taking C to be a *universal circuit*. When the primary goal of the protocol is to hide the computation, it is referred to as a *blind quantum computing protocol* [3, 9].

Here we focus on a protocol by Morimae [38]. This protocol achieves verifiability by combining a protocol for blind quantum computation (or QCED) with the *history state construction*, which is a special encoding of a quantum circuit into a quantum state [31, 30]. In Morimae's protocol, for a given circuit C the verifier uses a QCED protocol to delegate to the prover the preparation of two such history states (one for C and one for the complement of C, where the output qubit is negated). The verifier then requests these states from the prover and proceeds to measure them in the computational or Hadamard basis. This allows the verifier to determine the output of the computation. The history state construction guarantees that malicious behavior on the prover's part would be detected by the verifier's measurement. Additionally, the use of a QCED protocol ensures that the prover is "blind", i.e. does not know which computation the verifier delegated.

To dequantise this protocol, we use our QCED protocol with a classical client to delegate the preparation of the two history states to the quantum prover. We then replace the verifier's measurements on this state by a *measurement protocol* due to Mahadev [33], which allows the classical verifier to delegate these measurements to the prover in a way that forces the prover to report the correct outcomes. We thus obtain a VDBQC protocol with a classical verifier. Crucially, through the use of the classical client QCED protocol and Mahadev's measurement protocol, the prover is "computationally blind", i.e. unable to distinguish which computation the verifier has performed. In contrast, Mahadev's verification protocol [33] does not have this property.[4]

4 Related work

A number recent of works starting with [7, 33] have developed techniques that allow a classical verifier to use post-quantum cryptography to force an untrusted (but computationally bounded) quantum prover to behave in a certain way. Here, we briefly describe these works and explain their relation to our parallel RSP protocol.

In a breakthrough result [33], Mahadev introduced a protocol that allows a classical verifier to delegate a quantum computation to a quantum computer and be able to verify the correctness of the result. The key ingredient for this protocol is a *measurement protocol*, which allows the verifier to securely delegate single-qubit measurements in the standard or Hadamard basis to a quantum prover, assuming that the prover cannot break the LWE assumption. This can then be applied to so-called *prepare-and-measure protocols*: if one has a protocol that involves a quantum prover preparing and sending a quantum state to the verifier and the verifier performing single-qubit measurements on this state, one can use Mahadev's measurement protocol to delegate these quantum measurements to the prover itself. This yields a protocol in which the prover only sends classical measurement outcomes to the verifier, hence making the verifier classical.

This measurement protocol is in many ways similar to what we seek to do in this paper: it removes the need for quantum communication between a fully quantum prover and a verifier with very limited quantum capabilities (only measuring single qubits in the computational or Hadamard basis). The difference to our work is that we are concerned with *prepare-and-send protocols*, in which the verifier sends random BB84 states to the prover instead of receiving them.

[4] In [24], the authors also construct a blind verification protocol based on RSP. However, they approach the problem in a composable framework, which requires them to make an additional assumption on the prover (called the *measurement buffer* in [24]). In contrast, our protocol requires no extra assumptions on the prover. We describe the issue with the measurement buffer assumption in more detail in Section 4.

It turns out that replacing the quantum communication of prepare-and-send protocols requires significantly stronger control over the untrusted prover. At a high level, the reason is the following: for Mahadev's measurement protocol, it suffices to show that *there exists* a quantum state that is consistent with the distribution of measurement outcomes reported by the prover, in the sense that the measurement outcomes for different bases could have been obtained by measurements on (copies of) the same state. In contrast, if we want to replace the step of the verifier sending a physical quantum state to the prover, we need to show that the prover has *actually constructed* a certain quantum state, not just that such a quantum state exists mathematically.[5] We give a more detailed description of what it means to "actually construct" a quantum state in Section 2.1.

The first classical protocol that provably forced a quantum prover to prepare a certain quantum state was the single-qubit RSP protocol of [24] (see also [13] for a related result). This protocol essentially achieves our informal theorem as stated above for a single qubit, i.e. $n = 1$.[6] At first sight, it might seem as though a simple hybrid argument, which replaces each BB84 qubit with a (sequential) instance of [24], suffices to achieve the multi-qubit task. However, the single-qubit RSP protocol of [24] only ensures that each BB84 qubit can be individually replaced by an RSP protocol up to a *global* isometry. Because the prover's state can be entangled in arbitrary ways between intermediate applications of the protocol, it is difficult to justify that all of the individual replacements together form an actual n-qubit BB84 state; as we explain below, the fact that the protocol from [24] is composable does not remedy this situation, either. While some prior work [19] showed that composable *single-qubit* RSP suffices in the context of quantum verification, one would have to show a similar result for each application of interest. Our parallel RSP protocol, in contrast, can be used in a plug-and-play manner for many cryptographic protocols and applications. In addition, our protocol has fewer rounds than a sequential repetition of [24] and also immediately yields a *proof of quantum space* (a certificate that the prover has a certain number of qubits). We give a brief outline of [24] and its soundness proof in Section 2.1.

The main difficulty in going from [24] to our parallel RSP result is enforcing a tensor product structure on the prover's space: we would like to show that, if we execute multiple instances of a single-qubit RSP protocol in parallel, a successful prover must treat each of these copies independently. Mathematically, this means that we need to be able to split the prover's a priori uncharacterised Hilbert space into a *tensor product*, where each tensor factor is supposed to correspond to one instance of the RSP protocol. This is a more demanding version of the classic question of parallel repetition: there, one is interested in showing that any prover's winning probability in the protocol decays in essentially the same way as it would for a prover who executes the instances independently. In contrast, we need to show that the prover really does execute the different instances independently in a physically meaningful sense. We call this stronger requirement *parallel rigidity*.

In [24], the authors show that their protocol has composable security. This may suggest that one can obtain a parallel rigidity statement simply by composing the protocol with itself in sequence or in parallel. However, this is not the case because the composable security

[5] In fact, in [49] it was shown that Mahadev's measurement protocol does ensure that the prover *knows* (in the sense of a proof of knowledge) the state it is measuring, not just that it exists mathematically. The notion of "knowing" a quantum state is quite subtle to define and we forego a detailed description here, but point out that this is weaker than showing that the prover actually constructed the state and (to the best of our knowledge) not sufficient to use Mahadev's protocol for prepare-and-measure scenarios.

[6] The protocol in [24] allows for the qubit to be prepared in one of 10 possible states which includes the 4 BB84 states. Here, we only focus on the 4 BB84 states as this is the case we will deal with in our parallel RSP protocol.

statement in [24] requires an additional assumption called a *measurement buffer*, which effectively acts as a trusted intermediary between the verifier and the prover. A sequential or parallel composition of the protocol in [24] would utilise a different measurement buffer for each instance, thereby forcing the prover to treat the different instances in a (largely) independent way. In particular, this means that one already assumes a tensor product structure with n separate qubits in the prover's space, whereas in our work enforcing this tensor product structure is the key technical challenge. For cryptographic applications, we do not want to place any such assumption on the prover and instead allow the prover to perform arbitrary global operations involving all instances. This is what our parallel RSP protocol achieves. Furthermore, as shown in [4], achieving a composable single-qubit RSP without the measurement buffer is impossible. This means that one cannot hope to achieve parallel RSP by showing a stronger composable version of single-qubit RSP; instead, it is necessary to directly analyse parallel executions of the protocol, as we do in this paper.

The question of parallel rigidity has been studied extensively in the literature on quantum self-testing [17, 14, 39, 40], where one considers a setting of two non-communicating provers. Unfortunately, those techniques are not immediately transferable to the setting we consider here, namely a single computationally bounded prover.

Some progress towards the question of parallel rigidity for single computationally bounded provers was made in [36], which gives a protocol that allows a classical verifier to certify that a quantum prover must have prepared and measured a Bell state, i.e. an entangled 2-qubit quantum state. This has since been applied to device-independent quantum key distribution [35] and oblivious transfer [12], and been extended to work for magic states [37]. The protocol from [36] uses a 2-fold parallel repetition of [24] (with additional steps to allow for the certification of an entangled state, not just product states). As part of their soundness proof, [36] do show a kind of parallel rigidity result for 2 instances of the RSP protocol. However, their method does not generalise to an n-fold parallel repetition without an exponential decay in parameters. Hence, for our n-fold parallel rigidity proof, new techniques are needed. A more detailed comparison between our new parallel rigidity proof and the method in [36] can be found at the end of Section 2.1. We note that in independent concurrent work, [21] also gave an n-fold parallel rigidity proof in the computational setting, but the class of states they deal with is different from random BB84 pairs and they do not consider the dequantisation of cryptographic protocols.

In addition to this line of work focused on rigidity statements, application-specific dequantisations were already considered for private-key quantum money [22, 42], certifiable deletion of quantum encryption [28] and secure software leasing [32]. In all these cases the authors derived the desired security statement from properties of trapdoor claw-free functions, a cryptographic primitive which is also the basis of our RSP protocol. While this is less generic and modular than our approach and requires a new analysis for each application, it does have the advantage that one can obtain *negligible* security, whereas with RSP we obtain inverse polynomial security. We comment more on the possibility of negligible security from RSP-like primitives in Section 5.

5 Discussion

We have shown how a classical verifier can certify a tensor product of BB84 states in the memory of a quantum prover, assuming the quantum-intractability of the LWE problem. Importantly, the prover does not know which BB84 states it has prepared, whereas the verifier does. Hence, the result at the end of the protocol is as if the verifier had sent random

BB84 states to the prover. This allows us to dequantise a number of quantum cryptographic primitives, yielding a generic and modular way of translating these protocols to a setting where only classical communication is used. We have demonstrated the versatility of this approach by applying it to unclonable encryption, quantum copy-protection, computing on encrypted data, and blind verification. Naturally, we expect that other primitives that rely on BB84 states can also be dequantised using our approach. Examples of this include quantum encryption with certified deletion [10, 41] and private key quantum money [50, 42]. We leave these and other applications to future work.

Apart from applying our technique to dequantise additional cryptographic primitives, our work raises a number of further open problems. Firstly, while our RSP primitive is based on the hardness of LWE, we can ask whether it is possible to achieve this functionality from weaker computational assumptions. For instance, would it be possible to perform an RSP-like protocol assuming only the existence of quantum-secure one-way functions? This is of particular interest because recent results have shown that secure two-party computation can be achieved from one-way functions and quantum communication [5, 27]. These results are based on the fact that an oblivious-transfer protocol can be implemented from one-way functions and quantum communication that consists of BB84 states. However, an RSP primitive like ours would allow one to generically dequantise that quantum communication. Hence if RSP (with sufficiently strong parameters) can be obtained from quantum-secure one-way functions, then secure two-party computation can also be obtained from those functions, together with classical communication. In light of earlier work [29, 45] we conjecture that this is impossible. Formalising this intuition could lead to a better understanding of the minimum assumptions required for performing RSP-like protocols.

Secondly, a more technical open problem concerns the parameters of our rigidity theorem, Theorem 1. As stated above, provided the prover accepts, the state the verifier certifies is $1/\operatorname{poly}(n)$-close to a tensor product of n BB84 states (up to an isometry). The $1/\operatorname{poly}(n)$ closeness means that the soundness error of our dequantised protocols also scales as $1/\operatorname{poly}(n)$. It would be desirable to achieve *negligible* soundness error, particularly when considering composable instances of these protocols. This is not possible with the approach taken in this paper as the statistical argument used in deriving our main theorem will necessarily introduce $1/\operatorname{poly}(n)$ factors. However, it might be possible to circumvent an explicit RSP statement: the advantage of the RSP statement in our paper is that one can use it to dequantise existing protocols easily, but these existing protocols typically only use BB84 states because of their no-cloning properties. Therefore, instead of using an RSP protocol to prepare those states, one could instead try to show a "post-quantum cryptographic no-cloning property" directly that could plausibly be used to dequantise these protocols while preserving negligible soundness.

Finally, we mention that our derivation of the parameters in the rigidity theorem is likely not optimal and could be optimised to improve the efficiency of our protocol. The situation here is similar to that of parallel self-testing in the multi-prover setting, with the first works having round complexity that scaled as a high-degree polynomial [44] and more recent works achieving quasilinear scaling [39, 15]. It would be interesting to see whether ideas from these newer works are also applicable in the setting of parallel remote state preparation.

References

1 Scott Aaronson. Quantum copy-protection and quantum money. *2009 24th Annual IEEE Conference on Computational Complexity*, July 2009. doi:10.1109/ccc.2009.42.

2 Scott Aaronson, Jiahui Liu, Qipeng Liu, Mark Zhandry, and Ruizhe Zhang. New approaches for quantum copy-protection. In *Annual International Cryptology Conference*, pages 526–555. Springer, 2021.

3 Pablo Arrighi and Louis Salvail. Blind quantum computation. *International Journal of Quantum Information*, 4(05):883–898, 2006.

4 Christian Badertscher, Alexandru Cojocaru, Léo Colisson, Elham Kashefi, Dominik Leichtle, Atul Mantri, and Petros Wallden. Security limitations of classical-client delegated quantum computing. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 667–696. Springer, 2020.

5 James Bartusek, Andrea Coladangelo, Dakshita Khurana, and Fermi Ma. One-way functions imply secure computation in a quantum world. In *Annual International Cryptology Conference*, pages 467–496. Springer, 2021.

6 C. H. Bennett and G. Brassard. Quantum cryptography: Public key distribution and coin tossing. In *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, pages 8, vol. 175, 1984.

7 Z. Brakerski, P. Christiano, U. Mahadev, U. Vazirani, and T. Vidick. A cryptographic test of quantumness and certifiable randomness from a single quantum device. *IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 320–331, 2018. doi:10.1109/FOCS.2018.00038.

8 Anne Broadbent. Delegating private quantum computations. *Canadian Journal of Physics*, 93(9):941–946, 2015.

9 Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 517–526. IEEE, 2009.

10 Anne Broadbent and Rabib Islam. Quantum encryption with certified deletion. In *Theory of Cryptography*, pages 92–122. Springer International Publishing, 2020. doi:10.1007/978-3-030-64381-2_4.

11 Anne Broadbent and Sébastien Lord. Uncloneable Quantum Encryption via Oracles. In Steven T. Flammia, editor, *15th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2020)*, volume 158 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:22, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.TQC.2020.4.

12 Anne Broadbent and Peter Yuen. Device-independent oblivious transfer from the bounded-quantum-storage-model and computational assumptions. *arXiv preprint*, 2021. arXiv:2111.08595.

13 Alexandru Cojocaru, Léo Colisson, Elham Kashefi, and Petros Wallden. QFactory: Classically-instructed remote secret qubits preparation. *Advances in Cryptology - ASIACRYPT 2019, Lecture Notes in Computer Science, Springer*, pages 615–645, 2019. doi:10.1007/978-3-030-34578-5_22.

14 Andrea Coladangelo. Parallel self-testing of (tilted) EPR pairs via copies of (tilted) CHSH. *arXiv preprint*, 2016. arXiv:1609.03687.

15 Andrea Coladangelo, Alex B Grilo, Stacey Jeffery, and Thomas Vidick. Verifier-on-a-leash: new schemes for verifiable delegated quantum computation, with quasilinear resources. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 247–277. Springer, 2019.

16 Andrea Coladangelo, Christian Majenz, and Alexander Poremba. Quantum copy-protection of compute-and-compare programs in the quantum random oracle model. *arXiv preprint*, 2020. arXiv:2009.13865.

17 Matthew Coudron and Anand Natarajan. The parallel-repeated magic square game is rigid. *arXiv preprint*, 2016. arXiv:1609.06306.

18 W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. doi:10.1109/TIT.1976.1055638.

19 Vedran Dunjko and Elham Kashefi. Blind quantum computing with two almost identical states, 2016. doi:10.48550/arXiv.1604.01586.

20 Joseph F Fitzsimons. Private quantum computation: an introduction to blind quantum computing and related protocols. *npj Quantum Information*, 3(1):1–11, 2017.

21 Honghao Fu, Daochen Wang, and Qi Zhao. Computational self-testing of multi-qubit states and measurements. *arXiv preprint*, 2022. arXiv:2201.13430.

22 Dmitry Gavinsky. Quantum money with classical verification. In *2012 IEEE 27th Conference on Computational Complexity*, pages 42–52. IEEE, 2012.

23 Alexandru Gheorghiu, Theodoros Kapourniotis, and Elham Kashefi. Verification of quantum computation: An overview of existing approaches. *Theory of computing systems*, 63(4):715–808, 2019.

24 Alexandru Gheorghiu and Thomas Vidick. Computationally-secure and composable remote state preparation. *IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1024–1033, 2019. doi:10.1109/FOCS.2019.00066.

25 Daniel Gottesman. Uncloneable encryption. *Quantum Information and Computation*, pages 3:581–602, 2003.

26 William Timothy Gowers and Omid Hatami. Inverse and stability theorems for approximate representations of finite groups. *Sbornik: Mathematics*, 208(12):1784, 2017. doi:10.1070/SM8872.

27 Alex B Grilo, Huijia Lin, Fang Song, and Vinod Vaikuntanathan. Oblivious transfer is in miniqcrypt. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 531–561. Springer, 2021.

28 Taiga Hiroka, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa. Quantum encryption with certified deletion, revisited: Public key, attribute-based, and classical communication. In *Advances in Cryptology – ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part I*, pages 606–636, Berlin, Heidelberg, 2021. Springer-Verlag. doi:10.1007/978-3-030-92062-3_21.

29 Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 44–61, 1989.

30 Julia Kempe, Alexei Kitaev, and Oded Regev. The complexity of the local hamiltonian problem. *Siam journal on computing*, 35(5):1070–1097, 2006.

31 Alexei Yu Kitaev, Alexander Shen, Mikhail N Vyalyi, and Mikhail N Vyalyi. *Classical and quantum computation*, volume 47. American Mathematical Soc., 2002.

32 Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa. Secure software leasing from standard assumptions. In *Theory of Cryptography Conference*, pages 31–61. Springer, 2021.

33 Urmila Mahadev. Classical verification of quantum computations. *IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 259–267, 2018. doi:10.1109/FOCS.2018.00033.

34 M McKague, T H Yang, and V Scarani. Robust self-testing of the singlet. *Journal of Physics A: Mathematical and Theoretical*, 45(45):455304, October 2012. doi:10.1088/1751-8113/45/45/455304.

35 Tony Metger, Yfke Dulek, Andrea Coladangelo, and Rotem Arnon-Friedman. Device-independent quantum key distribution from computational assumptions. *New Journal of Physics*, 23(12):123021, 2021.

36 Tony Metger and Thomas Vidick. Self-testing of a single quantum device under computational assumptions. *arXiv preprint*, 2020. arXiv:2001.09161.

37 Akihiro Mizutani, Yuki Takeuchi, Ryo Hiromasa, Yusuke Aikawa, and Seiichiro Tani. Computational self-testing for entangled magic states. *arXiv preprint*, 2021. arXiv:2111.02700.

38 Tomoyuki Morimae. Blind quantum computing can always be made verifiable. *arXiv preprint*, 2018. arXiv:1803.06624.

39 Anand Natarajan and Thomas Vidick. A quantum linearity test for robustly verifying entanglement. *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1003–1015, 2017. doi:10.1145/3055399.3055468.

40 Anand Natarajan and Thomas Vidick. Low-degree testing for quantum states, and a quantum entangled games PCP for QMA. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 731–742. IEEE, 2018.

41 Alexander Poremba. Quantum proofs of deletion for learning with errors, 2022. doi:10.48550/ARXIV.2203.01610.

42 Roy Radian and Or Sattath. Semi-quantum money. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, AFT '19, pages 132–146, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3318041.3355462.

43 Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009. doi:10.1145/1568318.1568324.

44 Ben W Reichardt, Falk Unger, and Umesh Vazirani. Classical command of quantum systems. *Nature*, 496(7446):456–460, 2013.

45 Steven Rudich. The use of interaction in public cryptosystems. In *Annual International Cryptology Conference*, pages 242–251. Springer, 1991.

46 Marco Tomamichel, Serge Fehr, Jedrzej Kaniewski, and Stephanie Wehner. A monogamy-of-entanglement game with applications to device-independent quantum cryptography. *New Journal of Physics*, 15(10):103002, October 2013. doi:10.1088/1367-2630/15/10/103002.

47 Thomas Vidick. *The complexity of entangled games*. PhD thesis, UC Berkeley, 2011. URL: https://digitalassets.lib.berkeley.edu/etd/ucb/text/Vidick_berkeley_0028E_11907.pdf.

48 Thomas Vidick. Course FSMP, Fall'20: Interactions with quantum devices. http://users.cms.caltech.edu/~vidick/teaching/fsmp/fsmp.pdf, 2020.

49 Thomas Vidick and Tina Zhang. Classical proofs of quantum knowledge. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 630–660. Springer, 2021.

50 Stephen Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, January 1983. doi:10.1145/1008908.1008920.

51 William K Wootters and Wojciech H Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, 1982.

Parameterised and Fine-Grained Subgraph Counting, Modulo 2*

Leslie Ann Goldberg
Department of Computer Science, University of Oxford, UK

Marc Roth
Department of Computer Science, University of Oxford, UK

Abstract

Given a class of graphs $\mathcal{H}$, the problem $\oplus\textsc{Sub}(\mathcal{H})$ is defined as follows. The input is a graph $H \in \mathcal{H}$ together with an arbitrary graph G. The problem is to compute, modulo 2, the number of subgraphs of G that are isomorphic to H. The goal of this research is to determine for which classes $\mathcal{H}$ the problem $\oplus\textsc{Sub}(\mathcal{H})$ is fixed-parameter tractable (FPT), i.e., solvable in time $f(|H|) \cdot |G|^{O(1)}$.

Curticapean, Dell, and Husfeldt (ESA 2021) conjectured that $\oplus\textsc{Sub}(\mathcal{H})$ is FPT if and only if the class of allowed patterns $\mathcal{H}$ is *matching splittable*, which means that for some fixed B, every $H \in \mathcal{H}$ can be turned into a matching (a graph in which every vertex has degree at most 1) by removing at most B vertices.

Assuming the randomised Exponential Time Hypothesis, we prove their conjecture for (I) all hereditary pattern classes $\mathcal{H}$, and (II) all tree pattern classes, i.e., all classes $\mathcal{H}$ such that every $H \in \mathcal{H}$ is a tree. We also establish almost tight fine-grained upper and lower bounds for the case of hereditary patterns (I).

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Mathematics of computing → Discrete mathematics

Keywords and phrases modular counting, parameterised complexity, fine-grained complexity, subgraph counting

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.68

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `http://arxiv.org/abs/2301.01696`

Acknowledgements We want to thank Radu Curticapean, Holger Dell and Thore Husfeldt for insightful discussions on an early draft of this work.

1 Introduction

The last two decades have seen remarkable progress in the classification of subgraph counting problems: Given a small *pattern* graph H and a large *host* graph G, how often does H occur as a subgraph if G? Since it was discovered that subgraph counts from small patterns reveal global properties of complex networks [26, 27], subgraph counting has also found several applications in fields such as biology [2, 30] genetics [32], phylogeny [25], and data mining [33]. Moreover, the theoretical study of subgraph counting and related problems has led to many deep structural insights, establishing both new algorithmic techniques and tight lower bounds under the lenses of fine-grained and parameterised complexity theory [19, 16, 10, 14, 13, 6, 4].

* For the purpose of Open Access, the authors have applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission. All data is provided in full in the results section of this paper.

Without any additional restrictions, the subgraph counting problem is infeasible. The complexity class #W[1] is the parameterised complexity class analogous to NP (see Section 2 for more detail). Under standard assumptions, problems that are #W[1]-hard are not *fixed-parameter tractable* (FPT). The canonical complete problem for #W[1], the problem of counting k-cliques, corresponds to the special case of the subgraph counting problem where H is a clique of size k. This problem cannot be solved in time $f(k) \cdot n^{o(k)}$ for any function f unless the Exponential Time Hypothesis (ETH) fails [8, 9]. Due to this hardness result, the research focus in this area shifted to the question: Under which restrictions on the patterns H and the hosts G is algorithmic progress possible? More precisely, under which restrictions can the problem be solved in time $f(|H|) \cdot |G|^{O(1)}$, for some computable function f? Instances that can be solved within such a run time bound are called *fixed-parameter tractable* (FPT); allowing a potential super-polynomial overhead in the size of the pattern $|H|$ formalises the assumption that H is assumed to be (significantly) smaller than G.

If only the patterns are restricted, then the situation is fully understood. Formally, given a class $\mathcal{H}$ of patterns, the problem #Sub($\mathcal{H}$) asks, given as input a graph $H \in \mathcal{H}$ and an arbitrary graph G, to compute the number of subgraphs of G that are isomorphic to H. Following initial work by Flum and Grohe [19] and by Curticapean [11], Curticapean and Marx [14] proved that, under standard assumptions, #Sub($\mathcal{H}$) is FPT if and only if $\mathcal{H}$ has bounded matching number, that is, if there is a positive integer B such that the size of any matching in any graph in $\mathcal{H}$ is at most B. They also proved that all FPT cases are polynomial-time solvable.

In stark contrast, almost nothing is known for the decision version Sub($\mathcal{H}$). Here, the task is to correctly decide whether there is a copy of $H \in \mathcal{H}$ in G, rather than to count the copies. It is known that Sub($\mathcal{H}$) is FPT whenever $\mathcal{H}$ has bounded treewidth (see e.g. [20, Chapter 13]), and it is conjectured that those are all FPT cases. However, resolving this conjecture belongs to the "most infamous" open problems in parameterised complexity theory [18, Chapter 33.1].

1.1 Counting Modulo 2

To interpolate between the fully understood realm of (exact) counting and the barely understood realm of decision, Curticapean, Dell and Husfeldt proposed the study of counting subgraphs, modulo 2 [12]. Formally, they introduced the problem $\oplus$Sub($\mathcal{H}$), which expects as input a graph $H \in \mathcal{H}$ and an arbitrary graph G, and the goal is to compute *modulo 2* the number of subgraphs of G isomorphic to H.

The study of counting modulo 2 received significant attention from the viewpoint of classical, structural, and fine-grained complexity theory. For example, one way to state Toda's Theorem [31] is PH $\subseteq$ $\mathrm{P}^{\oplus\mathrm{P}}$, implying that counting satisfying assignments of a CNF, modulo 2, is at least as hard as the polynomial hierarchy. Another example is the quest to classify the complexity of counting modulo 2 the homomorphisms to a fixed graph, which was very recently resolved by Bulatov and Kazeminia [7]. There has also been work by Abboud, Feller, and Weimann [1] on the fine-grained complexity of counting modulo 2 the number of triangles in a graph that satisfy certain weight constraints.

In their work [12], Curticapean, Dell and Husfeldt proved that the problem of counting k-matchings modulo 2, that is, the problem $\oplus$Sub($\mathcal{H}$) where $\mathcal{H}$ is the class of all 1-regular graphs, is fixed-parameter tractable, where the parameter k is $|H|$. Since the exact counting version of this problem is #W[1]-hard [11], their result provides an example where counting modulo 2 is *strictly* easier than exact counting (subject to complexity assumptions). The complexity class $\oplus$W[1] can be defined via the complete problem of counting k-cliques

modulo 2. Crucially, $\oplus$W[1]-hard problems are not fixed-parameter tractable, unless the randomised ETH (rETH) fails. Curticapean et al. [12] proved that counting k-paths modulo 2 is $\oplus$W[1]-hard. Since *finding* a k-path in a graph G is fixed-parameter tractable via colour-coding [3], this hardness result provides an example where counting modulo 2 is *strictly* harder than decision (subject to complexity assumptions). Combining those observations, it appears that counting subgraphs modulo 2 may lie strictly in between the complexity of decision and the complexity of exact counting.

A *matching* is a graph whose maximum degree is at most 1. The *matching-split number* of a graph H is the minimum size of a set $S \subseteq V(H)$ such that $H \setminus S$ is a matching. A class of graphs $\mathcal{H}$ is called *matching splittable* if there is a positive integer B such that the matching-split number of any $H \in \mathcal{H}$ is at most B. For example, the class of all matchings is matching splittable while the class of all cycles is not. Curticapean, Dell and Husfeldt extended their FTP algorithm for counting k-matchings modulo 2 to obtain an FPT algorithm for $\oplus\textsc{Sub}(\mathcal{H})$ for any matching-splittable class $\mathcal{H}$. On this basis, they then made the following conjecture.

▶ **Conjecture 1** ([12]). *$\oplus\textsc{Sub}(\mathcal{H})$ is FPT if and only if $\mathcal{H}$ is matching splittable.*

A class $\mathcal{H}$ of graphs is called *hereditary* if it is closed under vertex removal. Intriguingly, if Conjecture 1 is true, then the FPT criterion for counting subgraphs modulo 2 ($\oplus\textsc{Sub}(\mathcal{H})$) would coincide with the polynomial-time criterion for finding subgraphs ($\textsc{Sub}(\mathcal{H})$) for hereditary pattern classes $\mathcal{H}$ as established by Jansen and Marx.

▶ **Theorem 2** ([24]). *Let $\mathcal{H}$ be a hereditary class of graphs and assume* P $\neq$ NP. *Then $\textsc{Sub}(\mathcal{H})$ is solvable in polynomial time if and only if $\mathcal{H}$ is matching splittable.*

Jansen and Marx also conjecture that the condition of $\mathcal{H}$ being hereditary can be removed.

▶ **Conjecture 3** ([24]). *$\textsc{Sub}(\mathcal{H})$ is solvable in polynomial time if and only if $\mathcal{H}$ is matching splittable.*

Conjectures 1 and 3 have the remarkable consequence that $\oplus\textsc{Sub}(\mathcal{H})$ is FPT if and only if $\textsc{Sub}(\mathcal{H})$ is solvable in polynomial time. In the current work we establish this consequence for all hereditary pattern classes.

1.2 Our Contributions

We resolve Conjecture 1 for all hereditary classes $\mathcal{H}$, as well as for every class $\mathcal{H}$ consisting only of trees; note that the upper bounds were shown in [12] and that the lower bounds are the novel part.

▶ **Theorem 4.** *Let $\mathcal{H}$ be a hereditary class of graphs. If $\mathcal{H}$ is matching splittable, then $\oplus\textsc{Sub}(\mathcal{H})$ is fixed-parameter tractable. Otherwise, the problem is $\oplus$W[1]-complete and, assuming rETH, cannot be solved in time $f(|H|) \cdot |G|^{o(|V(H)|/\log|V(H)|)}$ for any function f.*

▶ **Theorem 5.** *Let $\mathcal{T}$ be a recursively enumerable class of trees. If $\mathcal{T}$ is matching splittable, then $\oplus\textsc{Sub}(\mathcal{T})$ is fixed-parameter tractable. Otherwise $\oplus\textsc{Sub}(\mathcal{T})$ is $\oplus$W[1]-complete.*

The requirement that the class of trees $\mathcal{T}$ needs to be recursively enumerable is a standard technicality - the reason for it is that the function f in the running time in the standard definition of an FPT algorithm is required to be computable. It turns out that having $\mathcal{T}$ recursively enumerable is enough for this.

In order to prove our classifications, we adapt the by-now-standard technique for analysing subgraph counting problems established by Curticapean, Dell and Marx [13]. Let $\#\mathsf{Sub}(H \to G)$ denote the number of subgraphs of a graph G that are isomorphic to a

graph H and let $\#\mathsf{Hom}(F \to G)$ denotes the number of homomorphisms (edge-preserving mappings) from a graph F to a graph G. Given a graph H, there is a function a_H from graphs to rationals with finite support such that the following holds for any graph G:

$$\#\mathsf{Sub}(H \to G) = \sum_F a_H(F) \cdot \#\mathsf{Hom}(F \to G)\,, \tag{1}$$

where the sum is over all (isomorphism types of) graphs. Since a_H has finite support, $a_H(F) = 0$ for all but finitely-many graphs F. Thus, equation (1) allows us to express the solution to the *exact* counting problem as a finite linear combination of homomorphism counts. In a nutshell, the framework of [13] states that computing the function $G \mapsto \#\mathsf{Sub}(H \to G)$ is hard to compute if and only if there is a graph F of high treewidth with $a_H(F) \neq 0$. This translates the complexity of (exact) subgraph counting to the purely combinatorial problem of understanding the coefficients a_H. One might hope that this strategy transfers to counting modulo 2 as well. Unfortunately, this is not possible as Equation (1) might not be well-defined if arithmetic is done modulo 2. The reason for this is the fact that the coefficients $a_H(F)$ are of the form $\mu(F,H) \times |\mathsf{Aut}(H)|^{-1}$, where $\mu(F,H)$ is an integer, and $\mathsf{Aut}(H)$ is the automorphism group of the graph H [13]. Thus there is, a priori, no hope to extend the framework to counting modulo 2 for pattern graphs with an even number of automorphisms. In fact, according to Curticapean, Dell and Husfeldt [12], the absence of a comparable framework for counting modulo 2 is one of the main challenges for establishing the hardness part of Conjecture 1, and it is the main reason why the reductions in [12] use more classical, gadget-based reductions.

In this work, we solve the problem of patterns with an even number of automorphisms by considering a colourful intermediate problem. More concretely, we will equip each edge of the pattern H with a distinct colour and show that it will be sufficient to consider only automorphisms that preserve the colours. If H has no isolated vertices, then this is only the trivial automorphism. Formally, the coloured approach will be based on the notion of so-called *fractured* graphs introduced by Peyerimhoff et al. [29].

In what follows (Section 2), we will first introduced all required notions and previous results. In Section 3, we will prove the classification for hereditary pattern classes (Theorem 4). On a technical level, this proof can be considered a warm-up for the significantly harder challenge of establishing the classification for trees (Theorem 5), which we defer to the full version due to the space constraints.

2 Preliminaries

Let $f : A_1 \times A_2 \to B$ be a function. For each $a_1 \in A_1$ we write $f(a_1, \star) : A_2 \to B$ for the function that maps $a_2 \in A_2$ to $f(a_1, a_2)$.

Graphs in this work are undirected and without self loops. A *homomorphism* from a graph H to a graph G is a mapping φ from the vertices $V(H)$ of H to the vertices $V(G)$ of G such that for each edge $e = \{u, v\} \in E(H)$ of H, the image $\varphi(e) = \{\varphi(u), \varphi(v)\}$ is an edge of G. A homomorphism is called an *embedding* if it is injective. We write $\mathsf{Hom}(H \to G)$ and $\mathsf{Emb}(H \to G)$ for the sets of homomorphisms and embeddings, respectively, from H to G. An embedding $\varphi \in \mathsf{Emb}(H \to G)$ is called an *isomorphism* if it is bijective and $\{u, v\} \in E(H) \Leftrightarrow \{\varphi(u), \varphi(v)\} \in E(G)$. We say that H and G are *isomorphic*, denoted by $H \cong G$, if an isomorphism from H to G exists. A *graph invariant* ι is a function from graphs to rationals such that $\iota(H) = \iota(G)$ for each pair of isomorphic graphs H and G.

A *subgraph* of G is a graph G' with $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. We write $\mathsf{Sub}(H \to G)$ for the set of all subgraphs of G that are isomorphic to H. Given a subset of vertices $S \subseteq V(G)$ of a graph G, we write $G[S]$ for the graph induced by S, that is, $G[S]$ has vertices S and edges $\{\{u,v\} \subseteq S \mid \{u,v\} \in E(G)\}$.

We denote by $\mathsf{tw}(G)$ the *treewidth* of the graph G. Since we will rely on treewidth purely in a black-box manner, we omit the technical definition and refer the reader to [15, Chapter 7].

Given any graph invariant ι (such as treewidth) and a class of graphs $\mathcal{G}$, we say that ι is *bounded* in $\mathcal{G}$ if there is a non-negative integer B such that, for all $G \in \mathcal{G}$, $\iota(G) \leq B$. Otherwise we say that ι is *unbounded* in $\mathcal{G}$.

Given a graph $H = (V, E)$, a *splitting set* of H is a subset of vertices S such that every vertex in $H[V \setminus S]$ has degree at most 1. The *matching-split number* of H is the minimum size of a splitting set of H. A class of graphs $\mathcal{H}$ is called *matching splittable* if the matching-split number of $\mathcal{H}$ is bounded.

2.1 Colour-Preserving Homomorphisms and Embeddings

A homomorphism c from a graph G to a graph Q is sometimes called a "Q-colouring" of G. A *Q-coloured* graph is a pair consisting of a graph G and a homomorphism c from G to Q. Note that the identity function id_Q on $V(Q)$ is a Q-colouring of Q. If a homomorphism c from G to Q is vertex surjective, then we call (G, c) a *surjectively* Q-coloured graph.

▶ **Definition 6** (c_{E}). *A Q-colouring c of a graph G induces a (not necessarily proper) edge-colouring $c_E\colon E(G) \to E(Q)$ given by $c_E(\{u,v\}) = \{c(u), c(v)\}$.*

Notation. Given a Q-coloured graph (G, c) and a vertex $u \in V(Q)$, we will use the capitalised letter U to denote the subset of vertices of G that are coloured by c with u, that is, $U := c^{-1}(u) \subseteq V(G)$.

Given two Q-coloured graphs (H, c_H) and (G, c_G), we call a homomorphism φ from H to G *colour-preserving* if for each $v \in V(H)$ we have $c_G(\varphi(v)) = c_H(v)$. We note the special case in which $Q = H$ and c_H is the identity id_Q; then the condition simplifies to $c_G(\varphi(v)) = v$. A colour-preserving embedding of (H, c_H) in (G, c_G) is a vertex injective colour-preserving homomorphism from (H, c_H) to (G, c_G). We write $\mathsf{Hom}((H, c_H) \to (G, c_G))$ and $\mathsf{Emb}((H, c_H) \to (G, c_G))$ for the sets of all colour-preserving homomorphisms and embeddings, respectively, from (H, c_H) to (G, c_G).

Let k be a positive integer, let H be a graph with k edges, and let (G, γ) be a pair consisting of a graph G and a function that maps each edge of G to one of k distinct colours. We refer to γ as a "k-edge colouring" of G. For example, in most of our applications we will fix a graph Q with k edges and a Q-colouring c of G and we will take γ to be the edge-colouring c_E from Definition 6. We write $\mathsf{ColSub}(H \to (G, \gamma))$ for the set of all subgraphs of G that are isomorphic to H and that contain each of the k edge colours precisely once.

2.2 Fractures and Fractured Graphs

In this work, we will crucially rely on and extend the framework of *fractured* graphs as introduced in [29].

▶ **Definition 7** (Fractures). *Let Q be a graph. For each vertex v of Q, let $E_Q(v)$ be the set of edges of Q that are incident to v. A* fracture *of Q is a tuple $\rho = (\rho_v)_{v \in V(Q)}$, where for each vertex v of Q, ρ_v is a partition of $E_Q(v)$.*

Figure 1 Illustration of the construction of a fractured graph from [29]. The left picture shows a vertex v of a graph Q with incident edges $E_Q(v) = \{\bullet, \bullet, \bullet, \bullet, \bullet, \bullet\}$. The right picture shows the splitting of v in the construction of the fractured graph $Q\sharp\sigma$ for a fracture σ satisfying that the partition σ_v contains two blocks $B_1 = \{\bullet, \bullet, \bullet\}$, and $B_2 = \{\bullet, \bullet, \bullet\}$.

Note that a fracture describes how to split (or how to *fracture*) each vertex of a given graph: for each vertex v, create a vertex v^B for each block B in the partition ρ_v; edges originally incident to v are made incident to v^B if and only if they are contained in B. We call the resulting graph the *fractured graph* $H\sharp\rho$; a formal definition is given in Definition 8, a visualisation is given in Figure 1.

▶ **Definition 8** (Fractured Graph $Q\sharp\rho$)**.** *Given a graph Q, we consider the matching M_Q containing one edge for each edge of Q; formally,*

$$V(M_Q) := \bigcup_{e=\{u,v\}\in E(Q)} \{u_e, v_e\} \quad \text{and} \quad E(M_Q) := \{\{u_e, v_e\} \mid e = \{u, v\} \in E(Q)\}.$$

For a fracture ρ of Q, we define the graph $Q\sharp\rho$ to be the quotient graph of M_Q under the equivalence relation on $V(M_Q)$ which identifies two vertices v_e, w_f of M_Q if and only if $v = w$ and e, f are in the same block B of the partition ρ_v of $E_Q(v)$. We write v^B for the vertex of $Q\sharp\rho$ given by the equivalence class of the vertices v_e (for which $e \in B$) of M_Q.

▶ **Definition 9** (Canonical Q-colouring c_ρ)**.** *Let Q be a graph and let ρ be a fracture of Q. The* canonical Q-colouring *of the fractured graph $Q\sharp\rho$ maps v^B to v for each $v \in V(Q)$ and block $B \in \rho_v$, and is denoted by c_ρ.*

Observe that c_ρ is the identity in $V(Q)$ if ρ is the coarsest fracture (that is, each partition ρ_v only contains one block, in which case $Q\sharp\rho = Q$).

2.3 Parameterised and Fine-grained Computation

A *parameterised computational problem* is a pair consisting of a function $P : \Sigma^* \to \{0, 1\}$ and a computable parameterisation $\kappa : \Sigma^* \to \mathbb{N}$. A *fixed-parameter tractable* (FPT) *algorithm* for (P, κ) is an algorithm that computes P and runs, on input $x \in \Sigma^*$, in time $f(\kappa(x)) \cdot |x|^{O(1)}$ for some computable function f. We call (P, κ) *fixed-parameter tractable* (FPT) if an FPT algorithm for (P, κ) exists.

A parameterised Turing-reduction from (P, κ) to (P', κ') is an FPT algorithm for (P, κ) that is equipped with oracle access to P' and for which there is a computable function g such that, on input x, each oracle query y satisfies $\kappa'(y) \leq g(\kappa(x))$. We write $(P, \kappa) \leq_{\mathrm{T}}^{\mathrm{fpt}} (P', \kappa')$ if a parameterised Turing-reduction from (P, κ) to (P', κ') exists. This guarantees that fixed-parameter tractability of (P', κ') implies fixed-parameter tractability of (P, κ). For a more comprehensive introduction, we refer the reader the standard textbooks [15] and [20].

Counting modulo 2 and the rETH

The lower bounds in this work will rely on the hardness of the parameterised complexity class $\oplus\mathsf{W}[1]$, which can be considered a parameterised equivalent of $\oplus\mathsf{P}$. Following [12], we define $\oplus\mathsf{W}[1]$ via the complete problem $\oplus$Clique: Given as input a graph G and a positive integer k, the goal is to compute the number of k-cliques in G modulo 2, i.e., to compute $\oplus\mathsf{Sub}(K_k \to G)$. The problem is parameterised by k. A parameterised problem (P, κ) is called $\oplus\mathsf{W}[1]$-*hard* if $\oplus$Clique $\leq_{\mathrm{T}}^{\mathrm{fpt}} (P, \kappa)$, and it is called $\oplus\mathsf{W}[1]$-*complete* if, additionally, $(P, \kappa) \leq_{\mathrm{T}}^{\mathrm{fpt}} \oplus$Clique.

Modifications of the classical Isolation Lemma (see e.g. [5] and [34]) yield a *randomised* parameterised Turing reduction from finding a k-clique to computing the parity of the number of k-cliques. In combination with existing fine-grained lower bounds for finding a k-clique [8, 9], it can then be shown that $\oplus$Clique cannot be solved in time $f(k) \cdot |G|^{o(k)}$ for any function f, unless the randomised Exponential Time Hypothesis fails:

▶ **Definition 10** (rETH, [23]). *The* randomised Exponential Time Hypothesis *(rETH) asserts that 3-SAT cannot be solved by a randomised algorithm in time* $\exp o(n)$*, where n is the number of variables of the input formula.*

As an immediate consequence, the rETH implies that $\oplus\mathsf{W}[1]$-hard problems are not fixed-parameter tractable.

For the lower bounds in this work, we won't reduce from $\oplus$Clique directly, but instead from the following, more general problem:

▶ **Definition 11** ($\oplus$cp-Hom). *Let $\mathcal{H}$ be a class of graphs. The problem $\oplus$cp-Hom($\mathcal{H}$) has as input a graph $H \in \mathcal{H}$ and a surjectively H-coloured graph (G, c). The goal is to compute $\oplus\mathsf{Hom}((H, \mathsf{id}_H) \to (G, c))$. The problem is parameterised by $|H|$.*

The following lower bound was proved independently in [28, 29] and [12].

▶ **Theorem 12.** *Let $\mathcal{H}$ be a recursively enumerable class of graphs. If the treewidth of $\mathcal{H}$ is unbounded then $\oplus$cp-Hom($\mathcal{H}$) is $\oplus\mathsf{W}[1]$-hard and, assuming the rETH, it cannot be solved in time $f(|H|) \cdot |G|^{o(\mathsf{tw}(H)/\log \mathsf{tw}(H))}$ for any function f.*

Next is the central problem in this work.

▶ **Definition 13** ($\oplus$Sub). *Let $\mathcal{H}$ be a class of graphs. The problem $\oplus$Sub($\mathcal{H}$) has as input a graph $H \in \mathcal{H}$ and a graph G. The goal is to compute $\oplus\mathsf{Sub}(H \to G)$. The problem is parameterised by $|H|$.*

For example, writing $\mathcal{K}$ for the set of all complete graphs, the problem $\oplus$Sub($\mathcal{K}$) is equivalent to $\oplus$Clique.

Complexity Monotonicity and Inclusion-Exclusion

Throughout this work, we will rely on two important tools introduced in [29]. For the sake of being self-contained, we encapsulate them below in individual lemmas.

The first tool is an adaptation of the so-called Complexity Monotonicity principle to the realm of fractured graphs and modular counting (see [29, Sections 4.1 and 6.3] for a detailed treatment and for a proof). Intuitively, the subsequent lemma states that evaluating, modulo 2, a linear combination of colour-prescribed homomorphism counts from fractured graphs, is as hard as evaluating its hardest term with an odd coefficient.

▶ **Lemma 14** ([29]). *There is a deterministic algorithm $\mathbb{A}$ and a computable function f such that the following conditions are satisfied:*

1. *$\mathbb{A}$ expects as input a graph Q and a Q-coloured graph (G, c).*
2. *$\mathbb{A}$ is equipped with oracle access to a function*

$$(G', c') \mapsto \sum_{\rho} a(\rho) \cdot \oplus\mathsf{Hom}((Q\sharp\rho, c_\rho) \to (G', c')) \mod 2\,,$$

   where the sum is over all fractures of Q and a is a function from fractures of Q to integers.
3. *Each oracle query (G', c') is of size at most $f(|Q|) \cdot |G|$.*
4. *$\mathbb{A}$ computes $\oplus\mathsf{Hom}((Q\sharp\rho, c_\rho) \to (G, c))$ for each fracture ρ with $a(\rho) \neq 0 \mod 2$.*
5. *The running time of $\mathbb{A}$ is bounded by $f(|Q|) \cdot |G|^{O(1)}$.*

The second tool is a standard application of the inclusion-exclusion principle (see e.g. [29, Sections 4.2 and 6.3]). It will be used in the final steps of our reductions to remove the colourings.

▶ **Lemma 15** ([29]). *There is a deterministic algorithm $\mathbb{A}$ that satisfies the following conditions:*

1. *$\mathbb{A}$ expects as input a graph H with k edges, a graph G and a k-edge colouring γ of G.*
2. *$\mathbb{A}$ is equipped with oracle access to the function $\oplus\mathsf{Sub}(H \to \star)$, and each oracle query G' satisfies $|G'| \leq |G|$.*
3. *$\mathbb{A}$ computes $\oplus\mathsf{ColSub}(H \to (G, \gamma))$.*
4. *The running time of $\mathbb{A}$ is bounded by $2^{|H|} \cdot |G|^{O(1)}$.*

3 Classification for Hereditary Graph Classes

In this section, we will completely classify the complexity of $\oplus\textsc{Sub}(\mathcal{H})$ for hereditary classes. Let us start by restating the classification theorem.

▶ **Theorem 4.** *Let $\mathcal{H}$ be a hereditary class of graphs. If $\mathcal{H}$ is matching splittable, then $\oplus\textsc{Sub}(\mathcal{H})$ is fixed-parameter tractable. Otherwise, the problem is $\oplus\mathsf{W}[1]$-complete and, assuming rETH, cannot be solved in time $f(|H|) \cdot |G|^{o(|V(H)|/\log|V(H)|)}$ for any function f.*

The proof of Theorem 4 is split in four cases, which stem from a structural property of non matching splittable hereditary graph classes $\mathcal{H}$ due to Jansen and Marx [24]. For the statement, we need to consider the following classes:

- $\mathcal{F}_\omega$ is the class of all complete graphs.
- $\mathcal{F}_\beta$ is the class of all complete bipartite graphs.
- $\mathcal{F}_{P_2}$ is the class of all P_2-packings, that is, disjoint unions of paths with two edges.[1]
- $\mathcal{F}_{K_3}$ is the class of all triangle packings, that is, disjoint unions of the complete graph of size 3.

▶ **Theorem 16** (Theorem 3.5 in [24]). *Let $\mathcal{H}$ be a hereditary class of graphs. If $\mathcal{H}$ is not matching splittable then at least one of the following are true: (1.) $\mathcal{F}_\omega \subseteq \mathcal{H}$, (2.) $\mathcal{F}_\beta \subseteq \mathcal{H}$, (3.) $\mathcal{F}_{P_2} \subseteq \mathcal{H}$, or (4.) $\mathcal{F}_{K_3} \subseteq \mathcal{H}$.*

[1] To avoid confusion, we remark that [24] uses P_3 to denote the path of two edges (and three vertices). In the current work, it will be more convenient to use the number of edges of a path as index.

Thus, it suffices to consider cases 1.–4. to prove Theorem 4. We start with the easy cases of cliques and bicliques; they follow implicitly from previous works [12, 17, 28] and we only include a proof for completeness. Note that a tight bound under rETH is known for those cases:

▶ **Lemma 17.** *Let $\mathcal{H}$ be a hereditary class of graphs. If $\mathcal{F}_\omega \subseteq \mathcal{H}$ or $\mathcal{F}_\beta \subseteq \mathcal{H}$ then $\oplus\text{Sub}(\mathcal{H})$ is $\oplus\mathsf{W}[1]$-hard and, assuming rETH, cannot be solved in time $f(|H|) \cdot |G|^{o(|V(H)|)}$ for any function f.*

Proof. If $\mathcal{F}_\omega \subseteq \mathcal{H}$ then $\oplus\mathsf{W}[1]$-hardness follows immediately from the fact that $\oplus\text{Clique}$ is the canonical $\oplus\mathsf{W}[1]$-complete problem [12]. For the rETH lower bound, we can reduce from the problem of *deciding* the existence of a k-clique via a (randomised) reduction using a version of the Isolation Lemma due to Williams et al. [34, Lemma 2.1]. This reduction does not increase k or the size of the host graph and is thus tight with respect to the well-known lower bound for the clique problem due to Chen et al. [8, 9]: Deciding the existence of a k-clique in an n-vertex graph cannot be done in time $f(k) \cdot n^{o(k)}$ for any function f, unless ETH fails. Our lower bound under rETH follows since the reduction is randomised.

If $\mathcal{F}_\beta \subseteq \mathcal{H}$, then the claim holds by [17, Theorem 5], which established the problem of counting, modulo 2, the induced copies of a k-by-k-biclique in an n-vertex bipartite graph to be $\oplus\mathsf{W}[1]$-hard and not solvable in time $f(k) \cdot n^{o(k)}$ for any function f, unless rETH fails. Since a copy of a biclique (with at least one edge) in a bipartite graph must always be induced, the claim follows. This concludes the proof of Lemma 17. ◀

The more interesting cases are $\mathcal{F}_{P_2} \subseteq \mathcal{H}$ and $\mathcal{F}_{K_3} \subseteq \mathcal{H}$. One reason for this is that, in contrast to cliques and bicliques, the decision version of those instances are fixed-parameter tractable. Hence a reduction from the decision version via e.g. an isolation lemma does not help. In other words, establishing hardness for those cases requires us to rely on the full power of counting modulo 2. More precisely, we will rely on the framework of fractures graphs (see Section 2). Both cases can be considered simpler applications of the machinery used in the later sections, so we will present all steps in great detail. While this might seem unnecessary given the simplicity of the constructions, we hope that it enables the reader to make themselves familiar with the general reduction strategies which will be used throughout the later sections of this work.

3.1 Triangle Packings

The goal of this subsection is to establish hardness of $\oplus\text{Sub}(\mathcal{F}_{K_3})$. To this end, let Δ be an infinite computable class of cubic bipartite expander graphs, and let $\mathcal{Q} = \{L(H) \mid H \in \Delta\}$ where $L(H)$ is constructed as follows: Each $v \in V(H)$ becomes a triangle with vertices v_x, v_y, and v_z corresponding to the three neighbours x, y, and z of v. Finally, for every edge $\{u, v\} \in E(H)$ we identify v_u and u_v. In fact, $L(H)$ is just the *line graph* of H: Every edge of H becomes a vertex in $L(H)$, and two vertices of $L(H)$ are made adjacent if and only if the corresponding edges in H are incident. Since all $H \in \Delta$ are bipartite (and thus triangle-free), we can easily observe the following.[2]

▶ **Observation 18.** *The mapping $v \mapsto (v_x, v_y, v_z)$ is a bijection from vertices of H to triangles in $L(H)$.*

[2] Observation 18 is also an immediate consequence of Whitney's Isomorphism Theorem implying that a triangle of a line graph corresponds to either a claw or to a triangle in its primal graph.

We also consider the fracture of $L(H)$ that splits $L(H)$ back into $|V(H)|$ triangles; consider Figure 2 for an illustration.

▶ **Definition 19** ($\tau(H)$)**.** *Let $H \in \Delta$ and recall that each vertex w of $L(H)$ is obtained by identifying v_u and u_v for some edge $\{u, v\} \in E(H)$. Moreover, w has four incident edges e_x, e_y, e_a, e_b, to v_x, v_y, u_a, u_b, respectively, where x, y, u are the neighbours of v in H and v, a, b are the neighbours of u in H. We define $\tau(H)_w := \{\{e_x, e_y\}, \{e_a, e_b\}\}$, and we proceed similar for all vertices of $L(H)$.*

Next, we use that $\mathsf{tw}(L(H)) = \Omega(\mathsf{tw}(H))$ (see e.g. [22]). Moreover, $\mathsf{tw}(L(H)) \leq |V(L(H))|$ since the treewidth of a graph is always bounded by the number of its vertices. Additionally, $|V(L(H))| = |E(H)|$ by construction. Since the graphs in Δ are cubic, we further have that $|E(H)| = \Theta(|V(H)|)$ for $H \in \Delta$. We combine those bounds with the fact that expander graphs have treewidth linear in the number of vertices (see e.g. [21]); therefore Δ and thus $\mathcal{Q}$ have unbounded treewidth. Putting these facts together, we obtain the following.

▶ **Fact 20.** *$\mathcal{Q}$ has unbounded treewidth and $\mathsf{tw}(L(H)) = \Theta(|V(L(H))|) = \Theta(|V(H)|)$ for $H \in \Delta$.*

We are now able to establish hardness of $\oplus\textsc{Sub}(\mathcal{F}_{K_3})$. The proof will heavily rely on the transformation from edge-coloured subgraphs to homomorphisms established in [29].

▶ **Lemma 21.** *The problem $\oplus\textsc{Sub}(\mathcal{F}_{K_3})$ is $\oplus\mathsf{W}[1]$-hard. Furthermore, on input kK_3 and G, the problem cannot be solved in time $f(k) \cdot |G|^{o(k/\log k)}$ for any function f, unless rETH fails.*

Proof. We reduce from $\oplus\textsc{cp-Hom}(\mathcal{Q})$, which, by Fact 20 and Theorem 12, is $\oplus\mathsf{W}[1]$-hard and for $L(H) \in \mathcal{Q}$, it cannot be solved in time $f(|L(H)|) \cdot |G|^{o(|V(L(H))|/\log |V(L(H))|)}$, unless rETH fails.

Let L and (G, c) be an input instance to $\oplus\textsc{cp-Hom}(\mathcal{Q})$. Recall that Δ is computable – that is, there is an algorithm that takes a graph H and determines whether it is in Δ. Thus, there is an algorithm that takes input $L \in \mathcal{Q}$ and finds a graph $H \in \Delta$ with $L = L(H)$. The run time of this algorithm depends on $|L|$ but clearly not on (G, c). Let $k = |V(H)|$ and note that $|E(L(H))| = 3k$, since, by construction, each vertex v of H becomes a triangle of $L(H)$. We consider the graph G as a $3k$-edge-coloured graph, coloured by c_E. That is, each edge $e = \{x, y\}$ of G is assigned the colour $c_E(e) = \{c(x), c(y)\}$ which is an edge of L (see Figure 2 for an illustration).

Now, for *any* L-coloured graph (G', c') recall that $\mathsf{ColSub}(kK_3 \to (G', c'_E))$ is the set of subgraphs of G' that are isomorphic to kK_3 and that include each edge colour (each edge of L) precisely once. We will see later that $\oplus\mathsf{ColSub}(kK_3 \to (G', c'_E))$ can be computed using our oracle for $\oplus\textsc{Sub}(\mathcal{F}_{K_3})$ using the principle of inclusion and exclusion.

It was shown in [29, Lemma 4.1] that there is a unique function a such that for every L-coloured graph (G', c') we have[3]

$$\#\mathsf{ColSub}(kK_3 \to (G', c'_E)) = \sum_{\rho} a(\rho) \cdot \mathsf{Hom}(L\sharp\rho \to (G', c'))\,. \tag{2}$$

where the sum is over all fractures of L. Additionally, it was shown in [29, Corollary 4.3] that

$$a(\top) = \sum_{\rho \in \mathsf{F}(kK_3, L)} \prod_{w \in V(L)} (-1)^{|\rho_w|-1} \cdot (|\rho_w| - 1)!\,, \tag{3}$$

[3] In the language of [29], Equation (2) is obtained by choosing Φ as the property of being isomorphic to kK_3.

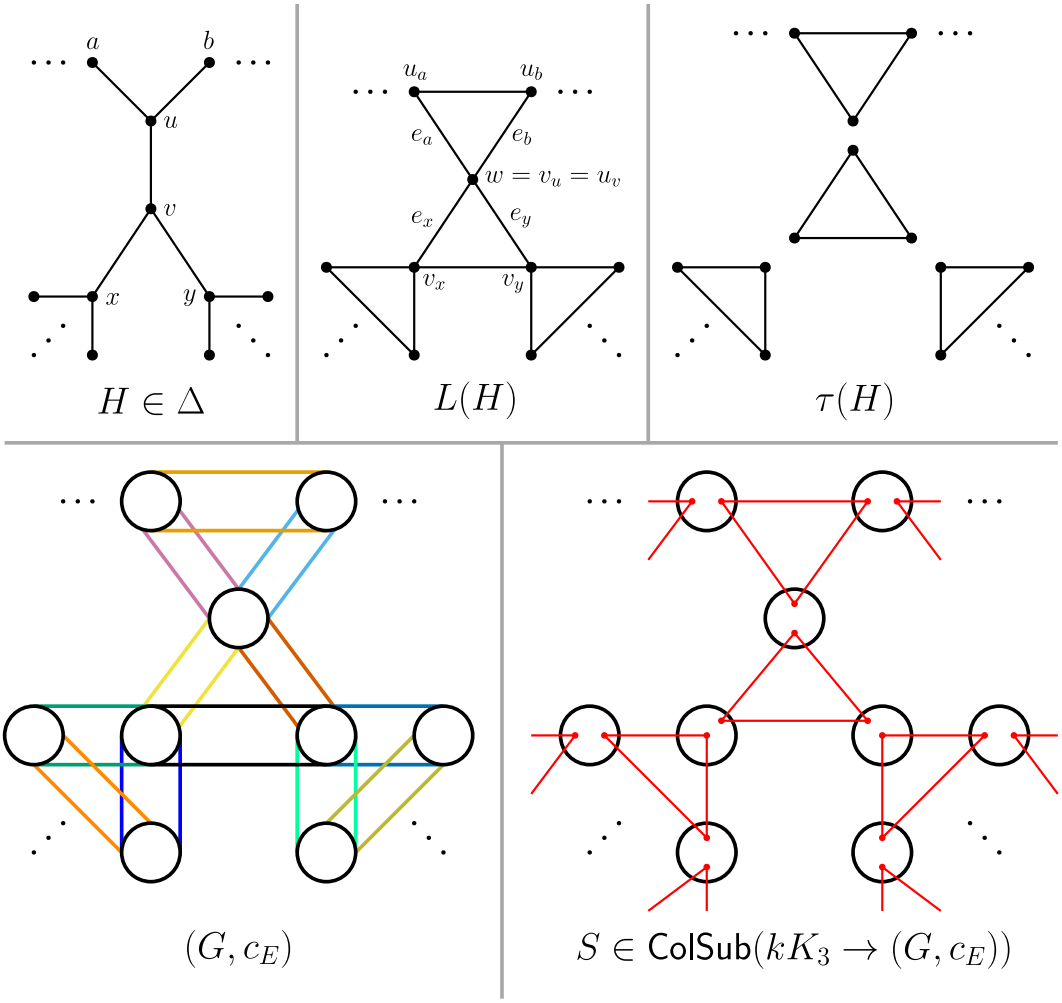


Figure 2 (*Top*:) A cubic bipartite graph $H \in \Delta$, its line graph $L(H)$, and the fractured graph induced by $\tau(H)$. (*Below*:) An $L(H)$-coloured graph (G, c); emphasised in distinct colours is the edge-colouring c_E of G induced by the mapping $\{u, v\} \mapsto \{c(u), c(v)\}$. Additionally we depict an element $S \in \mathsf{ColSub}(kK_3 \to (G, c_E))$, that is, a subgraph of G isomorphic to kK_3 that contains each edge colour of G precisely once.

where $\top$ is the fracture in which each partition consists only of one block (that is, $L\sharp\top = L$), and $\mathsf{F}(kK_3, L)$ is the set of all fractures ρ of L such that $L\sharp\rho \cong kK_3$. However, note that, by Observation 18, there is only way to fracture L into k disjoint triangles, and this fracture is given by $\tau(H)$. Thus, (3) simplifies to

$$a(\top) = \prod_{w \in V(L)} (-1)^{|\tau(H)_w|-1} \cdot (|\tau(H)_w| - 1)!\,, \tag{4}$$

which is odd since each partition of $\tau(H)$ consists of precisely two blocks (so in fact the expression in (4) is $(-1)^{|V(L)|}$).

Note that the algorithm for $\oplus$CP-Hom$(\mathcal{Q})$ is supposed to compute $\oplus\mathsf{Hom}((L, \mathsf{id}_L) \to (G, c))$ which is equal to $\oplus\mathsf{Hom}(L\sharp\top \to (G, c_\top))$. Since $a(\top)$ is odd, we can invoke Lemma 14 to recover this term by evaluating the entire linear combination (2), that is, by evaluating the function $\oplus\mathsf{ColSub}(kK_3 \to \star)$. More concretely, this means that we need to compute

$\oplus\mathsf{ColSub}(kK_3 \to (G', c'_E))$ for some L-coloured graphs (G', c') of size at most $f(|L|) \cdot |G|$ for some computable function f (see 3. in Lemma 14). This can easily be done using Lemma 15 since we have oracle access to the function $\oplus\mathsf{Sub}(kK_3 \to \star)$. We emphasise that, by condition 2. of Lemma 15, each oracle query $\hat{G}$ satisfies $|\hat{G}| \leq |G'|$, where (G', c') is the L-coloured graph for which we wish to compute $\oplus\mathsf{ColSub}(kK_3 \to (G', c'_E))$. Since $|(G', c')| \leq f(|L|) \cdot |G|$, we obtain that $|\hat{G}| \leq f(|L|) \cdot |G|$ as well.

Since, by Fact 20, $k = \Theta(|kK_3|) = \Theta(|V(L)|) = \Theta(\mathsf{tw}(L))$, our reduction yields $\oplus\mathsf{W}[1]$-hardness and transfers the conditional lower bound under rETH as desired. ◀

3.2 P_2-packings

Next we establish hardness for the case of P_2-packings. The strategy will be similar in spirit to the construction for triangle packings; however, rather then identifying a unique fracture for which the technique applies, we will encounter an *odd* number of possible fractures in the current section.

Let Δ be a computable infinite class of 4-regular expander graphs, and let $\mathcal{Q}$ be the class of all subdivisions of graphs in Δ, that is $\mathcal{Q} = \{H^2 \mid H \in \Delta\}$, where H^2 is obtained from H by subdividing each edge once.

We start by establishing an easy but convenient fact on the treewidth of the graphs in $\mathcal{Q}$.

▶ **Lemma 22.** *$\mathcal{Q}$ has unbounded treewidth and $\mathsf{tw}(H^2) = \Theta(|V(H)|)$ for $H \in \Delta$.*

Proof. As in Section 3.1, $\mathsf{tw}(H) = \Theta(|V(H)|)$ for $H \in \Delta$, since expanders have treewidth linear in the number of vertices. Since H is a minor of H^2, and since taking minors cannot increase treewidth (see [15, Exercise 7.7]), we thus have that $\mathsf{tw}(H^2) = \Omega(|V(H)|))$. Finally, we have $\mathsf{tw}(H^2) \leq |V(H^2)|$ since the treewidth is at most the number of vertices, and $|V(H^2)| = O(|V(H)|)$ since H is 4-regular. In combination, we obtain $\mathsf{tw}(H^2) = \Theta(|V(H)|)$ for $H \in \Delta$. Note that this also implies that $\mathcal{Q}$ has unbounded treewidth (as Δ is infinite). ◀

For what follows, given a subdivision H^2 of a graph H, it will be convenient to assume that $V(H^2) = V(H) \cup S_E$, where $S_E = \{s_e \mid e \in E(H\})$ is the set of the subdivision vertices.

▶ **Definition 23** (Odd Fractures)**.** *Let $H \in \Delta$ and let τ be a fracture of H^2. We say that τ is* odd *if the following two conditions are satisfied:*

1. *For each $s \in S_E$ the partition τ_s consists of two singleton blocks.*
2. *For each $v \in V(H)$ the partition τ_v consists of two blocks of size 2.*

Consider Figure 3 for a depiction of an odd fracture.

The following two lemmas are crucial for our construction.

▶ **Lemma 24.** *Let $H \in \Delta$. The number of odd fractures of H^2 is odd.*

Proof. The first condition in Definition 23 leaves only one choice for subdivision vertices. Let us thus consider a vertex $v \in V(H) = V(H^2) \setminus S_E$. Since H is 4-regular, there are 4 incident edges to v. Now note that there are precisely 3 partitions of a 4-element set with two blocks of size 2. Thus the total number of odd fractures of H^2 is $3^{|V(H)|}$, which is odd. ◀

▶ **Lemma 25.** *Let $H \in \Delta$, let $k = 2|V(H)|$ and let τ be a fracture of H^2 such that τ_v consists of at most 2 blocks for each $v \in V(H^2)$. Then $H^2 \sharp \tau \cong kP_2$ if and only if τ is odd.*

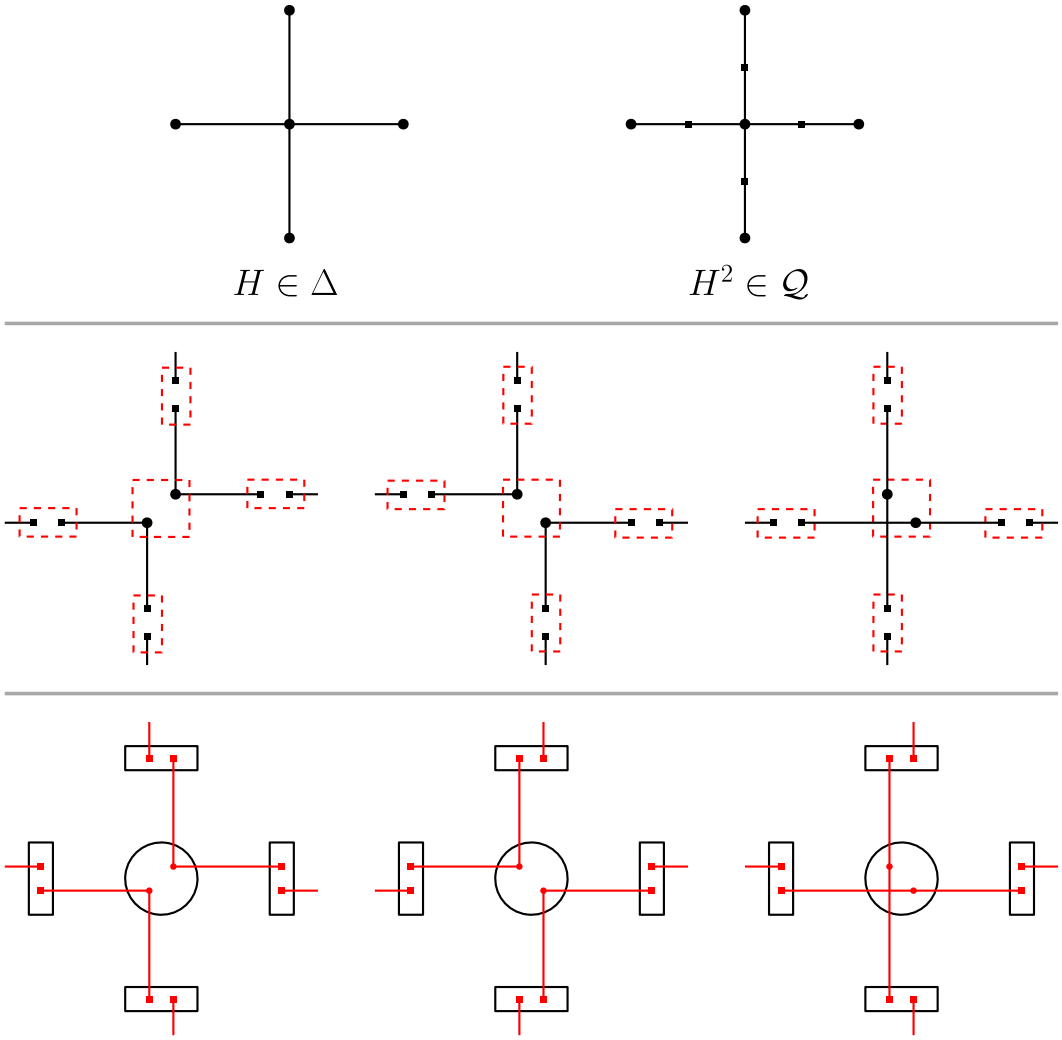


Figure 3 (*Top*:) Subdividing a 4-regular expander in Δ depicted by the neighbourhood of an individual vertex. (*Centre*:) Illustrations of odd fractures (Definition 23). For each non-subdivision vertex, there are only three ways to satisfy 2. in Definition 23. This observation is used in Lemma 24 to show that the number of odd fractures is a power of 3. (*Bottom*:) Elements of $\mathsf{ColSub}(kP_2 \to (G, c_E))$ inducing fractures of H^2 such that each partition has at most two blocks. Lemma 25 shows that those are precisely the odd fractures of H^2.

Proof. First observe that $|E(H^2)| = 2|E(H)| = 4|V(H)| = 2k$. Thus the number of edges of $H^2 \sharp \tau$ is equal to $2k$ (for each fracture τ of H^2), which is also equal to the number of edges of kP_2.

Thus, $H^2 \sharp \tau$ is isomorphic to kP_2 if and only if each connected component of $H^2 \sharp \tau$ is a path of length 2. It follows immediately by Definition 23 that τ being odd implies that $H^2 \sharp \tau$ consists only of disjoint P_2. It thus remains to show the other direction.

Assume for contradiction that there is a subdivision vertex $s \in S_E$ of H^2 such that τ_s consists of only one block (recall that s has degree 2, thus τ_s either consists of two singleton blocks, or of one block of size 2). Let $e = \{u, v\} \in E(H)$ be the edge corresponding to s, that is, s was created by subdividing e. Since $H^2 \sharp \tau$ is a union of P_2, we can infer that τ_v and τ_u contain a singleton block (otherwise we would have created a connected component which is

not isomorphic to P_2). Now recall that both u and v have degree 4, since H is 4-regular. We obtain a contradiction as follows: By assumption of the lemma, we know that τ_v and τ_u can have at most two blocks. Since we have just shown that both contain a singleton block, it follows that both τ_v and τ_u contain one further block of size 3. However, a block of size 3 yields a vertex of degree 3 in the fractured graph $H^2\sharp\tau$, contradicting the fact that $H^2\sharp\tau$ consists only of disjoint P_2.

Thus we have established that, for each $s \in S_E$, the partition τ_s consists of two singleton blocks. Given this fact, the only way for $H^2\sharp\tau$ being a disjoint union of P_2 is that each partition τ_v, for $v \in V(H) = V(H^2) \setminus S_E$, consists of two blocks of size 2. ◀

We are now able to prove our hardness result.

▶ **Lemma 26.** *The problem $\oplus\textsc{Sub}(\mathcal{F}_{P_2})$ is $\oplus\mathsf{W}[1]$-hard. Furthermore, on input kP_2 and G, the problem cannot be solved in time $f(k)\cdot|G|^{o(k/\log k)}$ for any function f, unless rETH fails.*

Proof. We reduce from $\oplus\textsc{cp-Hom}(\mathcal{Q})$, which, by Lemma 22 and Theorem 12, is $\oplus\mathsf{W}[1]$-hard and for $H' \in \mathcal{Q}$, it cannot be solved in time $f(|H'|)\cdot|G|^{o(|V(H')|/\log|V(H')|)}$, unless rETH fails.

Let H' and (G,c) be an input instance to $\oplus\textsc{cp-Hom}(\mathcal{Q})$. There is an algorithm that takes as input a graph $H' \in \mathcal{Q}$ and finds a graph $H \in \Delta$ with $H' = H^2$ – this is basically 2-colouring. The run time of this algorithm depends on $|H'|$ but clearly not on (G,c). Let $k = 2|V(H)|$ and note that $|E(H^2)| = 2|E(H)| = 4|V(H)| = 2k$. We consider the graph G as a $2k$-edge-coloured graph, coloured by c_E. That is, each edge $e = \{x,y\}$ of G is assigned the colour $c_E(e) = \{c(x),c(y)\}$ which is an edge of $H' = H^2$.

Now, for *any* H^2-coloured graph (G',c') recall that $\mathsf{ColSub}(kP_2 \to (G',c'_E))$ is the set of subgraphs of G' that are isomorphic to kP_2 and that include each edge colour (each edge of H^2) precisely once. We will see later that $\oplus\mathsf{ColSub}(kP_2 \to (G',c'_E))$ can be computed using our oracle for $\oplus\textsc{Sub}(\mathcal{F}_{P_2})$ using the principle of inclusion and exclusion.

It was shown in [29, Lemma 4.1] that there is a unique function a such that, for every H^2-coloured graph (G',c'),

$$\#\mathsf{ColSub}(kP_2 \to (G',c'_E)) = \sum_{\rho} a(\rho)\cdot\mathsf{Hom}(H^2\sharp\rho \to (G',c'))\,. \tag{5}$$

where the sum is over all fractures of H^2. As in Section 3.1 from [29, Corollary 4.3] we know that

$$a(\top) = \sum_{\rho\in\mathsf{F}(kP_2,H^2)}\prod_{w\in V(H^2)}(-1)^{|\rho_w|-1}\cdot(|\rho_w|-1)!\,, \tag{6}$$

where $\top$ is the fracture in which each partition consists only of one block and $\mathsf{F}(kP_2,H^2)$ is the set of all fractures ρ of H^2 such that $H^2\sharp\rho \cong kP_2$.

Our next goal is to show that $a(\top) = 1 \mod 2$. First, suppose that a fracture ρ contains a partition ρ_w with at least three blocks. Then $(|\rho_w|-1)! = 0 \mod 2$. Thus such fractures do not contribute to $a(\top)$ if arithmetic is done modulo 2. Next, note that if, for each w, the partition ρ_w contains at most 2 blocks, then

$$\prod_{w\in V(H^2)}(-1)^{|\rho_w|-1}\cdot(|\rho_w|-1)! = 1 \mod 2.$$

Let $\mathsf{Odd}(kP_2, H^2)$ be the set of all fractures ρ of H^2 such that $H^2 \sharp \rho \cong kP_2$ and each partition of ρ consists of at most 2 blocks. Our analysis then yields $a(\top) = |\mathsf{Odd}(kP_2, H^2)| \mod 2$. Finally, Lemma 25 states that $\mathsf{Odd}(kP_2, H^2)$ is precisely the set of odd fractures, and Lemma 24 thus implies that $|\mathsf{Odd}(kP_2, H^2)| = 1 \mod 2$. Consequently, $a(\top) = 1 \mod 2$ as well, and we have achieved the goal.

Next we can proceed similarly to the case of triangle packings. As in that case, the goal is to compute $\oplus\mathsf{Hom}((H^2, \mathsf{id}_{H^2}) \to (G, c)))$ which is equal to $\oplus\mathsf{Hom}((H^2 \sharp \top, c_\top) \to (G, c))$. Since $a(\top)$ is odd, we can invoke Lemma 14 to recover this term by evaluating the entire linear combination (5), that is, if we can evaluate the function $\oplus\mathsf{ColSub}(kP_2 \to \star)$. This can be done by using Lemma 15. Each call to the oracle is of the form $\oplus\mathsf{Sub}(kP_2 \to \hat{G})$ where $|\hat{G}|$ is bounded by $f(k) \cdot |G|$.

Now recall that $k \in \Theta(|V(H)|)$. By Lemma 22, we thus have $k = \Theta(\mathsf{tw}(H^2))$. Hence our reduction yields $\oplus\mathsf{W}[1]$-hardness and transfers the conditional lower bound under rETH as desired. ◀

We can now conclude the treatment of hereditary pattern classes by proving Theorem 4, which we restate for convenience.

▶ **Theorem 4.** *Let $\mathcal{H}$ be a hereditary class of graphs. If $\mathcal{H}$ is matching splittable, then $\oplus\textsc{Sub}(\mathcal{H})$ is fixed-parameter tractable. Otherwise, the problem is $\oplus\mathsf{W}[1]$-complete and, assuming rETH, cannot be solved in time $f(|H|) \cdot |G|^{o(|V(H)|/\log|V(H)|)}$ for any function f.*

Proof. The fixed-parameter tractability result was shown in [12]. For the hardness result, using the fact that $\mathcal{H}$ is not matching splittable and Theorem 16 we obtain four cases.

- If $\mathcal{H}$ contains all cliques or all bicliques, then hardness follows from Lemma 17.
- If $\mathcal{H}$ contains all triangle packings, then hardness follows from Lemma 21.
- If $\mathcal{H}$ contains all P_2-packings, then hardness follows from Lemma 26.

Since the case distinction is exhaustive, the proof is concluded. ◀

References

1. Amir Abboud, Shon Feller, and Oren Weimann. On the fine-grained complexity of parity problems. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 5:1–5:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.5.
2. Noga Alon, Phuong Dao, Iman Hajirasouliha, Fereydoun Hormozdiari, and S. Cenk Sahinalp. Biomolecular network motif counting and discovery by color coding. *Bioinformatics*, 24(13):i241–i249, 2008. doi:10.1093/bioinformatics/btn163.
3. Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.
4. Suman K. Bera, Lior Gishboliner, Yevgeny Levanzov, C. Seshadhri, and Asaf Shapira. Counting subgraphs in degenerate graphs. *J. ACM*, 69(3):23:1–23:21, 2022. doi:10.1145/3520240.
5. Andreas Björklund, Holger Dell, and Thore Husfeldt. The parity of set systems under random restrictions with applications to exponential time problems. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 231–242. Springer, 2015. doi:10.1007/978-3-662-47672-7_19.
6. Marco Bressan. Faster algorithms for counting subgraphs in sparse graphs. *Algorithmica*, 83(8):2578–2605, 2021. doi:10.1007/s00453-021-00811-0.

7 Andrei A. Bulatov and Amirhossein Kazeminia. Complexity classification of counting graph homomorphisms modulo a prime number. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1024–1037. ACM, 2022. doi:10.1145/3519935.3520075.

8 Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David W. Juedes, Iyad A. Kanj, and Ge Xia. Tight lower bounds for certain parameterized NP-hard problems. *Inf. Comput.*, 201(2):216–231, 2005. doi:10.1016/j.ic.2005.05.001.

9 Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.*, 72(8):1346–1367, 2006. doi:10.1016/j.jcss.2006.04.007.

10 Yijia Chen, Marc Thurley, and Mark Weyer. Understanding the Complexity of Induced Subgraph Isomorphisms. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 587–596. Springer, 2008. doi:10.1007/978-3-540-70575-8_48.

11 Radu Curticapean. Counting matchings of size k is w[1]-hard. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, volume 7965 of *Lecture Notes in Computer Science*, pages 352–363. Springer, 2013. doi:10.1007/978-3-642-39206-1_30.

12 Radu Curticapean, Holger Dell, and Thore Husfeldt. Modular counting of subgraphs: Matchings, matching-splittable graphs, and paths. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPIcs*, pages 34:1–34:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ESA.2021.34.

13 Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms are a good basis for counting small subgraphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 210–223. ACM, 2017. doi:10.1145/3055399.3055502.

14 Radu Curticapean and Dániel Marx. Complexity of counting subgraphs: Only the boundedness of the vertex-cover number counts. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 130–139. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.22.

15 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

16 Víctor Dalmau and Peter Jonsson. The complexity of counting homomorphisms seen from the other side. *Theoret. Comput. Sci.*, 329(1-3):315–323, 2004. doi:10.1016/j.tcs.2004.08.008.

17 Julian Dörfler, Marc Roth, Johannes Schmitt, and Philip Wellnitz. Counting induced subgraphs: An algebraic approach to #W[1]-hardness. *Algorithmica*, 84(2):379–404, 2022. doi:10.1007/s00453-021-00894-9.

18 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

19 Jörg Flum and Martin Grohe. The parameterized complexity of counting problems. *SIAM J. Comput.*, 33(4):892–922, 2004. doi:10.1137/S0097539703427203.

20 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. doi:10.1007/3-540-29953-X.

21 Martin Grohe and Dániel Marx. On tree width, bramble size, and expansion. *J. Comb. Theory, Ser. B*, 99(1):218–228, 2009. doi:10.1016/j.jctb.2008.06.004.

22 Daniel J. Harvey and David R. Wood. The treewidth of line graphs. *J. Comb. Theory, Ser. B*, 132:157–179, 2018. doi:10.1016/j.jctb.2018.03.007.

23 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

24 Bart M. P. Jansen and Dániel Marx. Characterizing the easy-to-find subgraphs from the viewpoint of polynomial-time algorithms, kernels, and turing kernels. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 616–629. SIAM, 2015. doi:10.1137/1.9781611973730.42.

25 Oleksii Kuchaiev, Tijana Milenković, Vesna Memišević, Wayne Hayes, and Nataša Pržulj. Topological network alignment uncovers biological function and phylogeny. *Journal of the Royal Society Interface*, 7(50):1341–1354, 2010. doi:10.1098/rsif.2010.0063.

26 R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network Motifs: Simple Building Blocks of Complex Networks. *Science*, 298(5594):824–827, 2002. doi:10.1126/science.298.5594.824.

27 Ron Milo, Shalev Itzkovitz, Nadav Kashtan, Reuven Levitt, Shai Shen-Orr, Inbal Ayzenshtat, Michal Sheffer, and Uri Alon. Superfamilies of evolved and designed networks. *Science*, 303(5663):1538–1542, 2004. doi:10.1126/science.1089167.

28 Norbert Peyerimhoff, Marc Roth, Johannes Schmitt, Jakob Stix, and Alina Vdovina. Parameterized (modular) counting and cayley graph expanders. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021, August 23-27, 2021, Tallinn, Estonia*, volume 202 of *LIPIcs*, pages 84:1–84:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.MFCS.2021.84.

29 Norbert Peyerimhoff, Marc Roth, Johannes Schmitt, Jakob Stix, Alina Vdovina, and Philip Wellnitz. Parameterized Counting and Cayley Graph Expanders. *SIAM J. Discrete Math.*, to appear.

30 Benjamin Schiller, Sven Jager, Kay Hamacher, and Thorsten Strufe. StreaM – A Stream-Based Algorithm for Counting Motifs in Dynamic Graphs. In *Proceedings of the 2nd International Conference on Algorithms for Computational Biology (AlCoB)*, pages 53–67. Springer International Publishing, 2015. doi:10.1007/978-3-319-21233-3_5.

31 Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991. doi:10.1137/0220053.

32 Ngoc Hieu Tran, Kwok Pui Choi, and Louxin Zhang. Counting motifs in the human interactome. *Nature communications*, 4(1):1–8, 2013. doi:10.1038/ncomms3241.

33 Charalampos E. Tsourakakis, Jakub Pachocki, and Michael Mitzenmacher. Scalable motif-aware graph clustering. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*, pages 1451–1460, 2017. doi:10.1145/3038912.3052653.

34 Virginia Vassilevska Williams, Joshua R. Wang, Richard Ryan Williams, and Huacheng Yu. Finding four-node subgraphs in triangle time. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1671–1680. SIAM, 2015. doi:10.1137/1.9781611973730.111.

Efficient Data Structures for Incremental Exact and Approximate Maximum Flow

Gramoz Goranci
Faculty of Computer Science, Universität Wien, Austria

Monika Henzinger
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Abstract

We show an $(1+\epsilon)$-approximation algorithm for maintaining maximum s-t flow under m edge insertions in $m^{1/2+o(1)}\epsilon^{-1/2}$ amortized update time for directed, unweighted graphs. This constitutes the first sublinear dynamic maximum flow algorithm in general sparse graphs with arbitrarily good approximation guarantee.

Furthermore we give an algorithm that maintains an exact maximum s-t flow under m edge insertions in an n-node graph in $\tilde{O}(n^{5/2})$ total update time. For sufficiently dense graphs, this gives to the first exact incremental algorithm with sub-linear amortized update time for maintaining maximum flows.

2012 ACM Subject Classification Theory of computation → Dynamic graph algorithms

Keywords and phrases dynamic graph algorithms, maximum flow, data structures

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.69

Category Track A: Algorithms, Complexity and Games

Related Version *Previous Version*: https://arxiv.org/abs/2211.09606

Funding This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101019564 "The Design of Modern Fully Dynamic Data Structures (MoDynStruct)" and from the Austrian Science Fund (FWF) project "Static and Dynamic Hierarchical Graph Decompositions", I 5982-N, and project "Fast Algorithms for a Reactive Network Layer (ReactNet)", P 33775-N, with additional funding from the *netidee SCIENCE Stiftung*, 2020–2024.

Acknowledgements This work was done in part while Gramoz Goranci was at Institute for Theoretical Studies, ETH Zurich, Switzerland. There, he was supported by Dr. Max Rössler, the Walter Haefner Foundation and the ETH Zürich Foundation. We also thank Richard Peng, Thatchaphol Saranurak, Sebastian Forster and Sushant Sachdeva for helpful discussions, and the anonymous reviewers for their insightful comments.

1 Introduction

The maximum flow problem and its dual, the minimum cut problem, are one of the cornerstones problems in combinatorial optimization. They are often used as subroutine for solving other prominent graph problems (e.g., Gomory-Hu Trees [11], Sparsest Cut [24]), performing divide-and-conquer on graphs [8] and have found several applications across many areas including computer vision [2], clustering [29] and scientific computing. Designing fast maximum flow algorithms has been an active area of research for decades, with recent advances making tremendous progress towards the quest of designing a near-linear time algorithm [6, 30, 26, 23, 28, 27, 31, 25, 4]. This has culminated in a recent breakthrough result due to Chen, Kyng, Liu, Peng, Probst Gutenberg, and Sachdeva [4], which computes a maximum flow in $m^{1+o(1)}$ time, where m is the number of edges of the input graph.

Recently, we have witnessed a growing interest in designing dynamic algorithms for computing maximum flows in dynamically changing graphs [20, 5, 1, 7, 14, 13, 3, 21]. Despite this, the current fastest algorithms either incur super-constant approximation factors [13, 3] or achieve competitive update times only for sufficiently dense graphs [1]. Moreover, all previous works on dynamic flows are restricted to undirected graphs. From a (conditional) lower bound perspective, for any $\delta > 0$, it is known [7] that no algorithm can exactly maintain maximum flow in $O(m^{1-\delta})$ amortized time per operation, even when restricted to algorithms that support edge insertions, unless the OMv conjecture [16] is false. Nevertheless, the lower bound construction from [7] is on dense graphs, i.e., $m = \Omega(n^2)$, and thus for sparse graphs yields only an $\Omega(m^{1/2})$ lower bound on the update time.

In this paper, we show a simple generic algorithmic framework for maintaining *approximate* maximum flows under edge insertions.

▶ **Theorem 1.** *Let $G = (V, E)$ be an initially empty, directed, unweighted n-vertex graph, s and t be any two vertices, and $\epsilon > 0$, $\mu \in [0, n]$ be two parameters. If there is*

(a) *an incremental algorithm* IncBMF(G, s, t, μ) *for inserting m edges and maintaining s-t maximum flow whose value is bounded by μ in $t_{total}(m, n, \mu)$ total update time and $q(m, n)$ query time, and*

(b) *a static algorithm for computing exact s-t maximum flow in an n-vertex, m'-edge graph in $t_{static}(m', n)$ time, where $m' \leq m$,*

then we can design an incremental algorithm for maintaining a $(1 + \epsilon)$-approximate s-t maximum flow under m edge insertions in

$$\frac{t_{total}(m, n, \mu + 1)}{m} + \frac{t_{static}(m, n)}{\epsilon\mu} + q(m, n)$$

amortized update time and $q(m, n)$ query time.

For maintaining exact maximum flows whose value is bounded by μ, we slightly adapt an incremental version of the Ford-Fulkerson [9] algorithm, which was initially observed by Henzinger [17] and later by Gupta and Khan [14] (cf. Lemma 9). This gives an incremental algorithm with $O(m\mu)$ total update time and $O(1)$ query time. The recent breakthrough result due to Chen, Kyng, Liu, Peng, Probst Gutenberg, and Sachdeva [4] (cf. Theorem 6) gives a static exact maximum flow algorithm that runs in $m^{1+o(1)}$ time. Plugging these bounds in Theorem 1 and choosing $\mu = m^{1/2+o(1)}\epsilon^{-1/2}$ yields the main result of this paper, which we summarize in the theorem below.

▶ **Theorem 2.** *Given an initially empty, directed, unweighted graph $G = (V, E)$, any two vertices s and t in V, and any $\epsilon > 0$, there is an incremental algorithm that maintains a $(1 + \epsilon)$-approximate maximum s-t flow in G under m edge insertions in $m^{1/2+o(1)}\epsilon^{-1/2}$ amortized update time. The algorithm supports queries about the value of the maintained flow in $O(1)$ time.*

When the underlying graph is undirected and unweighted, we additionally show an improved incremental version of an algorithm due to Karger and Levine [22] for maintaining exact maximum flows whose value is bounded by μ. Concretely, our algorithm achieves $\tilde{O}(m + n\mu^{3/2})$ total update time for handling m edge insertions (cf. Lemma 15). Since $\mu \leq n$ always holds in unweighted graphs, we immediately obtain the following result.

▶ **Theorem 3.** *Given an initially empty, undirected, unweighted graph $G = (V, E)$, any two vertices s and t in V, and any $\epsilon > 0$, there is an incremental algorithm that maintains an* exact *maximum s-t flow in G under m edge insertions in $\tilde{O}(n^{5/2})$ total update time. The algorithm supports queries about the value of the maintained flow in $O(1)$ time.*

For sufficiently dense graphs, this gives to the first exact incremental algorithm with sub-linear amortized update time for maintaining maximum flows.

We believe that our approach to dynamic flows may serve as the basis for designing new fully-dynamic maximum flow algorithms with competitive approximation ratio.

Independet Work. A recent independent work by Brand, Liu and Sidford [32] provides an algorithm for incremental approximate maximum flow with $n^{1/2+o(1)}\epsilon^{-1}$ amortized update time on directed graphs. They achieve this result by implementing a dynamic variant of the recent maximum flow algorithm based on the Interior Point Method (IPM) [4]. For comparison, their result extends to capacitated graphs with polynomially bounded capacitites, and achieves a speed up on the running time (albeit only on dense graphs). However, these improvements come at the cost of employing the complicated machinery of IPMs. Our result from Theorem 2 is simpler, matches their running time guarantee on sparse graphs and gives a slightly better dependncy on the accuracy parameter ϵ.

2 Preliminaries

In the following, we settle some basic notation, as well as review definitions and algorithms for computing flows on graphs.

Maximum Flow

Let $G = (V, E)$ be a directed, unweighted graph with n vertices and m edges, let $s \in V$ be a *source* vertex, and let $t \in V$ be a *target* vertex. A *flow* from s to t in G is a function $f : E \to \mathbb{R}^+$ that maps each edge to a non-negative real number; the value $f(e)$ represents the amount of flow sent along e. A flow must satisfy the following properties: (i) for each $e \in E$, we have $f(e) \leq 1$, known as *capacity constraints*, and (ii) for each $v \in V \setminus \{s, t\}$, $\sum_{(u,v)\in E} f(u, v) = \sum_{(v,u)} f(v, u)$, known as *conservation constraints*. The *value* of a flow f is the amount of flow leaving the source s minus the amount flow entering s, i.e., $v(f) = \sum_{(s,u)\in E} f(s, u) - \sum_{(u,s)\in E} f(u, s)$. In the maximum s-t flow problem, the goal is to find a flow f with the largest $v(f)$, called the *maximum s-t flow*. Let $F^* = \max\{v(f) \mid f \text{ is a flow in } G\}$. Note that while F^* is unique, there might be multiple maximum s-t flows attaining F^*.

Residual graph and Augmenting Paths

Given a directed, unweighted graph $G = (V, E)$, and a flow f from s to t in G, we let $G_f = (V, E_f)$ be the *residual graph* of G with respect to f, where E_f contains all edges of E, except that their direction is reversed if $f(e) = 1$. An edge whose direction in G_f is reversed is referred to as a *backward* edge. Otherwise, the edge is a *forward* edge. An augmenting path P from s to t in G is a simple directed path from s to t in G_f. We next review a powerful result relating the residual graphs and optimal flows.

▶ **Lemma 4** ([9]). *If there is no directed path from s to t in the residual graph G_f, then the flow f is a maximum s-t flow.*

Another useful fact is given in the lemma below.

▶ **Lemma 5.** *If there exists a directed path P from s to t in G_f, pushing flow along P in G increases the value of the flow f by at exactly one.*

Exact Maximum Flow in Directed Graphs

Our incremental approximate algorithm heavily relies on the ability to compute a maximum s-t flow quickly. Hence, we use the current fastest result [4] that achieves an exact, almost-linear algorithm for the maximum flow problem on directed graphs.[1]

▶ **Theorem 6** ([4]). *For any directed, unweighted graph $G = (V, E)$ and any two vertices s and t, there is an algorithm that computes an* exact *maximum s-t flow in G in $m^{1+o(1)}$ time.*

Approximate Maximum Flow

To measure the quality of approximate maximum flows, we will use the notion of α-approximations, which indicates that the value of the current flow solution is at least $1/\alpha$ of the optimum value. In other words, a flow f is α-approximate if $F := v(f) \geq \frac{1}{\alpha}F^*$.

3 A framework for Incremental Approximate Maximum Flow

In this section we show a simple generic algorithmic framework for maintaining a $(1+\epsilon)$-approximate maximum s-t flow under edge insertions, i.e., prove Theorem 1. Our construction is based on two important components: (i) incrementally maintaining maximum s-t flows whose value is upper bounded by some parameter μ (one should think of μ being small relative to the size of the network) and (ii) performing periodical rebuilds whenever the maximum s-t flow of the current flow is larger than the parameter μ. The latter is a common approach in dynamic algorithms and was used by Gupta and Peng [15] in their dynamic algorithm for maintaining approximate matchings, and also recently leveraged in the context of *exact* algorithms for the dynamic minimum cut problem [12]. We next further elaborate on the precise requirements of both components and discuss how they lead to our algorithm.

To implement component (i), given a directed, unweighted n-vertex, m-edge graph G, any two vertices s, t, and a parameter $\mu \in [0, n]$, the goal is to construct a data structure denoted by IncBMF(G, s, t, μ), or simply IncBMF, that supports the following operations

- Initialize(G, s, t, μ): Initializes the data structure.
- Insert(u, v): Insert the edge (u, v) to G.
- MaxFlow(s, t): Return the value of the maximum s-t flow in the current graph G if this value is smaller than μ.

Ideally, we would like that IncBMF(G, s, t, μ) supports edge insertions in amortized time proportional to the parameter μ, and queries in constant time. This would alone lead to an efficient incremental maximum flow algorithm whenever the current flow value is bounded by μ.

When the maximum flow is large (i.e., component (ii)), we can make use of the *stability* of maximum flow and periodically invoke a fast static algorithm. Concretely, first note that the value of the maximum s-t flow changes by at most one per insertion. Therefore, if we have a large flow that is close to the maximum one, it will remain close to the maximum flow over a large number of updates. This naturally leads to the following simple but powerful approach: compute a flow at a certain time in the update sequence and do nothing for a certain number of updates as long as the flow is a good approximation to the maximum flow. This idea together with the data structure IncBMF(G, s, t, μ) yields an incremental algorithm for approximating s-t maximum flow, which we formally describe below.

[1] The algorithm extends to graphs with polynomially bound weights, but for the purposes of this paper and simplifying the presentation, we only state the unweighted version of this result

Given a directed, unweighted graph $G = (V, E)$, any two vertices s, t, and two parameters $\epsilon > 0, \mu \in [0, n]$, our data structure maintains:

- a flow estimate F to the maximum s-t flow F^*,
- a counter τ indicating the number of operations since the last rebuild,
- an incremental algorithm IncBMF for maintaining graphs with the maximum flow bounded by some parameter $(\mu + 1)$.

Initially, G is an empty graph, $F \leftarrow 0$, $\tau \leftarrow 0$, and we invoke the operation Initialize of IncBMF with $(G, s, t, \mu + 1)$ as an input. Upon insertion of an edge (u, v) to G, we query IncBMF to determine whether the s-t maximum flow in the current graph is at most μ. If so, we pass the edge insertion to the IncBMF data structure and update F accordingly.

On the other hand, if the current maximum s-t flow is larger than μ (and our algorithm always correctly detects this since IncBMF run with the parameter $(\mu + 1)$ returns the correct answer), we increment τ, which counts the number of insertions since the last reset of τ that fall into this case. If $\tau \geq \epsilon\mu$, we compute an exact maximum flow F^* for the current graph from scratch using a static algorithm, update F using the value of F^* and set $\tau = 0$. We call such a step a *rebuild* step. Observe that since we are in the insertions-only setting, once a maximum flow is larger than μ, it will always remain larger than that value. Finally, to answer a query about the maximum s-t flow, we return F as an estimate. These procedures are summarized in Algorithm 1.

Algorithm 1 Incremental Approximate Maximum Flow (IncApproxMF.)

```
1  Procedure Initialize(G = (V, E), s, t, ϵ, μ)
2      Set E ← 0 and F ← 0
3      Invoke IncBMF.Initialize(G, s, t, μ + 1)
4  Procedure Insert(u, v)
5      E ← E ∪ {(u, v)}
6      if IncBMF.MaxFlow(s, t) ≤ μ then
7          Invoke IncBMF.Insert(u, v)
8          Set F ← IncBMF.MaxFlow(s, t)
9      else
10         Set τ ← τ + 1
11         if τ ≥ ϵμ then
12             Compute an s-t maximum flow in G using a static algorithm
13             Set F to be the value of the flow computed in the previous step
14             Set τ ← 0
15 Procedure MaxFlow(s, t)
16     return F
```

▶ **Theorem 7** (Restatement of Theorem 1). *Let $G = (V, E)$ be an initially empty, directed, unweighted n-vertex graph, s and t be any two vertices, and $\epsilon > 0$, $\mu \in [0, n]$ be two parameters. If there is*

(a) *an incremental algorithm* IncBMF(G, s, t, μ) *for inserting m edges and maintaining s-t maximum flow whose value is bounded by μ in total update time $t_{total}(m, n, \mu)$ and $q(m, n)$ query time, and*

(b) *a static algorithm for computing exact s-t maximum flow in an n-vertex, m'-edge graph in $t_{static}(m', n)$ time, where $m' \leq m$,*

then we can design an incremental algorithm for maintaining a $(1+\epsilon)$-approximate s-t maximum flow under m edge insertions in

$$\frac{t_{total}(m, n, \mu+1)}{m} + \frac{t_{static}(m, n)}{\epsilon\mu} + q(m, n)$$

amortized update time and $q(m, n)$ query time.

Proof. We first prove the correctness of the algorithm. Let G be the current graph and let F be the estimate maintained by the algorithm to the value of the maximum s-t flow F^* in G. We will show that F is an $(1+\epsilon)$-approximation to F^*. To this end, we distinguish the following three cases.

(1) If $F^* \leq \mu$, then by assumption of the theorem, the data structure IncBMFensures that $F = F^*$ and thus our claim trivially holds.

(2) If $F^* = \mu + 1$, the call IncBMF.MaxFlow(s, t) returns the value $\mu + 1$ and, thus, the algorithm reaches the else-case for the first time (here we slightly abuse the notation and denote this as a *rebuild* step).

(3) If $F^* > \mu + 1$, then this is not the first time that the algorithm reaches the else-case and, thus, there was a prior rebuild. Note that F corresponded to the value of some s-t maximum flow at the last prior rebuild. This in turn implies that F must be larger than μ. Let F_0^* be the value of the maximum s-t flow of the graph at that rebuild. Since each edge insertion can increase the value of the maximum flow by at most 1 and we recompute a new maximum flow every $\epsilon\mu$ insertions, we have that $F^* \leq F_0^* + \epsilon\mu$. Since $F_0^* > \mu$ and $F = F_0^* \geq 1$, bringing these together yields:

$$\frac{F^*}{F} \leq \frac{F_0^* + \epsilon\mu}{F} \leq \frac{(1+\epsilon)F_0^*}{F_0^*} \leq 1 + \epsilon,$$

which proves our claimed approximation guarantee.

We next study the running time. Note that our algorithm passes the edge insertions to the incremental algorithm IncBMF (invoked with the parameter $\mu + 1$) only if the value of the maximum flow in the current graph is bounded by μ. Hence, by the theorem assumption, the total update time to handle these insertions is $t_{\text{total}}(m, n, \mu+1) + mq(m, n)$. Amortizing the latter over m insertions gives an amortize cost of $t_{\text{total}}(m, n, \mu+1)/m + q(m, n)$, which in turn gives the first and the third term of our claimed running time guarantee.

It remains to analyze the cost of periodical rebuilds. Note that if the current maximum flow value is larger than μ, our algorithm updates the estimate F every $\epsilon\mu$ operations. By assumption of the theorem, the time to compute an exact maximum flow is $t_{\text{static}}(m', n) \leq t_{\text{static}}(m, n)$ as $m' \leq m$. Charging this time over $\epsilon\mu$ insertions, yields an amortized cost of $t_{\text{static}}(m, n)/(\epsilon\mu)$, which in turn gives the second term our of claimed runtime guarantee and completes the proof of the theorem. ◀

4 Incremental Bounded Maximum Flow

In this section we give two incremental algorithms for exactly maintaining the maximum flow as long as its value is bounded by a predefined parameter μ. The first is an incremental version of the Ford-Fulkerson [9] algorithm, applies to directed graphs, and runs in $O(m\mu)$ total update time, while the second one is an incremental version of an algorithm due to Karger and Levine [22], applies to undirected graphs and runs in $\tilde{O}(m + n\mu^{3/2})$ total update time.

4.1 Directed Graphs

The algorithm we are about to discuss applies to directed, unweighted graphs, was initially observed by Henzinger [17] and later by Gupta and Khan [14], and can be thought of as an incremental version of the celebrated Ford-Fulkserson algorithm [9]. We review it below and slightly adapt it for our purposes.

Henzinger [17] showed how to incrementally maintain maximum s-t flow in $O(F^*)$ amortized update time, where F^* is the value of the maximum flow in the final graph. As there are graphs where $F^* = \Omega(n)$, her running time guarantee is competitive only when F^* is small, e.g., sub-linear on the size of the graph. We next show to slightly adapt her algorithm so that it maintains a maximum flow as long as its value is bounded by a parameter μ.

The key observation behind this algorithm is that the insertions of an (unit-capacitated) edge can only increase the maximum flow value by at most 1. To check whether this value has increased, she uses Lemma 4 as a certificate, i.e., one determines whether the insertion of the (forward) edge in the residual graph G_f creates a directed path from s to t in G_f. A naive way to determine this is to run a graph search algorithm on G_f after each insertion, which requires $\Omega(m)$ for a single update and is thus prohibitively expensive for our purposes. However, one can exploit a data structure due to Italiano [19] for incrementally maintaining single source reachability information from a source s which requires $O(m)$ total update time for handling m insertions. Let us briefly review this data structure before presenting the incremental algorithm.

Incremental Single Source Reachability

In the incremental single source reachability problem, given an (initially empty) directed, unweighted graph $G = (V, E)$ and a distinguished vertex s, the goal is to construct a data structure IncSSR that supports the following operations: (i) Initialize(G, s): initialize the data structure in G with source s, (ii) Insert(u, v): insert the edge (u, v) in G, and (iii) Reach(u): return `True` if u is reachable from s, and `False` otherwise.

Italiano [19] observed that an incremental version of graph search leads to an efficient incremental IncSSR data structure. The main idea is to maintain a reachability tree T from s. Initially, the tree is initialized to $\{s\}$. Upon insertion of an edge (u, v) to G, we need to update T iff $u \in T$ and $v \notin T$. If this is the case, we add (u, v) to T and make the v the child of u. Moreover, the algorithm examines all outgoing neighbors w incident to v, and if $w \notin T$, processes the edge (v, w) recursively using the same procedure. To answer queries, we return `True` if $u \in T$, and `False` otherwise. These procedures are summarized in Algorithm 2.

The correctness of the data structure immediately follows by construction as we always maintain a correct rechability tree T from s for the current graph. For the running time, note that the total time over all insertions is $O(m)$ as each edge is processed at most $O(1)$ times; once when it is inserted into the graph and once when it is added to T.

▶ **Lemma 8** ([19]). *Given an initially empty directed, unweighted graph $G = (V, E)$ and a source vertex s, the incremental algorithm* IncSSR *maintains reachability information from s to every other node in V while supporting insertions in $O(1)$ amortized update time and queries in $O(1)$ in worst-case time.*

The Algorithm

We now have all the necessary tools to present an incremental algorithm maintaining the maximum flow whose value is bounded by μ. Let F^* denote the maximum s-t flow value on the current graph.

Algorithm 2 Incremental Single Source Reachability (IncSSR).

```
1  Procedure Initialize(G = (V, E), s)
2      Set T ← {s} and E ← ∅
3  Procedure Insert(u, v)
4      E ← E ∪ {(u, v)}
5      UpdateTree(u, v)
6  Procedure UpdateTree(u, v)
7      if u ∈ T and v ∉ T then
8          Make v a child of u in T
9          foreach (v, w) ∈ E do
10             UpdateTree(v, w)
11 Procedure Reach(u)
12     if u ∈ T then
13         return True
14     else
15         return False
```

Initially, G and the residual graph G_f are empty graphs, $F^* \leftarrow 0$ and $f(e) \leftarrow 0$ for each $e \in E$. The algorithm proceeds in μ rounds, where a round ends when the value of the current maximum flow increases by one. Each round starts by initializing an incremental single source reachability data structure IncrSSR from the source s (Lemma 8) on the residual graph G_f. Upon an edge insertion (u, v) to G, we pass the directed edge (u, v) to the data structure IncrSSR and test whether t is reachable from s using this data structure. If the latter holds, then we find a simple directed s-t path P in G_f, which in turn serves as an augmenting path for G. We then send one unit of flow along the path P in G and update the current flow and its value accordingly. To answer a query about the maximum flow between s and t, we simply return F^*. These procedures are summarized in Algorithm 3.

The correctness of this algorithm is immediate by Lemma 4, which correctly tells us when to increase the value of the maximum flow, and Lemma 5, which asserts that the sending one unit of flow along an augmenting path increases the value of the flow by exactly one.

For the running time, note that each round requires $O(m)$ total time. As there are exactly μ rounds, we get a total update time of $O(m\mu)$. The query time is $O(1)$ as we simply return the value of current maximum flow.

▶ **Lemma 9.** *Given an initially empty directed, unweighted graph $G = (V, E)$ with n vertices, any two vertices s and t, and a parameter $\mu \in [0, n]$, the algorithm* IncBMF(G, s, t, μ) *exactly maintains, under m edge insertions, the maximum s-t flow in G whose value is bounded by μ in $O(m\mu)$ total update time and $O(1)$ query time.*

4.2 Undirected Graphs

We next give an incremental variant of the deterministic maximum flow algorithm for unweighted, *undirected* graphs due to Karger and Levine [22]. For a threshold parameter μ on the maximum flow value, we obtain a total update time of $\tilde{O}(m + n\mu^{3/2})$ for handling m insertions.

Algorithm 3 Incremental Bounded Maximum Flow (IncBMF).

```
1  Procedure Initialize(G = (V, E), s, t, μ)
2      Set E ← ∅
3      Set f(e) ← 0 for each e ∈ E, G_f ← (V, E) and F* ← 0
4      Invoke IncSSR.Initialize(G_f, s)
5  Procedure Insert(u, v)
6      if F* ≤ μ then
7          Set E ← E ∪ {(u, v)}
8          Invoke IncSSR.Insert(u, v)
9          if IncSSR.Reach(t) then
10             Find a simple directed s-t path P in G_f
11             Augment f along the path P in G and let f' be the resulting flow
12             Set f ← f' and G_f ← G_f'
13             Set F* ← F* + 1
14             Invoke IncSSR.Initialize(G_f, s)
15 Procedure MaxFlow(s, t)
16     return F*
```

The basic idea behind this improvement is to sparsify the residual graph on a flow problem so that the augmenting paths can be found more efficiently than paying $O(m)$ per path, as we did in the incremental version of the Ford-Fulkerson algorithm. Two core components that allow for a faster algorithm are: (i) using spanning forests for edges that do not carry any flow in the residual graph (i.e, edges that remain undirected) and (ii) removing cycles from the current flow after each augmentation step to make sure that the flow does not use too many edges.

We next elaborate more on these two components. First, since we will need a different treatment for directed and undirected edges, setting up some additional notation is useful. For a graph $G = (V, E)$ and a flow f on the edges of G, we let E_f^u denote the "undirected edges" of G, i.e., edges e for which $f(e) = 0$, and let E_f^d denote the "directed edges" of G, i.e., edges e for which $f(e) = 1$. Component (i) involves replacing the edges in E_f^u with a spanning forest T. It is known that T captures the connectivity information among any pair of vertices in E_f^u, and thus whenever searching for an augmenting path, it suffices to do so in the graph induced by edge edges E_f^d and T. Another advantage is that T can have at most $(n-1)$ edges, which is potentially much smaller than the size of E_f^u. One challenge with this approach is that E_f^u evolves over time, i.e., edges might have flow added to it or flow is sent on the reserve direction during an augmentation step. Fortunately, we have efficient data structures to maintain such dynamic updates.

▶ **Lemma 10** ([18]). *Given an undirected graph $G = (V, E)$, there is an algorithm* DynSpanF *to maintain a spanning forest T of G that supports operations edge insertions and deletions (i.e., operations Insert(u, v) and Delete(u, v)) in $O(\log^2 n)$ amortized time per operation.*

Unfortunately the above idea alone is not sufficient. The problem is that we do not have any control on the size of E_f^d. It can be well the case that all edges in the graph become eventually directed, which defeats the purpose of treating undirected edges differently. To get around this, we first introduce the notion of acyclic flows and then review a result that shows that integral acyclic flows use very few edges. This lays the foundations of component (ii).

Algorithm 4 Incremental Bounded Maximum Flow for Undirected Graphs (IncBMFU).

```
1  Procedure Initialize(G = (V, E), s, t, μ)
2      Set f(e) ← 0 for each e ∈ E, E_f^u ← ∅, E_f^d ← ∅ and F* ← 0
3      Invoke DynSpanF.Initialize(G = (V, E_f^u)) to maintain a spanning forest T
4      Invoke IncSSR.Initialize(E_f^d ∪ T, s)
5  Procedure Insert(u, v)
6      if F* ≤ μ then
7          Set E_f^u ← E_f^u ∪ {(u, v)}
8          Invoke DynSpanF.Insert(u, v)
9          if (u, v) ∈ T then
10             Invoke IncSSR.Insert(u, v) and IncSSR.Insert(v, u)
11             if IncSSR.Reach(t) then
12                 Find a simple directed s-t path P in E_f^d ∪ T
13                 Augment f along the path P in G, let f' be the resulting flow and set
                    f ← f'
14                 Set f ← Decycle(f)
                   // delete any edge no longer in E_f^u because flow added
15                 for each e ∈ E_f^u with f(e) > 0 do
16                     Set E_f^u ← E_f^u \ {e} and E_f^d ← E_f^d ∪ {e}
17                     Invoke DynSpanF.Delete(e)
                   // insert an edge to E_f^u because flow removed
18                 for each e ∈ E_f^d with f(e) = 0 do
19                     Set E_f^d ← E_f^d \ {e} and E_f^u ← E_f^u ∪ {e}
20                     Invoke DynSpanF.Insert(e)
21                 Set F* ← F* + 1
22                 Invoke IncSSR.Initialize(E_f^d ∪ T, s)
23 Procedure MaxFlow(s, t)
24     return F*
```

▶ **Definition 11.** *We say that a flow f is* acyclic *if there is no directed cycle on which every edge has positive flow in the direction of the cycle.*

▶ **Lemma 12** ([10]). *Any integral acylic flow f uses at most $O(n\sqrt{v(f)})$ edges.*

Taking cue from the lemma above, our goal would be to ensure that at any time, the current flow we maintain is acyclic. Note that even if a flow is initially acylic, an augmentation step may destroy this property. This suggests that we need a *decycling* step to bring back the flow to the desired state. More importantly, for unweighted, undirected graphs, the *decycling* procedure takes time that is proportional to the number of edges that carry non-zero flow on the current graph.

▶ **Lemma 13** ([22]). *Let G be an unweighted, undirected graph, and let f be a flow of G that is non-zero on exactly x edges. Then there is an algorithm* Decycle(f) *that returns an* acyclic *flow f' with $v(f) = v(f')$ and runs in $O(x)$ time.*

The Algorithm

We now show how the above ideas lead to an incremental algorithm that maintains a maximum flow whose value is bounded by μ. As before, let F^* denote the maximum s-t flow value on the current graph.

Initially, G and the edges sets E_f^u, E_f^d are empty, $F^* \leftarrow 0$ and $f(e) \leftarrow 0$ for each $e \in E$. The algorithm initializes a dynamic spanning forest data structure DynamicSpanF on E_f^u to maintain a spanning forest T (Lemma 10). There are μ rounds, and each round ends when the value of the current maximum flow increases by one. Each round starts by initializing an incremental single source reachability data structure IncrSSR on $E_f^d \cup T$ from the source s (Lemma 8).

Upon an edge insertion (u, v) to G, we first pass this insertion to the data structure DynamicSpanF. If the edge (u, v) ends up being added to T, we then pass this insertion as two edge insertions (u, v) and (v, u) to the data structure IncrSSR and test whether t is reachable from s using this data structure. If the latter holds, then we find a simple directed s-t path in $E_f^d \cup T$, which in turn serves as an augmenting path for G. We then send one unit of flow along the path P in G. To make sure that the flow remains acyclic, we invoke procedure Decycle to remove potential directed cycles and update the current flow to be acyclic. Using the dynamic data structure DynamicSpanF, we delete all edges that no longer belong to E_f^u (because they now carry non-zero flow), and insert all new edges to E_f^u (because flow was removed from them). Finally, we increment the current flow by exactly 1.

To answer a query about the maximum flow between s and t, we simply return F^*. These procedures are summarized in Algorithm 4.

We next argue about the correctness of the algorithm. We start by reviewing the result below which shows that it is safe to restrict our attention to the graph $E_f^d \cup T$ when searching for an augmenting path.

▶ **Lemma 14** ([22]). *Let G_f be the residual graph of an undirected, unweighted graph G with respect to the flow f. Then $E_f^d \cup T$ has an augmenting path if and only if G_f does.*

In light of the lemma above, Lemma 4 and Lemma 5, it suffices to show that our incremental algorithm correctly maintains $E_f^d \cup T$. To this end, observe that this directly follows from (i) the correctness of DynSpanF data structure for maintaining T (Lemma 10) and (ii) by Lines 18-20 in Algorithm 4 which makes sure that the set E_f^d is correctly updated after each augmentation step. This completes the correctness argument.

We prove the running time complexity of the algorithm in the lemma below.

▶ **Lemma 15.** *Given an initially empty undirected, unweighted graph $G = (V, E)$ with n vertices, any two vertices s and t, and a parameter $\mu \in [0, n]$, the algorithm* IncB-MFU(G, s, t, μ) *exactly maintains, under m edge insertions, the maximum s-t flow in G whose value is bounded by μ in $\tilde{O}(m + n\mu^{3/2})$ total update time and $O(1)$ query time.*

Proof. Let us first study the work done to find augmenting paths. Since we decycle flows after each augmentation and the spanning forest T can have at most $2(n-1)$ edges (two edges in reverse direction for each undirected edge), by Lemma 12, each augmentation step is done on a graph with $O(n\sqrt{\mu})$ edges and thus takes $O(n\sqrt{\mu})$ time. Similarly, note that before an augmentation step, the set E_f^d does not change, and we only report to IncSSR data structure the edge insertions that ended up being added to T. There can be at most $2(n-1)$ such edge insertions. Therefore, the total cost of running IncSSR per round is $O(|E_f^d \cup T|) = O(n\sqrt{\mu})$. Since there are μ rounds, the total time is $O(n\mu^{3/2})$.

It remains to account for the dynamic operations handled by DynSpanF data structure. Consider the cost of deletions. An edge is deleted from the data structure whenever we put some non-zero flow on it. Since an augmenting path can have at most n edges, and there are at most μ rounds, this can happen to at most $n\mu$ edges. The latter in turn leads to at most $n\mu$ deletions for a total time of $\tilde{O}(n\mu)$ for handling them (Lemma 10).

We now turn our attention to the cost of insertions. Over the course of the incremental algorithm we pass m edges insertions to DynSpanF, for a total time of $\tilde{O}(m)$ (Lemma 10)). We also also pass insertions to DynSpanF whenever flow has been removed on the edges. However, for flow to be removed from an edge, it must have been first added an on edge, i.e., this edge was passed as a deletion to the data structure. Therefore, we can charge the total cost of these insertions to the total cost of deletions, which we bounded by $\tilde{O}(n\mu)$. This completes the proof of the lemma. ◀

5 Conclusion

In this paper we showed two algorithms for maintaining approximate and exact flows in dynamic graphs undergoing edge insertions. Our dynamic approximation algorithm first showed how to maintain small maximum flows efficiently in the incremental setting, and then employed the well-known technique of periodical rebuilds. For the exact result, we showed that the sparsifiers of residual graphs in the undirected setting can be maintained efficiently under edge insertions.

In general, the dynamic complexity of maximum flows is a largely unexplored area, with many fundamental questions remaining unanswered. For example, do there exist *decremental* algorithms achieving comparable guarantees to the ones we obtained in the incremental setting? Our framework from Theorem 6 readily extends to the graphs undergoing edge deletions only. However, it is not known how to maintain small maximum flows in the decremental setting.

Another fundamental open question is the existence of a fast *fully dynamic* algorithm that approximates maximum flows up to a constant factor. For general undirected graphs, recent research suggests that this question is intimately connected to efficient sparsifiers constructions that (approximately) preserve the cut structure between terminal subset of vertices on graphs. Thus, beyond dynamic graphs, any progress in answering this question would potentially lead to understanding other fundamental problems in graph algorithms.

References

1 Ittai Abraham, David Durfee, Ioannis Koutis, Sebastian Krinninger, and Richard Peng. On fully dynamic graph sparsifiers. In *Symposium on Foundations of Computer Science (FOCS)*, pages 335–344, 2016.

2 Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001.

3 Li Chen, Gramoz Goranci, Monika Henzinger, Richard Peng, and Thatchaphol Saranurak. Fast dynamic cuts, distances and effective resistances via vertex sparsifiers. In *Symposium on Foundations of Computer Science (FOCS)*, 2020.

4 Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 612–623, 2022.

5 Ho Yee Cheung, Tsz Chiu Kwok, and Lap Chi Lau. Fast matrix rank algorithms and applications. *J. ACM*, 60(5):31:1–31:25, 2013.

6 Paul Christiano, Jonathan A. Kelner, Aleksander Madry, Daniel A. Spielman, and Shang-Hua Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Symposium on Theory of Computing (STOC)*, pages 273–282, 2011.

7 Søren Dahlgaard. On the hardness of partially dynamic graph problems and connections to diameter. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 48:1–48:14, 2016.

8 Gary William Flake, Robert E Tarjan, and Kostas Tsioutsiouliklis. Graph clustering and minimum cut trees. *Internet Mathematics*, 1(4):385–408, 2004.

9 L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

10 Zvi Galil and Xiangdong Yu. Short length versions of menger's theorem (extended abstract). In Frank Thomson Leighton and Allan Borodin, editors, *Symposium on Theory of Computing (STOC)*, pages 499–508. ACM, 1995.

11 R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.

12 Gramoz Goranci, Monika Henzinger, Danupon Nanongkai, Thatchaphol Saranurak, Mikkel Thorup, and Christian Wulff-Nilsen. Fully dynamic exact edge connectivity in sublinear time. In *Symposium on Discrete Algorithms (SODA) 2023*, pages 70–86, 2023.

13 Gramoz Goranci, Harald Räcke, Thatchaphol Saranurak, and Zihan Tan. The expander hierarchy and its applications to dynamic graph algorithms. In *Symposium on Discrete Algorithms (SODA)*, 2021.

14 Manoj Gupta and Shahbaz Khan. Simple dynamic algorithms for maximal independent set, maximum flow and maximum matching. In *Symposium on Simplicity in Algorithms (SOSA) 2021*, pages 86–91, 2021.

15 Manoj Gupta and Richard Peng. Fully dynamic (1+ e)-approximate matchings. In *Symposium on Foundations of Computer Science (FOCS)*, pages 548–557, 2013.

16 Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Symposium on Theory of Computing (STOC)*, pages 21–30, 2015.

17 Monika Rauch Henzinger. A static 2-approximation algorithm for vertex connectivity and incremental approximation algorithms for edge and vertex connectivity. *J. Algorithms*, 24(1):194–220, 1997.

18 Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *J. ACM*, 48(4):723–760, 2001.

19 Giuseppe F. Italiano. Amortized efficiency of a path retrieval data structure. *Theor. Comput. Sci.*, 48(3):273–281, 1986.

20 Giuseppe F. Italiano, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen. Improved algorithms for min cut and max flow in undirected planar graphs. In *Symposium on Theory of Computing (STOC)*, pages 313–322, 2011.

21 Adam Karczmarz. Fully dynamic algorithms for minimum weight cycle and related problems. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 198 of *LIPIcs*, pages 83:1–83:20, 2021.

22 David R. Karger and Matthew S. Levine. Finding maximum flows in undirected graphs seems easier than bipartite matching. In *Symposium on the Theory of Computing (STOC)*, pages 69–78. ACM, 1998.

23 Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Symposium on Discrete Algorithms (SODA)*, pages 217–226, 2014.

24 Rohit Khandekar, Satish Rao, and Umesh V. Vazirani. Graph partitioning using single commodity flows. *J. ACM*, 56(4):19:1–19:15, 2009.

25 Yang P. Liu and Aaron Sidford. Faster divergence maximization for faster maximum flow. In *Symposium on Foundations of Computer Science (FOCS)*, 2020.

26 Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *Symposium on Foundations of Computer Science (FOCS)*, pages 253–262, 2013.

27 Aleksander Madry. Computing maximum flow with augmenting electrical flows. In *Symposium on Foundations of Computer Science (FOCS)*, pages 593–602, 2016.

28 Richard Peng. Approximate undirected maximum flows in $O(m\text{polylog}(n))$ time. In *Symposium on Discrete Algorithms (SODA)*, pages 1862–1867, 2016.

29 Thatchaphol Saranurak and Di Wang. Expander decomposition and pruning: Faster, stronger, and simpler. In Timothy M. Chan, editor, *Symposium on Discrete Algorithms (SODA)*, pages 2616–2635. SIAM, 2019.

30 Jonah Sherman. Nearly maximum flows in nearly linear time. In *Symposium on Foundations of Computer Science (FOCS)*, pages 263–269, 2013.

31 Jonah Sherman. Area-convexity, l_∞ regularization, and undirected multicommodity flow. In *Symposium on Theory of Computing (STOC)*, pages 452–460, 2017.

32 Jan van den Brand, Yang P. Liu, and Aaron Sidford. Dynamic maxflow via dynamic interior point methods. *CoRR*, abs/2212.06315, 2022. to appear at STOC'23. `arXiv:2212.06315`.

Low Sample Complexity Participatory Budgeting

Mohak Goyal[1]
Stanford University, CA, USA

Sukolsak Sakshuwong[1]
Stanford University, CA, USA

Sahasrajit Sarmasarkar[1]
Stanford University, CA, USA

Ashish Goel
Stanford University, CA, USA

Abstract

We study low sample complexity mechanisms in participatory budgeting (PB), where each voter votes for a preferred allocation of funds to various projects, subject to project costs and total spending constraints. We analyse the distortion that PB mechanisms introduce relative to the minimum-social-cost outcome in expectation. The Random Dictator mechanism for this problem obtains a distortion of 2. In a special case where every voter votes for exactly one project, [11] obtain a distortion of 4/3. We show that when PB outcomes are determined as any convex combination of the votes of two voters, the distortion is 2. When three uniformly randomly sampled votes are used, we give a PB mechanism that obtains a distortion of at most 1.66, thus breaking the barrier of 2 with the smallest possible sample complexity.

We give a randomized Nash bargaining scheme where two uniformly randomly chosen voters bargain with the disagreement point as the vote of a voter chosen uniformly at random. This mechanism has a distortion of at most 1.66. We provide a lower bound of 1.38 for the distortion of this scheme. Further, we show that PB mechanisms that output a median of the votes of three voters chosen uniformly at random, have a distortion of at most 1.80.

2012 ACM Subject Classification Applied computing

Keywords and phrases Social Choice, Participatory budgeting, Nash bargaining

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.70

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2302.05810` [17]

Supplementary Material *Software (Source Code)*: `https://github.com/Sahasrajit123/Low-sample-complexity-PB`

Acknowledgements We would like to thank Lodewijk Gelauff, (Stanford University), Geoff Ramseyer(Stanford University) and Kamesh Munagala(Duke University) for valuable insights and discussions on the paper and the introduction.

1 Introduction

More than 1500 cities around the globe have begun adopting *Participatory Budgeting* (PB) [22, 13], a process through which residents can vote directly on a city government's use of public funds. Residents might, for example, vote directly on how to allocate a budget of reserved funds between projects like street repairs or library renovations. PB has been shown to promote government transparency, resident engagement, and good governance [23].

[1] In alphabetical order.

We study a PB setup similar to [12] where each vote is an allocation of funds to projects (we call it a "preferred budget") subject to the constraint that the sum of allocations to all projects is equal to one. Projects have a fixed cost, and allocations to any project cannot exceed its cost. However, allocations less than the project's cost are allowed. ([12] consider all project costs equal to one). In this model, therefore, every vote and the outcome of the PB election can be represented as a point on the unit simplex.

We study the *distortion* (Definition 5) that PB mechanisms introduce in expectation relative to the social cost minimizing allocation in the worst case of PB instances, following the lines of [1]. We adopt the ℓ_1 distance as the cost function where a voter with preferred budget a experiences a cost of $d(a, b) = \|a - b\|_1$ from an outcome budget b (Definition 2).

Several preference elicitation methods have been studied for PB [7, 2, 15, 5]. Policymakers must then transform a list of votes into a real-world allocation of funds. Furthermore, even though there may be an "optimal" allocation (under natural notions of social welfare), this allocation may be intractable to compute [19, 21] or difficult to reliably estimate if turnout is low [8]. In some situations, policymakers need to obtain a quick estimate of the budgetary region in which preferences may lie. In these cases, and when running a fully-fledged PB election is costly or difficult, low-sample complexity PB mechanisms are an attractive choice.

Low-sample complexity preference elicitation mechanisms have also been of interest recently in computational social choice [11, 10, 1, 9] – in this work, we give low-sample complexity mechanisms (using the preferred budgets of a small number of sampled voters) for PB, which achieve a distortion of less than 2. Note that 2 is a natural barrier for the distortion in this problem since the *Random Dictator* mechanism achieves a distortion of 2 in our model of PB. The Random Dictator mechanism chooses the outcome as the preferred budget of a uniformly randomly chosen voter. From Theorem 5 of [1], its distortion is at most 2, and from our Lemma 7, it is 2. We further prove that a mechanism that chooses any linear combination of two randomly sampled votes (*Random Diarchy*) also attains a distortion of 2 (Lemma 8). Another low sample-complexity mechanism, *Random Referee* [10], asks a randomly chosen voter ("the referee") to choose one out of two possible outcomes, which are random samples from the preferred budgets of the voters. This mechanism also attains a distortion of at least 2 in our setup (Lemma 9). We give a PB mechanism which samples three voters uniformly at random and attains a distortion of at most 1.66.

1.1 Our Contributions

When the PB mechanism samples three voters uniformly at random, we show that aggregation schemes that choose a median of their preferred budgets achieve a distortion of at most 1.80. We refer to such schemes as the *median schemes* and denote this class of schemes by $\mathcal{M}$.

We then turn to the case where two uniformly randomly chosen voters can come together and "bargain" with a third voter's preferred budget (again chosen uniformly at random) as the "disagreement point." We formulate the bargaining rules for the voters via the well-studied Nash bargaining framework [6]. When these bargaining rules can be further specified by a randomized rule (§4.2), we show that the distortion of the resulting mechanism is at most 1.66 (Theorem 35). We call this mechanism the *randomized Nash bargaining scheme* $\mathfrak{n}_{rand}$.

A key technical tool we use is the analysis of *pessimistic distortion* (PD) (Definition 26) first proposed by [10]. PD is a form of distortion where the comparison is made with a counterfactual which chooses a separate outcome for every small subset of voters (of a fixed size κ), thereby attaining a lower social cost than the true "optimal". In this work, we use $\kappa = 6$. This choice is due to computational constraints. We show that the PD with $\kappa = 6$ is an upper bound on the distortion of our proposed mechanisms with any number of voters n.

We then reduce the problem of computing the PD into a set of linear programs for the median schemes $\mathcal{M}$ and bilinear programs of constant size for the randomized Nash

bargaining scheme $\mathfrak{n}_{rand}$. For this, we use a projection of the preferred budgets of voters into a space (we call it the *incremental allocation space* (§3)) that captures the common preferences of a subset of voters relative to other voters. In the median schemes, funds are allocated to projects ensuring that the final outcome is the median of the preferred budgets of three randomly sampled voters. In the randomized Nash bargaining scheme $\mathfrak{n}_{rand}$, the expected funds allocated to a project satisfy additional proportionality constraints (§4.2), resulting in bilinear programs. Since the proportionality constant is not fixed, this results in another variable in the optimization formulation. The problem has a complex combinatorial structure due to the nuances of Nash bargaining. However, we are able to exploit symmetries of the problem, enabling us to solve it efficiently. Since the bilinear programs are of a constant size (depends on κ, which we set to 6), we can solve these in fixed time.

Same as *Random Dictator* and *Random Referee*, our PB mechanism $\mathfrak{n}_{rand}$ also naturally respects project interactions such as complementarity and substitution as long as the voters are aware of these interactions. This is because the bargaining outcome between two voters is guaranteed to be Pareto optimal for them. We describe this point in detail in §8.

1.2 Related Work

The *sequential deliberation* (SD) mechanism for social choice was proposed in [11] where the two uniformly randomly chosen voters deliberate in each round under the rules of Nash bargaining, and the outcome for every round is the disagreement point for the next round. The SD for one round corresponds to the randomized Nash bargaining scheme $\mathfrak{n}_{rand}$. They analyzed the mechanism in median spaces, which include median graphs and trees, and found an upper bound of the distortion of the mechanism to be 1.208. They also analyze the distortion in the *budget space* (or unit simplex) in a special setting where each voter only approved funds for a single project. In this case, they show that the distortion in the equilibrium of SD is 4/3. This paper extends their work in the case of the unit simplex, such that voters in our model do not have to restrict their vote to one project.

The authors of [16] study a model where voters' opinions evolve via deliberations in small groups over multiple rounds. Opinions in their model correspond to preferred budgets in our model; however, unlike preferred budgets, opinions change as a result of deliberations. They study the distortion in single-winner elections setting and show that it is bounded by $O\left(1+\sqrt{\frac{\log n}{n}}\right)$ when voters deliberate in groups of 3 (n is the number of voters).

The work most closely related to ours is [10]; they study the *random referee* mechanism. We use their technique of analyzing the PD of 6 voters. However, they apply this technique where the underlying decision space is the Euclidean plane and use the underlying geometric structure to perform a grid search. In contrast, we study the PD with Nash bargaining, which leads to a complex structure of outcomes that we capture in linear or bilinear programs.

The authors of [9] analyze low sample-complexity randomized mechanisms for PB. They obtain constant factor guarantees for higher moments of distortion, and the distortion bound they provide is much larger than 2. Several additional results and research directions in PB are described in the survey [4].

1.3 Future Directions

A natural direction for future work is to analyze the distortion for multiple rounds of deliberation in our model, with every round's outcome serving as the next round's disagreement point. Another interesting modelling question is to study the deliberation or bargaining process with more than two agents participating together. Closing the gap of the distortion of $\mathfrak{n}_{rand}$ also remains an interesting open problem.

1.4 Roadmap

We describe the model and preliminaries in §2, introduce a projection operation and give some technical results in §3, characterize the outcome of different schemes in §4. We derive the distortion of the class of median schemes in §5. We derive the distortion under $\mathfrak{n}_{rand}$ in §6. We give empirical results on real-world Participatory Budget (PB) data in §7, and discuss project interactions in §8.

2 Model and Preliminaries

Suppose we have m projects and are required to design a budget. A *budget* denotes the fraction of the total funds that are to be spent on each project. Projects have a maximum possible allocation or "project costs." All votes respect these *project costs*, and consequently, the outcomes of all our mechanisms also respect the project costs.[2] For notational simplicity, we drop the project costs from the model henceforth and operate under the assumption that project costs are 1. All our results trivially follow for general project costs.

▶ **Definition 1.** *Let b_j denote the funds allotted to project j in budget b. We define the* budget simplex *as the set of valid budgets i.e.,* $\mathbb{B} = \{b \in \mathbb{R}^m | \sum_{j=1}^m b_j = 1$ *and* $b_j \geq 0, \forall j \in [m]\}$.

There are n voters, each with a *preferred budget* $v_i \in \mathbb{B}$. A *vote profile* P denotes the list of preferred budgets of all voters, i.e, $P = (v_1, v_2, \ldots, v_n)$. The funds allotted to project j by voter i is $v_{i,j}$. A vote profile defines an *instance of PB*. The outcome of an instance of PB is a budget in $\mathbb{B}$. Voters adopt the ℓ_1 distance as the cost function. (Not to be confused with the project costs, which is a different concept here.)

▶ **Definition 2.** *For $a, b \in \mathbb{B}$, the cost of an outcome b for a voter with preferred budget a is $d(a,b) = \sum_{j=1}^m |a_j - b_j|$. The sum of cost over all budgets, $\sum_{i\in[n]} d(v_i, b)$, is the social-cost of budget b.*

We define the *overlap utility* which is closely related to the *cost*. Note that this notion of overlap utility has been studied in knapsack voting [15, 12].

▶ **Definition 3** (Overlap Utility)**.** $u(a,b) = \sum_{j=1}^m \min(a_j, b_j)$.

▶ **Lemma 4.** *For budgets $a, b \in \mathbb{B}$, $d(a,b) = 2 - 2u(a,b)$.*

A proof is in Appendix A.13 of the extended version [17]. Lemma 4 implies that for a voter, maximizing overlap utility is the same as minimizing the cost. Note that overlap utility is *symmetric*, i.e, $u(a,b) = u(b,a)$.

2.1 Distortion

Here we define *distortion,* which we use as a metric to quantify how good a outcome is in comparison to the optimal solution for minimizing social-cost. We define distortion through the *cost* $d(\cdot,\cdot)$.

▶ **Definition 5.** *The **distortion** of budget b for vote profile P is*

$$\text{Distortion}_P(b) = \frac{\sum_{v\in P} d(v,b)}{\min_{b^*\in\mathbb{B}} \sum_{v\in P} d(v,b^*)}.$$

[2] There is no constraint on the minimum allocation to a project other than that it must be non-negative.

Let $h(P)$ be the output of mechanism h for vote profile P.

▶ **Definition 6.** *The distortion of a class of voting mechanisms $\mathcal{H}$ is:* $\text{Distortion}(\mathcal{H}) = \sup_{n\in\mathbb{Z}^+,\ P\in\mathbb{B}^n,\ h\in\mathcal{H}} \mathbb{E}[\text{Distortion}_P(h(P))]$.

Note that distortion is defined as a supremum over all instances of PB and all mechanisms in class $\mathcal{H}$.[3] The expectation is over the randomness of the mechanism, which also includes the randomness in the selection of voters.

The distortion of a voting mechanism is widely used to evaluate its performance regarding how close its output is to the social cost-minimizing outcome in expectation [1, 20, 3, 14, 10]. The *Random Dictator* [1] voting mechanism has a distortion of 2, as shown in Lemma 7. A proof is given in Appendix A.1 in the extended version [17].

▶ **Lemma 7.** *Any aggregation method constrained to choose its outcome as the preferred budget of a uniformly randomly chosen voter has distortion* 2*.*

Now, consider a mechanism that chooses the outcome via the deliberation between two voters chosen uniformly at random with preferred budgets a and b. Within this class, we consider mechanisms constrained to choose the outcome as a convex combination of budgets a and b.

Now, consider a mechanism constrained to choose the outcome as a linear combination of budgets a and b where a and b denote the preferred budgets of randomly sampled voters. That is, $\alpha(P)a + (1-\alpha(P))b$ for $\alpha(P) \in [0,1]$. Note that $\alpha(P)$ may be optimized over the entire vote profile.[4] We refer to this class of mechanisms as *Random Diarchy* and denote it by $\mathcal{Q}$. Interestingly, the distortion of $\mathcal{Q}$ is 2, the same as that of *Random Dictator*.

▶ **Lemma 8.** *For Random Diarchy* $\inf_{\mathfrak{q}\in\mathcal{Q}} \text{Distortion}(\mathfrak{q}) = 2$.

A proof is given in Appendix A.2 in the extended version [17]. We further show that *Random Referee* scheme described in [11] where one of the two preferred budgets of the bargaining voters is chosen based on the preferred budget of third sampled voter also has a distortion ratio of at least 2 in Lemma 9, proven in Appendix A.3 in the extended version [17]. We denote the class of such mechanisms by $\mathcal{R}$.

▶ **Lemma 9.** *For Random Referee* $\inf_{\mathfrak{q}\in\mathcal{R}} \text{Distortion}(\mathfrak{q}) \geq 2$.

2.2 Model of preference aggregation

Let us define the mechanism formally in steps and we call it **Triadic scheme**.

1. Pick a voter i uniformly at random and set the disagreement point c as the preferred budget of voter i.
2. Now choose two voters a and b uniformly at random with replacement and they bargain with c as the disagreement point.

All our theoretical results in this paper are for the outcome of the triadic scheme. However, as discussed in [11], we can extend this bargaining scheme to multiple rounds by setting the outcome of the previous round as the disagreement point for the next round and sampling the two bargaining voters uniformly at random without replacement. We provide empirical results for this setup for multiple rounds (upto 10 rounds) in § 7.

[3] We will often study the distortion of a single mechanism, i.e., not a class. In that case, $\text{Distortion}(h)$ simply denotes the distortion of the mechanism h.

[4] All such outcomes maximize the sum of the overlap utilities of the deliberating agents.

Our bound on pessimistic distortion assumes that the voters are chosen with replacement as done in [10]. This directly gives us a bound on the distortion when voters are sampled without replacement. It is easy to see that the difference in these bounds is of $O(\frac{1}{n})$. The case where two or more identical voters are sampled out of three defaults to the *Random Dictator* mechanism, which has constant distortion – the probability of this event is of $O(\frac{1}{n})$.

We consider bargaining schemes satisfying one or more of the following constraints, namely a) Pareto efficiency, b) Invariance to Affine Transformation, c) Symmetry, and d) Independence of Irrelevant Alternatives. Bargaining schemes that satisfy all of these constraints are the class of Nash bargaining schemes denoted by $\mathcal{N}$ [6].

▶ **Definition 10.** *An outcome of $\mathcal{N}(a,b,c)$, the **Nash bargaining** between two voters with preferred budgets a and b and the disagreement point c, is a budget z which maximizes the Nash product $(u(a,z)-u(a,c))\times(u(b,z)-u(b,c))$, subject to individual rationality $u(a,z)\geq u(a,c)$ and $u(b,z)\geq u(b,c)$, and in case of a tie between possible outcomes, maximizes $u(c,z)$.*

The fact that $\mathcal{N}$ breaks ties in favor of the disagreement point is crucial for the distortion of triadic scheme with bargaining schemes in $\mathcal{N}$ to be smaller than 2. It is also crucial for the membership of $\mathcal{N}$ in a class of bargaining schemes that maximize the sum of overlap utilities of the bargaining agents and the disagreement point. We now define this class of bargaining schemes.

▶ **Definition 11.** *$\mathcal{M}$ is the class of **median schemes** if any outcome $z\in\mathcal{M}(a,b,c)$ maximises the sum of utilities with budgets a,b and c i.e. $u(z,a)+u(z,b)+u(z,c)$.*

The following important result is proved in Appendix A.11 in the extended version [17].

▶ **Theorem 12.** *Every scheme in $\mathcal{N}$ is also a median scheme i.e. $\mathcal{N}\subseteq\mathcal{M}$*

3 Incremental allocation space

We now give a function that captures the marginal preferences of a subset of voters S regarding the allocation to project j, relative to the preference of the other voters (i.e., $P\setminus S$). This function will be useful as an analytical tool in the paper. Specifically,

▶ **Definition 13.** *Given a vote profile $P=(v_1,v_2,\ldots,v_n)$, and project j, the **incremental project allocation** $X_{j,P}:2^{[n]}\to[0,1]$ maps a subset of budgets S to*

$$X_{j,P}(S)=\max\left(\left(\min_{i\in S}v_{i,j}\right)-\left(\max_{i\in P\setminus S}v_{i,j}\right),0\right).$$

Here max and min over $\emptyset$ are defined as 0 and 1, respectively. $X_{j,P}(S)$ denotes the amount by which the budgets in S *all* agree on *increasing* the allocation to project j above the *maximum* allocation to j by any budget in $P\setminus S$. Summing this quantity over all projects $j\in[m]$ gives us $X_P(S)$, which is defined in the following.

▶ **Definition 14.** *For a vote profile P, the **incremental allocation** $X_P:2^{[n]}\to\mathbb{R}$ is $X_P(S)=\sum_{j=1}^{m}X_{j,P}(S)$ for all $S\subseteq P$.*

We use $X_P(.)$ in §5 since its complexity is dependent only on the number of voters n and not on the number of projects m. This helps us give results valid for arbitrarily large values of m. We illustrate the functions $X_{j,P}(\cdot)$ and $X_P(\cdot)$ in the following example.

▶ **Example 15.** Consider an instance of PB with three projects and a vote profile P with three budgets $a = \langle 1, 0, 0\rangle$, $b = \langle 0, 1, 0\rangle$, and $c = \langle 0.25, 0.25, 0.5\rangle$. [5] Then, $X_{1,P}(a) = 0.75$. This is because the budget a has allocation 1 to project 1, out of which only 0.75 is *incremental* on top of $\max(b_1, c_1)$. Also, $X_{2,P}(a) = X_{3,P}(a) = 0$. As a result, $X_P(a) = 0.75$. Similarly $X_{2,P}(b) = X_P(b) = 0.75$. Also, $X_{3,P}(c) = X_P(c) = 0.5$. Further, $X_P(ac) = 0.25$. This is because the subset $\{a, c\}$ has a minimum allocation of 0.25 to project 1 among themselves. It is also incremental since $b_1 = 0$. Further, we have $X_{1,P}(abc) = X_{2,P}(abc) = X_{3,P}(abc) = X_P(abc) = 0$. This is because the group of all three budgets has no allocation that is common to all. Finally, $X_P(\emptyset) = X_{3,P}(\emptyset) = 0.5$ because no budget allocated funds more than 0.5 to project 3.

We use $\mathcal{P}(P)$ to denote the power set of P. We now give an important corollary regarding the function $X_{j,P}(\cdot)$.

▶ **Corollary 16.** $\sum_{S\in\mathcal{P}(P)} X_{j,P}(S) = 1$, $\forall j \in [m]$.

A proof is given in Appendix A.8 in the extended version [17]. Corollary 16 says that every incremental allocation to project j by budgets in S adds up to 1 when summed over all subsets S (this includes the empty set; $X_{j,P}(\emptyset) > 0$ implies that no voter allocated the full 1 unit budget to project j).

3.1 Projection On Incremental Allocations

We now give a *projection* of $X_{j,P}$ from P to $Q \subseteq P$ to get $X_{j,Q}$. This operation has two applications in this paper. First, it enables us to study the allocations of an outcome z relative to the vote profile P by making projections from $P \cup \{z\}$ to P. Second, it is used to study the outcomes of bargaining with a subset $Q \subseteq P$ of voters with respect to the entire vote profile P via projections from P to Q.

▶ **Lemma 17.** *For any vote profile P and $Q \subseteq P$, the* ***projection from P to Q*** *is $X_{j,Q}(S) = \sum_{\hat{S}\in\mathcal{P}(P\setminus Q)} X_{j,P}(\hat{S} \cup S)$ for all $S \in \mathcal{P}(Q)$, and all $j \in [m]$. Summing over $j \in [m]$, $X_Q(S) = \sum_{\hat{S}\in\mathcal{P}(P\setminus Q)} X_P(\hat{S} \cup S)$.*

A proof is given in Appendix A.9 in the extended version [17]. Lemma 17 captures an important technical fact. To calculate the incremental project allocation function on S over a vote profile $Q \subseteq P$, i.e., $X_Q(S)$, we may sum $X_P(\cdot)$ over all subsets of budgets in P which contain all elements of S but no element of $Q \setminus S$. Note that here $S \subseteq Q \subseteq P$.

We now consider the problem of analyzing an outcome z with the help of the incremental allocation function. Towards this, we define the function $Z_{j,P}(S)$ with respect to an outcome z with the help of the projection operation described in Lemma 17.

▶ **Definition 18.** *For vote profile P and budget z, define $Z_{j,P} : 2^{[n]} \to [0, 1]$ as $Z_{j,P}(S) = X_{j,P\cup\{z\}}(S \cup \{z\})$ $\forall S \subseteq P$.*

Recall from Definition 13 that $X_{j,P}(S)$ denotes the amount by which *all* budgets in S want to increase the allocation to project j over the maximum allocation to j by any budget in $P \setminus S$. The quantity $Z_{j,P}(S)$ denotes the amount by which the outcome budget z "accepts" this preference of S. Naturally, $Z_{j,P}(S) \leq X_{j,P}(S)$.

Analogous to summing $X_{j,P}(S)$ over all $j \in [m]$ to get $X_P(S)$, we can sum $Z_{j,P}(S)$ over all $j \in [m]$ to get $Z_P(S)$.

[5] For brevity, we omit braces and commas in the argument of X.

▶ **Definition 19.** *For vote profile P and budget z, define $Z_P(S) : 2^{[n]} \to [0,1]$ as $Z_P(S) = \sum_{j=1}^{m} Z_{j,P}(S)$ $\forall S \subseteq P$.*

$Z_P(S)$ informally denotes the amount by which outcome budget z "accepts" the preference of S for increasing allocations above the allocations of $P \setminus S$ across all projects. See that $Z_P(S) \leq X_P(S)$.

▶ **Corollary 20.** *For any vote profile P and budget z, $Z_{j,P}(S) \leq X_{j,P}(S)$ for all $j \in [m]$. Summing over $j \in [m]$, $Z_P(S) \leq X_P(S)$.*

Proof. Follows directly from Lemma 17 since $X_{j,P}(S) = Z_{j,P}(S) + X_{j,P\cup\{z\}}(S)$ (we are projecting from $P \cup \{z\}$ to P). ◀

▶ **Corollary 21.** *$\sum_{S\in\mathcal{P}(P)} Z_{j,P}(S) = z_j$ for all vote profiles P and $z \in \mathbb{B}$. Summing over all projects $j \in [m]$, we get $\sum_{S\in\mathcal{P}(P)} Z_P(S) = 1$.*

Proof. We have $z_j = X_{j,\{z\}}(\{z\})$ [Definition 13]. Apply Lemma 17 by doing a projection from $P \cup \{z\}$ to $\{z\}$. ◀

This result captures, in the incremental common budget space, the fact that the total funds allocated by a budget z to projects $j \in [m]$ is 1. The following example illustrates $Z_{j,P}(S)$.

▶ **Example 22.** Consider vote profile $P = \{a, b, c\}$ with two projects. Let the budgets a, b, and c be $\langle 0.2, 0.8\rangle, \langle 0.5, 0.5\rangle$, and $\langle 0.8, 0.2\rangle$ respectively. Let the outcome budget z be $\langle 0.4, 0.6\rangle$. In this case, $X_{2,P}(ab) = 0.3$ and $Z_{2,P}(ab) = 0.3$ since the excess allocation by outcome z to project 2 over the allocation by budget c (i.e., 0.4) is larger than the least excess allocation to project 2 by budgets a and b over allocation in budget c (i.e., $X_{2,P}(ab)$ which is 0.3). In other words, the entire incremental allocation to project 2 by budgets a and b is accepted by outcome z. However, $X_{2,P}(a) = 0.3$ but $Z_{2,P}(a) = 0.1$ since the incremental allocation to project 2 by budget z over budgets b and c is 0.1. Thus only a partial incremental allocation to project 2 by budget a is "accepted" by budget z.

4 Overview of Median and Nash bargaining schemes

Recall the triadic mechanism from § 2.2 and we characterize its outcome. Let the disagreement point be c and the preferred budgets of the agents chosen randomly for the mechanism be a and b. For simplicity of notation, we denote $X_{\{a,b,c\}}(S)$ by $\mathfrak{X}(S)$ for S being any subset of $\{a, b, c\}$[6]. We also denote the outcome budget of the bargaining by z and $Z_{(a,b,c)}(S) = X_{\{a,b,c,z\}}(S\cup\{z\})$ by $\mathcal{Z}(S)$ for $S \subseteq \{a, b, c\}$.

4.1 Overview of class of schemes $\mathcal{M}$ and $\mathcal{N}$

In Figure 1, we illustrate the incremental allocations $\{\mathfrak{X}(S)\}_{S\subseteq\{a,b,c\}}$ with budgets a, b, and c on a Venn diagram. Recall from Definition 19 that $\mathcal{Z}(S)$ denotes what incremental allocation from $\mathfrak{X}(S)$ is "accepted" by outcome z. For the construction of $\mathcal{Z}(\cdot)$, the bargaining agents first select all the allocations "agreed" to by at least two of the three budgets. In Figure 1, this corresponds to the area of the overlaps. Now, we have two cases, i.e. the total allocation to $\mathcal{Z}$ is less than 1 or exceeds 1. We denote the difference between 1 and the total allocation to $\mathcal{Z}$ by Excess.

[6] Note that we do not consider $X_P(\cdot)$ in this section where P is the set of the preferred budgets of all the voters, even those not involved in the bargaining.

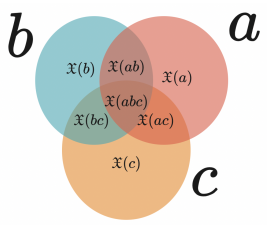


Figure 1 Incremental allocations with two preferred budgets of bargaining agents a and b, and disagreement point c.

Consider the case when the total allocation to $\mathcal{Z}(S)$ is less than 1. Here, the agents need to make further allocations worth Excess. Under the class of median schemes $\mathcal{M}$ [described in §4.3], they may select project allocations from $\mathfrak{X}(a), \mathfrak{X}(b)$, and $\mathfrak{X}(c)$ arbitrarily into the outcome z and thus into $\mathcal{Z}(a), \mathcal{Z}(b)$ and $\mathcal{Z}(c)$. In Figure 1, this corresponds to the area covered by exactly one of the budgets. However, under Nash bargaining schemes $\mathcal{N}$, they select allocations worth $\frac{\text{Excess}}{2}$ from each of $\mathfrak{X}(a)$ and $\mathfrak{X}(b)$.

Now, consider the case when the total allocation to $\mathcal{Z}(S)$ is more than 1. In this case, under median schemes, $\mathcal{M}$, the participating agents select total project allocations worth Excess arbitrarily from $\mathfrak{X}(ab), \mathfrak{X}(bc)$ and $\mathfrak{X}(ca)$ and remove allocations to these projects. In Figure 1, this corresponds to the area of the overlap of exactly two budgets. However, under Nash bargaining schemes $\mathcal{N}$, they select allocations worth $\frac{|\text{Excess}|}{2}$ from each of $\mathfrak{X}(ac)$ and $\mathfrak{X}(bc)$ and remove allocations to these projects from the outcome z.

The following lemma characterizes the overlap of the outcome $z \in \mathcal{N}(a, b, c)$ with the budgets a, b, and c, in terms of the incremental allocation functions $\mathfrak{X}(\cdot)$ and $\mathcal{Z}(\cdot)$.

▶ **Lemma 23.** *For any preferred budgets of bargaining agents a and b, disagreement point c, and outcome z of $\mathcal{N}(a, b, c)$,*

$$\begin{aligned}
\mathcal{Z}(abc) &= \mathfrak{X}(abc), \quad \mathcal{Z}(ab) = \mathfrak{X}(ab),\\
\mathcal{Z}(ac) &= \mathfrak{X}(ac) + \min(\textit{Excess}/2, 0),\\
\mathcal{Z}(bc) &= \mathfrak{X}(bc) + \min(\textit{Excess}/2, 0),\\
\mathcal{Z}(a) &= \mathcal{Z}(b) = \max(0, \textit{Excess}/2),\\
\mathcal{Z}(c) &= \mathcal{Z}(\emptyset) = 0.
\end{aligned}$$

Where, $\textit{Excess} = (1 - \mathfrak{X}(abc) - \mathfrak{X}(ab) - \mathfrak{X}(ac) - \mathfrak{X}(bc))$.

Proof Sketch. In Nash bargaining, no part of z is such that it is not preferred by both a and b. That is, $\mathcal{Z}(c) = \mathcal{Z}(\emptyset) = 0$. Otherwise, we could construct a new outcome z' that reallocated the funds from $\mathcal{Z}(c)$ or $\mathcal{Z}(\emptyset)$ to $\mathcal{Z}(a)$ and $\mathcal{Z}(b)$. This would increase $u(a, z)$ and $u(b, z)$ and thus z would not be Pareto optimal. The parts of z that benefit both a and b must be maximized. That is, $\mathcal{Z}(abc) = \mathfrak{X}(abc)$ and $\mathcal{Z}(ab) = \mathfrak{X}(ab)$. Otherwise, we could construct a new outcome z' that reallocates funds from any other project to the project that benefits both a and b, thus showing that z is not Pareto optimal. The remaining part of the proof is technical and is in Appendix A.12 in the extended version [17]. ◀

We give an explanation of the construction of the Nash bargaining solution z (and correspondingly $\mathcal{Z}$) in three steps.[7]

[7] The steps are only for illustration purposes. There is no chronology or structure required in bargaining processes. We can only characterize the outcome.

Step 1: The voters with preferred budgets a and b mutually decide to allocate funds to projects that benefit both of them. This means, for all projects $j \in [m]$, $z_j = \min(a_j, b_j)$. In terms of $\mathfrak{X}(\cdot)$ and $\mathcal{Z}(\cdot)$, this corresponds to $\mathcal{Z}(abc) = \mathfrak{X}(abc)$ and $\mathcal{Z}(ab) = \mathfrak{X}(ab)$. At this point, $\mathcal{Z}(\cdot)$ is zero for all other subsets of $\{a, b, c\}$.

Step 2: At this point, the total allocation to projects in the bargaining outcome z may be less than 1. The bargaining agents now allocate more funds to the projects $j \in [m]$ for which $z_j < \max(a_j, b_j)$ *and* $z_j < c_j$. Now z_j is set to the "median" of (a_j, b_j, c_j) for *all* projects $j \in [m]$. In terms of $\mathfrak{X}(\cdot)$ and $\mathcal{Z}(\cdot)$, this corresponds to setting $\mathcal{Z}(ac) = \mathfrak{X}(ac)$ and $\mathcal{Z}(bc) = \mathfrak{X}(bc)$.

Step 3: Now, two possibilities arise for the total amount of funds allocated in z so far, i.e., the bargaining agents have either over-spent or under-spent the total funds. These cases are central to the analysis in the paper and will be revisited several times.

Case 1: The total funds currently allocated in z is at most 1, i.e., $\mathcal{Z}(ab) + \mathcal{Z}(bc) + \mathcal{Z}(ac) + \mathcal{Z}(abc) \leq 1$. This is same as:

$$\mathfrak{X}(ab) + \mathfrak{X}(bc) + \mathfrak{X}(ac) + \mathfrak{X}(abc) \leq 1. \tag{1}$$

Recall the definition of Excess in Lemma 23. In this case, since there is a positive Excess, the bargaining agents now allocate more funds to projects with $z_j < \max(a_j, b_j)$. Since in Nash bargaining we assume equal importance of the overlap utilities of both the bargaining agents, they divide the Excess equally. They incrementally fund projects with $z_j < a_j$ and the projects with $z_j < b_j$ with Excess/2 amount each. They ensure that $z_j \leq \max(a_j, b_j)$. The precise manner of doing so is not important to satisfy the axioms of Nash bargaining. In terms of $\mathcal{Z}(\cdot)$, this corresponds to setting $\mathcal{Z}(a) = \mathcal{Z}(b) = \text{Excess}/2$.

Case 2: The total funds currently allocated in z exceeds 1, i.e., $\mathcal{Z}(ab) + \mathcal{Z}(bc) + \mathcal{Z}(ac) + \mathcal{Z}(abc) \geq 1$. This is same as:

$$\mathfrak{X}(ab) + \mathfrak{X}(bc) + \mathfrak{X}(ac) + \mathfrak{X}(abc) \geq 1. \tag{2}$$

If we are in this case, then the bargaining agents have overspent the funds and Excess is negative. They need to remove $-$Excess amount of allocations from z. Recall that at this point, z_j is set to the median of (a_j, b_j, c_j) for *all* projects $j \in [m]$. They remove funds from projects with $(z_j > a_j)$ and the projects with $(z_j > b_j)$ with Excess/2 amount each. They ensure that $z_j \geq \min(a_j, b_j)$. The precise manner of doing so is not important to satisfy the axioms of Nash bargaining. In terms of $\mathcal{Z}(\cdot)$, this corresponds to setting $\mathcal{Z}(ac) = \mathfrak{X}(ac) + \text{Excess}/2$, and $\mathcal{Z}(bc) = \mathfrak{X}(bc) + \text{Excess}/2$.

We now give a randomized way of allocating the Excess funds in **Step 3** while satisfying the axioms of Nash bargaining.

4.2 Randomised Nash bargaining solution $\mathfrak{n}_{\mathsf{rand}}$

Case 1: Denote $s_j^a = \max\{a_j - z_j, 0\}$ for all projects j.[8] To projects with $s_j^a > 0$, allocate incremental funds r_j^a at random such that $\mathbb{E}[r_j^a]$ is proportional to s_j^a. The sum of r_j^a over all $j \in [m]$ is Excess/2 and no incremental allocation r_j^a is more than s_j^a.[9] A similar process is followed for projects j with $z_j < b_j$ by defining $s_j^b = \max\{b_j - z_j, 0\}$ and making incremental allocations r_j^b summing to Excess/2, $\mathbb{E}[r_j^b]$ proportional to s_j^b, and with $r_j^b \leq s_j^b$.

[8] This precisely corresponds to $X_{j,Q}(a)$ in the incremental allocation space.

[9] The randomness of this process is the same as the hypergeometric distribution with (discretized) s_j^a balls corresponding to each project $j \in [m]$ in an urn, and we pick (discretized) Excess/2 balls without replacement to provide incremental allocations.

Case 2: Denote $t_j^a = \max\{z_j - a_j, 0\}$ for all projects $j \in [m]$.[10] From projects with $t_j^a > 0$, remove r_j^a amount of previously allocated funds at random such that $\mathbb{E}[r_j^a]$ is proportional to t_j^a. The sum of r_j^a over all $j \in [m]$ is $-\text{Excess}/2$ and with $r_j^a \leq t_j^a$. A similar process is followed for projects with $z_j > b_j$ by defining $t_j^b = \max\{z_j - b_j, 0\}$ and removing allocations r_j^b from project j summing to $\text{Excess}/2$, $\mathbb{E}[r_j^b]$ proportional to t_j^b, and with $r_j^b \leq t_j^b$.

We now give a characterization of median schemes $\mathcal{M}$ in terms of $\mathcal{Z}$ [recall that $\mathcal{N} \subseteq \mathcal{M}$ from Theorem 12 in §2.2].

4.3 Median schemes $\mathcal{M}$

▶ **Theorem 24.** *For any budgets $a, b, c \in \mathbb{B}$, a budget $z \in \mathbb{B}$ is in $\mathcal{M}(a, b, c)$ if and only if it satisfies the following conditions.*

1. $\mathcal{Z}(abc) = \mathfrak{X}(abc)$ *and* $\mathcal{Z}(\emptyset) = 0$.
2. *In* ***Case 1****:* $\mathcal{Z}(ab) = \mathfrak{X}(ab)$, $\mathcal{Z}(bc) = \mathfrak{X}(bc)$, $\mathcal{Z}(ca) = \mathfrak{X}(ca)$.
3. *In* ***Case 2****:* $\mathcal{Z}(a) = \mathcal{Z}(b) = \mathcal{Z}(c) = 0$.

The proof of this theorem is technical and is given in Appendix A.10 in the extended version [17].

Note that all the conditions on the outcomes of the bargaining schemes in $\mathcal{M}$ are *symmetric* in all three of $\{a, b, c\}$. However, outcomes in $\mathcal{N}$ also satisfy some additional conditions which may not be symmetric in all three of $\{a, b, c\}$.

We now give a lower bound on Distortion($\mathcal{N}$). Since $\mathcal{M}$ contains $\mathcal{N}$, this bound also applies to Distortion($\mathcal{M}$). Moreover, the same bound also holds for the distortion of $\mathfrak{n}_{\text{rand}}$.

▶ **Theorem 25.** Distortion($\mathcal{M}$) $\geq$ Distortion($\mathcal{N}$) > 1.38.
Also, Distortion($\mathfrak{n}_{rand}$) > 1.38.

Proof. The proof is by the following example of a PB instance. Suppose there are $n_A + n_B$ voters and $n_A + 1$ projects for some $n_A, n_B \geq 1$. Let o_i denote the budget where the i-th project receives allocation 1 and all the other projects get allocation 0. Each voter i in group A ($i \in [n_A]$) prefers budget o_i. Each voter i in group B ($i \in [n_A + n_B] \setminus [n_A]$) prefers budget o_{n_A+1}. The analysis of this example is in Appendix A.4 in the extended version [17] where we set $n_A = 2200$; $n_B = 3000$. ◀

We now give upper bounds of the distortion of $\mathcal{M}$.

5 Distortion Of Schemes in $\mathcal{M}$

To find an upper bound of the distortion of triadic scheme with any bargaining scheme, we use a technique introduced in [10], called *pessimistic distortion* (PD). In this technique, we first analyze the distortion for a small group of voters, call it PD, and then show that the distortion over all voters cannot be more than the PD. Specifically, in this paper, we analyze the PD for a group of 6 voters. The idea is that we allow the counterfactual solution to choose a separate "optimal" budget for every 6-tuple of voters, thereby attaining a smaller social cost than a common outcome for all voters. On the other hand, for our mechanism, we consider the expected social cost under one outcome. This is why the distortion calculated is *pessimistic*. Formally:

[10] This precisely corresponds to $X_{j,Q}(bc)$ in the incremental allocation space.

▶ **Definition 26.** *The **pessimistic distortion (PD)** of the class of mechanisms $\mathcal{M}$ with triadic scheme with 6 voters is:*

$$PD(\mathcal{M}) = \sup_{P\in\mathbb{B}^6;\ h\in\mathcal{M}} \frac{\frac{1}{20}\sum_{Q\in\mathcal{C}([6],3)} \frac{1}{3}\sum_{i\in[6]\setminus Q} d(h(Q),P_i)}{\min_{p\in\mathbb{B}} \frac{1}{6}\sum_{i\in[6]} d(p,P_i)}.$$

Here $\mathcal{C}(S,k)$ denotes the set of all k-combinations of set S.[11]

Notice that in the definition of PD, we only consider the cost for the non-bargaining agents (same as in [10]). We illustrate the PD in Figure 2, where the bargaining is over budgets $\{a,b\}$, the disagreement point is c, and the cost is computed only for $\{d,e,f\}$, the budgets not involved in the bargaining. This definition is more pessimistic than considering all agents' costs. Further, since the outcome of $\mathcal{M}$ is *symmetric* in $\{a,b,c\}$, we can use any *combination* Q of three voters to compute the outcome of bargaining without designating one of the budgets as the disagreement point. The next result, proved in Appendix A.14 in

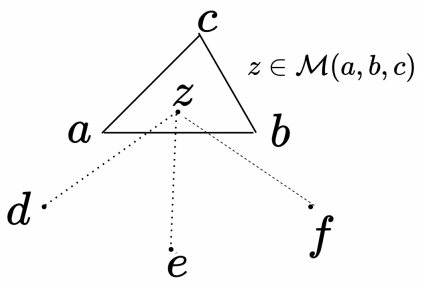


Figure 2 Illustration of PD where a,b,c are sampled for the mechanism $\mathcal{M}$, and $\{d,e,f\}$ are the other budgets for which we measure the cost of outcome z.

the extended version [17], is that the distortion of any bargaining scheme in $\mathcal{M}$ with triadic scheme cannot be more than its PD with triadic scheme with only 6 voters.

▶ **Lemma 27.** $\text{Distortion}(\mathcal{M}) \le PD(\mathcal{M})$.

We now give a representation of the overlap utilities $u(\cdot,\cdot)$ (equivalently the cost $d(\cdot,\cdot)$), in terms of the incremental allocations $X_P(S)$. This representation is of technical importance for proofs.

▶ **Lemma 28.** *For budgets $\{a,b\}$, and a vote profile P that includes $\{a,b\}$, we have $u(a,b) = X_{(ab)}(ab) \overset{(1)}{=} \sum_{\hat{S}\in\mathcal{P}(P\setminus\{a,b\})} X_P(\hat{S}\cup\{a,b\})$.*

Proof. From Definition 3, we have $u(a,b) = \sum_{j=1}^m \min(a_j,b_j)$. From Definition 13 we have $\sum_{j=1}^m \min(a_j,b_j) = \sum_{j=1}^m X_{j,(a,b)}(ab) = X_{(ab)}(ab)$. Now apply Lemma 17 with $Q = S = \{a,b\}$, to obtain equality (1). ◀

[11] For simplicity of notation, we use $\mathfrak{n}(Q)$ in place of $\mathfrak{n}(P_{Q_1},P_{Q_2},P_{Q_3})$ in PD.

Lemma 28 shows that the overlap utility between two budgets a, b is the same as the sum of what a, b, and all subsets of the other budgets in P have in common via the incremental allocation function $X_P(S)$. For example, if $P = (a, b, c, d)$, then $u(a, b) = X_P(ab) + X_P(abc) + X_P(abd) + X_P(abcd)$.

Lemma 28 is useful for the proof of the following important result, which is an upper bound for $PD(\mathcal{M})$.

▶ **Lemma 29.** $PD(\mathcal{M}) \leq 1.80$.

We give a sketch of the proof here. The detailed proof is in Appendix A.15 in the extended version [17].

Proof Sketch. Let p^Q denote a budget obtained on bargaining with budgets in set Q using a bargaining scheme in $\mathcal{M}$. Note that mechanisms in $\mathcal{M}$ are *symmetric* in Q therefore, we do not need to designate a disagreement point in Q for analysis.

$$\begin{aligned} PD(\mathcal{M}) &= \sup_{P\in\mathbb{B}^6;\ h\in\mathcal{M}} \frac{\frac{1}{60}\sum_{Q\in\mathcal{C}([6],3)}\sum_{i\in[6]\setminus Q} d(h(Q),P_i)}{\frac{1}{6}\min_{v\in\mathbb{B}}\sum_{i\in[6]} d(v,P_i)}, \\ &\leq \sup_{P\in\mathbb{B}^6} \frac{\frac{1}{60}\sum_{Q\in\mathcal{C}([6],3)}\sup_{p^Q\in\mathcal{M}(Q)}\sum_{i\in[6]\setminus Q} d(p^Q,P_i)}{\frac{1}{6}\min_{v\in\mathbb{B}}\sum_{i\in[6]} d(v,P_i)}. \end{aligned}$$

Suppose that $PD(\mathcal{M}) > 1.80$. Then the following optimization problem has an optimal objective value strictly greater than 0.

$$\begin{aligned} \text{maximize} \quad & \frac{1}{60}\sum_{Q\in\mathcal{C}([6],3)}\sum_{i\in[6]\setminus Q} d(p^Q,P_i) - 1.80\cdot\frac{1}{6}\sum_{i\in[6]} d(v,P_i), \\ \text{subject to} \quad & P\in\mathbb{B}^6, \\ & p^Q\in\mathcal{M}(Q) \qquad\qquad \forall Q\in\mathcal{C}([6],3), \\ & v\in\mathbb{B}. \end{aligned} \tag{3}$$

To convert this problem into a linear program, we map it to the incremental allocation space of the set of 6 budgets $P = \{P_1, P_2, \ldots, P_6\}$. Denote $X_P(\cdot)$ by $X(\cdot)$ for simplicity of notation in the optimization programs. Similar to Definition 19, we define $V(S) = X_{(P\cup\{v\})}(S\cup\{v\})$ via the "optimal" budget v and $Z^Q(S) = X_{(P\cup\{p^Q\})}(S\cup\{p^Q\})$ using the outcome of our mechanism p^Q, for each $Q\in\mathcal{C}([6],3)$.

By Lemma 4, we write the cost in terms of the overlap utility $d(p^Q,P_i) = 2-2u(p^Q,P_i)$, which, by Lemma 28 and the definition of $Z^Q(S)$, equals $2-2\sum_{S\in\mathcal{P}(P\setminus P_i)} Z^Q(S\cup P_i)$. Similarly, we have $d(v,P_i) = 2-2\sum_{S\in\mathcal{P}(P\setminus P_i)} V(S\cup P_i)$. To make the $p^Q\in\mathcal{M}(Q)$ constraints linear, we use case analysis.

Consider a given $Q = \{q_1,q_2,q_3\}\in C([6],3)$ and a budget $p^Q\in\mathbb{B}$. Let $\mathfrak{X}(S) = X_Q(S)$ and $\mathcal{Z}(S) = X_{(Q\cup\{p^Q\})}(S\cup\{p^Q\})$. Theorem 24 implies that $p^Q\in\mathcal{M}(Q)$ if and only if the following holds:

- **Case 1**: If $\mathfrak{X}(q_1q_2q_3)+\mathfrak{X}(q_1q_2)+\mathfrak{X}(q_1q_3)+\mathfrak{X}(q_2q_3)\geq 1$, $\mathcal{Z}(q_1)=\mathcal{Z}(q_2)=\mathcal{Z}(q_3)=0$.
- **Case 2**: If $\mathfrak{X}(q_1q_2q_3)+\mathfrak{X}(q_1q_2)+\mathfrak{X}(q_1q_3)+\mathfrak{X}(q_2q_3)\leq 1$, $\mathcal{Z}(q_1q_2)=\mathfrak{X}(q_1q_2), \mathcal{Z}(q_1q_3)=\mathfrak{X}(q_1q_3), \mathcal{Z}(q_2q_3)=\mathfrak{X}(q_2q_3)$.

We break each $p^Q\in\mathcal{M}(Q)$ constraint into two cases. Since there are $\binom{6}{3}$ such constraints in the optimization problem, there are $2^{\binom{6}{3}}$ cases overall. We represent each case by a binary string of length 20 where a 0 or 1 at each position denotes whether the triplet Q corresponding to that position is in **Case 1** or **Case 2**.

However, most of these $2^{\binom{6}{3}}$ cases are not unique up to the permutation of preferred budgets, i.e. when the preferred budgets of different voters are permuted, we may move from one case to another. Since these cases have the same objective value, we do not need to solve all the cases. Exploiting further symmetries, we have 2136 unique cases, each of which is formulated as a linear program with precise details in Appendix A.15 in the extended version [17]. We obtain the optimal value for each case to be 0 hence, a contradiction. ◀

Using Lemmas 27 and 29, we get the following key result.

▶ **Theorem 30.** Distortion$(\mathcal{M}) \leq 1.80$.

6 Distortion of $\mathfrak{n}_{\text{rand}}$

Recall the randomized Nash bargaining scheme $\mathfrak{n}_{\text{rand}}$ explained in § 4.2. In this section, we derive an upper bound for it. Towards this, we first define a *hypothetical* bargaining scheme $\tilde{\mathfrak{n}}_{\text{rand}}$. This scheme is hypothetical because it assumes that the bargaining agents use some knowledge about the preferred budgets of the non-bargaining agents to break ties among potential outcomes. We then show in Lemma 32 that the Distortion of $\mathfrak{n}_{\text{rand}}$ is at most as much as that of $\tilde{\mathfrak{n}}_{\text{rand}}$. We then bound the Distortion of $\tilde{\mathfrak{n}}_{\text{rand}}$ by its *expected pessimistic distortion* (EPD), a quantity similar in essence to the PD. We define the EPD in Definition 33. Our main technical contribution in this section is the analysis of the EPD of $\tilde{\mathfrak{n}}_{\text{rand}}$, which we do by expressing it as the solution of a bilinear optimization problem.

6.1 Construction of bargaining solution in $\tilde{\mathfrak{n}}_{\text{rand}}$

Recall Definition 18 of $Z_{j,P}(\cdot)$ for an outcome budget z. Also recall that $Z_{j,P}(\cdot)$ satisfies Corollaries 20 and 21. For $\tilde{\mathfrak{n}}_{\text{rand}}$, we characterize the outcome in the incremental allocation space; denoted by $\tilde{Z}_{j,P}(\cdot)$. Same as $Z_{j,P}(\cdot)$, $\tilde{Z}_{j,P}(\cdot)$ also satisfies Corollaries 20 and 21, i.e.,

$$0 \leq \tilde{Z}_{j,P}(S) \leq X_{j,P}(S) \forall S \in \mathcal{P}(P) \text{ and all } j \in [m]. \tag{4}$$

$$\sum_{j=1}^{m} \tilde{Z}_{j,P}(S) = \tilde{Z}_P(S) \quad \text{and,} \quad \sum_{S \in \mathcal{P}(P)} \tilde{Z}_P(S) = 1. \tag{5}$$

Before describing the construction of $\tilde{\mathfrak{n}}_{\text{rand}}$, we now give the following result on the overlap utility $u(a, z)$ of outcome budget z and any budget $a \in P$ in terms of $Z_P(S)$.

▶ **Lemma 31.** *For a vote profile P, a budget $a \in P$, and any budget z, the overlap utility is $u(a, z) = \sum_{S \in \mathcal{P}(P) | S \ni a} Z_P(S)$.*

Proof. In Lemma 28, use z for b, a for a, and $P \cup \{z\}$ for P. ◀

By Lemma 31, $u(v, \tilde{Z}_P) = \sum_{S \in \mathcal{P}(P) | S \ni a} \tilde{Z}_P(S)$.[12] Similarly, the cost can be given by $d(v, \tilde{Z}_P) = 2 - 2u(v, \tilde{Z}_P)$.

Let c be the disagreement point, and $\{a, b\}$ be the preferred budgets of the agents chosen to bargain. Denote $Q = \{a, b, c\}$. For the construction of $\tilde{Z}_P(\cdot)$, we first do **Step 1** and **Step 2** from § 4. We then have for all $j \in [m]$, $\tilde{Z}_{j,P}(S) = X_{j,P}(S)$ for all $S \in \mathcal{P}(P)$ such that S contains at least 2 elements of Q and $\tilde{Z}_{j,P}(S) = 0$ for all other $S \in \mathcal{P}(P)$. We then encounter either **Case 1** or **Case 2**, as in § 4.

[12] Note the overload in the notation of the overlap utility; it was initially defined for a pair of budgets v and z, here we define it for v and $\tilde{Z}$ where $\tilde{Z}$ captures z.

Case 1: Here we need to allocate more funds to projects. Recall the construction of z for $\mathfrak{n}_{\text{rand}}$ in § 4.2. Recall the random incremental allocations r_j^a and r_j^b used in $\mathfrak{n}_{\text{rand}}$. For the incremental allocations in $\tilde{\mathfrak{n}}_{\text{rand}}$ we construct $\alpha_{j,P}(S) = r_j^a \cdot (X_{j,P}(S)/X_{j,Q}(a))$[13] for all $\{S \mid a \in S; b, c \notin S\}$ for all projects $j \in [m]$. Intuitively, this may be thought of as a proportional selection of projects from every subset of budgets S. Similarly we construct $\beta_{j,P}(S) = r_j^b \cdot (X_{j,P}(S)/X_{j,Q}(b))$ for all $\{S \mid b \in S; a, c \notin S\}$ and all projects $j \in [m]$.

Now, set $\tilde{Z}_{j,P}(S) = \tilde{Z}_{j,P}(S) + \alpha_{j,P}(S) \;\; \forall \{S \mid a \in S; b, c \notin S\}$ and $\tilde{Z}_{j,P}(S) = \tilde{Z}_{j,P}(S) + \beta_{j,P}(S) \;\; \forall \{S \mid b \in S; a, c \notin S\}$ and $\forall\, j \in [m]$.

Case 2: In this case we need to remove allocations from projects. Recall the construction of z for $\mathfrak{n}_{\text{rand}}$ in § 4.2. Recall the removals of allocations r_j^a and r_j^b used in $\mathfrak{n}_{\text{rand}}$. For the removals of allocations in $\tilde{\mathfrak{n}}_{\text{rand}}$, we construct $\alpha_{j,P}(S) = r_j^a \cdot (X_{j,P}(S)/X_{j,Q}(bc))$ for all $\{S \mid b, c \in S, a \notin S\}$ for all $j \in [m]$. Similarly we construct $\beta_{j,P}(S) = r_j^b \cdot (X_{j,P}(S)/X_{j,Q}(ac))$ for all $\{S \mid a, c \in S; b \notin S\}$.

Now, set $\tilde{Z}_{j,P}(S) = \tilde{Z}_{j,P}(S) - \alpha_{j,P}(S) \;\forall \{S \mid b, c \in S; a \notin S\}$, and $\tilde{Z}_{j,P}(S) = \tilde{Z}_{j,P}(S) - \beta_{j,P}(S) \;\forall \{S \mid a, c \in S; b \notin S\} \;\forall\, j \in [m]$.

We can now construct $\tilde{Z}_P(S)$ via $\tilde{Z}_P(S) = \sum_{j=1}^m \tilde{Z}_{j,P}(S)$. With this, we now construct $\tilde{Z}_Q$ as the outcome of the hypothetical bargaining process, via the projection from P to Q That is, $\tilde{Z}_Q(S) = \sum_{\hat{S} \in \mathcal{P}(P \setminus Q)} \tilde{Z}_P(S \cup \hat{S})$ [recall projection in Lemma 17].

See that $\{\tilde{Z}_{j,P}(.)\}_{j \in [m]}$ satisfies Corollaries 20 and 21. Further, $\tilde{Z}_Q(.)$ satisfies all equations of Lemma 23 [proof in Appendix A.16 in the extended version [17]].

6.2 Distortion under $\mathfrak{n}_{\text{rand}}$

We now bound the distortion of the triadic scheme with bargaining scheme $\mathfrak{n}_{\text{rand}}$ by that of the hypothetical scheme $\tilde{\mathfrak{n}}_{\text{rand}}$. A proof is in the Appendix A.18 in the extended version [17].

▶ **Lemma 32.** Distortion($\mathfrak{n}_{rand}$) $\leq$ Distortion($\tilde{\mathfrak{n}}_{rand}$).

We now follow a similar approach as in §5 and define expected pessimistic distortion under bargaining scheme $\tilde{\mathfrak{n}}_{\text{rand}}$ as follows.

▶ **Definition 33.** *The **expected pessimistic distortion** of $\tilde{\mathfrak{n}}_{rand}$ with triadic scheme with 6 voters, EPD($\tilde{\mathfrak{n}}_{rand}$) is*

$$\sup_{P \in \mathbb{B}^6} \frac{\frac{1}{60} \sum_{c \in [6]} \sum_{\substack{\{a,b\} \in \\ \mathcal{C}([6]\setminus\{c\},2)}} \frac{1}{3} \sum_{i \in [6]\setminus\{a,b,c\}} \mathbb{E}[d(\tilde{\mathfrak{n}}_{rand}(a,b,c), P_i)]}{\min_{p \in \mathbb{B}} \frac{1}{6} \sum_{i \in [6]} d(p, P_i)}.$$

▶ **Lemma 34.** Distortion($\tilde{\mathfrak{n}}_{rand}$) $\leq$ *EPD*($\tilde{\mathfrak{n}}_{rand}$).

The proof is similar to Lemma 27 and is in Appendix A.19 in the extended version [17].

▶ **Lemma 35.** *EPD*($\tilde{\mathfrak{n}}_{rand}$) ≤ 1.66.

[13] Note that $\alpha_{j,P}(S) \leq r_j^a$ since $X_{j,P}(S) \leq X_{j,Q}(a)$ [follows from Lemma 17] and $\sum_{j=1}^m \sum_{\substack{S \in \mathcal{P}(P) \\ S \ni a, S \not\ni b,c}} \alpha_{j,P}(S) = \frac{\text{EXCESS}}{2}$ since $\sum_{\substack{S \in \mathcal{P}(P) \\ S \ni a, S \not\ni b,c}} X_{j,P}(S) = X_{j,Q}(S)$ [follows from Lemma 17] and the fact that $\sum_{j=1}^m r_j^a = \text{EXCESS}/2$ [as defined in **Case 1** in §4.2].

The proof is similar to that of Lemma 29 and is presented in Appendix A.20 in the extended version [17]. We present the key ideas of the proof here.

Proof Sketch. Recall the construction of $\tilde{Z}_{j,P}(.) \sim \tilde{\mathfrak{n}}_{\text{rand}}(a,b,c)$ and consider **Case 1**. A similar analysis holds for **Case 2** as well.

We show in Appendix A.20 in the extended version [17] that $\mathbb{E}[\tilde{Z}_P(S)] = \gamma_a^1 X_P(S)$ for all $\{S : S \ni a; S \not\ni b, c\}$ and $\mathbb{E}[\tilde{Z}_P(S)] = \gamma_b^1 X_P(S)$ for all $\{S : S \ni b; S \not\ni a, c\}$ for some variables $0 \le \gamma_a^1, \gamma_b^1 \le 1$. Here, γ_a^1 and γ_b^1 denote what fraction of allocation from the incremental allocation $X_P(S)$ is "accepted" into $\tilde{Z}_P(S)$. In our optimization problem formulation equation (29) in Appendix A.20 in the extended version [17], we use γ_b^1, γ_a^1 as variables of our optimization formulation, together with $X_P(S)$ and therefore we obtain a bilinear program. We solve it with the Gurobi solver [18]. Similar to the proof of Lemma 27, we remove the cases that are not unique to permutations of voters and use further symmetries of the problem to reduce number of bilinear programs from $2^{\binom{6}{3}}$ to 1244. ◀

Using Lemmas 32, 34, and 35, we get the following result.

▶ **Theorem 36.** Distortion$(\mathfrak{n}_{rand}) \le 1.66$.

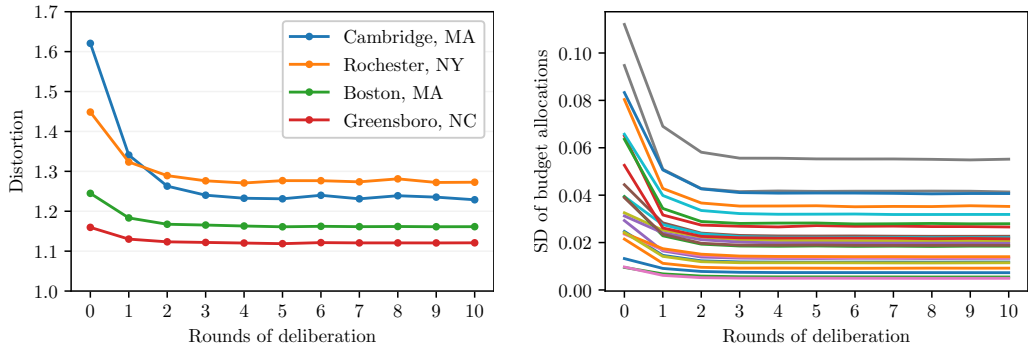
7 Empirical Results

Recall triadic scheme as described in §2.2. We now define a sequential deliberation mechanism that could run bargaining over multiple rounds by setting the disagreement point for each round as the outcome of the previous round as proposed in [11].

1. Pick a voter i uniformly at random. Set the disagreement point for the deliberation c to their preferred budget v_i.
2. Repeat the following process T times,
   a. Pick two voters independently and uniformly at random with replacement. They bargain with c as the disagreement point.
   b. Set the disagreement point c to the outcome of the bargaining.
3. The outcome of the process is c.

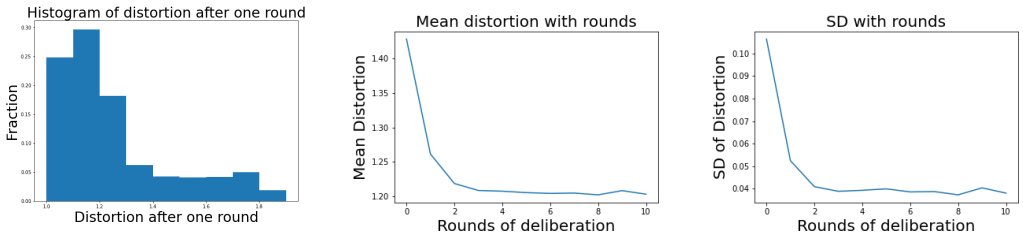
Observe that on setting $T = 1$, we exactly get triadic scheme as §2.2. To evaluate the distortion of sequential deliberation in PB empirically, we ran a simulation from the online participatory budgeting elections in Boston in 2016 ($n = 4,482$), Cambridge in 2015 ($n = 3,273$), Greensboro in 2019 ($n = 512$), and Rochester in 2019 ($n = 1,563$) where the data were obtained from `https://budget.pbstanford.org/`. In these elections, projects had a fixed cost, and voters participated in knapsack voting [15], in which they could choose any number of projects as long as they fitted within the fund limits. Note that in this simulation setup partial project funding is not allowed, unlike the setup in the theoretical model. We further present simulation results in Figures 4a, 4b, 4c on real dataset from a PB (participatory budgeting) process run by a non-profit organisation in Boston in 2016 where they used a fractional allocation setting, more aligned with our theoretical work.

To simulate sequential deliberation, we picked a voter uniformly at random to set their preferred budget as the disagreement point. We then picked another two voters independently and uniformly at random and calculated a Nash bargaining solution between them. We assumed that everyone voted truthfully. We then made the bargaining outcome the new disagreement point and repeated the deliberation process for $T = 10$ rounds. We repeated this entire simulation $10,000$ times for each PB election. The average distortion after each round of deliberation is shown in Figure 3a. The point corresponding to 0 rounds of deliberation is



(a) Average distortion. **(b)** SD of distortion.

Figure 3 (a) The average distortion after each round of sequential deliberation in a simulation using the data from PB elections in four cities. The simulation was run 10,000 times for each city. (b) The standard deviation (SD) of the fund allocation to each project in the simulation of sequential deliberation in the PB election in Cambridge. Each line represents a project.



(a) Histogram of triadic scheme. **(b)** Average distortion. **(c)** SD of distortion.

Figure 4 Distortion results on PB platform in Boston under the fractional allocation setup

the first disagreement point and is selected uniformly at random. Since voters did not have to use all the budget available, we added an "unspent" project and allocated the unspent budget of each voter to this project. We normalized the budget to sum to 1 in each election.

The mean and standard deviation of the distortion after each round of sequential deliberation for the fractional allocation setting as in the PB process in Boston is shown in Figures 4b and 4c, respectively. A histogram plot of the distortion after one round of deliberation is in Figure 4a. As before, we observe a quick convergence within three rounds of sequential deliberation with the point corresponding to zero rounds of deliberation being the first disagreement point.

The results from all the PB elections show that the average distortion is quite low, even after only two rounds of deliberation. It also shows that the distortion converges quickly within three rounds. Further, we measured the stability of the fund allocation to the projects after each round of deliberation. We simulated sequential deliberation on the data from the PB in Cambridge $1,000,000$ times, each time with 10 rounds of deliberation. The fund allocation to each project after each round was recorded. The fund allocation's standard deviation (SD) is shown in Figure 3b. We can see that the SD stabilizes after only three rounds of deliberation.

8 Triadic Scheme With Project Interactions

Mathematically, we model project interactions as follows: if projects in group q are **perfect complements** of each other, then the overlap utility that voters can derive from *each* project in q is the minimum funding of any project in q. For example, consider a proposal of buying some computers for the community. Within this, one project is for buying hardware and another one is for buying software. If the software and hardware projects are funded 0.2 and 0.5, then the community members can only use 0.2 each, and the extra funding of 0.3 for the hardware project is wasted [14].

On the other hand, if the projects in group r are **perfect substitutes**, then the utility that voters can derive from group r is the maximum funding of a project in r. Thus, if two companies are paid 0.2 and 0.5 to do the same work, only 0.5 will be used, and 0.2 is wasted.

We now give a formal model of the set of projects. Let m_c denote the number of groups of *perfect complementary* projects, m_s denote the number of groups of *perfect substitute* projects, and m_r denote the number of *regular* projects. Let $s(q)$ denote the number of projects in group q. For groups of perfect complementary and perfect substitute projects, $s(q) \geq 2$ and for regular projects $s(q) = 1$. The total number of projects is $m = \left(\sum_{q=1}^{m_c+m_s} s(q)\right) + m_r$. For simplicity, project groups are arranged such that groups $1, \ldots, m_c$ are perfect complementary, groups $m_c + 1, \ldots, m_c + m_s$ are perfect substitutes, and $m_c + m_s + 1, \ldots, m_c + m_s + m_r$ are regular projects.

Let $f(b)$ be the *efficiency function* which quantifies how much budget b respects the project interactions. Specifically, $f(b)$ takes a budget $b \in \mathbb{R}^m$ and outputs a vector in $\mathbb{R}^{m_c+m_s+m_r}$, where

$$f(b)_q = \begin{cases} s(q) \cdot \min(\{b_j \mid j \in \text{group } q\}) & \text{if } q \in [1, m_c] \quad \text{(perfect complementary groups)}, \\ \max(\{b_j \mid j \in \text{group } q\}) & \text{if } q \in [m_c + 1, m_c + m_s], \quad \text{(perfect substitute groups)} \\ \{b_j \mid j \in \text{group } q\} & \text{otherwise.} \qquad \text{(regular projects)}. \end{cases}$$

For a group of perfect complementary projects, the corresponding output element is the bottle-neck allocation in the group, multiplied by the number of projects in the group. For a group of perfect substitute projects, the corresponding output element is the largest allocation in that group. For regular projects, the corresponding output elements are the same as the allocation to the project. We now give a modified definition of the overlap utility, accounting for project interactions.

▶ **Definition 37.** *The **overlap utility** of budgets a and b, accounting for project interactions is* $u(a, b) = \sum_{q=1}^{m_c+m_s+m_r} \min(f(a)_q, f(b)_q)$.

In the following definition we formally state the requirements for a budget to be consistent with the project interactions.

▶ **Definition 38.** *A budget b **respects the project interactions** if and only if projects in each perfect complementary group are all funded equally, and at most one project in each perfect substitute group is funded at all.*

The following lemma states that the efficiency function $f(b)$ sums to 1 if and only if the budget b *respects the project interactions.*

[14] This is a stylized model and in general, the scale of the funds required for each project can be very different.

▶ **Lemma 39.** *Budget b respects the project interactions iff the efficiency function $f(b)$ satisfies $\sum_q f(b)_q = 1$. Otherwise, $\sum_q f(b)_q < 1$.*

We now give a result that a Pareto improvement exists over a budget that does not respect the project interactions.

▶ **Lemma 40.** *If $\sum_q f(b)_q < 1$, then for some $k \in [m_c + m_s + m_r]$, there exists a budget b' for which $f(b')_k > f(b)_k$ and $f(b')_q \geq f(b)_q$ for all project groups q.*

The proofs of lemmas 39 and 40 are presented in Appendix A.5 and A.6 in [17].

We now give the main result of this section. We show that if either of the budgets of the bargaining agents respect the project interactions (which will be true for rational agents), then the outcome of any median scheme respects project interactions. Since the class of median schemes contains the class of Nash bargaining schemes (Theorem 12), this result also applied to $\mathcal{N}$ and therefore also to our randomized bargaining scheme $\mathfrak{n}_{\text{rand}}$.

▶ **Theorem 41.** *If budget a or b respects the project interactions, then for any budget $c \in \mathbb{B}$, $\mathcal{M}(a, b, c)$ respects the project interactions.*

Proof. Let z be an outcome from $\mathcal{M}(a, b, c)$. Assume without loss of generality that budget a respects the project interactions, and suppose that outcome z does not. By Lemma 39, $\sum_q f(a)_q = 1$ and $\sum_q f(z)_q < 1$. Thus, there exists some k where $f(a)_k > f(z)_k$. By Lemma 40, there exists a budget z' which respects the project interactions and $f(z')_q \geq f(z)_q$ for all project groups q and $f(z')_k > f(z)_k$. The overlap utility functions satisfy:

$$u(a, z') = \sum_q \min(f(a)_q, f(z')_q) > \sum_q \min(f(a)_q, f(z)_q) = u(a, z),$$
$$u(b, z') = \sum_q \min(f(b)_q, f(z')_q) \geq \sum_q \min(f(b)_q, f(z)_q) = u(b, z).$$
$$u(c, z') = \sum_q \min(f(c)_q, f(z')_q) \geq \sum_q \min(f(c)_q, f(z)_q) = u(c, z).$$

This implies that the sum of overlap utilities of a, b, and C with z' is higher than that with z, a contradiction for an outcome of $\mathcal{M}$. ◀

Theorem 41 implies that if every voter has a preferred budget that respects the project interactions, then the outcome of the sequential deliberation mechanism will also respect the project interactions, no matter how many rounds it runs.

9 Conclusion

We study low sample-complexity mechanisms for PB, which are particularly attractive when the policymakers are interested in obtaining a quick estimate of the voter's preferences or when a full-fledged PB election is difficult or costly to conduct. In our PB setup, the distortion of mechanisms that obtain and use the votes of only one uniformly randomly sampled voter is 2. Extending this result, we show that when two voters are sampled, and a convex combination of their votes is used by the mechanism, the distortion cannot be made smaller than 2. We then show that with 3 samples, there is a significant improvement in the distortion – we give a PB mechanism that obtains a distortion of 1.66. Our mechanism builds on the existing works on Nash bargaining between two voters with a third voter's preferred outcome as the disagreement point. We also give a lower bound of 1.38 for our mechanism.

References

1. Elliot Anshelevich and John Postl. Randomized social choice functions under metric preferences. *Journal of Artificial Intelligence Research*, 58:797–827, 2017.
2. Kenneth J Arrow, Amartya Sen, and Kotaro Suzumura. *Handbook of social choice and welfare*, volume 2. Elsevier, 2010.
3. Haris Aziz, Bo Li, and Xiaowei Wu. Approximate and strategyproof maximin share allocation of chores with ordinal preferences. *Mathematical Programming*, pages 1–27, 2022.
4. Haris Aziz and Nisarg Shah. Participatory budgeting: Models and approaches. In *Pathways Between Social Science and Computational Social Science*, pages 215–236. Springer, 2021.
5. Gerdus Benade, Swaprava Nath, Ariel D Procaccia, and Nisarg Shah. Preference elicitation for participatory budgeting. *Management Science*, 67(5):2813–2827, 2021.
6. Ken Binmore, Ariel Rubinstein, and Asher Wolinsky. The nash bargaining solution in economic modelling. *The RAND Journal of Economics*, pages 176–188, 1986.
7. Steven J Brams and Peter C Fishburn. Voting procedures. *Handbook of social choice and welfare*, 1:173–236, 2002.
8. Hendrik Ewens and Joris van der Voet. Organizational complexity and participatory innovation: participatory budgeting in local government. *Public Management Review*, 21(12):1848–1866, 2019.
9. Brandon Fain, William Fan, and Kamesh Munagala. Concentration of distortion: The value of extra voters in randomized social choice. *IJCAI*, 2020.
10. Brandon Fain, Ashish Goel, Kamesh Munagala, and Nina Prabhu. Random dictators with a random referee: Constant sample complexity mechanisms for social choice. *AAAI*, 2019.
11. Brandon Fain, Ashish Goel, Kamesh Munagala, and Sukolsak Sakshuwong. Sequential deliberation for social choice. In *Web and Internet Economics*, pages 177–190. Springer, 2017.
12. Rupert Freeman, David M Pennock, Dominik Peters, and Jennifer Wortman Vaughan. Truthful aggregation of budget proposals. *Journal of Economic Theory*, 193:105234, 2021.
13. Ernesto Ganuza and Gianpaolo Baiocchi. The power of ambiguity: How participatory budgeting travels the globe. *Journal of Public Deliberation*, 8, 2012.
14. Nikhil Garg, Vijay Kamble, Ashish Goel, David Marn, and Kamesh Munagala. Iterative local voting for collective decision-making in continuous spaces. *Journal of Artificial Intelligence Research*, 64:315–355, 2019.
15. Ashish Goel, Anilesh K. Krishnaswamy, Sukolsak Sakshuwong, and Tanja Aitamurto. Knapsack voting for participatory budgeting. *ACM Transactions on Economics and Computation*, 7(2), 2019.
16. Ashish Goel and David T Lee. Towards large-scale deliberative decision-making: Small groups and the importance of triads. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 287–303, 2016.
17. Mohak Goyal, Sukolsak Sakshuwong, Sahasrajit Sarmasarkar, and Ashish Goel. Low sample complexity participatory budgeting, 2023. `arXiv:2302.05810`.
18. Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022. URL: `https://www.gurobi.com`.
19. Pallavi Jain, Krzysztof Sornat, and Nimrod Talmon. Participatory budgeting with project interactions. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 386–392, 2020.
20. Reshef Meir, Fedor Sandomirskiy, and Moshe Tennenholtz. Representative committees of peers. *Journal of Artificial Intelligence Research*, 71:401–429, 2021.
21. Dominik Peters, Grzegorz Pierczyński, and Piotr Skowron. Proportional participatory budgeting with cardinal utilities. *arXiv preprint*, 2020. `arXiv:2008.13276`.
22. Hilary Wainwright. Making a people's budget in porto alegre. *NACLA Report on the Americas*, 36:37–42, March 2003.
23. Brian Wampler. A guide to participatory budgeting. In *Participatory Budgeting*. World Bank, 2007.

The Impacts of Dimensionality, Diffusion, and Directedness on Intrinsic Cross-Model Simulation in Tile-Based Self-Assembly

Daniel Hader ✉
Department of Computer Science and Computer Engineering,
University of Arkansas, Fayetteville, AR, USA

Matthew J. Patitz ✉
Department of Computer Science and Computer Engineering,
University of Arkansas, Fayetteville, AR, USA

Abstract

Algorithmic self-assembly occurs when components in a disorganized collection autonomously combine to form structures and, by their design and the dynamics of the system, are forced to intrinsically follow the execution of algorithms. Motivated by applications in DNA-nanotechnology, theoretical investigations in algorithmic tile-based self-assembly have blossomed into a mature theory with research strongly leveraging tools from computability theory, complexity theory, information theory, and graph theory to develop a wide range of models and to show that many are computationally universal, while also exposing a wide variety of powers and limitations of each. In addition to computational universality, the abstract Tile-Assembly Model (aTAM) was shown to be intrinsically universal (FOCS 2012), a strong notion of completeness where a single tile set is capable of simulating the full dynamics of all systems within the model; however, this result fundamentally required non-deterministic tile attachments. This was later confirmed necessary when it was shown that the class of directed aTAM systems, those in which all possible sequences of tile attachments eventually result in the same terminal assembly, is not intrinsically universal (FOCS 2016). Furthermore, it was shown that the non-cooperative aTAM, where tiles only need to match on 1 side to bind rather than 2 or more, is not intrinsically universal (SODA 2014) nor computationally universal (STOC 2017). Building on these results to further investigate the impacts of other dynamics, Hader et al. examined several tile-assembly models which varied across (1) the numbers of dimensions used, (2) restrictions imposed on the diffusion of tiles through space, and (3) whether each system is directed, and determined which models exhibited intrinsic universality (SODA 2020). Such results have shed much light on the roles of various aspects of the dynamics of tile-assembly and their effects on the universality of each model. In this paper we extend that previous work to provide direct comparisons of the various models against each other by considering intrinsic simulations between models. Our results show that in some cases, one model is strictly more powerful than another, and in others, pairs of models have mutually exclusive capabilities. This direct comparison of models helps expose the impacts of these three important aspects of self-assembling systems, and further helps to define a hierarchy of tile-assembly models analogous to the hierarchies studied in traditional models of computation.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Tile-Assembly, Tiles, aTAM, Intrinsic Simulation, Simulation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.71

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2305.01877` [8]

Funding *Daniel Hader*: Supported in part by National Science Foundation Grant CAREER-1553166.
Matthew J. Patitz: Supported in part by National Science Foundation Grant CAREER-1553166.

1 Introduction

Self-assembling systems are those in which a disorganized collection of simple components spontaneously combine to form complex, organized structures through random motion and local interactions. From the pristine, periodic arrangements formed by crystallizing atoms to the robust coordination of dividing cells in developing organisms, such systems are the source of much complexity in nature and a topic of critical importance to many fields of research. Among them is the field of DNA nanotechnology, wherein artificial DNA strands are used as structural units that self-assemble according to the dynamics of DNA base pairing, which has seen immense success over the past several decades in harnessing the power of self-assembly to create microscopic structures with incredible precision [3, 11, 12, 18] and even perform algorithmic tasks at the nano-scale [4, 6, 13, 14, 17, 20, 24, 25]. Because it's difficult and expensive to accurately model the chemistry of DNA, a variety of simplifying models have been proposed to facilitate the design of DNA-based self-assembling systems. Among the more popular and effective ones are tile-assembly (TA) models where components, made of several bound DNA strands exposing small unbound portions with which other components can bind, are abstractly represented as geometric tiles whose labeled sides attach to one another according to predefined affinity rules [5, 16, 22]. The advantage of these models lies not only in their success as design tools, but in their similarity to existing models studied heavily in computer science such as Wang tiles and cellular automata. This similarity isn't a coincidence either; the first TA model proposed, the abstract Tile-Assembly Model (aTAM), was designed, at least in part, to show that the dynamics of DNA-based self-assembly are algorithmically universal [22]. Consequently, DNA nanotechnology shares a unique relationship with the theory of computation, with theorists frequently borrowing ideas from complexity, computability, and information theory to study questions regarding, among many other things, what kinds of structures can be self-assembled, the relative difficulty of assembling different shapes, and how variations in a model's dynamics affect its algorithmic power. This paper is particularly focused on that latter question. As with more conventional

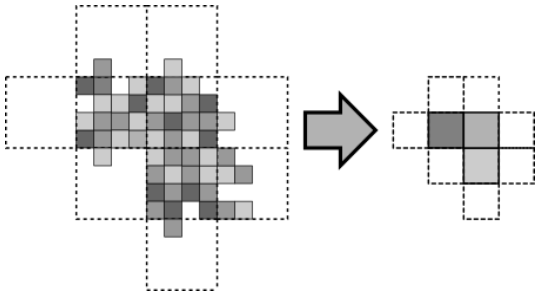


Figure 1 During an intrinsic simulation, the dynamics of individual tile attachments are simulated so that blocks of tiles in the simulating system "look like" individual tiles at scale.

models of computation, we generally study such questions by proving whether one model is capable of simulating all systems of another. We have to be careful about our definition of simulation however, as it's generally straightforward to show that many TA models are capable of universal computation. Consequently, most TA models are capable of "simulating" all others in that they can simulate a Turing machine which can in turn simulate the other model. To learn something useful about the relative power of two TA models therefore, we have to consider the geometry of the tile-assembly dynamics. We do this by adapting a tool from the theory of cellular automata, namely *intrinsic simulation*. For a simulation

to be *intrinsic*, we require that the simulation is not merely symbolic (i.e. how a Turing machine can simulate an aTAM system by storing an internal representation of the tiles as symbols on its tape), but rather geometric wherein blocks of tiles in the simulating system correspond to individual tiles in the simulated system and the order of tile attachments in these blocks follow those in the simulated system up to a fixed scale factor. In other words, such a simulation would appear identical to the system being simulated if we "zoomed out" sufficiently far. This approach is not novel to our results, in fact there is already a relatively mature theory of intrinsic simulations in tile-assembly which has resulted in a "kind of computational complexity theory for self-assembly" [23]. Such efforts have been instrumental in characterizing the relative power of TA models and has lead to a deeper understanding how different dynamics can be used for the same algorithmic purpose.

Our results

In an attempt to extend several previous results regarding intrinsic simulation, here we consider 3 specific variations of the aTAM: *dimensionality*, where both 2D and 3D systems are considered, *diffusion*, where tiles cannot attach in regions which have been surrounded by previously attached tiles, and *directedness*, where tile attachments in a system are required to result in exactly one terminal assembly. It's important to note that these variations aren't arbitrary either. The difference between directed and undirected systems is analogous to the difference between deterministic and probabilistic algorithms and, among other things, plays a role in the study of the complexity of shape assembly [21, 10]. The diffusion restriction on the other hand is often used to make 3D tile-assembly models more "realistic" by limiting tile attachments to those locations in which a tile could reasonably diffuse (i.e. not in a region completely surrounded by other tiles). These variations can be introduced into the aTAM in any combination to yield 8 different models and, considering all ordered pairs of these 8 models gives rise to a table consisting of 64 entries each representing one model's ability or inability to intrinsically simulate the dynamics of another. Generally speaking, results regarding these *cross-model simulations* are complex, involving intricate tile-assembly constructions and counterexamples; consequently, only a handful of these entries have been proved in past literature.

In this paper, we fill a considerable number of missing entries. Table 1 lays out our results along with past results denoted by an asterisk. In it, entries are labeled to indicate whether the model in the row's header can simulate the model in the column's header. There are of course a few entries for which the answer is obvious, which we state as observations with justification rather than full theorems, but many of our results are distinctly non-trivial and some were rather unexpected. For instance, while we initially suspected that the diffusion restricted version of the aTAM (i.e. the Planar aTAM or PaTAM) was, as it's name suggests, a weaker version of the aTAM, we found that both models exhibit dynamics which cannot be simulated by the other. While the table is still missing a few entries, our contributions have brought the number of known entries up to 52 from the 16 which previously existed in published literature (8 of which were technically not explicitly stated, but were trivial observations based on the tile sets and proofs presented in [7]).[1] The rest of our paper is laid out as follows. In Section 2, we provide definitions of the various models and concepts used.

[1] It should also be noted that most of the remaining unknown entries involve simulating directed, diffusion restricted systems. While we do hope to fill these entries in the future, we suspect that their proofs will be quite complicated since simulating diffusion restricted systems is tricky and counterexamples are often harder to find in directed systems.

Table 1 Table of our results, outlining whether the row's model can intrinsically simulate the column's model. PaTAM is the Planar aTAM, 3DaTAM the 3-dimensional aTAM, and SaTAM is the Spatial aTAM (see Section 2.2 for full definitions). *All* refers to the set of all systems in a model and *dir* refers to the subset of directed systems. Cells marked with an asterisk (*) are existing results and those marked with a dagger (†) are trivial observations using tile sets from existing results. All other results are novel.

		aTAM		**PaTAM**	
		all	**dir**	**all**	**dir**
aTAM	**all**	yes* [2]		no (thm. 11, obs. 5)	?
	dir	no (thm. 9)	no* [9]		?
PaTAM	**all**	no (thm. 10, obs. 5)		no* [7]	yes (thm. 13)
	dir			no (thm. 9)	no* [7]
3DaTAM	**all**	yes†(obs. 7)		?	?
	dir	no (thm. 9)	yes† (obs. 7)	no (thm. 9)	?
SaTAM	**all**	yes† (obs. 7)		?	?
	dir	no (thm. 9)	yes† (obs. 7)	no (thm. 9)	?

		3DaTAM		**SaTAM**	
		all	**dir**	**all**	**dir**
aTAM	**all**	no (obs. 6)			
	dir				
PaTAM	**all**				
	dir				
3DaTAM	**all**	yes* [7]		no (thm. 12, obs. 5)	?
	dir	no (thm. 9)	yes* [7]		?
SaTAM	**all**	yes† (obs. 8)		yes* [7]	?
	dir	no (thm. 9)	yes† (obs. 8)	no (thm. 9)	?

Then in Section 3 we state our results explicitly and sketch their proofs. That section is perhaps the most important since it is where we intuitively explain our results and describe how they follow from the dynamics of the model. Complete proofs and technical details can be found in a full version of the paper on arXiv [8].

2 Preliminary definitions

Throughout this paper we will use $\mathbb{Z}$, $\mathbb{Z}^+$, and $\mathbb{N}$ to denote the set of integers, positive integers, and non-negative integers respectively. We will also assume $\mathbb{Z}^d$ has the additional structure of a lattice graph so that each point is a vertex and two points are adjacent (i.e. share an edge) exactly when their Euclidean distance is 1.

2.1 Definition of the abstract Tile-Assembly Model

In this section, we define the abstract Tile-Assembly Model in 2 and 3 dimensions. We will use the abbreviation *aTAM* to refer to the 2D model and *3DaTAM* for the 3D model. These definitions are borrowed from [7] and we note that [19] is a good introduction to the model for unfamiliar readers.

Fix $d \in \{2, 3\}$ to be the number of dimensions and Σ to be some alphabet with Σ^* its finite strings. A *glue* $g \in \Sigma^* \times \mathbb{N}$ consists of a finite string *label* and non-negative integer *strength*. A *tile type* is a tuple $t \in (\Sigma^* \times \mathbb{N})^{2d}$, thought of as a unit square or cube with a glue on each side. A *tile set* is a finite set of tile types. We always assume a finite set of tile types, but allow an infinite number of copies of each tile type to occupy locations in the $\mathbb{Z}^d$ lattice, each called a *tile*.

Given a tile set T, a *configuration* is an arrangement (possibly empty) of tiles in the lattice $\mathbb{Z}^d$, i.e. a partial function $\alpha : \mathbb{Z}^d \dashrightarrow T$. Two adjacent tiles in a configuration *interact*, or are *bound* or *attached*, if the glues on their abutting sides are equal (in both label and strength) and have positive strength. Each configuration α induces a *binding graph* B_α whose vertices are those points occupied by tiles, with an edge of weight s between two vertices if the corresponding tiles interact with strength s. An *assembly* is a configuration whose domain (as a graph) is connected and non-empty. The *shape* $S_\alpha \subseteq \mathbb{Z}^d$ of assembly α is the domain of α. For some $\tau \in \mathbb{Z}^+$, an assembly α is *τ-stable* if every cut of B_α has weight at least τ, i.e. a τ-stable assembly cannot be split into two pieces without separating bound tiles whose shared glues have cumulative strength τ. Given two assemblies α, β, we say α is a *subassembly* of β (denoted $\alpha \sqsubseteq \beta$) if $S_\alpha \subseteq S_\beta$ and for all $p \in S_\alpha$, $\alpha(p) = \beta(p)$.

A *tile-assembly system* (TAS) is a triple $\mathcal{T} = (T, \sigma, \tau)$, where T is a tile set, σ is a finite τ-stable assembly called the *seed assembly*, and $\tau \in \mathbb{Z}^+$ is called the *binding threshold*. Given a TAS $\mathcal{T} = (T, \sigma, \tau)$ and two τ-stable assemblies α and β, we say that *α $\mathcal{T}$-produces β in one step* (written $\alpha \to_1^{\mathcal{T}} \beta$) if $\alpha \sqsubseteq \beta$ and $|S_\beta \setminus S_\alpha| = 1$. That is, $\alpha \to_1^{\mathcal{T}} \beta$ if β differs from α by the addition of a single tile. The *$\mathcal{T}$-frontier* is the set $\partial^{\mathcal{T}} \alpha = \bigcup_{\alpha \to_1^{\mathcal{T}} \beta} S_\beta \setminus S_\alpha$ of locations in which a tile could τ-stably attach to α.

We use $\mathcal{A}^T$ to denote the set of all assemblies of tiles in tile set T. Given a TAS $\mathcal{T} = (T, \sigma, \tau)$, a sequence of $k \in \mathbb{Z}^+ \cup \{\infty\}$ assemblies $\alpha_0, \alpha_1, \ldots$ over $\mathcal{A}^T$ is called a *$\mathcal{T}$-assembly sequence* if, for all $1 \leq i < k$, $\alpha_{i-1} \to_1^{\mathcal{T}} \alpha_i$. The *result* of an assembly sequence is the unique limiting assembly of the sequence. For finite assembly sequences, this is the final assembly; whereas for infinite assembly sequences, this is the assembly consisting of all tiles from any assembly in the sequence. We say that *α $\mathcal{T}$-produces β* (denoted $\alpha \to^{\mathcal{T}} \beta$) if there is a $\mathcal{T}$-assembly sequence starting with α whose result is β. We say α is *$\mathcal{T}$-producible* if $\sigma \to^{\mathcal{T}} \alpha$ and write $\mathcal{A}[\mathcal{T}]$ to denote the set of $\mathcal{T}$-producible assemblies. We say α is *$\mathcal{T}$-terminal* if α is τ-stable and there exists no assembly which is $\mathcal{T}$-producible from α. We denote the set of $\mathcal{T}$-producible and $\mathcal{T}$-terminal assemblies by $\mathcal{A}_\square[\mathcal{T}]$.

When $\mathcal{T}$ is clear from context, we may omit $\mathcal{T}$ from the notation above.

Cooperative Attachment

Given a TAS $\mathcal{T} = (T, \sigma, \tau)$, for a tile to attach to an assembly it must match glues whose cumulative strength is at least τ in order to result in a τ-stable assembly. This can happen if, for instance, one of the matched glues has strength at least τ, in which case any other matching glues are superfluous. Alternatively, a tile may still attach without any τ-strength glues though this requires multiple glues to match whose strengths sum to at least τ. We refer to such attachments as *cooperative*.

2.2 Model Variations

In this paper we consider 3 variations of the aTAM. Other than the 3D aTAM, these include *directed* and *diffusion restricted* versions of the models. We say that a TAS $\mathcal{T}$ is *directed* if $|\mathcal{A}_\square[\mathcal{T}]| = 1$, i.e. $\mathcal{T}$ admits only a single producible terminal assembly. When we refer to a *directed model* we simply mean the set of all directed systems in a model. Directed systems are desirable for self-assembly since we often want our tiles to grow into a single target shape.

For diffusion restricted models, we note that in the aTAM it's possible for tiles to attach within a region of space which has been completely surrounded by other tiles. In 2D, we can imagine that the tiles are able to navigate around the assembly through the 3rd dimension, but in 3D such attachments are difficult to justify. Consequently, we also consider models where such attachments are forbidden. In 2D, this restriction could model a self-assembly process on the surface of a droplet of water where surface tension prevents the components from taking advantage of the 3rd dimension. We call the 2D diffusion restricted aTAM the *Planar aTAM* or PaTAM, and we call the 3D diffusion restricted aTAM the *Spatial aTAM* or SaTAM. In these models, and their directed subsets, we refer to regions which have been completely surrounded (in which no tile attachments are allowed to occur) *constrained*. To formally model this restriction, we first note that given a finite d-dimensional assembly α, the graph $\mathbb{Z}^d \setminus S_\alpha$ consists of a finite number of connected components, exactly one of which will be infinite in size. We say that this component graph is the *outside* of α while the finite-sized components are *constrained*. In a diffusion restricted system we only allow tile attachments on the outside of an assembly.

2.3 Intrinsic Simulation

First we provide a high-level definition of the notion of *intrinsic simulation* which should be sufficient for understanding our results. A full technical definition follows afterward. For brevity, in this paper, unless explicitly stated, "simulation" will refer to intrinsic simulation.

High-Level Description of Simulation

Simulation of system $\mathcal{T}$ by system $\mathcal{S}$ occurs at a scale factor m, so that $m \times m$ (or $m \times m \times m$ in 3D) blocks of tiles from $\mathcal{S}$, which we refer to as *macrotiles*, correspond to individual tiles in $\mathcal{T}$. For a given simulation, we define a *macrotile representation function* R which describes this mapping of macrotiles to tiles. Additionally for convenience, using R we define an *assembly representation function* R^* which maps entire assemblies from $\mathcal{S}$ to assemblies in $\mathcal{T}$, essentially evaluating R on each macrotile location for a given assembly in $\mathcal{S}$. Note that we don't require all locations within a macrotile to contain a tile and macrotile blocks containing tiles can still be mapped to empty space under R. When a tile attachment causes the representation of a macrotile location to map to a tile for the first time, we say that the attachment has caused the macrotile to *resolve* and once a macrotile has resolved, any additional tile attachments within the macrotile cannot change its representation under R. While we do allow macrotile locations to map to empty space, for a simulation to be valid there must be restrictions on where tiles are allowed to attach in $\mathcal{S}$. For our notion of simulation to be useful as a metric of comparing the relative capabilities of models, we require that $\mathcal{S}$ only place tiles within the macrotile regions immediately adjacent (not diagonally) to those which have already resolved, and we call such locations *fuzz*. This allows tiles in $\mathcal{S}$ to attach only in macrotiles which could potentially resolve during a valid simulation, since only the locations in $\mathcal{T}$ mapped to by the fuzz locations could possibly receive tiles in $\mathcal{T}$. If a class of systems C can all be simulated by another class of systems C' sharing a single tile set (though each may have a different seed assembly), we say that class C' *intrinsically simulates* C with a *universal tile set*. We can also say that C' is *intrinsically universal* (IU) for C.

Formal Definition of Simulation

Now we provide formal definitions for *intrinsic simulation*. The definitions here are taken from [7] and specifically refer to 3D systems. Similar definitions for 2D intrinsic simulation are given in [9]. For simulation of a 2D system by a 3D system, we use the 3D definitions and assume that all systems in the 2D system are defined in 3D so that assemblies occupy only the $z = 0$ plane.

From this point on, let T be a tile set and let the scale factor be $m \in \mathbb{Z}^+$. An *m-block macrotile* over T is a partial function $\alpha : \mathbb{Z}^3_m \dashrightarrow T$, where $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$. Let B^T_m be the set of all m-block macrotiles over T. The m-block with no domain is said to be *empty*. For a general assembly $\alpha : \mathbb{Z}^3 \dashrightarrow T$ and $(x', y', z') \in \mathbb{Z}^3$, define $\alpha^m_{(x',y',z')}$ to be the m-block macrotile defined by $\alpha^m_{(x',y',z')}(i_x, i_y, i_z) = \alpha(mx' + i_x, my' + i_y, mz' + i_z)$ for $0 \leq i_x, i_y, i_z < m$. For some tile set S, a partial function $R : B^S_m \dashrightarrow T$ is said to be a *valid m-block macrotile representation* from S to T if for any $\alpha, \beta \in B^S_m$ such that $\alpha \sqsubseteq \beta$ and $\alpha \in \text{dom } R$, then $R(\alpha) = R(\beta)$.

For a given valid m-block macrotile representation function R from tile set S to tile set T, define the *assembly representation function*[2] $R^* : \mathcal{A}^S \to \mathcal{A}^T$ such that $R^*(\alpha') = \alpha$ if and only if $\alpha(x, y, z) = R\left(\alpha'^m_{(x,y,z)}\right)$ for all $(x, y, z) \in \mathbb{Z}^3$. For an assembly $\alpha' \in \mathcal{A}^S$ such that $R^*(\alpha') = \alpha$, α' is said to map *cleanly* to $\alpha \in \mathcal{A}^T$ under R^* if for all non empty blocks $\alpha'^m_{(x,y,z)}$, $(x, y, z) + (u_x, u_y, u_z) \in \text{dom } (\alpha)$ for some $(u_x, u_y, u_z) \in U_3$ such that $u_x^2 + u_y^2 + u_z^2 \leq 1$, or if α' has at most one non-empty m-block $\alpha^m_{0,0}$. In other words, α' may have tiles on macrotile blocks representing empty space in α, but only if that position is adjacent to a tile in α. We call such growth "around the edges" of α' *fuzz* and thus restrict it to be adjacent to only valid macrotiles, but not diagonally adjacent (i.e. we do not permit *diagonal fuzz*).

In the following definitions, let $\mathcal{T} = (T, \sigma_T, \tau_T)$ be a TAS, let $\mathcal{S} = (S, \sigma_S, \tau_S)$ be a TAS, and let R be an m-block representation function $R : B^S_m \to T$.

▶ **Definition 1.** *We say that $\mathcal{S}$ and $\mathcal{T}$ have* equivalent productions *(under R), and we write $\mathcal{S} \Leftrightarrow \mathcal{T}$ if the following conditions hold:*

1. $\{R^*(\alpha') | \alpha' \in \mathcal{A}[\mathcal{S}]\} = \mathcal{A}[\mathcal{T}]$.
2. $\{R^*(\alpha') | \alpha' \in \mathcal{A}_\square[\mathcal{S}]\} = \mathcal{A}_\square[\mathcal{T}]$.
3. *For all $\alpha' \in \mathcal{A}[\mathcal{S}]$, α' maps cleanly to $R^*(\alpha')$.*

▶ **Definition 2.** *We say that $\mathcal{T}$* follows *$\mathcal{S}$ (under R), and we write $\mathcal{T} \dashv_R \mathcal{S}$ if $\alpha' \to^{\mathcal{S}} \beta'$, for some $\alpha', \beta' \in \mathcal{A}[\mathcal{S}]$, implies that $R^*(\alpha') \to^{\mathcal{T}} R^*(\beta')$.*

The next definition essentially specifies that every time $\mathcal{S}$ simulates an assembly $\alpha \in \mathcal{A}[\mathcal{T}]$, there must be at least one valid growth path in $\mathcal{S}$ for each of the possible next steps that $\mathcal{T}$ could make from α which results in an assembly in $\mathcal{S}$ that maps to that next step. While this definition is unfortunately dense, it accommodates subtle situations such as where $\mathcal{S}$ must "commit to" a subset of possible representations in $\mathcal{T}$ before being explicitly mapped, under R, to any one in particular.

▶ **Definition 3.** *We say that $\mathcal{S}$* models *$\mathcal{T}$ (under R), and we write $\mathcal{S} \models_R \mathcal{T}$, if for every $\alpha \in \mathcal{A}[\mathcal{T}]$, there exists $\Pi \subset \mathcal{A}[\mathcal{S}]$ where $\Pi \neq \varnothing$ and $R^*(\alpha') = \alpha$ for all $\alpha' \in \Pi$, such that, for every $\beta \in \mathcal{A}[\mathcal{T}]$ where $\alpha \to^{\mathcal{T}} \beta$, (1) for every $\alpha' \in \Pi$ there exists $\beta' \in \mathcal{A}[\mathcal{S}]$ where $R^*(\beta') = \beta$ and $\alpha' \to^{\mathcal{S}} \beta'$, and (2) for every $\alpha'' \in \mathcal{A}[\mathcal{S}]$ where $\alpha'' \to^{\mathcal{S}} \beta'$, $\beta' \in \mathcal{A}[\mathcal{S}]$, $R^*(\alpha'') = \alpha$, and $R^*(\beta') = \beta$, there exists $\alpha' \in \Pi$ such that $\alpha' \to^{\mathcal{S}} \alpha''$.*

[2] Note that R^* is a total function since every assembly of S represents *some* assembly of T; the functions R and α are partial to allow undefined points to represent empty space.

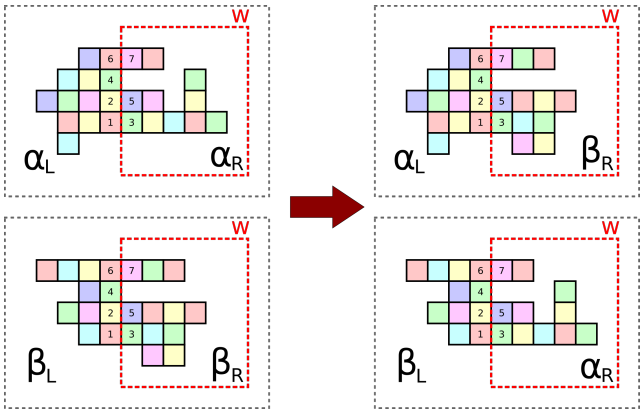


Figure 2 An illustration of the window movie lemma. On the left are two producible assemblies $\alpha = \alpha_L \cup \alpha_R$ and $\beta = \beta_L \cup \beta_R$ made from the same tile set, which are each divided into two subassemblies by the window w. For both assemblies, the window w has the same window movie, i.e. the order in which tiles present glues along the window, depicted by numbers on the tiles describing the relative order in which they attached. Since all growth within the windowed regions depends only on the glues presented along the window, we can splice these assemblies to get $\alpha_L \cup \beta_R$ or $\beta_L \cup \alpha_R$ (illustrated on the right). The window movie lemma then guarantees that both of these assemblies are producible.

▶ **Definition 4.** *We say that* $\mathcal{S}$ intrinsically simulates $\mathcal{T}$ *(under* R*) if* $\mathcal{S} \Leftrightarrow_R \mathcal{T}$ *(equivalent productions),* $\mathcal{T} \dashv_R \mathcal{S}$ *and* $\mathcal{S} \models_R \mathcal{T}$ *(equivalent dynamics).*

2.4 Window Movie Lemma

In [15], the authors proved the Window Movie Lemma, a pumping lemma of sorts for the aTAM (and its variants) which has since seen much use as a powerful tool for proving that certain tile-assembly simulations are impossible. Since it appears in several of our proofs, we first informally describe the lemma, then explicitly state it. A *window* is an edge cut which partitions the lattice graph ($\mathbb{Z}^2$ in 2D or $\mathbb{Z}^3$ in 3D) into two regions. Given some window w and some assembly sequence $\vec{\alpha}$ in a TAS $\mathcal{T}$, a *window movie* M is defined to be the ordered sequence of glues presented along w by tiles in $\mathcal{T}$ during the assembly sequence $\vec{\alpha}$. Informally, if we think of the window w as a thin pane dividing two regions of tile locations and imagine stepping through the assembly sequence $\vec{\alpha}$ one tile attachment at a time, M is constructed by recording the glues which appear on the surface of the pane and their relative order. More formally, a *window movie* is the sequence $M_w^{\vec{\alpha}} = \{(v_i, g_i)\}$ of pairs of grid graph vertices v_i and glues g_i, given by order of appearance of the glues along window w during $\vec{\alpha}$. Furthermore, if k glues appear along w during the same assembly step in $\vec{\alpha}$, then these glues appear contiguously and are listed in lexicographical order of the unit vectors describing their orientation in $M_w^{\vec{\alpha}}$.

Informally, the Window Movie Lemma states that any tile attachments that occur within the region bounded by a window are possible in a region bounded by the same window (up to translation) with an identical window movie. This allows us to splice assembly sequences together and, consequently, pump a sequence of tile attachments so long as we can ensure the existence of identical window movies. Figure 3 illustrates how the Window Movie Lemma can be used to pump growth.

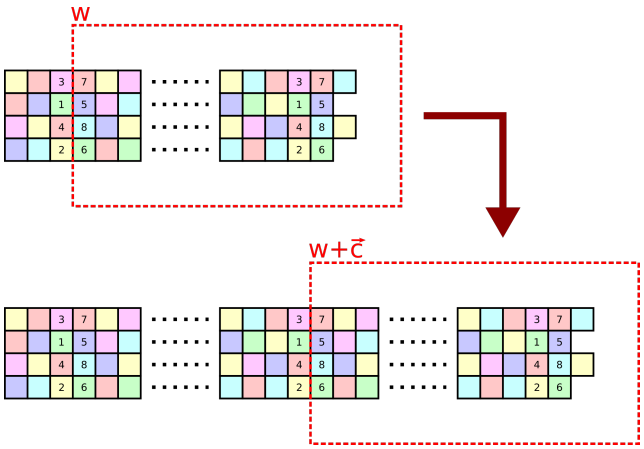


Figure 3 Using the Window Movie Lemma to "pump" assembly sequences. The top assembly depicts a ribbon of tiles growing horizontally to the right and numbers on tiles describe a relative order of attachment. If such a ribbon of tiles grows long enough, then by pigeonhole principle, eventually there must exist two identical vertical slices along its length. Because every tile attachment inside a window w depends only on the tiles and their relative order of attachment along the window, we can thus find an assembly sequence where growth repeats after the second identical vertical slice. This can be performed indefinitely to "pump" the ribbon.

Window Movie Lemma

Let $\vec{\alpha} = \{\alpha_i\}$ and $\vec{\beta} = \{\beta_i\}$ be assembly sequences in TAS $\mathcal{T}$ and let α, β be the result assemblies of each respectively. Let w be a window that partitions α into two configurations α_L and α_R and let $w' = w + \vec{c}$ be a translation of w that partitions β into two configurations β_L and β_R (with α_L and β_L being the configurations containing their respective seed tiles). Furthermore define $M_w^{\vec{\alpha}}$ and $M_w^{\vec{\beta}}$ to be the window movies for $\vec{\alpha}, w$ and $\vec{\beta}, w'$ respectively. Then if $M_w^{\vec{\alpha}} = M_w^{\vec{\beta}}$, the assemblies $\alpha_L \cup \beta'_R$ and $\beta'_L \cup \alpha_R$ (where $\beta'_L = \beta_L - \vec{c}$ and $\beta'_R = \beta_R - \vec{c}$) are also producible.

3 Results

In this section we sketch our results. Detailed proofs can be found in the full version on arXiv [8]. We begin with some trivial observations which allow us to fill in several boxes from Table 1.

▶ **Observation 5.** *If there exists a directed system $\mathcal{T}$ in tile-assembly model M which cannot be simulated by any system in tile-assembly model M', then (1) there exists a system in M which cannot be simulated by any system in M', (2) there exists a system in M which cannot be simulated by any directed system in M', and (3) there exists a directed system in M which cannot be simulated by any directed system in M'.*

▶ **Observation 6.** *There exists systems, both directed and undirected, in the 3D models (3DaTAM and SaTAM) which cannot be simulated by any systems in any of the 2D models (aTAM and PaTAM, both directed and undirected).*

Observation 5 holds because the set of directed systems in a model is a subset of all systems in that model. Consequently, $\mathcal{T}$ is a system in both M and in the directed subset of M. By assumption, $\mathcal{T}$ cannot be simulated by any system in M' and therefore cannot

be simulated by any subset of systems of M', particularly the subset of directed systems. Regarding Observation 6, while we restrict the notion of simulation to use square macrotiles, simulations of systems on triangular lattices have been implemented using roughly hexagonal macrotiles made from square tiles [1], so one might imagine the possibility that by loosening our definition of simulation to use more interesting macrotiles, it could be possible to capture the geometry of 3D square tiles using 2D tiles. In our case however, we note that there can exist no planar embedding of the lattice graph of $\mathbb{Z}^3$ as a consequence of Kuratowski's theorem. Consequently, there can be no way to divide $\mathbb{Z}^2$ into connected regions of macrotile locations which preserves the adjacency of points in $\mathbb{Z}^3$ and therefore simulation could not be possible even if we generalized our notion of macrotiles. This is true for any 3D systems which have producible assemblies whose domains, as graphs, are non-planar as is trivially possible in all 3D models considered.

3.1 Simulations using existing tile sets

In [7], it was shown that there exists IU tile sets for the 3DaTAM, SaTAM, and both models' subsets of directed systems. While the main focus of that result was intrinsic simulation within a model, those IU tile sets can be used to trivially fill in a few boxes of Table 1. First we note that any aTAM system can also be thought of as a 3DaTAM system (or even SaTAM system since tiles occupying only a single plane of 3D space can't constrain a 3D region) with glues only appearing on 4 of the 6 faces of any tile. Second, we note that the IU tile sets for the 3DaTAM and SaTAM differed only by the addition of a few tile types responsible for growing a wall around each face of a macrotile before resolving. This was necessary for intrinsic universality in the SaTAM since without them, the tiles making up a macrotile were sparse enough to necessarily allow a diffusion path for tiles to pass through a resolved macrotile. Consequently, if we don't include those tile types, then the IU tile set can simulate 3DaTAM systems even in the SaTAM since without walls surrounding each macrotile, the diffusion restriction does not interfere with the attachment of any tiles. Finally, by design, this tile set preserves directedness when simulating a directed system. Therefore, using the IU tile set and proofs from [7], the following observations hold.

▶ **Observation 7.** *There exists a universal tile set in both the 3DaTAM and SaTAM which intrinsically simulates all systems in the aTAM, preserving directedness.*

▶ **Observation 8.** *There exists a universal tile set in the SaTAM which intrinsically simulates all systems in the 3DaTAM, preserving directedness.*

3.2 Directed systems cannot simulate undirected systems

▶ **Theorem 9.** *There exist systems in the aTAM, 3DaTAM, PaTAM, and SaTAM, which cannot be simulated by any directed system in any of these models.*

Whereas directed systems only have one terminal assembly, undirected systems can have several. Figure 4 illustrates the tile set and terminal assemblies of a simple undirected system $\mathcal{T}$ which can be a system in the aTAM, 3DaTAM, PaTAM, or SaTAM without modification as it does not use any dynamics unique to any of those models. Because directed systems can only have a single terminal assembly, any directed system attempting to simulate $\mathcal{T}$ would necessarily fail since any assembly representation function R^* could not map one terminal assembly to both terminal assemblies of $\mathcal{T}$.

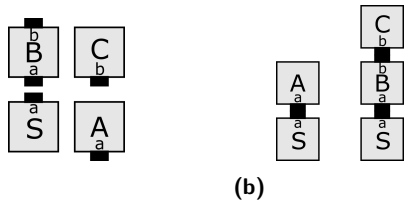


Figure 4 (a) Tile set of an undirected system for the proof of Theorem 9 and (b) Its two terminal assemblies.

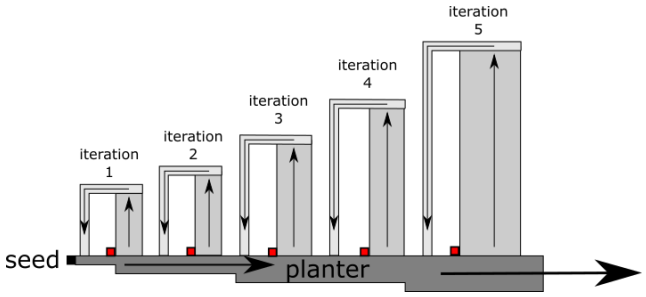


Figure 5 System $\mathcal{T}$ of the proof of Theorem 10. An infinite planter grows to the east from the seed and initiates upward growth of an infinite series of counters, each taller than the last, which initiate single-tile-wide paths that grow to the left then crash downward into the planter. To the left of each counter, at its base, it is possible for a red tile to attach.

3.3 The PaTAM cannot simulate the aTAM

Here we show that there are aTAM systems which cannot be correctly simulated by any PaTAM systems. To show this, we take advantage of the fact that aTAM systems are capable of growth inside of constrained regions while PaTAM systems are not. Specifically, we show that the PaTAM can't simulate the directed aTAM and, by Observation 5, note that this also implies that the PaTAM can't simulate the aTAM.

▶ **Theorem 10.** *There exists a system $\mathcal{T}$ which is a directed aTAM system, and therefore also an aTAM system, which cannot be simulated by any PaTAM system.*

Figure 5 is a schematic diagram of the terminal assembly of $\mathcal{T}$, a directed aTAM system which we claim is impossible to simulate in the PaTAM. Note that $\mathcal{T}$ is more complex than a system in which tiles attach to constrain a region which could have another tile attach within. This is because the definition of intrinsic simulation allows for macrotiles to resolve even when they aren't completely filled with tiles. Consequently, while macrotiles may map to tiles constraining a region, the tiles making up the macrotiles may not constrain a region. Our construction is designed to ensure that at some point, any supposed simulating system must constrain a region before the tiles inside are able to attach. In our directed aTAM system $\mathcal{T}$, this is done by first initiating the growth of a *planter*, a gadget that counts up in binary as it grows eastward, initiating the growth of increasingly tall arms at defined intervals. These arms are essentially binary counter gadgets which each grow upward to a distance, encoded in the glues of the tiles provided by the planter, and initiate the growth of thin arms when they finish. The thin arms are just a single tile wide and begin by growing a fixed distance to the west before growing south to crash into the planter below. By this process, each arm initiated by the planter constrains increasingly large regions of space which each contain a single location between the planter and arms, in which a single tile

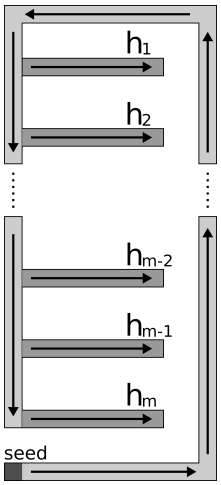


Figure 6 A schematic of system $\mathcal{P}$ for the proof of Theorem 11. Tiles grow in a rectangular shape, periodically spawning arms which can crash into the walls and constrain a region. It is undirected and its size depends non-deterministically on the number of tiles that attach between each corner.

can cooperatively attach (denoted by the red squares in Figure 5). Each of the tiles making up the southward growing portion of the thin arms are of the same tile type, each with identical glues on their north and south faces. While it is possible for different macrotiles to map to the same tile in $\mathcal{T}$, there are only so many combinations of tiles that make up a macrotile. Consequently, regardless of scale factor, if we look far enough down the planter, there will be an arm which grows tall enough that the simulating set must repeat a macrotile representation in two places along the same thin arm. We can then use the Window Movie Lemma to show that this arm "pumps" in our supposed simulating system, before crashing into the planter. It is therefore impossible for any simulating PaTAM system to prevent a region from becoming constrained before the macrotile inside is able to resolve, yielding terminal assemblies which aren't correctly mapped to a terminal assembly in $\mathcal{T}$.

3.4 The aTAM cannot simulate the PaTAM

Given that the PaTAM is just the aTAM with an added restriction on tile attachment, it's not terribly surprising that the PaTAM can't simulate the full dynamics of the aTAM; however, less obvious is the fact that the planarity restriction *also* gives the PaTAM some capabilities not possible in the aTAM, namely the ability to constrain a region and stop growth within. We utilize this ability in our proof of Theorem 11 which is sketched here. Also, by Observation 5, this also holds for the directed aTAM.

▶ **Theorem 11.** *There exists a PaTAM system $\mathcal{P}$ which cannot be simulated by any aTAM system.*

As with the proof for Theorem 10, in the definition of intrinsic simulation, we consider all possible representation functions and scale factors to prove impossibility. Figure 6 is a schematic diagram of PaTAM system $\mathcal{P}$ which is impossible to correctly simulate in the aTAM. Growth of $\mathcal{P}$ begins with tiles attaching in a row growing east. The length of this row is non-deterministic as at any point along the row, it's possible for a corner tile to attach, initiating growth to the north. Consequently, $\mathcal{P}$ is an undirected system so any potential simulating system must be able to simulate all possible assemblies of $\mathcal{P}$. Similarly,

northward and eastward growing rows of tiles attach with some length depending on how many tiles attached before each corner. Finally, a column of tiles begins growing south and, as it does, initiates the growth of several arms eastward, each spaced 4 tiles apart. Both the southward growing column of tiles and the arms continue growth until they are constrained or crash into another part of the assembly. To show that $\mathcal{P}$ cannot be simulated in the aTAM, we assume the existence of a simulating aTAM system $\mathcal{T}$ and prove that it must admit some assembly sequences which don't correspond to those in $\mathcal{P}$. To do this, we consider an assembly sequence in $\mathcal{P}$ where the rectangle of tiles grows to a size, based on the scale factor of the simulation, so that a sufficiently large number of sufficiently long arms are spawned by the south growing column of tiles. We also choose an assembly sequence where the south growing column will eventually collide with the seed tile, constraining the region containing the arms. Because we've chosen the assembly to be sufficiently large, each arm is capable of being "pumped" as per the window movie lemma. We then grow the bottom arm until just after it has collided with the east wall and note that, while $\mathcal{T}$ is an aTAM system and can still grow tiles inside of the constrained region, tiles on the inside and outside will no longer be able to affect each other's growth. There are a few cases to be considered, depending on whether or not the representation function has resolved the last tile of the bottom arm, but essentially we then show that we can continue the growth of the west wall until its macrotiles have resolved to tiles in $\mathcal{P}$ that constrain the rectangle's interior. By a counting argument and our choice of the number of arms, we can then show that one of the other arms must be able to continue growth within the constrained region, and that the assembly sequence in $\mathcal{T}$ maps to one invalid in $\mathcal{P}$.

3.5 The 3DaTAM cannot simulate the SaTAM

The proof of Theorem 12 is similar in principle to the proof of Theorem 11, albeit with a slightly different system which takes advantage of the differences between 2D and 3D. We sketch the proof here.

▶ **Theorem 12.** *There exists an SaTAM system $\mathcal{S}$ which cannot be simulated by any 3DaTAM system.*

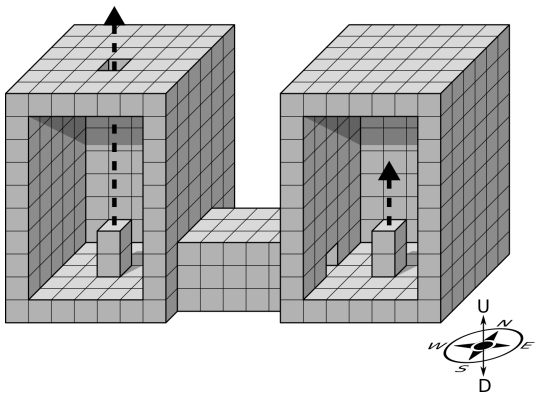


Figure 7 Cut-away view of system $\mathcal{S}$ from the proof of Theorem 12. Two chambers are connected by a thin tunnel. Pillars growing inside the outer west chamber will eventually constrain the region within the chambers, at which point, the pillar growing in the inner east chamber will no longer be able to continue growth.

The system $\mathcal{S}$ for this result, as illustrated in Figure 7, initially grows 2 nearly sealed chambers connected by a thin tunnel which allows for a diffusion path between them. These chambers both have a fixed base size of 9×9, but they can grow to have an arbitrary height in a way similar to the frame of the system used in the proof of Theorem 11. Once fully grown, the ceiling of one chamber contains a single tile wide opening which is the only way for tiles to diffuse into the chambers from outside; we call the chamber with this hole the *outer chamber* and the other one the *inner chamber*. Additionally, from the bottoms of both chambers, pillars can grow upwards to an arbitrary height by the attachment of copies of tiles with identical tile types. The pillar in the inner chamber will eventually crash into the ceiling *or* until the pillar in the outer chamber grows tall enough to plug the opening in its ceiling and constrain the space inside. We show that $\mathcal{S}$ cannot be simulated by any 3DaTAM system by showing that, in any potential simulating system, under the right conditions, although unwanted, it must still possible for the inner chamber pillar to continue growth even after the outer chamber pillar has sealed the chambers. To do this, we note that during some supposed simulation, the only way for the pillar in the inner chamber to "know" that the chambers have been sealed, is for tiles to attach inside of the tunnel. Consequently, because the tunnel is thin with a cross-section made of a hollow 3×3 square, the chambers can only communicate with each other a finite amount of times during a simulation. Specifically, if the scale factor of the simulation is c, then the number of tiles that can be placed in any x-coordinate corresponding to the tunnel is bounded by $5c \times 5c$ which includes any potential tiles growing in the fuzz adjacent to the macrotiles of the tunnel. Therefore, by a simple counting argument, if we initially grew our chambers to have a sufficiently large height, then there must exist some assembly sequence where both pillars grow by any desired number of macrotiles (which we choose to be long enough to allow pumpable growth) and during which no tile is placed in the center of the tunnel. Using the Window Movie Lemma, we then construct an assembly sequence where the outer chamber pumps to constrain the chambers. Because during this assembly sequence, no tiles are placed in the center of the tunnel, there is nothing to stop the inner chamber pillar from also being pumped. Such an assembly sequence must be possible in any 3DaTAM system which supposedly simulates our system $\mathcal{S}$, and since this assembly sequence corresponds to one which is invalid in the SaTAM, such a simulation is impossible.

3.6 The PaTAM can simulate the directed PaTAM

▶ **Theorem 13.** *There exists a universal Planar aTAM tile set S that can simulate any directed PaTAM system.*

Despite the fact that both the PaTAM and directed PaTAM are not intrinsically universal for themselves[7], using tools from [7] and [2] we are able to construct a PaTAM tile set capable of simulating arbitrary directed PaTAM systems. Here we outline the process by which a PaTAM tileset S can simulate any given directed PaTAM system $\mathcal{T}$. The tileset S is universal, meaning that regardless of the directed PaTAM system $\mathcal{T}$, the same tileset will be used at a fixed binding threshold, with only the seed of the simulating system changing to accommodate $\mathcal{T}$.

Given a directed PaTAM system $\mathcal{T}$, we define a simulating system $\mathcal{S}$ using a fixed tile set at binding threshold 2. The seed of $\mathcal{S}$ consists of already-resolved macrotiles in the same configuration as the seed of $\mathcal{T}$. Each macrotile in $\mathcal{S}$ consists of a 9×9 grid of structures we call *component blocks* (CBs) which are each made of many smaller tile-based constructions and which each store an encoding of the system $\mathcal{T}$ along with a bit of extra data in the form

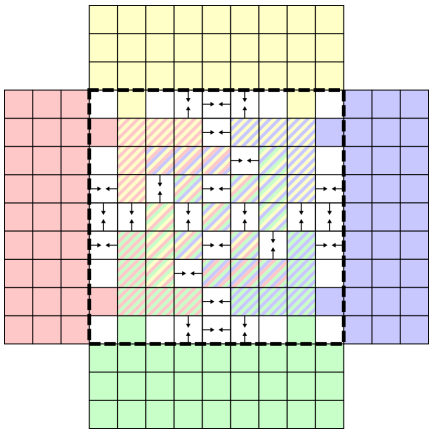


Figure 8 A schematic describing the 9×9 grid of potential component blocks which may appear in a macrotile location. Squares containing two arrows indicate a grid location which may contain a probe region. The surrounding macrotiles are illustrated using colored tiles to represent their relative direction from the current macrotile. Colors of CB locations indicate which surrounding macrotile the CB may have information about.

of specific glues on some of its tiles. The CBs of a macrotile each perform calculations using tiles which emulate Turing machines to determine how they should grow and whether or not the macrotile can resolve given the current information regarding the surrounding macrotiles.

Each CB essentially behaves like an individual tile on the 9×9 grid and we can think of CBs as growing in one of two ways. Either the CB grows using tile attachments from another adjacent CB in a way analogous to a τ-strength tile attachment, or a CB can grow in the gap between two adjacent CBs in certain locations of the grid designated as *probe regions*. This is analogous to a tile attachment that occurs by cooperative binding between two opposing tiles (which we refer to as *across-the-gap* cooperation). These "cooperative attachments" between CBs are used to consolidate information between the CBs. For instance, one CB might contain information encoded about the north adjacent macrotile and one might contain information about the west; in the probe region between them, a new CB can grow which will contain the information about both which it can then use to determine if a tile attachment in $\mathcal{T}$ would be possible in the tile location corresponding to the macrotile. Figure 8 illustrates the layout of a macrotile into CB locations with these probe regions indicated by squares with two opposing arrows.

Probe regions are CB locations in which two adjacent CBs, on opposite sides, can present structures called *probes* which are long, thin structures that grow from the surrounding CBs towards the center of a CB location. Each probe that grows in a probe region, indicates some possible combination of information from surrounding macrotiles and grows in a unique position according to this information. The length of a probe is chosen to be just shy of the center of the CB location, so that when two probes align from opposing sides of the probe region, there will be exactly a single tile wide gap between them. This gap allows a tile to cooperatively attach and grow along the sides of the probes to recover the information from both. Otherwise, if no probes in a probe region align, there will be enough room for the components that make up a CB to squeeze in between the probes from one side of the probe region to another. Figure 9 illustrates two scenarios involving probe regions.

Probe regions were introduced in [2] to solve the problem illustrated in Figure 10. Naively when simulating a tile system, to check for macrotiles which may cooperate across-the-gap, tiles must grow to query both adjacent macrotiles and determine if the attachment is possible.

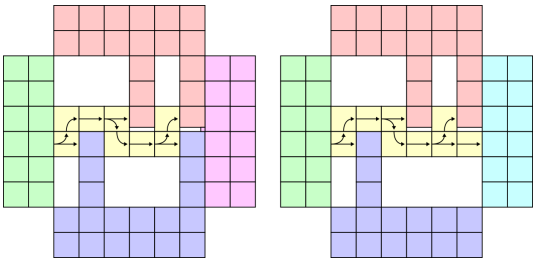


Figure 9 Probe regions between component blocks. The red and blue CBs grow probes to the center of the CB location in the middle while the green CB attempts to grow through the probe region. On the left, two probes happen to align, in which case a path of tiles containing information from the green CB cannot pass and the CB to the east results from the cooperative tile attachment between the probes. On the right, no probes align meaning the path of tiles from the green CB can squeeze between the probes to influence the growth of the CB to the east.

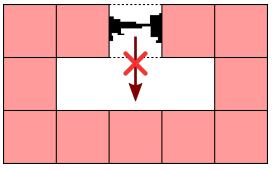


Figure 10 When checking for across-the-gap cooperation during a simulation, tiles can't naively span the entire gap without disconnecting two regions of space.

This however necessarily separates regions of space and in the case of planar systems also constrains one before it has been determined if the attachment can even occur. If it cannot, then tiles will no longer be able to attach in the constrained region and the simulation will likely end up being invalid. Probes avoid this problem by aligning exactly when across-the-gap cooperation is possible while still allowing tile structures to grow through if they don't align.

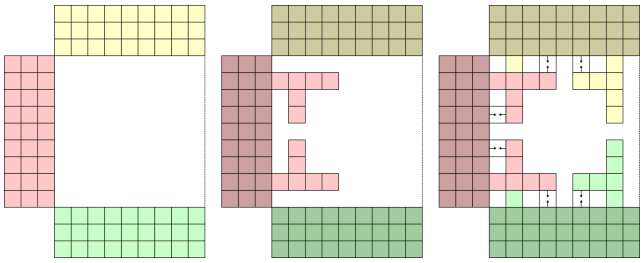


Figure 11 Hands made of component blocks growing from surrounding macrotiles.

Now that we have an idea of how the component blocks and probe regions behave we describe the protocol for resolving a macrotile by highlighting a few important cases. Growth within a macrotile begins when one or more of the surrounding macrotiles resolve and tiles begin to attach within the macrotile. From a surrounding macrotile, the protocol always begins by the growth of two "T"-shaped structures made from CBs called *hands* illustrated in Figure 11. Note that two adjacent surrounding macrotiles may both attempt to grow hands in the same location. This is handled by a single point of competition and the first surrounding macrotile for placing a tile in the closest corner of the shared hand locations is allowed to place theirs. Between the hands and the surrounding macrotiles probes are grown in the regions indicated on the right of Figure 11 which allows a CB to "attach"

cooperatively to combine information from both the hand and nearby macrotile. In some cases this information may be redundant, but with two or more surrounding macrotiles at least one location will always be able to combine information from two macrotiles.

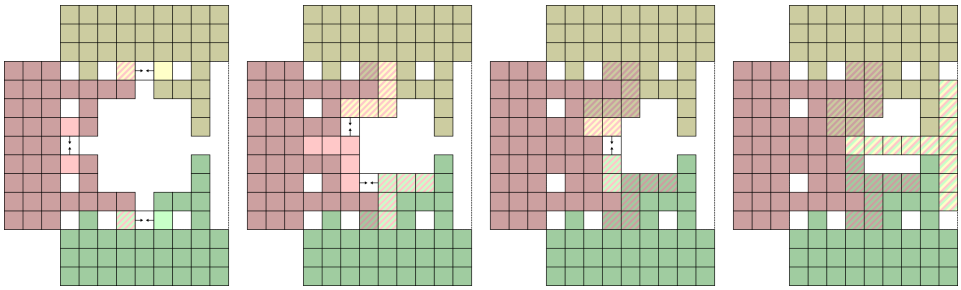


Figure 12 Once the hands have grown, CBs cooperate until information from all sides has been combined into a single CB. Then the macrotile can resolve.

The CBs resulting from cooperation between the hands and surrounding macrotiles then cooperate once again and CBs grow along the hands to form clockwise elbows with additional probe regions between them. CBs then cooperatively attach between these elbows and cooperate again near the center of the tile to eventually combine all of the information from the surrounding macrotiles. Once this occurs, the CB which "attaches" in the center of the tile contains the information from all sides. If the surrounding macrotiles represent tiles in $\mathcal{T}$ capable of placing a tile, additional CBs can grow to the remaining sides to present this information to the remaining sides and repeat the procedure in the adjacent macrotile locations.

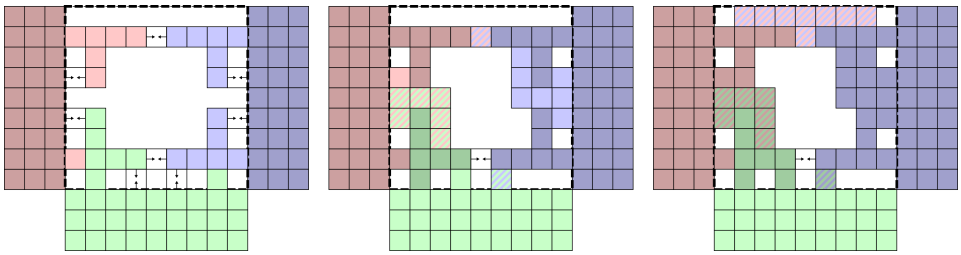


Figure 13 Probe regions between opposing macrotiles can check for across-the-gap cooperation.

In the case that an across-the-gap cooperation is possible in $\mathcal{T}$, the protocol deviates slightly. Illustrated in Figure 13, if across-the-gap cooperation is possible between the east and west macrotiles, their hands will share a probe region with aligned probes. Consequently, a CB can grow in that location and resolve the macrotile. This growth may constrain the region to the south, halting any tile attachments and CB growth in the south side of the macrotile, but this doesn't matter since the macrotile will only need to start the process in adjacent macrotiles that haven't yet resolved. The described protocol is robust to different orders of hand growth and different numbers of surrounding macrotiles, including those that don't end up contributing to macrotile resolution. If at any time a CB has sufficient information to determine how the macrotile should resolve, it begins growth to the center and then surrounding edges of the macrotile. This process will not be interrupted by other CBs since we are simulating a directed system where at most one unique tile can attach in each location.

References

1 John Calvin Alumbaugh, Joshua J Daymude, Erik D Demaine, Matthew J Patitz, and Andréa W Richa. Simulation of programmable matter systems using active tile-based self-assembly. In *International Conference on DNA Computing and Molecular Programming*, pages 140–158. Springer, 2019.

2 David Doty, Jack H. Lutz, Matthew J. Patitz, Robert T. Schweller, Scott M. Summers, and Damien Woods. The tile assembly model is intrinsically universal. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science*, FOCS 2012, pages 302–310, 2012.

3 Shawn M. Douglas, Hendrik Dietz, Tim Liedl, Björn Högberg, Franziska Graf, and William M. Shih. Self-assembly of DNA into nanoscale three-dimensional shapes. *Nature*, 459:414–418, May 2009. doi:10.1038/nature08016.

4 Constantine Glen Evans. *Crystals that count! Physical principles and experimental investigations of DNA tile self-assembly.* PhD thesis, California Institute of Technology, 2014.

5 Bin Fu, Matthew J. Patitz, Robert T. Schweller, and Robert Sheline. Self-assembly with geometric tiles. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming – 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, volume 7391 of *LNCS*, pages 714–725. Springer, 2012.

6 Hongzhou Gu, Jie Chao, Shou-Jun Xiao, and Nadrian C. Seeman. A proximity-based programmable dna nanoscale assembly line. *Nature*, 465(7295):202–205, May 2010. doi:10.1038/nature09026.

7 Daniel Hader, Aaron Koch, Matthew J. Patitz, and Michael Sharp. The impacts of dimensionality, diffusion, and directedness on intrinsic universality in the abstract tile assembly model. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2607–2624. SIAM, 2020.

8 Daniel Hader and Matthew J. Patitz. The impacts of dimensionality, diffusion, and directedness on intrinsic cross-model simulation in tile-based self-assembly, 2023. arXiv:2305.01877.

9 Jacob Hendricks, Matthew J. Patitz, and Trent A. Rogers. Universal simulation of directed systems in the abstract tile assembly model requires undirectedness. In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2016), New Brunswick, New Jersey, USA* October 9-11, 2016, pages 800–809, 2016.

10 Ming-Yang Kao and Robert T. Schweller. Randomized self-assembly for approximate shapes. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP (1)*, volume 5125 of *Lecture Notes in Computer Science*, pages 370–384. Springer, 2008. doi:10.1007/978-3-540-70575-8_31.

11 Yonggang Ke, Luvena L Ong, William M Shih, and Peng Yin. Three-dimensional structures self-assembled from DNA bricks. *Science*, 338(6111):1177–1183, 2012.

12 Wenyan Liu, Hong Zhong, Risheng Wang, and Nadrian C. Seeman. Crystalline two-dimensional dna-origami arrays. *Angewandte Chemie International Edition*, 50(1):264–267, 2011. doi:10.1002/anie.201005911.

13 Kyle Lund, Anthony J. Manzo, Nadine Dabby, Nicole Michelotti, Alexander Johnson-Buck, Jeanette Nangreave, Steven Taylor, Renjun Pei, Milan N. Stojanovic, Nils G. Walter, Erik Winfree, and Hao Yan. Molecular robots guided by prescriptive landscapes. *Nature*, 465(7295):206–210, May 2010. doi:10.1038/nature09012.

14 Kyle Lund, Anthony T. Manzo, Nadine Dabby, Nicole Micholotti, Alexander Johnson-Buck, Jeanetter Nangreave, Steven Taylor, Renjun Pei, Milan N. Stojanovic, Nils G. Walter, Erik Winfree, and Hao Yan. Molecular robots guided by prescriptive landscapes. *Nature*, 465:206–210, 2010.

15 Pierre-Étienne Meunier, Matthew J. Patitz, Scott M. Summers, Guillaume Theyssier, Andrew Winslow, and Damien Woods. Intrinsic universality in tile self-assembly requires cooperation. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA 2014), (Portland, OR, USA, January 5-7, 2014)*, pages 752–771, 2014.

16 Jennifer E. Padilla, Matthew J. Patitz, Raul Pena, Robert T. Schweller, Nadrian C. Seeman, Robert Sheline, Scott M. Summers, and Xingsi Zhong. Asynchronous signal passing for tile self-assembly: Fuel efficient computation and efficient assembly of shapes. In *UCNC*, volume 7956 of *Lecture Notes in Computer Science*, pages 174–185. Springer, 2013. doi:10.1007/978-3-642-39074-6_17.

17 Jennifer E. Padilla, Ruojie Sha, Martin Kristiansen, Junghuei Chen, Natasha Jonoska, and Nadrian C. Seeman. A signal-passing DNA-strand-exchange mechanism for active self-assembly of DNA nanostructures. *Angewandte Chemie International Edition*, 54(20):5939–5942, March 2015.

18 Paul W. K. Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440(7082):297–302, March 2006. doi:10.1038/nature04586.

19 Paul W. K. Rothemund and Erik Winfree. The program-size complexity of self-assembled squares (extended abstract). In *STOC '00: Proceedings of the thirty-second annual ACM Symposium on Theory of Computing*, pages 459–468, Portland, Oregon, United States, 2000. ACM.

20 Paul WK Rothemund, Nick Papadakis, and Erik Winfree. Algorithmic self-assembly of dna sierpinski triangles. *PLoS biology*, 2(12):e424, 2004.

21 David Soloveichik and Erik Winfree. Complexity of self-assembled shapes. *SIAM Journal on Computing*, 36(6):1544–1569, 2007. doi:10.1137/S0097539704446712.

22 Erik Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, California Institute of Technology, June 1998.

23 Damien Woods. Intrinsic universality and the computational power of self-assembly. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 373(2046), 2015. doi:10.1098/rsta.2014.0214.

24 Damien Woods, David Doty, Cameron Myhrvold, Joy Hui, Felix Zhou, Peng Yin, and Erik Winfree. Diverse and robust molecular algorithms using reprogrammable dna self-assembly. *Nature*, 567(7748):366–372, 2019.

25 Yin Zhang, Angus McMullen, Lea-Laetitia Pontani, Xiaojin He, Ruojie Sha, Nadrian C. Seeman, Jasna Brujic, and Paul M. Chaikin. Sequential self-assembly of dna functionalized droplets. *Nature Communications*, 8(1):21, 2017. doi:10.1038/s41467-017-00070-0.

Parameter Estimation for Gibbs Distributions

David G. Harris
Department of Computer Science, University of Maryland, College Park, MD, USA

Vladimir Kolmogorov
Institute of Science and Technology Austria, Klosterneuburg, Austria

Abstract

A central problem in computational statistics is to convert a procedure for *sampling* combinatorial objects into a procedure for *counting* those objects, and vice versa. We will consider sampling problems which come from *Gibbs distributions*, which are families of probability distributions over a discrete space Ω with probability mass function of the form $\mu_\beta^\Omega(\omega) \propto e^{\beta H(\omega)}$ for β in an interval $[\beta_{\min}, \beta_{\max}]$ and $H(\omega) \in \{0\} \cup [1, n]$.

The *partition function* is the normalization factor $Z(\beta) = \sum_{\omega \in \Omega} e^{\beta H(\omega)}$, and the *log partition ratio* is defined as $q = \frac{\log Z(\beta_{\max})}{Z(\beta_{\min})}$

We develop a number of algorithms to estimate the counts c_x using roughly $\tilde{O}(\frac{q}{\varepsilon^2})$ samples for general Gibbs distributions and $\tilde{O}(\frac{n^2}{\varepsilon^2})$ samples for integer-valued distributions (ignoring some second-order terms and parameters), We show this is optimal up to logarithmic factors. We illustrate with improved algorithms for counting connected subgraphs and perfect matchings in a graph.

2012 ACM Subject Classification Mathematics of computing → Probabilistic algorithms; Applied computing → Physics

Keywords and phrases Gibbs distribution, sampling

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.72

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2007.10824`

Acknowledgements We thank Heng Guo for helpful explanations of algorithms for sampling connected subgraphs and matchings, Maksym Serbyn for bringing to our attention the Wang-Landau algorithm and its use in physics.

1 Introduction

A central problem in computational statistics is to convert a procedure for *sampling* combinatorial objects into a procedure for *counting* those objects, and vice versa. We will consider sampling algorithms for Gibbs distributions. Formally, given a real-valued function $H(\cdot)$ over a finite set Ω, the *Gibbs distribution* is defined as a family of distributions μ_β^Ω over Ω, parameterized by β, of the form

$$\mu_\beta^\Omega(\omega) = \frac{e^{\beta H(\omega)}}{Z(\beta)}$$

These distributions occur in a number of sampling algorithms, as we describe shortly; they also frequently occur in physics, where the parameter $-\beta$ corresponds to the inverse temperature, the function $H(\omega)$ is called the *Hamiltonian* of the system, and the normalizing constant $Z(\beta) = \sum_{\omega \in \Omega} e^{\beta H(\omega)}$ is called the *partition function*.

Suppose we have access to an oracle which return a sample from μ_β^Ω for any chosen query value $\beta \in [\beta_{\min}, \beta_{\max}]$. We will seek to estimate the vector of counts (also known as the *(discrete) density of states (DOS)*), defined as

$$c_x = |H^{-1}(x)|, \qquad x \geq 0$$

In statistical physics, for instance, this essentially gives full information about the system and physically relevant quantities such as entropy, free energy, etc. Another parameter, whose role is less intuitive, is the partition ratio function

$$Q(\beta) = \frac{Z(\beta)}{Z(\beta_{\min})},$$

and in particular the value $Q(\beta_{\max}) = \frac{Z(\beta_{\max})}{Z(\beta_{\min})}$. This can interpreted as a measure of the "diversity" of the distribution as β varies.

As is common in this setting, we assume that (after rescaling if necessary) we are given known parameters n, q with

$$\log Q(\beta_{\max}) \leq q, \qquad H(\Omega) \subseteq \mathcal{F} \stackrel{\text{def}}{=} \{0\} \cup [1, n]$$

In some cases, the domain is integer-valued, i.e. $H(\Omega) \subseteq \mathcal{H} \stackrel{\text{def}}{=} \mathcal{F} \cap \mathbb{Z} = \{0, 1, \ldots, n\}$ for integer n. We call this the *general integer setting*. A special case of the integer setting, which we call the *log-concave setting*, is when the counts $c_0, c_1, c_2, \ldots, c_{n-1}, c_n$ are non-zero and satisfy $c_k/c_{k-1} \geq c_{k+1}/c_k$ for $k = 1, \ldots, n-1$. The general case, where $H(\omega)$ takes values in $\mathcal{F}$, is called the *continuous setting*.[1]

There is an associated probability distribution we call the *gross Gibbs distribution* $\mu_\beta(x)$ over $\mathcal{F}$ given by

$$\mu_\beta(x) = \frac{c_x e^{\beta x}}{Z(\beta)}, \qquad Z(\beta) = \sum_x c_x e^{\beta x}$$

We will only require oracle access to μ_β, for any chosen query value $\beta \in [\beta_{\min}, \beta_{\max}]$; this is provided automatically given access to μ_β^Ω. We let γ denote the target failure probability and ε the target accuracy of our algorithms, i.e. with probability at least $1-\gamma$, the algorithms should return estimates within a factor of $[e^{-\varepsilon}, e^{\varepsilon}]$ of the correct value. Throughout, "sample complexity" refers to the number of calls to the oracle; for brevity, we also define the *cost* of a sampling algorithm to be its *expected sample complexity*.

To avoid degenerate cases, we assume $n, q \geq 2$ and $\varepsilon, \gamma \in (0, \frac{1}{2})$ throughout. If upper bounds n and/or q are not available directly, they can often be estimated by simple algorithms (up to constant factors), or can be guessed by exponential back-off strategies.

1.1 Algorithmic sampling-to-counting

To make our problem setting more concrete, consider the following scenario: we have a combinatorial system, where the objects have a "weight", and we have an algorithm to sample objects from the corresponding Gibbs distribution for any parameter β. This may be an exact sampler, or it may be an approximate sampler such as a Markov chain whose stationary distribution is the Gibbs distribution. The runtime (e.g. the mixing time of the Markov chain) may depend on β. As a few prominent examples:

[1] The log-concave algorithms still work if some of the counts c_i are equal to zero; in this case, the non-zero counts must form a discrete interval $\{i_0, i_0+1, \ldots, i_1-1, i_1\}$ and the required bound must hold for $k = i_0+1, \ldots, i_1-1$.

1. Connected subgraphs of a given graph; the weight of a subgraph is its cardinality [10].
2. Matchings of a given graph; the weight of a matching, again, is its cardinality [15, 13].
3. Independent sets in bounded-degree graphs; the weight is the size of the independent set [7, 13].
4. Assignments to a given k-SAT instance; the weight is the number of unsatisfied clauses [8].
5. Vertex cuts for the ferromagnetic Ising model; the weight is the imbalance of the cut [5].

We may wish to know the number of objects of a given weight class, e.g. connected subgraphs of a given size. This can be viewed in terms of estimating the counts c_i. In a number of these applications, such as connected subgraphs and matchings, the count sequence is further known to be log-concave.

Our estimation algorithms can be combined with these prior sampling algorithms to yield improved algorithmic results, essentially for free. As some examples, we will show the following:

▶ **Theorem 1.** *Let $G = (V, E)$ be a connected graph, and for each $i = 0, \dots, |E| - |V| + 1$ let c_i be the number of connected subgraphs with $|E| - i$ edges. There is an fully-polynomial randomized approximation scheme (FPRAS) to estimate all values c_i in time complexity $\tilde{O}(|E|^3|V|/\varepsilon^2)$.*

▶ **Theorem 2.** *Let $G = (V, E)$ be a graph of maximum degree D and for each $i = 0, \dots, |V|$ let c_i be the number of independent sets of size i. For any constant $\xi > 0$ there is an FPRAS with runtime $\tilde{O}(|V|^2/\varepsilon^2)$ to simultaneously estimate all values $c_0, \dots, c_t$ for $t = (\alpha_c - \xi)|V|$, where α_c is the computational hardness threshold shown in [7].*

▶ **Theorem 3.** *Let $G = (V, E)$ be a graph with $|V| = 2v$ and for each $i = 0, \dots, v$ let c_i be the number of matchings in G with i edges. Suppose $c_v > 0$ and $c_{v-1}/c_v \le f$ for a known parameter f. There is an FPRAS for all c_i running in time $\tilde{O}(|E||V|^3 f/\varepsilon^2)$. In particular, if G has minimum degree at least $|V|/2$, the time complexity is $\tilde{O}(|V|^7/\varepsilon^2)$.*

Theorem 1 improves by a factor of $|E|$ over the algorithm in [11]. Similarly, Theorem 3 improves by a factor of $|V|$ compared to the FPRAS for counting matchings in [15]. Theorem 2 matches the runtime of an FPRAS for a *single* value i_k given in [13].

There are two minor technical issues we should clarify here. First, to obtain a randomized estimation algorithm, we must also bound the computational complexity of our procedures in addition to the number of oracle calls. In all the algorithms we develop, the computational complexity is a small logarithmic factor times the query complexity. The computational complexity of the oracle is typically much larger than this overhead. Thus, our sampling procedures translate directly into efficient randomized algorithms, whose runtime is the expected sample complexity multiplied by the oracle's computational complexity. We will not comment on computational issues henceforth.

Second, we may only have access to some approximate oracle $\tilde{\mu}_\beta$ that is close to μ_β in terms of total variation distance (e.g. by running an MCMC sampler). By a standard coupling argument (see e.g. [19, Remark 5.9]), our results remain valid if exact oracles are replaced with sufficiently close approximate oracles.

1.2 Our contributions

Before we can formally describe our algorithm for count estimation, we need to clear up two technical issues. The first is that counts can only be recovered up to scaling, so some (arbitrary) normalization must be chosen. For sake of consistency with other algorithms, we use the parameter $\pi(x)$ defined as:

$$\pi(x) \stackrel{\text{def}}{=} \mu_{\beta_{\min}}(x) = \frac{c_x e^{\beta_{\min} x}}{Z(\beta_{\min})}$$

The second, much trickier, issue is that if a count c_x is relatively small, then it is inherently hard to estimate accurately. To explain this, suppose that $\max_{\beta\in[\beta_{\min},\beta_{\max}]} \mu_\beta(x) = \mu_*$. In this case, $\Omega(1/\mu_*)$ samples are clearly needed to distinguish between $c_x = 0$ and $c_x > 0$; with fewer samples, we will never draw x from the oracle. Moreover, $\Omega(\frac{1}{\mu_* \varepsilon^2})$ samples are needed to estimate c_x to relative error ε. Since we can vary β, the complexity of estimating c_x must depend on the *best case* $\mu_\beta(x)$, over all allowed values of β. This gives rise to the parameter $\Delta(x)$ defined as

$$\Delta(x) \stackrel{\text{def}}{=} \max_{\beta\in[\beta_{\min},\beta_{\max}]} \mu_\beta(x)$$

With these two provisos, let us define the problem $P^{\delta,\varepsilon}_{\texttt{count}}$ for parameters $\delta, \varepsilon \in (0,1)$ as follows. We seek to obtain a pair of vectors $(\hat{\pi}, u)$, to satisfy two properties:

- **(i)** for all $x \in \mathcal{F}$ with $c_x \neq 0$, there holds $|\hat{\pi}(x) - \pi(x)| \leq u(x) \leq \varepsilon\pi(x)(1 + \delta/\Delta(x))$.
- **(ii)** for all $x \in \mathcal{F}$ with $c_x = 0$, there holds $\hat{\pi}(x) = 0$, and $u(x)$ can be set to an arbitrary value.

In other words, $[\hat{\pi}(x) - u(x), \hat{\pi}(x) + u(x)]$ should be a confidence interval for $\pi(x)$. In particular, if $\Delta(x) \geq \delta$, then $P^{\delta,\varepsilon}_{\texttt{count}}$ provides a $(1 \pm O(\varepsilon))$ relative approximation to $\pi(x)$. When $\Delta(x) \ll \delta$, then it still provides meaningful approximation guarantees which are critical in some of our other algorithms.

We develop three main algorithmic results:

▶ **Theorem 4.** *$P^{\delta,\varepsilon}_{\texttt{count}}$ can be solved with the following complexities:*

- *In the continuous setting, with cost $O\Big(\frac{q\log n + \sqrt{q\log n}/\delta}{\varepsilon^2}\log\frac{q}{\delta\gamma}\Big)$.*
- *In the general integer setting, with cost $O\Big(\frac{n^2+n/\delta}{\varepsilon^2}\log^2\frac{nq}{\gamma}\Big)$.*
- *In the log-concave setting, with cost $O\Big(\frac{\min\{(q+n)\log n, n^2\}+1/\delta}{\varepsilon^2}\log\frac{nq}{\gamma}\Big)$.*

where recall that cost refers to the expected number of queries to the oracle.

Our full results are somewhat more precise, see Theorems 13, 20 and 21 for more details.

We also show lower bounds for $P^{\delta,\varepsilon}_{\texttt{count}}$; we summarize these results here as follows:

▶ **Theorem 5.** *Let $n \geq n_0, q \geq q_0, \varepsilon < \varepsilon_0, \delta < \delta_0, \gamma < 1/4$ for certain absolute constants $n_0, q_0, \varepsilon_0, \delta_0$. There are problem instances μ which satisfy the given bounds n and q such that:*

- **(a)** *$P^{\delta,\varepsilon}_{\texttt{count}}$ requires cost $\Omega(\frac{(q+\sqrt{q}/\delta)\log\frac{1}{\gamma}}{\varepsilon^2})$.*
- **(b)** *$P^{\delta,\varepsilon}_{\texttt{count}}$ requires cost $\Omega(\frac{\min\{q+\sqrt{q}/\delta, n^2+n/\delta\}\log\frac{1}{\gamma}}{\varepsilon^2})$, and μ is integer-valued.*
- **(c)** *$P^{\delta,\varepsilon}_{\texttt{count}}$ requires cost $\Omega(\frac{(1/\delta+\min\{q,n^2\})\log\frac{1}{\gamma}}{\varepsilon^2})$, and μ is log-concave.*

These first two results match Theorem 4 up to logarithmic factors in n and q. The result for the log-concave setting has an additive discrepancy $\tilde{O}(\frac{n}{\varepsilon^2})$ in the regime when $1/\delta + q = o(n)$. (Throughout, we use the notation $\tilde{O}(x) = x\,\mathrm{polylog}(x)$.) See Theorem 36 for a more precise and general statement of these bounds. We emphasize that these lower bounds only apply to estimation algorithms which make use of the Gibbs oracle in a black-box way.

Some count-estimation algorithms have been considered for specific problems, e.g. in [14] for counting matchings or in [7] for counting independent sets. These procedures depended on specific properties of the Gibbs distribution, e.g. log-concavity. In addition, the algorithm in [14] was roughly worse by a factor of n compared to Theorem 4. By swapping in our new algorithm for $P_{\texttt{count}}^{\delta,\varepsilon}$, we will immediately obtain simpler, and more efficient, algorithms for these problems.

The general problem $P_{\texttt{count}}$ has not been theoretically analyzed, to our knowledge. In practice, the *Wang-Landau (WL)* algorithm [20] is a popular heuristic to estimate counts in physical applications. This uses a completely different methodology from our algorithm, based on a random walk on $\mathcal{F}$ with a running count estimate $\hat{c}$. As discussed in [18], there are more than 1500 papers on the WL algorithm as well as variants such as the $1/t$-WL algorithm [3]. These algorithms are not well understood; some variants are guaranteed to converge asymptotically [9], but bounds on convergence rate or accuracy seem to be lacking. For a representative example, see for example [17], which describes a Gibbs distribution model of protein folding, and uses the WL algorithm to determine relevant properties.

Estimating partition ratio

As a key building block, we develop new subroutines to estimate partition ratios. Formally, let us define the problem $P_{\texttt{ratio}}^{\texttt{all}}$ to compute a data structure $\mathcal{D}$ with an associated *deterministic* function $\hat{Q}(\alpha|\mathcal{D})$ satisfying the property

$$|\log \hat{Q}(\alpha|\mathcal{D}) - \log Q(\alpha)| \leq \varepsilon \qquad \text{for all } \alpha \in (\beta_{\min}, \beta_{\max}]$$

We say in this case that $\mathcal{D}$ is ε-*close*. We emphasize that, although generating $\mathcal{D}$ will require sampling from the Gibbs distribution, using it will not. Our main result here will be the following:

▶ **Theorem 6.** *$P_{\texttt{ratio}}^{\texttt{all}}$ can be solved with the following complexities:*

- *In the continuous setting, with cost $O\big(\frac{q \log n}{\varepsilon^2} \log \frac{1}{\gamma}\big)$.*
- *In the general integer setting, with cost $O\big(\frac{n^2 \log n}{\varepsilon^2} \log \frac{1}{\gamma} + n \log q\big)$.*
- *In the log-concave integer setting, with cost $O\big(\frac{n^2}{\varepsilon^2} \log \frac{1}{\gamma} + n \log q\big)$.*

A number of algorithms have been developed for *pointwise* estimation of $Q(\beta_{\max})$, with steadily improving sample complexities [4, 19, 12]. We denote this problem by $P_{\texttt{ratio}}^{\texttt{point}}$. The best prior algorithm for $P_{\texttt{ratio}}^{\texttt{point}}$ in the continuous setting [16] had cost $O\big(\frac{q \log n}{\varepsilon^2} \log \frac{1}{\gamma}\big)$ (matching our algorithm for $P_{\texttt{ratio}}^{\texttt{all}}$). No specialized algorithms were known for the integer setting. We also show matching lower bounds:

▶ **Theorem 7.** *Let $n \geq n_0, q \geq q_0, \varepsilon < \varepsilon_0, \delta < \delta_0, \gamma < 1/4$ for certain absolute constants $n_0, q_0, \varepsilon_0, \delta_0$. There are problem instances μ which satisfy the given bounds n and q such that:*

(a) *$P_{\texttt{ratio}}^{\texttt{point}}$ requires cost $\Omega(\frac{q \log \frac{1}{\gamma}}{\varepsilon^2})$.*

(b) *$P_{\texttt{ratio}}^{\texttt{point}}$ requires cost $\Omega(\frac{\min\{q, n^2\} \log \frac{1}{\gamma}}{\varepsilon^2})$. and μ is log-concave*

Thus, Theorem 6 is optimal up to logarithmic factors; this essentially settles the complexity of $P_{\texttt{ratio}}$ as functions of n and q.

The first algorithm of Theorem 6, for the continuous setting, is similar to the pointwise algorithm in [16]; we defer it the to the full version of the paper.

1.3 Overview

We will develop two, quite distinct, types of algorithms: the first uses "cooling schedules" similar to [12, 16], and the second is based on a new type of "covering schedule" for the integer setting. In Section 6, we use these algorithms for approximate counting of matching and connected subgraphs. In Section 7, we show the lower bounds for the problems $P_{\texttt{ratio}}$ and $P_{\texttt{count}}$.

We remark that when $q \leq n^2$ in the integer setting, general continuous algorithms may be more efficient than the specialized integer algorithms for some tasks. These will be used for our algorithms to count independent sets and connected subgraphs, for instance.

Before the technical details, let us provide a high-level roadmap. For simplicity, we assume that tasks need to be solved with constant success probability.

The continuous setting

We can use a variant on an algorithm of [16] to solve $P^{\texttt{all}}_{\texttt{ratio}}$. Assuming this problem can be solved, let us examine the problem $P^{\delta,\varepsilon}_{\texttt{count}}$. As a starting point, consider the identity

$$\pi(x) = e^{-(\beta-\beta_{\min})x} \cdot \mu_\beta(x) \cdot Q(\beta) \qquad \text{for all } x \in \mathcal{F}, \beta \in [\beta_{\min}, \beta_{\max}]. \tag{1}$$

For any value β, we can estimate $Q(\beta)$ using our algorithm for $P^{\texttt{all}}_{\texttt{ratio}}$, and we can estimate $\mu_\beta(x)$ by drawing $\Theta(\frac{1}{\mu_\beta(x)\varepsilon^2})$ samples from μ_β. We then make use of the following important result: if $\mu_\beta([0,x])$ and $\mu_\beta([x,n])$ are both bounded below by constants, then $\mu_\beta(x) \geq \Omega(\Delta(x))$.

Therefore, we do the following: (i) use binary search to find value β with $\mu_\beta([0,x]) \approx \mu_\beta([x,n])$; and (ii) estimate $\mu_\beta(x)$ using $O(\frac{1}{\delta\varepsilon^2})$ samples; (iii) use Eq. (1) to determine $\pi(x)$. From standard concentration bounds, this satisfies the conditions of $P^{\delta,\varepsilon}_{\texttt{count}}$; for example, if $\Delta(x) \geq \delta$, then $\mu_\beta(x)$, and hence $\pi(x)$, is estimated within relative error ε.

To estimate all the counts, we find cut-points $y_1, \ldots, y_t$, where each interval $[y_i, y_{i+1}]$ has a corresponding value β_i with $\mu_{\beta_i}([y_i, n]) \geq \Omega(1)$ and $\mu_{\beta_i}([0, y_{i+1}]) \geq \Omega(1)$. Any $x \in [y_{i+1}, y_i]$ then has $\mu_{\beta_i}(x) \geq \Omega(\Delta(x))$, so we can use samples from μ_{β_i} to estimate c_x simultaneously for all $x \in [y_i, y_{i+1}]$. We show that only $t = O(\sqrt{q \log n})$ distinct intervals are needed, leading to a cost of $O(\frac{\sqrt{q \log n}}{\delta\varepsilon^2})$ plus the cost of solving $P^{\texttt{all}}_{\texttt{ratio}}$. The formal analysis appears in Section 3.

The integer setting

To solve $P^{\delta,\varepsilon}_{\texttt{count}}$, we develop a new data structure we call a *covering schedule*. This consists of a sequence $\beta_{\min} = \beta_0, \beta_1, \ldots, \beta_t = \beta_{\max}$ and corresponding values $k_1, \ldots, k_t$ so that $\mu_{\beta_i}(k_i)$ and $\mu_{\beta_i}(k_{i+1})$ are large for all i. (The definition is adjusted slightly for the endpoints $i = 0$ and $i = t$). Define $w_i = \min\{\mu_{\beta_i}(k_i), \mu_{\beta_i}(k_{i+1})\}$ ("weight" of i). If we take $\Omega(1/w_i)$ samples from μ_{β_i}, we can accurately estimate the quantities $\mu_{\beta_i}(k_i), \mu_{\beta_i}(k_{i+1})$, in turn allowing us to estimate

$$\frac{Q(\beta_i)}{Q(\beta_{i-1})} = e^{(\beta_i - \beta_{i-1})k_i} \frac{\mu_{\beta_{i-1}}(k_i)}{\mu_{\beta_i}(k_i)}$$

By telescoping products, this in turn allows us to estimate every value $Q(\beta_i)$.

Next, for each index $x \in \mathcal{H}$, we use binary search to find α with $\mu_\alpha([0,x]) \approx \mu_\alpha([x,n])$ and then estimate $\mu_\alpha(x)$ by taking $O(\frac{1}{\delta\varepsilon^2})$ samples. If α lies in interval $[\beta_i, \beta_{i+1}]$ of the covering schedule, we can use the estimates for $Q(\beta_i)$ and $Q(\beta_{i+1})$ to estimate $Q(\alpha)$ and hence $\pi(x)$. Since we do this for each $x \in \mathcal{H}$, the overall cost of this second phase is roughly $O(\frac{n}{\delta\varepsilon^2})$.

There is a more efficient algorithm for $P^{\delta,\varepsilon}_{\texttt{count}}$ for log-concave counts. In this case, for a fixed β and $x \in [\sigma^-, \sigma^+]$ we have $\mu_\beta(x) \geq \min\{\mu_\beta(\sigma^-), \mu_\beta(\sigma^+)\}$. Thus, a single value β_i in a covering schedule "covers" the interval $[k_i, k_{i+1}]$. We can solve $P^{\delta,\varepsilon}_{\texttt{count}}$ with $O(\frac{1}{\delta\varepsilon^2} + \sum_i \frac{1}{w_i\varepsilon^2})$ samples, by drawing $\Theta(\frac{1}{w_i\varepsilon^2})$ samples at β_i and $\Theta(\frac{1}{\delta\varepsilon^2})$ samples at $\beta_{\min}$ and $\beta_{\max}$.

After solving $P^{\delta,\varepsilon}_{\texttt{count}}$, we can then solve $P^{\texttt{all}}_{\texttt{ratio}}$ essentially for free, by estimating $\hat{Q}(\alpha \mid \mathcal{D}) = \sum_i e^{(\alpha-\beta_{\min})i}\hat{\pi}(i)$. So $P^{\texttt{all}}_{\texttt{ratio}}$ in the integer setting reduces to a special case of $P_{\texttt{count}}$. (Interestingly, the continuous-case algorithm works very differently – there, $P^{\texttt{all}}_{\texttt{ratio}}$ is a subroutine used to solve $P_{\texttt{count}}$.)

Obtaining a covering schedule

The general $P_{\texttt{count}}$ algorithm described above uses $O(\sum_i \frac{n}{w_i\varepsilon^2})$ samples to estimate the values $Q(\beta_i)$, and similarly the log-concave algorithm uses $O(\frac{1}{\delta\varepsilon^2} + \sum_i \frac{1}{w_i\varepsilon^2})$ samples. We thus refer to the quantity $\sum_i \frac{1}{w_i}$ as the *inverse weight* of the schedule. In the most technically involved part of the paper, we produce a covering schedule with inverse weight $O(n \log n)$ (or $O(n)$ in the log-concave setting). Here we just sketch some key ideas.

First, we construct a "preschedule" where each interval can choose two different indices σ_i^-, σ_i^+ instead of a single index k_i, with the indices interleaving as $\sigma_i^- \leq \sigma_{i+1}^- \leq \sigma_i^+ \leq \sigma_{i+1}^+$. The algorithm repeatedly fill gaps: if some half-integer $\ell + 1/2$ is not currently covered, then we can select a value β with $\mu_\beta([0,\ell]) \approx \mu_\beta([\ell+1, n])$. For this β, there is a value $\sigma^+ \in [\ell+1, n]$ with $\mu_\beta(\sigma^+) \cdot (\sigma^+ - \ell) \geq \Omega(\frac{1}{\log n})$, and similarly a value $\sigma^- \in [0, k]$ with $\mu_\beta(\sigma^-) \cdot (\ell - \sigma^- + 1) \geq \Omega(\frac{1}{\log n})$. The interval $[\sigma^-, \sigma^+]$ then fills the gap and also has weight $w \geq \Omega(\frac{1}{(\sigma^+ - \sigma^-)\log n})$.

At the end of the process, we throw away redundant intervals so each x is covered by at most two intervals, and "uncross" them into a schedule with $k_i \in \{\sigma^+_{i-1}, \sigma^-_i\}$. Since $\frac{1}{w_i} \leq O((\sigma_i^+ - \sigma_i^-)\log n)$ for each i, this gives an $O(n \log n)$ bound of the inverse weight of the schedule.

2 Preliminaries

Define $z(\beta) = \log Z(\beta)$ and $z(\beta_1, \beta_2) = \log \frac{Z(\beta_2)}{Z(\beta_1)} = \log \frac{Q(\beta_2)}{Q(\beta_1)}$; note that $z(\beta_{\min}, \beta_{\max}) \leq q$ by definition. We write $z'(\beta)$ for the derivative of function z.

Define the Chernoff separation functions $F_+(x,t) = \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^x$ and $F_-(x,t) = \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^x$, where $\delta = t/x$. These are well-known upper bounds on the probability that a binomial random variable with mean x is larger than $x+t$ or smaller than $x-t$, respectively. We also define $F(x,t) = F_+(x,t) + F_-(x,t)$.

For a random variable X, we write $\mathbb{V}(X)$ for the variance of X, and $\mathbb{S}[X] = \frac{\mathbb{E}[X^2]}{(\mathbb{E}[X])^2} - 1 = \frac{\mathbb{V}(X)}{(\mathbb{E}[X])^2}$ for the relative variance of X.

We write $\mu_\beta(x,y), \mu_\beta[x,y)$ instead of $\mu_\beta((x,y)), \mu_\beta([x,y))$, etc. for readability.

2.1 The `Balance` subroutine

Given a target χ, we sometimes need to find a value β with $\mu_\beta[0,\chi] \approx 1/2 \approx \mu_\beta[\chi, n]$. That is, χ is the "balancing point" in the distribution μ_β. Formally, for values $\beta_{\texttt{left}} \leq \beta_{\texttt{right}}$, let us denote by $\Lambda_\tau(\beta_{\texttt{left}}, \beta_{\texttt{right}}, \chi)$ the set of values $\beta \in [\beta_{\texttt{left}}, \beta_{\texttt{right}}]$ which satisfy the following two properties:

- Either $\beta = \beta_{\texttt{left}}$ or $\mu_\beta[0,\chi) \geq \tau$
- Either $\beta = \beta_{\texttt{right}}$ or $\mu_\beta[\chi, n] \geq \tau$

To find this, we use a subroutine `Balance`. To summarize briefly, since $\mu_\beta[0,\chi]$ is a monotonic function of β and can be estimated by sampling, the value β is found via a noisy binary search. Our main result is the following:

▶ **Theorem 8.** *Suppose that τ is an arbitrary constant and $\beta_{\min} \le \beta_{\texttt{left}} < \beta_{\texttt{right}} \le \beta_{\max}$. Then $\beta \leftarrow \texttt{Balance}(\beta_{\texttt{left}}, \beta_{\texttt{right}}, \chi, \gamma, \tau)$ has cost $O(\log \frac{nq}{\gamma})$. With probability at least $1-\gamma$, there holds $\beta \in \Lambda_\tau(\beta_{\texttt{left}}, \beta_{\texttt{right}}, \chi)$ (we say in this case that the call is* good*).*

The details appear in the full paper. The following observation explains the motivation for the definition.

▶ **Proposition 9.** *If $\beta \in \Lambda_\tau(\beta_{\min}, \beta_{\max}, x)$, then $\mu_\beta(x) \ge \tau\Delta(x)$.*

Proof. Consider $\alpha \in [\beta_{\min}, \beta_{\max}]$ with $\mu_\alpha(x) = \Delta(x)$. The result is clear if $\alpha = \beta$. Suppose that $\alpha < \beta$; the case $\alpha > \beta$ is completely analogous. So $\beta > \beta_{\min} = \beta_{\texttt{left}}$, and since $\beta \in \Lambda_\tau(\beta_{\min}, \beta_{\max}, x)$, this implies that $\mu_\beta[0,x] \ge \tau$. We then have:

$$\begin{aligned}\mu_\alpha(x) &= \frac{c_x e^{\alpha x}}{\sum_y c_y e^{\alpha y}} \le \frac{c_x}{\sum_{y\le x} c_y e^{\alpha(y-x)}} \le \frac{c_x}{\sum_{y\le x} c_y e^{\beta(y-x)}} \\ &= \frac{c_x e^{\beta x}}{\sum_{y\le x} c_y e^{\beta y}} = \frac{\mu_\beta(x)}{\mu_\beta[0,x]} \le \frac{\mu_\beta(x)}{\tau}.\end{aligned}$$

◀

2.2 Statistical sampling

We can obtain an unbiased estimator of the probability vector μ_β by computing empirical frequencies $\hat\mu_\beta$ from N independent samples from μ_β; we denote this process as $\hat\mu_\beta \leftarrow \texttt{Sample}(\beta; N)$. We record the following standard concentration bound, which we will use repeatedly:

▶ **Lemma 10.** *For $\varepsilon, \gamma \in (0, \frac12), p_\circ \in (0,1]$, suppose we draw random variable $\hat p \sim \frac1N Binom(N,p)$ where $N \ge \frac{3e^\varepsilon \log(4/\gamma)}{(1-e^{-\varepsilon})^2 p_\circ}$. Then, with probability at least $1-\gamma$, the following two bounds both hold:*

$$|\hat p - p| \le \varepsilon(p + p_\circ), \qquad \text{and} \tag{2}$$

$$\hat p \in \begin{cases} [e^{-\varepsilon}p, e^{\varepsilon}p] & \text{if } p \ge e^{-\varepsilon}p_\circ \\ [0, p_\circ) & \text{if } p < e^{-\varepsilon}p_\circ \end{cases} \tag{3}$$

In particular, if Eq. (3) holds and $\min\{p,\hat p\} \ge p_\circ$, then $|\log\hat p - \log p| \le \varepsilon$.

Proof. See full paper. ◀

Many of our algorithms are based on calling $\hat\mu_\beta \leftarrow \texttt{Sample}(\beta; N)$ and making decisions depending on the values $\hat\mu_\beta(I)$ for certain sets $I \subseteq \mathcal{F}$; they succeed when the estimates $\hat\mu_\beta(I)$ are close to $\mu_\beta(I)$. We say the execution of `Sample` *well-estimates* I if Eqs. (2),(3) hold for $p = \mu_\beta(I)$ and $\hat p = \hat\mu_\beta(I)$; otherwise it *mis-estimates* I. Likewise we say `Sample` well-estimates k if it well-estimates the singleton set $I = \{k\}$. Since this comes up so frequently, we write

$$\hat\mu_\beta \leftarrow \texttt{Sample}(\beta; \varepsilon, \gamma, p_\circ)$$

as shorthand for $\hat\mu_\beta \leftarrow \texttt{Sample}(\beta; \lceil \frac{3e^\varepsilon \log(4/\gamma)}{(1-e^{-\varepsilon})^2 p_\circ} \rceil)$. Note that this has cost $O(\frac{\log(1/\gamma)}{\varepsilon^2 p_\circ})$, and each set I is well-estimated with probability at least $1-\gamma$.

As we have touched upon, our algorithms for $P^{\delta,\varepsilon}_{\texttt{count}}$ estimate each value $\pi(x)$ by sampling $\hat\mu_\alpha(x)$ for a well-chosen value α. We use similar formulas to produce the estimates $\hat\pi(x), u(x)$ in all these cases. We record the following general result:

▶ **Lemma 11.** *Suppose that for $x \in \mathcal{F}$, we are given $\alpha \in [\beta_{\min}, \beta_{\max}]$ and non-negative parameters $\hat{Q}(\alpha), \hat{\mu}_\alpha(x), p_\circ$ (all of which may depend upon x), satisfying the following bounds:*
(A1) $|\log \hat{Q}(\alpha) - \log Q(\alpha)| \le 0.1\varepsilon$.
(A2) $p_\circ \le \mu_\alpha(x)(1 + \delta/\Delta(x))$
(A3) $|\hat{\mu}_\alpha(x) - \mu_\alpha(x)| \le 0.1\varepsilon(\mu_\alpha(x) + p_\circ)$.

Then the estimated values

$$\hat{\pi}(x) = \hat{Q}(\alpha)e^{(\beta_{\min}-\alpha)x}\hat{\mu}_\alpha(x), \qquad u(x) = 0.4\hat{Q}(\alpha)e^{(\beta_{\min}-\alpha)x}\varepsilon(\hat{\mu}_\alpha(x) + p_\circ)$$

satisfy the criteria for the problem $P^{\delta,\varepsilon}_{\texttt{count}}$.

Proof. See full paper. ◀

Since this formula comes up so often, we write

$\texttt{EstimatePi}(x, \alpha, p_\circ)$

as shorthand for setting $\hat{\pi}(x), u(x)$ according to the formula in Lemma 11. The values $\hat{Q}(\alpha)$ and $\hat{\mu}_\alpha(x)$ should be clear from the context.

In a number of places, we need to estimate certain telescoping products. Direct Monte Carlo sampling does not give strong tail bounds, so we use a standard method based on median amplification. See the full paper for a description and proof.

▶ **Theorem 12.** *Suppose we can sample non-negative random variables $X_1, \dots, X_N$. The subroutine $\texttt{EstimateProducts}(X, \tau, \varepsilon, \gamma)$ takes input $\varepsilon, \gamma \in (0, 1)$ and $\tau > 0$, and returns a vector of estimates $(\hat{X}^{\texttt{prod}}_1, \dots, \hat{X}^{\texttt{prod}}_N)$. It uses $O(N(1 + \tau/\varepsilon^2)\log\frac{1}{\gamma})$ total samples of the X variables. If $\tau \ge \sum_{i=1}^N \mathbb{S}[X_i]$, then with probability at least $1 - \gamma$, it holds that $\frac{\hat{X}^{\texttt{prod}}_i}{\prod_{j=1}^i \mathbb{E}[X_j]} \in [e^{-\varepsilon}, e^{\varepsilon}]$ for all $i = 1, \dots, N$.*

In this case, it is also convenient to define $\hat{X}^{\texttt{prod}}_0 = 1 = \prod_{j=1}^0 \mathbb{E}[X_j]$.

3 Solving $P^{\delta,\varepsilon}_{\texttt{count}}$ in the continuous setting

In this section, we develop Algorithm 1 for $P^{\delta,\varepsilon}_{\texttt{count}}$. Here, we use a general algorithm to solve $P^{\texttt{all}}_{\texttt{ratio}}$ in the continuous setting with cost $O\big(\frac{q \log n}{\varepsilon^2} \log \frac{1}{\gamma}\big)$.

Algorithm 1 Solving $P^{\delta,\varepsilon}_{\texttt{count}}$ for error parameter γ.

call $\mathcal{D} \leftarrow \texttt{PratioAll}(\varepsilon/10, \gamma/4)$.
initialize $x_0 \leftarrow n, \alpha_0 \leftarrow \beta_{\max}$
for $t = 1$ **to** $T = 10\min\{q, \sqrt{q \log n}\}$ **do**
  set $\alpha_t \leftarrow \texttt{Balance}(\beta_{\min}, \alpha_{t-1}, x_{t-1}, \frac{\gamma}{100T}, 1/4)$
  set $\hat{\mu}_{\alpha_t} \leftarrow \texttt{Sample}(\alpha_t; \frac{10^8 \log \frac{50T}{\delta\gamma}}{\delta\varepsilon^2})$
  if $\alpha_t > \beta_{\min}$ **then**
    set x_t to be the minimum value with $\hat{\mu}_{\alpha_t}[0, x_t] \ge 1/100$
    foreach $y \in (x_t, x_{t-1}]$ **do** $\texttt{EstimatePi}(y, \alpha_t, \delta/200)$ with $\hat{Q}(\alpha_t) = \hat{Q}(\alpha_t \mid \mathcal{D})$
  else if $\alpha_t = \beta_{\min}$ **then**
    foreach $y \in [0, x_{t-1}]$ **do** $\texttt{EstimatePi}(y, \alpha_t, \delta/200)$ with $\hat{Q}(\alpha_t) = \hat{Q}(\alpha_t \mid \mathcal{D})$
    return

▶ **Theorem 13.** *Algorithm 1 solves* $P^{\delta,\varepsilon}_{\texttt{count}}$ *with cost*

$$O\Big(\frac{\min\{q, \sqrt{q \log n}\} \log \frac{q}{\delta\gamma}}{\delta\varepsilon^2} + \frac{q \log n \log \frac{1}{\gamma}}{\varepsilon^2}\Big).$$

The complexity bound follows immediately from specification of suborutines. We next analyze the success probability; this will require a number of intermediate calculations.

▶ **Proposition 14.** *With probability at least* $1 - \gamma/10$*, the following conditions hold for all iterations* t*:*

(i) $\alpha_t \in \Lambda_{1/4}(\beta_{\min}, \alpha_{t-1}, x_{t-1})$ *and* $x_t < x_{t-1}$ *and and* $\mu_{\alpha_t}[0, x_t) \le \frac{1}{70}$

(ii) $\alpha_t = \beta_{\min}$ *or* $\mu_{\alpha_t}[0, x_t] \ge \frac{1}{200}$.

Proof. See full paper. ◀

For the remainder of the analysis, we suppose that the bounds of Proposition 14 hold.

▶ **Proposition 15.** *For all iterations* t*, we have* $\alpha_{t+1} < \alpha_t$ *strictly and* $\mu_{\alpha_{t+1}}[x_t, n] \ge 1/4$*. Furthermore, if* $\alpha_{t+1} \ne \beta_{\min}$*, then* $z(\alpha_{t+1}, \alpha_t) \ge 2 + \frac{x_{t+1}}{x_{t-1} - x_{t+1}}$.

Proof. See full paper. ◀

▶ **Lemma 16.** *The loop at line 3 terminates before iteration* T*.*

Proof. Suppose not; by Proposition 14, we have $x_1 > x_2 > \cdots > x_{T-1} > \beta_{\min}$ strictly. Since each x_i comes from $\mathcal{F}$, we must have $x_1 \le n$ and $x_{T-2} \ge 1$. Let $g = T - 4$; note that due to bounds on n, q. we have $g \ge T/2 > 0$. For each $\ell = 1, \dots, g$, consider the non-negative value $a_\ell = \log(\frac{x_{\ell-1}}{x_{\ell+1}})$. We note the following bound:

$$\begin{aligned}\sum_{\ell=1}^{g} a_\ell &= \log\frac{x_1}{x_3} + \log\frac{x_2}{x_4} + \log\frac{x_3}{x_5} + \cdots + \log\frac{x_{g-2}}{x_g} + \log\frac{x_{g-1}}{x_{g+1}} \\ &= \log x_1 + \log x_2 - \log x_g - \log x_{g+1} \qquad \text{by telescoping sums} \\ &\le \log n + \log n - 0 - 0 = 2\log n\end{aligned}$$

By using Proposition 15 for each iteration $1, \dots, g$, we can compute:

$$q \ge z(\beta_{\max}, \beta_{\min}) \ge \sum_{i=1}^{g} z(\alpha_{i+1}, \alpha_i) \ge \sum_{i=1}^{g} 2 + \frac{x_{i+1}}{x_{i-1} - x_{i+1}} = 2g + \sum_{\ell=1}^{g} \frac{1}{e^{a_\ell} - 1}. \tag{4}$$

By Jensen's inequality applied to the concave function $y \mapsto \frac{1}{e^y - 1}$, we have

$$\sum_{\ell=1}^{g} \frac{1}{e^{a_\ell} - 1} \ge \frac{g}{\exp(\frac{1}{g}\sum_{\ell=1}^{g} a_\ell) - 1} \ge \frac{g}{\exp(\frac{2\log n}{g}) - 1}. \tag{5}$$

If $q > 2\log n$, then Eq. (4) shows $g \le q/2$. If $q \ge 2\log n$, then $\exp(\frac{2\log n}{g}) - 1 \le \frac{4e\log n}{g}$, and then Eq. (5) implies $q \ge \frac{g}{(4e\log n)/g} \ge g^2/20$, i.e. $g \le \sqrt{20q\log n}$. Either way, we have $g \ge \min\{\sqrt{20q\log n}, q/2\}$. Since $g \ge T/2$, this is a contradiction to the definition of T. ◀

▶ **Proposition 17.** *With probability at least* $1 - \gamma/10$*, the preconditions of Lemma 11 for* `EstimatePi` *(with* $p_\circ = \delta/200$*) hold for all* $y \in \mathcal{F}$*.*

Proof. See full paper. ◀

Overall, the total failure probability is at most $\gamma/10$ (from Proposition 14) plus $\gamma/10$ (from Proposition 17). This concludes the proof of Theorem 13. It also shows the first part of Theorem 4.

4 Solving $P_{\texttt{count}}$ and $P^{\texttt{all}}_{\texttt{ratio}}$ for integer-valued Gibbs distributions

The algorithms in the integer setting hinge on a data structure called the *covering schedule*. Formally, we define a covering schedule to be a sequence of the form

$$(\beta_0, w_0, k_1, \beta_1, w_1, k_2, \dots, \beta_{t-1}, w_{t-1}, k_t, \beta_t, w_t)$$

which satisfies the following additional constraints:

(i) $\beta_{\min} = \beta_0 < \dots < \beta_t = \beta_{\max}$;
(ii) $k_1 < k_2 < \dots < k_t$
(iii) $w_i \in [0,1]$ for $i = 0, \dots, t$.

Note that $t \le n+1$. We say that $\mathcal{I}$ is *proper* if for all $i = 1, \dots, t$ it satisfies

$$\mu_{\beta_{i-1}}(k_i) \ge w_{i-1} \text{ and } \mu_{\beta_i}(k_i) \ge w_i.$$

We define

$$\texttt{InvWeight}(\mathcal{I}) = \sum_{i=0}^{t} \frac{1}{w_i}.$$

Our algorithm to solve $P_{\texttt{count}}$ will have four stages. As a high-level summary, it proceeds as follows:

1. Construct a suitable covering schedule $\mathcal{I} = (\beta_0, w_0, k_1, \dots, k_t, \beta_t, w_t)$.
2. Estimate the values $Q(\beta_i)$ for $i = 0, \dots, t$.
3. Use these estimates $\hat{Q}(\beta_i)$ to estimate the counts c_i
4. Use the estimated counts $\hat{c}_i$ to estimate the entire function $Q(\beta)$

The first stage is quite involved, so we defer it to Section 5 where we show the following result:

▶ **Theorem 18.** *There is a procedure* $\texttt{FindCoveringSchedule}(\gamma)$ *which produces a covering schedule* $\mathcal{I}$*, which is proper with probability at least* $1-\gamma$*. In the general integer setting, the procedure has cost* $O(n \log^3 n + n \log n \log \frac{1}{\gamma} + n \log q)$ *and has* $\texttt{InvWeight}(\mathcal{I}) \le O(n \log n)$*. In the log-concave setting, the procedure has cost* $O(n \log^2 n + n \log \frac{1}{\gamma} + n \log q)$ *and has* $\texttt{InvWeight}(\mathcal{I}) \le O(n)$*.*

The second stage is summarized in the following result:

▶ **Theorem 19.** *There is an algorithm* $\texttt{PratioCoveringSchedule}(\mathcal{I}, \varepsilon, \gamma)$ *which takes as input a covering schedule* $\mathcal{I} = (\beta_0, w_0, k_1, \dots, k_t, \beta_t, w_t)$ *and produces estimates* $\hat{Q}(\beta_0), \dots, \hat{Q}(\beta_t)$*. The overall algorithm cost is* $O\Big(\frac{\min\{nW, q \log n\} \log \frac{1}{\gamma}}{\varepsilon^2}\Big)$ *where* $W = \texttt{InvWeight}(\mathcal{I})$*. If* $\mathcal{I}$ *is proper, then with probability at least* $1-\gamma$ *it satisfies* $|\log \hat{Q}(\beta_i) - Q(\beta_i)| \le \varepsilon$ *for all* i*. (When this latter condition holds, we say that the call to* $\texttt{PratioCoveringSchedule}$ *is* good*).*

Proof. To get the cost $O\big(\frac{q \log n}{\varepsilon^2} \log \frac{1}{\gamma}\big)$, we simply run the algorithm $\mathcal{D} \leftarrow \texttt{PratioAll}(\varepsilon, \gamma)$ for the continuous setting as in Theorem 6, and output $\hat{Q}(\beta_i \mid \mathcal{D})$ for all i. To get the other cost bound (in terms of W), we use the following algorithm:

Algorithm 2 Estimating values $Q(\beta_i)$ via `EstimateProducts`.

1 **for** $\boldsymbol{i = 1, \ldots t}$ form random variables $X_i \sim \text{Bernoulli}(\mu_{\beta_{i-1}}(k_i))$ and $Y_i \sim \text{Bernoulli}(\mu_{\beta_i}(k_i))$
2 set $\hat{X}^{\texttt{prod}} \leftarrow \texttt{EstimateProducts}(X, W, \varepsilon/2, \gamma/4)$
3 set $\hat{Y}^{\texttt{prod}} \leftarrow \texttt{EstimateProducts}(Y, W, \varepsilon/2, \gamma/4)$
4 for $i = 0, \ldots, t$ set $\hat{Q}(\beta_i) = \exp\left(\sum_{j=1}^{i}(\beta_j - \beta_{j-1})k_j\right) \cdot \hat{X}_i^{\texttt{prod}} / \hat{Y}_i^{\texttt{prod}}$

Here, assuming that $\mathcal{I}$ is proper, we have $\mathbb{S}[X_i] = \frac{1}{\mu_{\beta_{i-1}}(k_i)-1} \leq \frac{1}{w_{i-1}}$ for each i, so $\sum_i \mathbb{S}[X_i] \leq W$. Likewise $\sum_i \mathbb{S}[Y_i] \leq W$. So with probability at least $1 - \gamma/2$ the estimates $\hat{X}_i^{\texttt{prod}}, \hat{Y}_i^{\texttt{prod}}$ are all within $e^{\pm\varepsilon/2}$ of $\prod_{j=1}^{i}\mathbb{E}[X_j], \prod_{j=1}^{i}\mathbb{E}[Y_j]$ respectively. Observe that

$$\frac{\mathbb{E}[\prod_{j=1}^{i} X_j]}{\mathbb{E}[\prod_{j=1}^{i} Y_j]} = \prod_{j=1}^{i-1} \frac{\mu_{\beta_{j-1}}(k_j)}{\mu_{\beta_j}(k_j)} = \prod_j e^{(\beta_{j-1}-\beta_j)k_j} \frac{Z(\beta_j)}{Z(\beta_{j-1})} = \frac{Z(\beta_i)}{Z(\beta_0)} \cdot \exp\Big(\sum_{j=1}^{i}(\beta_{j-1} - \beta_j)k_j\Big)$$

so in that case, the values $\hat{Q}(\beta_i)$ are also within $e^{\pm\varepsilon}$ of $Z(\beta_i)/Z(\beta_0) = Q(\beta_i)$ as required. ◀

4.1 Solving $P^{\delta,\varepsilon}_{\texttt{count}}$

We now move on to the third stage, of using the covering schedule to solve $P_{\texttt{count}}$. There are two quite distinct algorithms here: one for generic integer-valued distributions, and a specialized algorithm for log-concave distributions. We begin with the following algorithm for general integer distributions:

Algorithm 3 Solving problem $P^{\delta,\varepsilon}_{\texttt{count}}$.

1 set $\mathcal{I} = (\beta_0, w_0, k_1, \ldots, k_t, \beta_t, w_t) \leftarrow \texttt{FindCoveringSchedule}(\gamma/10)$
2 set $(\hat{Q}(\beta_0), \ldots, \hat{Q}(\beta_t)) \leftarrow \texttt{PratioCoveringSchedule}(\mathcal{I}, \varepsilon/100, \gamma/10)$
3 **for** $i = 0, \ldots, t$ **do** let $\hat{\mu}_{\beta_i} \leftarrow \texttt{Sample}(\beta_i; \varepsilon/100, \frac{\gamma}{10(n+1)^2}, w_i)$
4 **for** $j \in \mathcal{H}$ **do**
5 set $\alpha \leftarrow \texttt{Balance}(\beta_{\min}, \beta_{\max}, j, \frac{\gamma}{10(n+1)^2}, 1/4)$
6 find index $i < t$ with $\alpha \in [\beta_i, \beta_{i+1}]$
7 let $\hat{\mu}_\alpha \leftarrow \texttt{Sample}(\alpha; \varepsilon/100, \frac{\gamma}{10(n+1)^2}, \delta/4)$
8 **if** $\boldsymbol{\hat{\mu}_\alpha(k_{i+1}) \geq \delta}$ **then** $\texttt{EstimatePi}(j, \alpha, \delta/4)$ where $\hat{Q}(\alpha) = \frac{\hat{\mu}_{\beta_i}(k_{i+1})}{\hat{\mu}_\alpha(k_{i+1})} e^{(\alpha-\beta_i)k_{i+1}} \hat{Q}(\beta_i)$
9 **else if** $\boldsymbol{j \geq k_{i+1}}$ **then** $\texttt{EstimatePi}(j, \beta_{i+1}, w_{i+1}/8)$ where $\hat{Q}(\beta_{i+1})$ is set at line 2.
10 **else if** $\boldsymbol{j < k_{i+1}}$ **then** $\texttt{EstimatePi}(j, \beta_i, w_i/8)$ where $\hat{Q}(\beta_i)$ is set at line 2.

▶ **Theorem 20.** *Algorithm 3 solves $P^{\delta,\varepsilon}_{\texttt{count}}$ with cost $O\Big(\frac{(n/\delta)\log\frac{n}{\gamma} + n^2 \log n \log\frac{1}{\gamma}}{\varepsilon^2} + n \log q\Big)$.*

Proof. See full paper. ◀

This gives the second part of Theorem 4. As we have mentioned, there is an alternative algorithm to estimate counts in the log-concave setting:

Algorithm 4 Solving $P^{\delta,\varepsilon}_{\texttt{count}}$ in the log-concave setting.

set $\mathcal{I} = (\beta_0, w_0, k_1, \dots, k_t, \beta_t, w_t) \leftarrow \texttt{FindCoveringSchedule}(\gamma/10)$
set $(\hat{Q}(\beta_0), \dots, \hat{Q}(\beta_t)) \leftarrow \texttt{PratioCoveringSchedule}(\mathcal{I}, 0.1\varepsilon, \gamma/6)$
update $\delta \leftarrow \min\{\delta, 1/n, 1/\texttt{InvWeight}(\mathcal{I})\}$.
for $i = 1, \dots t-1$ **do**
  let $\hat{\mu}_{\beta_i} \leftarrow \texttt{Sample}(\beta_i; 0.01\varepsilon, \frac{\gamma}{6(n+1)}, w_i)$
  foreach $j \in \{k_i + 1, k_i + 2, \dots, k_{i+1}\}$ **do** $\texttt{EstimatePi}(j, \beta_i, \delta/4)$
let $\hat{\mu}_{\beta_{\min}} \leftarrow \texttt{Sample}(\beta_{\min}; 0.01\varepsilon, \frac{\gamma}{6(n+1)}, w'_0)$ for $w'_0 = \min\{w_0, \delta/2\}$
foreach $j \in \{0, 1, \dots, k_1\}$ **do** $\texttt{EstimatePi}(j, \beta_{\min}, w'_0)$.
let $\hat{\mu}_{\beta_{\max}} \leftarrow \texttt{Sample}(\beta_{\max}; 0.01\varepsilon, \frac{\gamma}{6(n+1)}, w'_t)$ for $w'_t = \min\{w_t, \delta/2\}$)
foreach $j \in \{k_t + 1, k_t + 2, \dots, n\}$ **do** $\texttt{EstimatePi}(j, \beta_{\max}, w'_t)$.

▶ **Theorem 21.** *In the log-concave setting, Algorithm 4 solves $P^{\delta,\varepsilon}_{\texttt{count}}$ with cost*

$$O\Big(n \log^2 n + n \log q + \frac{\min\{n^2, q \log n\} \log \frac{1}{\gamma} + (n + 1/\delta) \log \frac{n}{\gamma}}{\varepsilon^2}\Big)$$

Proof. See full paper. ◀

Again, with some simplification of parameters, this gives the third part of Theorem 4.

4.2 Solving $P^{\texttt{all}}_{\texttt{ratio}}$

Finally, having estimated the counts, we can proceed to use these estimates to fill in the entire function $Q(\beta)$. This is a black-box reduction from $P_{\texttt{count}}$ to $P^{\texttt{all}}_{\texttt{ratio}}$.

▶ **Theorem 22.** *Given a solution $(\hat{\pi}, u)$ for $P^{1/n, 0.1\varepsilon}_{\texttt{count}}$ in the integer setting, we can solve $P^{\texttt{all}}_{\texttt{ratio}}$ with probability one and no additional queries to the oracle.*

Proof. The data structure $\mathcal{D}$ is the vector $\hat{\pi}$, and for a query value α we set $\hat{Q}(\alpha \mid \mathcal{D}) = \sum_{i \in \mathcal{H}} \hat{\pi}(i) e^{(\alpha - \beta_{\min})i}$. See full paper for proof details. ◀

Our $P^{\delta,\varepsilon}_{\texttt{count}}$ algorithms thus solve $P^{\texttt{all}}_{\texttt{ratio}}$ with cost $O\big(\frac{n^2 \log n \log \frac{1}{\gamma}}{\varepsilon^2} + n \log q\big)$ in the general integer setting, and $O\big(\frac{n^2 \log \frac{1}{\gamma}}{\varepsilon^2} + n \log q\big)$ in the log-concave setting. This shows the two bounds of Theorem 6.

5 Constructing a covering schedule

In the full paper, we show that any non-negative log-concave sequence $a_1, \dots, a_m$ satisfying $a_k \le \frac{1}{k}$ for each $k \in [m]$ satisfies $a_1 + \dots + a_m \le e$. Without the log-concavity assumption we would have $a_1 + \dots + a_m \le \sum_{k=1}^m \frac{1}{k} \le 1 + \log m$ (by a well-known inequality for the harmonic series). Motivated by these facts, we define the following parameter in this section:

$$\rho \stackrel{\text{def}}{=} \begin{cases} 1 + \log(n+1) & \text{in the general integer setting} \\ e & \text{in the log-concave setting} \end{cases}$$

We will show the following more precise bound on the weight of the schedule.

▶ **Theorem 23.** *In the integer setting, the procedure* `FindCoveringSchedule`(γ) *produces a covering schedule* $\mathcal{I}$ *with* `InvWeight`$(\mathcal{I}) \le a(n+1)\rho$ *and* $\mathbb{P}[\mathcal{I}$ *is proper*$] \ge 1-\gamma$*, where* $a > 4$ *is an arbitrary constant. It has cost* $O(n\rho(\log^2 n + \log\frac{1}{\gamma}) + n\log q)$.

This immediately implies Theorem 18. In order to build the covering schedule, we first build an object with relaxed constraints called a *preschedule*, discussed in Sections 5.1. In Section 5.2, we convert this into a schedule.

5.1 Constructing a preschedule

Let us fix constants $\tau \in (0, \frac{1}{2})$, $\lambda \in (0,1)$, and set $\phi = \tau\lambda^3/\rho$. Let us introduce basic terminology and definitions.

An $\mathcal{H}$-interval is a discrete set of points $\{\sigma^-, \sigma^-+1, \dots, \sigma^+-1, \sigma^+\}$, for integers $0 \le \sigma^- \le \sigma^+ \le n$. We also write this more compactly as $\sigma = [\sigma^-, \sigma^+]$. We define $\mathtt{span}(\sigma) = \sigma^+ - \sigma^- + 1$, i.e. the cardinality of σ when viewed as a subset of $\mathcal{H}$.

A *segment* is a tuple $\theta = (\beta, \sigma)$ where $\beta \in [\beta_{\min}, \beta_{\max}]$, and σ is an $\mathcal{H}$-segment. We say θ is ϕ-*proper* (or just proper if ϕ is understood) if it satisfies the following two properties:

- Either $\beta = \beta_{\min}$ or $\mu_\beta(\sigma^-) \ge \phi/\mathtt{span}(\sigma)$
- Either $\beta = \beta_{\max}$ or $\mu_\beta(\sigma^+) \ge \phi/\mathtt{span}(\sigma)$

A *preschedule* is a sequence of distinct segments $\mathcal{J} = ((\beta_0, \sigma_0), \dots, (\beta_t, \sigma_t))$ satisfying the following properties:

(`I0`) $\sigma^-_{i+1} \le \sigma^+_i$ for $i = 0, \dots, t-1$.
(`I1`) $\beta_{\min} = \beta_0 \le \dots \le \beta_t = \beta_{\max}$.
(`I2`) $0 = \sigma^-_0 \le \dots \le \sigma^-_t \le n$ and $0 \le \sigma^+_0 \le \dots \le \sigma^+_t = n$

We say that $\mathcal{I}$ is ϕ-*proper* if all segments θ_i are ϕ-proper.

The main idea of the algorithm is to maintain a sequence of proper segments satisfying properties (`I1`) and (`I2`), and grow it until it satisfies (`I0`). This uses an additional subroutine $\sigma \leftarrow \mathtt{FindInterval}(\beta, \sigma_{\mathtt{left}}, \sigma_{\mathtt{right}})$, where $\beta \in [\beta_{\min}, \beta_{\max}]$, and $\sigma_{\mathtt{left}}, \sigma_{\mathtt{right}}$ are two discrete intervals in $\mathcal{H}$ and the returned interval $\sigma = [\sigma^-, \sigma^+]$ has $\sigma^- \in \sigma_{\mathtt{left}}, \sigma^+ \in \sigma_{\mathtt{right}}$. Deferring for the moment the definition of `FindInterval`, the details are provided below.

Algorithm 5 Computing an initial preschedule.

call $\sigma_{\min} \leftarrow \mathtt{FindInterval}(\beta_{\min}, \{0\}, \mathcal{H})$ and $\sigma_{\max} \leftarrow \mathtt{FindInterval}(\beta_{\max}, \mathcal{H}, \{n\})$
initialize $\mathcal{J}$ to contain the two segments $(\beta_{\min}, \sigma_{\min}), (\beta_{\max}, \sigma_{\max})$
while $\mathcal{J}$ *does not satisfy* (`I0`) **do**
  pick arbitrary consecutive segments $\theta_{\mathtt{left}} = (\beta_{\mathtt{left}}, \sigma_{\mathtt{left}})$ and $\theta_{\mathtt{right}} = (\beta_{\mathtt{right}}, \sigma_{\mathtt{right}})$ in $\mathcal{J}$ with $\sigma^+_{\mathtt{left}} < \sigma^-_{\mathtt{right}}$.
  let $M = \left\lfloor \frac{\sigma^+_{\mathtt{left}} + \sigma^-_{\mathtt{right}}}{2} \right\rfloor + \frac{1}{2}$
  call $\beta \leftarrow \mathtt{Balance}(\beta_{\mathtt{left}}, \beta_{\mathtt{right}}, M, \frac{1}{4n}, \tau)$
  call

$$\sigma \leftarrow \begin{cases} \mathtt{FindInterval}(\beta, [\sigma^-_{\mathtt{left}}, M-\frac{1}{2}], [M+\frac{1}{2}, \sigma^+_{\mathtt{right}}]) & \text{if } \beta_{\mathtt{left}} < \beta < \beta_{\mathtt{right}} \\ \mathtt{FindInterval}(\beta, \{\sigma^-_{\mathtt{left}}\}, [M+\frac{1}{2}, \sigma^+_{\mathtt{right}}]) & \text{if } \beta = \beta_{\mathtt{left}} \\ \mathtt{FindInterval}(\beta, [\sigma^-_{\mathtt{left}}, M-\frac{1}{2}], \{\sigma^+_{\mathtt{right}}\}) & \text{if } \beta = \beta_{\mathtt{right}} \end{cases}$$

  insert (β, σ) into $\mathcal{J}$ between $\theta_{\mathtt{left}}$ and $\theta_{\mathtt{right}}$
return $\mathcal{J}$

Now let us say that a segment (β, σ, w) is *extremal* if it satisfies the following conditions:

$$\mu_\beta(k) \leq \frac{1}{\lambda} \cdot \frac{\texttt{span}(\sigma)}{\texttt{span}(\sigma) + (\sigma^- - k)} \cdot \mu_\beta(\sigma^-) \qquad \forall k \in \{0, \dots, \sigma^- - 1\} \tag{6a}$$

$$\mu_\beta(k) \leq \frac{1}{\lambda} \cdot \frac{\texttt{span}(\sigma)}{\texttt{span}(\sigma) + (k - \sigma^+)} \cdot \mu_\beta(\sigma^+) \qquad \forall k \in \{\sigma^+ + 1, \dots, n\} \tag{6b}$$

There are two additional invariants we hope to maintain in Algorithm 5:
(I3) Each segment θ of $\mathcal{J}$ is ϕ-proper.
(I4) Each segment θ of $\mathcal{J}$ is extremal.

We say the call $\sigma \leftarrow \texttt{FindInterval}(\beta, \sigma_{\texttt{left}}, \sigma_{\texttt{right}})$ is *good* if the segment $\theta = (\beta, \sigma)$ satisfies (I3) and (I4), and we say the call at line 7 is *valid* if $\beta \in \Lambda_\tau(\beta_{\texttt{left}}, \beta_{\texttt{right}}, M)$ and both $\theta_{\texttt{left}}$ and $\theta_{\texttt{right}}$ satisfy (I3), (I4). The calls at line 1 are always valid. The following result summarizes `FindInterval`.

▶ **Theorem 24.** $\texttt{FindInterval}(\beta, \sigma_{\texttt{left}}, \sigma_{\texttt{right}})$ *has cost* $O\big(\rho(\sigma^+_{\texttt{right}} - \sigma^-_{\texttt{left}} + 1) \log n\big)$. *If the call is valid, then the call is good with probability at least* $1 - \frac{1}{4(n+2)}$.

We defer the proof, which is quite technical, the full paper. Putting it aside for the moment, we have the following results:

▶ **Proposition 25.** *Algorithm 5 outputs a preschedule, and it is ϕ-proper with probability at least* $1/2$.

Proof. If all calls to `Balance` and `FindInterval` are good, then $\mathcal{J}$ maintains properties (I3) and (I4), and in particular it is ϕ-proper. The loop in lines 3 – 8 is executed at most n times, since each time it covers a new half-integer value M. So the algorithm calls `FindInterval` at most $n+2$ times and `Balance` at most n times. Since `Balance` or `FindInterval` fail with probability at most $\frac{1}{4n}$ and $\frac{1}{4(n+2)}$ respectively, properties (I3) and (I4) are maintained with probability at least $1/2$. ◀

▶ **Proposition 26.** *Algorithm 5 has cost* $O(n \log q + n\rho \log^2 n)$.

Proof. See full paper. ◀

5.2 Converting the preschedule into a covering schedule

There are two steps to convert the preschedule into a covering schedule. First, we throw away redundant intervals. Second, we "uncross" the adjacent intervals. While we are doing this, we also check if the resulting schedule is proper; if not, we will discard it and generate a new preschedule from scratch.

▶ **Proposition 27.** *Given a preschedule $\mathcal{J}$, there is a procedure* $\texttt{MinimizePreschedule}(\mathcal{J})$, *which has zero sample complexity, to generate a preschedule* $\mathcal{J}' = ((\beta_0, \sigma_0), \dots, (\beta_t, \sigma_t))$ *satisfying the following three properties:*
(J1) $\sigma_i^+ < \sigma_{i+2}^-$ *for* $i = 0, \dots, t-2$.
(J2) $\beta_0 < \beta_1 < \cdots < \beta_t$ *strictly.*
(J3) *For any $k \in \mathcal{H}$, there are at most two segments $\theta_i = (\beta_i, \sigma_i) \in \mathcal{J}'$ with $k \in \sigma_i$.*
Furthermore, if $\mathcal{J}$ is ϕ-proper, then so is $\mathcal{J}'$ with probability one.

Proof. Start with $\mathcal{J}$ and repeatedly apply two operations: (i) discard a segment $i \in \{1, \dots, t-1\}$ if $\sigma_{i+1}^- \leq \sigma_{i-1}^+$ or (ii) merge adjacent segments with $\beta_i = \beta_{i+1}$, namely, replace the two segments $(\beta_i, \sigma_i), (\beta_{i+1}, \sigma_{i+1})$ with a single segment $(\beta_i, [\sigma_i^-, \sigma_{i+1}^+])$. The operations are performed in any order until no further changes are possible; let $\mathcal{J}'$ be the result of this process. ◀

We next describe the procedure to uncross a preschedule. Here $\nu > 0$ is some arbitrary constant.

Algorithm 6 `UncrossSchedule`$(\mathcal{J}, \gamma)$ for preschedule $\mathcal{J} = ((\beta_0, \sigma_0), \dots, (\beta_t, \sigma_t))$.

for $i = 0, \dots, t$ **do** let $\hat{\mu}_{\beta_i} \leftarrow \texttt{Sample}(\beta_i; \frac{\nu}{2}, \frac{\gamma}{4(t+1)}, e^{-\nu/2} w_i)$ where $w_i = \phi / \texttt{span}(\sigma_i)$
for $i = 1, \dots, t$ **do**
  if $\exists\, k \in \{\sigma_{i-1}^+, \sigma_i^-\}$ *s.t.* $\hat{\mu}_{\beta_{i-1}}(k) \geq e^{-\nu/2} w_{i-1}$ *and* $\hat{\mu}_{\beta_i}(k) \geq e^{-\nu/2} w_i$ **then**
    set $k_i = k$ for arbitrary such k
  else return $\perp$
return *covering schedule* $\mathcal{I} = (\beta_0, e^{-\nu} w_0, k_1, \beta_1, e^{-\nu} w_1, k_2, \dots, k_t, \beta_t, e^{-\nu} w_t)$

▶ **Theorem 28.** *Suppose that preschedule $\mathcal{J}$ satisfies properties* `(J1)`, `(J2)`, `(J3)`. *Then:*
(a) *The output is either $\perp$ or a covering schedule $\mathcal{I}$ with* $\texttt{InvWeight}(\mathcal{I}) \leq \frac{2e^{\nu}(n+1)}{\phi}$.
(b) *It outputs an improper covering schedule with probability at most γ, irrespective of $\mathcal{J}$.*
(c) *If $\mathcal{J}$ is ϕ-proper, then it outputs a proper covering schedule with probability at least $1 - \gamma$.*
(d) *The cost is $O(n\rho \log \frac{n}{\gamma})$.*

Proof. See full paper. ◀

We can finish by combining all the preschedule processing algorithms, as follows:

Algorithm 7 Algorithm `FindCoveringSchedule`(γ).

while `true` **do**
  call Algorithm 5 with appropriate constants ν, λ, τ to compute preschedule $\mathcal{J}$
  call $\mathcal{J}' \leftarrow \texttt{MinimizePreschedule}(\mathcal{J})$
  call $\mathcal{I} \leftarrow \texttt{UncrossSchedule}(\mathcal{J}', \gamma/4)$
  if $\mathcal{I} \neq \perp$ **then return** $\mathcal{I}$

By Proposition 25 and Theorem 28, each iteration of Algorithm 7 terminates with probability at least $\frac{1}{2}(1 - \gamma/4) \geq 3/8$, so there are $O(1)$ expected iterations. Each call to `UncrossSchedule` has cost $O(n\rho \log \frac{n}{\gamma})$. By Proposition 26, each call to Algorithm 5 has cost $O(n \log q + n\rho \log n)$.

By Theorem 28(a), $\texttt{InvWeight}(\mathcal{I}) \leq 2\rho(n+1) \cdot \frac{e^{\nu}}{\tau\lambda^3}$. The term $\frac{e^{\nu}}{\tau\lambda^3}$ gets arbitrarily close to 2 for constants ν, λ, τ sufficiently close to $0, 1, \frac{1}{2}$ respectively.

Finally, by Proposition 28, each iteration of Algorithm 7 returns a non-proper covering schedule with probability at most $\gamma/4$ (irrespective of the choice of $\mathcal{J}$). Thus, the total probability of returning a non-proper covering schedule over all iterations is at most $\sum_{i=0}^{\infty} (3/8)^i \gamma/4 = 2\gamma/5 \leq \gamma$ as desired.

This shows Theorem 23.

6 Combinatorial applications

Consider a combinatorial setting with c_i objects of weights $i = 0, \dots, n$, and we can sample from a Gibbs distribution at rate β (for certain values of β). If we know at least one of the counts, then estimates for $\pi(x)$ directly translate into estimate of c_i. Our usual strategy here will be to solve $P^{\delta,\varepsilon}_{\texttt{count}}$ for $\delta = O(\min_x \Delta(x))$, for chosen boundary parameters $\beta_{\min}, \beta_{\max}$; in this case, it can easily be seen that the resulting estimated counts $\hat{c}_i = c_0 \hat{\pi}(i)/\hat{\pi}(0)$ are accurate within $e^{\pm\varepsilon}$ relative error.

In many of these combinatorial applications, the counts are known to be log-concave; in this case, there are natural choices for algorithm parameters which lead to particularly clean bounds. When counts are not log-concave, more involved properties of the Gibbs distribution (e.g. it approaches a normal distribution) must be used.

▶ **Theorem 29.** *Suppose the counts are log-concave and non-zero. If* $\beta_{\min} \le \log \frac{c_0}{c_1}$ *and* $\beta_{\max} \ge \log \frac{c_{n-1}}{c_n}$*, then* $\Delta(k) \ge \frac{1}{n+1}$ *for all* $k = 0, \dots, n$*, and* $\log Q(\beta_{\max}) \le q := 3n\Gamma$ *where* $\Gamma := \max\{\beta_{\max}, \log \frac{c_1}{c_0}, 1\}$*. In particular, for* $\delta = \frac{1}{n+1}$*, we can solve* $P^{\delta,\varepsilon}_{\texttt{count}}$ *with cost*

$$O\Big(\min\Big\{\frac{n\Gamma \log n \log \frac{1}{\gamma}}{\varepsilon^2}, \frac{n^2 \log \frac{1}{\gamma}}{\varepsilon^2} + n \log \Gamma\Big\}\Big)$$

Proof. See full paper. ◀

Theorem 3 follows directly from Theorem 29 combined with an MCMC sampler for matchings appearing [15]. (See full paper for details).

6.1 Counting connected subgraphs

Consider a connected graph $G = (V, E)$. In [11], Guo & Jerrum described an algorithm to sample a connected subgraph $G' = (V, E')$ with probability proportional to $\prod_{f \in E'}(1 - p(f)) \prod_{f \in E - E'} p(f)$, for any weighting function $p : E \to [0, 1]$. If we set $p(f) = \frac{1}{1+e^\beta}$ for all edges f, then their algorithm samples from the Gibbs distribution with c_i being the number of connected subgraphs of G with $|E| - i$ edges. Guo & He [10] subsequently improved the algorithm runtime; we summarize their result as follows:

▶ **Theorem 30** ([10], Corollary 10)**.** *There is an algorithm to sample from the Gibbs distribution with counts* c_i *for any value of* $\beta > 0$*; the expected runtime is* $O(|E| + |E||V|e^\beta)$*.*

Proof of Theorem 1. The sequence c_i counts the number of independent sets in the co-graphic matroid, where $n = |E| - |V| + 1$. By the result of [1], this sequence c_i is log-concave; also $c_0 = 1$ so it suffices to estimates counts up to any scaling. The ratios c_{n-1}/c_n and c_1/c_0 are both at most $|E|$, since to enumerate a connected graph with $|V|$ edges we may select a spanning tree and any other edge in the graph, and to enumerate a graph with $|E| - 1$ edges we simply select an edge of G to delete.

So we can apply Theorem 29, setting $\beta_{\max} = \log |E| \ge \log \frac{c_{n-1}}{c_n}, \beta_{\min} = -\log |E| \le \log \frac{c_0}{c_1}$ and $\Gamma = \log |E|$. The definition of an FPRAS traditionally sets $\gamma = O(1)$, and here $n = |E|$. So the algorithm uses $O(\frac{|E| \log^2 |E|}{\varepsilon^2})$ samples in expectation. With these parameters, each call to the sampling oracle of Theorem 30 has runtime $O(|E|^2|V|)$. The total runtime is then $O(\frac{|E|^3|V| \log^2 |E|}{\varepsilon^2})$. ◀

The work [11] sketches an FPRAS for this problem as well; the precise complexity is unspecified and appears to be much larger than Theorem 1. We also note that Anari et al. [2] provide a general FPRAS for counting the number of independent sets in arbitrary matroids, which would include the number of connected subgraphs. This uses a very different sampling method, which is not based on the Gibbs distribution. They do not provide concrete complexity estimates for their algorithm.

6.2 Counting independent sets in bounded-degree graphs

For a graph $G = (V, E)$ of maximum degree D, let I_k denote the collection of independent sets of size k for $k = 0, \dots, |V|$. A key problem in statistical physics is to sample efficiently from I_k. Here, there is critical hardness threshold defined by $\lambda_c = \frac{(D-1)^{D-1}}{(D-2)^D} \approx e/D$, such

that, for $\beta > \lambda_c$, it is intractable to sample from the Gibbs distribution at rate λ; on the other, for $\beta < \lambda_c$, there is a polynomial-time sampler for the Gibbs distribution. We quote the following result of [6].

▶ **Theorem 31** ([6]). *Let $D \geq 3$ and $\xi > 0$ be any fixed constants. There is an algorithm to approximately sample from the Gibbs distribution at $\beta \in [0, \lambda_c - \xi]$, up to total variation distance ρ, with runtime $O(n \log n \log(n/\rho))$.*

The related problem of estimating the values i_k was considered in [7]. Based on this sampling result, they identified a related computational threshold for estimating the counts $c_k = |I_k|$. Namely, they define the threshold value $\alpha_c = \frac{\lambda_c}{1+(D+1)\lambda_c}$ and then show that, for $k > \alpha_c|V|$, it is intractable to estimate c_k or to sample approximately uniformly from I_k; on the other hand, for constant $D \geq 3$ and $\xi > 0$ and $k < (\alpha_c - \xi)|V|$, then describe an algorithm to estimate c_k in polynomial time. A follow-up work [13] provided tighter estimates for the Gibbs distribution and improved algorithms; specifically, it showed how to estimate a given count c_i for $i < (\alpha - \xi)|V|$ with runtime $\tilde{O}(n^2/\varepsilon^2)$.

A key analytical technique of [13] was to show that the Gibbs distribution for independent sets closely approximated to a normal distribution, i.e. it obeyed a type of Central Limit Theorem. Using Theorem 3.1 of [13], we have the following crude estimate:

▶ **Lemma 32** ([13]). *Let $D \geq 3$ and $\xi > 0$ be any fixed constants. There is a constant $\xi' > 0$ such that, for any $k \leq (\alpha_c - \xi)|V|$, there is some value $\beta \in [0, \lambda_c - \xi']$ with $\mu_\beta(k) \geq \Omega(1/\sqrt{|V|})$.*

By using Lemma 32, we immediately get the following result:

▶ **Theorem 33.** *Let $D \geq 3$ and $\xi > 0$ be any fixed constants. There is an algorithm to estimate all counts $c_0, \dots c_{\lfloor(\alpha_c - \xi)|V|\rfloor}$ with runtime $\tilde{O}(\frac{n^2 \log(1/\gamma)}{\varepsilon^2})$.*

Proof. We set $\beta_{\min} = 0, \beta_{\max} = \lambda_c - \xi', n = |V|$. Note that the Gibbs distribution is *not* necessarily log-concave. Since $c_0 = 1$ and clearly $c_i \leq 2^n$ for all i, we have $Q(\beta_{\max})/Q(\beta_{\min}) \leq (2^n e^{\beta_{\max} n})/1$; in particular, since $\beta_{\max} = O(1)$ (for fixed D), we have $Q(\beta_{\max})/Q(\beta_{\min}) \leq e^{O(n)}$ and we can take $q = \Theta(n)$. By Lemma 32, we have $\Delta(k) \geq \Omega(1/\sqrt{n})$ for these parameters. Thus, it suffices to solve $P^{\delta, 0.1\varepsilon}_{\texttt{count}}$ for $\delta = \Omega(1/\sqrt{n})$.

For this purpose, we will actually use the continuous-setting algorithm – it is more efficient than the general integer-setting algorithm. By Theorem 13, this algorithm has cost

$$O\Big(\frac{\min\{q, \sqrt{q \log n}\} \log \frac{q}{\delta\gamma}}{\delta\varepsilon^2} + \frac{q \log n \log \frac{1}{\gamma}}{\varepsilon^2}\Big) = O\Big(\frac{n \log^{3/2} n + n \log n \log \frac{1}{\gamma}}{\varepsilon^2}\Big).$$

Accordingly, we need to run the approximate sampler of Theorem 31 with $\rho = \text{poly}(n, 1/\varepsilon, \log \frac{1}{\gamma})$ leading to a computational complexity of $O(n \log n \log(\frac{n \log 1/\gamma}{\varepsilon}))$. With some simplification of parameters, the overall runtime becomes

$$O\Big(\frac{n^2 \log^{5/2} n \log(n/\varepsilon) + n^2 \log^2 n \log \frac{1}{\gamma} \log(\frac{n \log 1/\gamma}{\varepsilon})}{\varepsilon^2}\Big) = \tilde{O}\Big(\frac{n^2 \log \frac{1}{\gamma}}{\varepsilon^2}\Big). \qquad \blacktriangleleft$$

We note that the algorithm in [13] has this same runtime, but only estimates a *single* count c_i; our algorithm simultaneously produces estimates for all values c_i up to the threshold value $i < (\alpha_c - \xi)|V|$ with the same runtime. It also does not depend on the precise distributional properties of the Gibbs distribution; it only requires the much cruder estimate in Lemma 32.

7 Lower bounds on sample complexity

Following [16], our strategy is to construct a target instance $c^{(0)}$ surrounded by an envelope of d alternate instances $c^{(1)}, \dots, c^{(d)}$, such that solving $P^{\texttt{point}}_{\texttt{ratio}}$ or $P_{\texttt{count}}$ on an unknown instance $c^{(r)}$ distinguishes between the cases $r = 0$ and $r > 0$. On the other hand, an "indistinguishability lemma" gives a lower bound on the sample complexity of any such procedure to distinguish the distributions.

Define $\mu^{(r)}_\beta$ to be the Gibbs distribution with parameter β for instance $c^{(r)}$, and $Z^{(r)}(\beta)$ to be its partition function, and $z^{(r)} = \log Z^{(r)}$, and $\Delta^{(r)}(x) = \max_{\beta \in [\beta_{\min}, \beta_{\max}]} \mu^{(r)}_\beta(x)$. We will require that the instances are *balanced*, namely, that they satisfy the property $\prod_{r=1}^d c^{(r)}_x = (c^{(0)}_x)^d$ for all $x \in \mathcal{F}$. We also define parameters

$$U(\beta) = \prod_{r=1}^{d} \frac{Z^{(r)}(\beta)}{Z^{(0)}(\beta)}, \qquad \Psi = \max_{\beta \in [\beta_{\min}, \beta_{\max}]} \log U(\beta).$$

▶ **Lemma 34** ([16]). *Let $\mathfrak{A}$ be an algorithm which generates queries $\beta_1, \dots, \beta_T \in [\beta_{\min}, \beta_{\max}]$ and receives values $x_1, \dots, x_T$, where each x_i is drawn from μ_{β_i}. At some point the procedure stops and outputs TRUE or FALSE. The queries β_i and the stopping time T may be adaptive and may be randomized.*

Suppose that $\mathfrak{A}$ outputs TRUE on input $c^{(0)}$ with probability at least $1 - \gamma$ and outputs FALSE on inputs $c^{(1)}, \dots, c^{(d)}$ with probability at least $1 - \gamma$, for some parameter $\gamma < 1/4$.

If the instances are balanced, then the cost of $\mathfrak{A}$ on instance $c^{(0)}$ is $\Omega(\frac{d \log(1/\gamma)}{\Psi})$.

To get more general lower bounds for count estimation, we consider a problem variant called $\check{P}^{\delta,\varepsilon}_{\texttt{count}}$, namely, to compute a vector $\hat{c} \in (\mathbb{R}_{>0} \cup \{?\})^{\mathcal{F}}$ satisfying the following two properties:

- **(i)** for all pairs x, y with $\hat{c}_x, \hat{c}_y \neq ?$, there holds $|\log \frac{\hat{c}_x}{\hat{c}_y} - \log \frac{c_x}{c_y}| \leq \varepsilon$
- **(ii)** for all x with $\Delta(x) \geq \delta$ there holds $\hat{c}_x \neq ?$.

Given a solution $(\hat{\pi}, u)$ to $P^{\delta, 0.1\varepsilon}_{\texttt{count}}$, we can solve $\check{P}^{\delta,\varepsilon}_{\texttt{count}}$ with zero sample complexity and probability one (see full paper for details). Problem $\check{P}^{\delta,\varepsilon}_{\texttt{count}}$ is easier than $P^{\delta,\varepsilon}_{\texttt{count}}$ (up to constant factors in parameters), in two ways: first, it does not require any specific normalization of the counts, only pairwise consistency. Second, it only provides approximation guarantees for c_x if $\Delta(x) \geq \delta$, while $P^{\delta,\varepsilon}_{\texttt{count}}$ provides meaningful bounds over a wide range of scales.

▶ **Corollary 35.**

- **(a)** *Suppose that $|z^{(0)}(\beta_{\min}, \beta_{\max}) - z^{(r)}(\beta_{\min}, \beta_{\max})| > 2\varepsilon$ for all $r = 1, \dots, d$. Then any algorithm for $P^{\texttt{point}}_{\texttt{ratio}}$ must have cost $\Omega(\frac{d \log(1/\gamma)}{\Psi})$ on instance $c^{(0)}$.*
- **(b)** *Suppose that for each $r = 1, \dots, d$ there are x, y with $\Delta^{(0)}(x), \Delta^{(0)}(y) \geq \delta$, and $|\log(c^{(0)}_x / c^{(0)}_y) - \log(c^{(r)}_x / c^{(r)}_y)| > 2\varepsilon$. (We refer to the values x, y as the* witnesses *for r.) Then any algorithm for $\check{P}^{\delta,\varepsilon}_{\texttt{count}}$ must have cost $\Omega(\frac{d \log(1/\gamma)}{\Psi})$ on instance $c^{(0)}$.*

Proof. We show how to convert these algorithms into procedures distinguishing $c^{(0)}$ from $c^{(1)}, \dots, c^{(d)}$:

- **(a)** Given a solution $\hat{Q}(\beta_{\max})$ to $P^{\texttt{point}}_{\texttt{ratio}}$, output TRUE if $|\log \hat{Q}(\beta_{\max}) - z^{(0)}(\beta_{\min}, \beta_{\max})| \leq \varepsilon$, else output FALSE.
- **(b)** Given a solution $\hat{c}$ to $\check{P}^{\delta,\varepsilon}_{\texttt{count}}$, output TRUE if $\hat{c} \neq ?$ for all x with $\Delta^{(0)}(x) \geq \delta$, and every pair x, y with $\Delta(x), \Delta(y) \geq \delta$ satisfy $|\log(\hat{c}_x / \hat{c}_y) - \log(c^{(0)}_x / c^{(0)}_y)| \leq \varepsilon$, else output FALSE. ◀

By applying Corollary 35 to carefully constructed instances, we will show the following:

▶ **Theorem 36.** *Let $n \geq n_0, q \geq q_0, \varepsilon < \varepsilon_0, \delta < \delta_0, \gamma < 1/4$ for certain absolute constants $n_0, q_0, \varepsilon_0, \delta_0$. There are problem instances μ which satisfy the given bounds n and q such that:*

(a) $\check{P}^{\delta,\varepsilon}_{\texttt{count}}$ *requires cost* $\Omega(\frac{\min\{q+\sqrt{q}/\delta, n^2+n/\delta\}\log\frac{1}{\gamma}}{\varepsilon^2})$*, and μ is integer-valued.*

(b) $\check{P}^{\delta,\varepsilon}_{\texttt{count}}$ *requires cost* $\Omega(\frac{(1/\delta+\min\{q,n^2\})\log\frac{1}{\gamma}}{\varepsilon^2})$*, and μ is log-concave.*

(c) $P^{\texttt{point}}_{\texttt{ratio}}$ *requires cost* $\Omega(\frac{\min\{q,n^2\}\log\frac{1}{\gamma}}{\varepsilon^2})$*, and μ is log-concave.*

(d) $\check{P}^{\delta,\varepsilon}_{\texttt{count}}$ *requires cost* $\Omega(\frac{(q+\sqrt{q}/\delta)\log\frac{1}{\gamma}}{\varepsilon^2})$.

(e) $P^{\texttt{point}}_{\texttt{ratio}}$ *requires cost* $\Omega(\frac{q\log\frac{1}{\gamma}}{\varepsilon^2})$.

The lower bounds on $\check{P}^{\delta,\varepsilon}_{\texttt{count}}$ immediately imply lower bounds on $P^{\delta,\varepsilon}_{\texttt{count}}$, in particular, they give Theorems 5 and 7. Result (e) was already shown in [16], but we include it here since it is a corollary of other results.

The proofs and constructions appear in the full paper.

References

1 K. Adiprasito, J. Huh, and E. Katz. Hodge theory for combinatorial geometries. *Annals of Mathematics*, 188(2):381–452, 2018.

2 N. Anari, K. Liu, S. O. Gharan, and C. Vinzant. Log-concave polynomials II: High-dimensional walks and an FPRAS for counting bases of a matroid. In *Proc. 51st annual ACM Symposium on Theory of Computing (STOC)*, pages 1–12, 2019.

3 R. E. Belardinelli and V. D. Pereyra. Wang-Landau algorithm: A theoretical analysis of the saturation of the error. *The Journal of Chemical Physics*, 127(18):184105, 2007.

4 I. Bezáková, D. Štefankovič, V. V. Vazirani, and E. Vigoda. Accelerating simulated annealing for the permanent and combinatorial counting problems. *SIAM J. Comput.*, 37:1429–1454, 2008.

5 Charlie Carlson, Ewan Davies, Alexandra Kolla, and Will Perkins. Computational thresholds for the fixed-magnetization ising model. In *Proc. 54th annual ACM Symposium on Theory of Computing (STOC)*, pages 1459–1472, 2022.

6 Zongchen Chen, Kuikui Liu, and Eric Vigoda. Optimal mixing of Glauber dynamics: Entropy factorization via high-dimensional expansion. In *Proc. 53rd annual ACM Symposium on Theory of Computing (STOC)*, pages 1537–1550, 2021.

7 Ewan Davies and Will Perkins. Approximately counting independent sets of a given size in bounded-degree graphs. In *Proc. 48th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 62:1–62:18, 2021.

8 W. Feng, H. Guo, Y. Yin, and C. Zhang. Fast sampling and counting k-SAT solutions in the local lemma regime. In *Proc. 52nd annual ACM Symposium on Theory of Computing (STOC)*, pages 854–867, 2020.

9 G. Fort, B. Jourdain, E. Kuhn, T. Leliévre, and G. Stoltz. Convergence of the Wang-Landau algorithm. *Mathematics of Computation*, 84(295):2297–2327, 2015.

10 H. Guo and K. He. Tight bounds for popping algorithms. *Random Struct. Algorithms*, 57(2):371–392, 2020.

11 H. Guo and M. Jerrum. A polynomial-time approximation algorithm for all-terminal network reliability. *SIAM J. Comput.*, 48(3):964–978, 2019.

12 M. Huber. Approximation algorithms for the normalizing constant of Gibbs distributions. *The Annals of Applied Probability*, 25(2):974–985, 2015.

13 Vishesh Jain, Will Perkins, Ashwin Sah, and Mehtaab Sawhney. Approximate counting and sampling via local central limit theorems. In *Proc. 54th annual ACM Symposium on Theory of Computing (STOC)*, pages 1473–1486, 2022.

14 M. Jerrum and A. Sinclair. Approximating the permanent. *SIAM J. Comput.*, 18(6):1149–1178, 1989.

15 M. Jerrum and A. Sinclair. The Markov Chain Monte Carlo method: an approach to approximate counting and integration. *Approximation algorithms for NP-hard problems*, pages 482–520, 1996.

16 V. Kolmogorov. A faster approximation algorithm for the Gibbs partition function. *Proceedings of Machine Learning Research*, 75:228–249, 2018.

17 Pedro Ojeda, Martin E Garcia, Aurora Londoño, and Nan-Yow Chen. Monte Carlo simulations of proteins in cages: influence of confinement on the stability of intermediate states. *Biophysical journal*, 96(3):1076–1082, 2009.

18 L. N. Shchur. On properties of the Wang-Landau algorithm. *Journal of Physics: Conference Series*, 1252, 2019.

19 D. Štefankovič, S. Vempala, and E. Vigoda. Adaptive simulated annealing: A near-optimal connection between sampling and counting. *J. of the ACM*, 56(3) Article #18, 2009.

20 F. Wang and D. P. Landau. Efficient, multiple-range random walk algorithm to calculate the density of states. *Phys. Rev. Lett.*, 86(10):2050–2053, 2001.

On Finding Constrained Independent Sets in Cycles

Ishay Haviv
School of Computer Science, The Academic College of Tel Aviv-Yaffo, Israel

Abstract

A subset of $[n] = \{1, 2, \ldots, n\}$ is called stable if it forms an independent set in the cycle on the vertex set $[n]$. In 1978, Schrijver proved via a topological argument that for all integers n and k with $n \geq 2k$, the family of stable k-subsets of $[n]$ cannot be covered by $n - 2k + 1$ intersecting families. We study two total search problems whose totality relies on this result.

In the first problem, denoted by SCHRIJVER(n, k, m), we are given an access to a coloring of the stable k-subsets of $[n]$ with $m = m(n, k)$ colors, where $m \leq n - 2k + 1$, and the goal is to find a pair of disjoint subsets that are assigned the same color. While for $m = n - 2k + 1$ the problem is known to be PPA-complete, we prove that for $m < d \cdot \lfloor \frac{n}{2k+d-2} \rfloor$, with d being any fixed constant, the problem admits an efficient algorithm. For $m = \lfloor n/2 \rfloor - 2k + 1$, we prove that the problem is efficiently reducible to the KNESER problem. Motivated by the relation between the problems, we investigate the family of *unstable* k-subsets of $[n]$, which might be of independent interest.

In the second problem, called Unfair Independent Set in Cycle, we are given ℓ subsets $V_1, \ldots, V_\ell$ of $[n]$, where $\ell \leq n - 2k + 1$ and $|V_i| \geq 2$ for all $i \in [\ell]$, and the goal is to find a stable k-subset S of $[n]$ satisfying the constraints $|S \cap V_i| \leq |V_i|/2$ for $i \in [\ell]$. We prove that the problem is PPA-complete and that its restriction to instances with $n = 3k$ is at least as hard as the Cycle plus Triangles problem, for which no efficient algorithm is known. On the contrary, we prove that there exists a constant c for which the restriction of the problem to instances with $n \geq c \cdot k$ can be solved in polynomial time.

2012 ACM Subject Classification Mathematics of computing → Graph theory; Mathematics of computing → Combinatorial algorithms; Theory of computation → Problems, reductions and completeness

Keywords and phrases Schrijver graph, Kneser graph, Stable sets, PPA-completeness

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.73

Category Track A: Algorithms, Complexity and Games

Funding *Ishay Haviv*: Research partly supported by the Israel Science Foundation (grant No. 1218/20).

1 Introduction

For integers n and k with $n \geq 2k$, the Kneser graph $K(n, k)$ is the graph whose vertices are all the k-subsets of $[n] = \{1, 2 \ldots, n\}$, where two such sets are adjacent in the graph if they are disjoint. The graph $K(n, k)$ admits a proper vertex coloring with $n - 2k + 2$ colors. This indeed follows by assigning the color i, for each $i \in [n - 2k + 1]$, to all the vertices whose minimal element is i, and the color $n - 2k + 2$ to the remaining vertices, those contained in $[n] \setminus [n - 2k + 1]$. In 1978, Lovász [22] proved, settling a conjecture of Kneser [20], that fewer colors do not suffice, that is, the chromatic number of the graph satisfies $\chi(K(n, k)) = n - 2k + 2$. Soon later, Schrijver [28] strengthened Lovász's result by proving that the subgraph $S(n, k)$ of $K(n, k)$ induced by the stable k-subsets of $[n]$, i.e., the vertices of $K(n, k)$ that form independent sets in the cycle on the vertex set $[n]$, has the same chromatic number. It was further shown in [28] that the graph $S(n, k)$ is vertex-critical, in the sense that any removal of a vertex from the graph decreases its chromatic number.

It is interesting to mention that despite the combinatorial nature of Kneser's conjecture [20], Lovász's proof [22] relies on the Borsuk–Ulam theorem [6], a fundamental result in the area of algebraic topology. Several alternative proofs and extensions were provided in the literature over the years (see, e.g., [24, 25]). Although they are substantially different from each other, they all essentially rely on topological tools.

The computational search problem associated with Kneser graphs, denoted by KNESER, was proposed by Deng, Feng, and Kulkarni [7] and is defined as follows. Its input consists of integers n and k with $n \geq 2k$ and an access to a coloring of the vertices of $K(n,k)$ with $n-2k+1$ colors. The goal is to find a monochromatic edge in the graph, i.e., two disjoint k-subsets of $[n]$ that are assigned the same color by the given coloring. Since the number of colors used by the input coloring is strictly smaller than the chromatic number of $K(n,k)$ [22], it follows that this search problem is total, in the sense that every input is guaranteed to have a solution. Note that the input coloring may be given as an oracle access that provides the color of any queried vertex, and that an algorithm for the problem is considered efficient if its running time is polynomial in n. In other variants of the problem, the input coloring is given by some succinct representation, e.g., a Boolean circuit or an efficient Turing machine. The computational search problem SCHRIJVER is defined similarly, where the input represents a coloring of the vertices of $S(n,k)$ with $n-2k+1$ colors, and the goal is to find a monochromatic edge, whose existence is guaranteed by the aforementioned result of Schrijver [28].

The computational complexity of the SCHRIJVER problem was determined in [15], where it was shown to be complete in the complexity class PPA. This complexity class, introduced in 1994 by Papadimitriou [26], is known to capture the complexity of several additional total search problems whose totality is based on the Borsuk–Ulam theorem, e.g., Consensus Halving, Bisecting Sandwiches, and Splitting Necklaces [12]. Note that this line of PPA-completeness results is motivated not only from the computational complexity perspective, but also from a mathematical point of view, as one may find those results as an indication for the necessity of topological arguments in the existence proof of the solutions of these problems. As for the KNESER problem, it is an open question whether it is also PPA-hard, as was suggested by Deng et al. [7]. We remark that its complexity is related to that of the Agreeable Set problem from the area of resource allocation (see [23, 16]). The KNESER and SCHRIJVER problems were also investigated in the framework of parameterized algorithms [16, 17], where it was shown that they admit randomized fixed-parameter algorithms with respect to the parameter k, namely, algorithms whose running time is $n^{O(1)} \cdot k^{O(k)}$ on input colorings of $K(n,k)$ and $S(n,k)$.

Before turning to our results, let us mention another computational search problem, referred to as the CYCLE-PLUS-TRIANGLES problem. Its input consists of an integer k and a graph on $3k$ vertices, whose edge set is the disjoint union of a Hamilton cycle and k pairwise vertex-disjoint triangles. The goal is to find an independent set of size k in the graph. The existence of a solution for every input of the problem follows from a result of Fleischner and Stiebitz [13], which settled in the early nineties a conjecture of Du, Hsu, and Hwang [9] as well as its strengthening by Erdös [10]. Their proof in fact shows that every such graph is 3-choosable, and thus 3-colorable, so in particular, it contains an independent set of size k. Here, however, the existence of a solution for every input of the problem is known to follow from several different arguments. While the proof of [13] relies on the polynomial method in combinatorics (see also [3]), an elementary proof was given slightly later by Sachs [27], and another proof, based on the chromatic number of $S(n,k)$, was provided quite recently by Aharoni et al. [1]. Yet, none of these proofs is constructive, in the sense that they do not

suggest an efficient algorithm for the Cycle-Plus-Triangles problem. The question of whether the problem admits an efficient algorithm was asked by several authors and is still open (see, e.g., [14, 1, 4]). Interestingly, the approach of [1] implies that the problem is not harder than the restriction of the Schrijver problem to colorings of $S(n,k)$ with $n = 3k$.

1.1 Our Contribution

In this paper, we introduce two total search problems concerned with finding stable sets under certain constraints. The totality of the problems relies on the chromatic number of the graph $S(n,k)$ [28]. We study these problems from algorithmic and computational perspectives. In what follows, we describe the two problems and our results on each of them.

1.1.1 The Generalized Schrijver Problem

We start by considering a generalized version of the Schrijver problem, which allows the number of colors used by the input coloring to be any prescribed number. Let Schrijver(n,k,m) denote the problem which asks to find a monochromatic edge in $S(n,k)$ for an input coloring that uses $m = m(n,k)$ colors. Note that every input of the problem is guaranteed to have a solution whenever $m \leq n - 2k + 1$, and that for $m = n - 2k + 1$, the problem coincides with the standard Schrijver problem.

The Schrijver(n,k,m) problem obviously becomes easier as the number of colors m decreases. For example, it is not difficult to see that for $m = \lfloor n/k \rfloor - 1$, the problem can be solved efficiently, in time polynomial in n. Indeed, the clique number of the graph $S(n,k)$ is $\lfloor n/k \rfloor$, which is strictly larger than m, so by querying the input coloring for the colors of the vertices of a clique of maximum size, one can find two adjacent vertices with the same color. Our first result extends this observation and essentially shows that the Schrijver(n,k,m) problem can be solved efficiently for any number of colors m satisfying $m = O(n/k)$.

▶ **Theorem 1.** *For every integer $d \geq 2$, there exists an algorithm for the* Schrijver(n,k,m) *problem with $m < d \cdot \lfloor \frac{n}{2k+d-2} \rfloor$ whose running time is $n^{O(d)}$.*

Our next result relates the generalized Schrijver(n,k,m) problem to the Kneser problem.

▶ **Theorem 2.** *Schrijver$(n,k,\lfloor n/2 \rfloor - 2k + 1)$ is polynomial-time reducible to Kneser.*

The simple proof of Theorem 2 involves a proper coloring of the subgraph of $K(n,k)$ induced by the *unstable* k-subsets of $[n]$, i.e., the vertices of $K(n,k)$ that do not form vertices of $S(n,k)$. This graph, which we denote by $U(n,k)$, can be properly colored using $\lceil n/2 \rceil$ colors. Indeed, every unstable k-subset of $[n]$ includes an odd element, hence by assigning to each vertex of $U(n,k)$ some odd element that belongs to its set, we obtain a proper coloring of the graph with the desired number of colors. Since $U(n,k)$ is a subgraph of $K(n,k)$, it follows that for all admissible values of n and k, we have $\chi(U(n,k)) \leq \min(n-2k+2, \lceil n/2 \rceil)$.

Motivated by the reduction given by Theorem 2, we further explore the graph $U(n,k)$, whose study may be of independent interest. We prove that the above upper bound on the chromatic number is essentially tight (up to an additive 1 in certain cases; see Corollary 19 and the discussion that follows it). The proof is topological and applies the Borsuk–Ulam theorem. We further determine the independence number of the graph $U(n,k)$ (see Theorem 20), using a structural result of Hilton and Milner [18] on the largest non-trivial intersecting families of k-subsets of $[n]$.

The motivation for Theorem 2 comes from the fact that the SCHRIJVER problem is known to be PPA-hard, whereas no hardness result is known for the KNESER problem. It would be interesting to figure out whether or not the SCHRIJVER(n, k, m) problem with $m = \lfloor n/2 \rfloor - 2k + 1$ admits an efficient algorithm. While this challenge is left open, the following result shows that the problem is not harder than the restriction of the standard SCHRIJVER problem to colorings of $S(n, k)$ with $n = 4k$.

▶ **Theorem 3.** *If there exists a polynomial-time algorithm for the restriction of the SCHRIJVER problem to colorings of $S(n, k)$ with $n = 4k$, then there exists a polynomial-time algorithm for the SCHRIJVER(n, k, m) problem where $m = \lfloor n/2 \rfloor - 2k + 1$.*

We finally observe that the restriction of SCHRIJVER(n, k, m) with $m = \lfloor n/2 \rfloor - 2k + 1$ to instances satisfying $n = \Omega(k^4)$ admits an efficient randomized algorithm. This essentially follows from the fixed-parameter algorithm presented in [17] (see Section 3 for details).

1.1.2 The Unfair Independent Set in Cycle Problem

The second problem studied in this paper is the Unfair Independent Set in Cycle problem, denoted by UNFAIR-IS-CYCLE and defined as follows. Its input consists of two integers n and k with $n \geq 2k$ and ℓ subsets $V_1, \ldots, V_\ell$ of $[n]$, where $\ell \leq n - 2k + 1$ and $|V_i| \geq 2$ for all $i \in [\ell]$. The goal is to find a stable k-subset S of $[n]$ that satisfies the constraints $|S \cap V_i| \leq |V_i|/2$ for $i \in [\ell]$. The name of the problem essentially borrows the terminology of [1], where a set is said to fairly represent a set V_i if it includes at least roughly half of its elements, hence the desired stable set in the UNFAIR-IS-CYCLE problem is required to *unfairly* represent each of the given sets V_i. It is not difficult to show, using the chromatic number of $S(n, k)$, that every input of the UNFAIR-IS-CYCLE problem has a solution (see Lemma 13). Note that the requirement that the input sets satisfy $|V_i| \geq 2$ for all $i \in [\ell]$ is discussed in Section 2.4.

It is natural to compare the definition of the UNFAIR-IS-CYCLE problem to that of the Fair Independent Set in Cycle problem, denoted by FAIR-IS-CYCLE and studied in [15] (see Definition 11). While the goal in the former is to find a stable subset of $[n]$ with a prescribed size k that includes *no more* than half of the elements of each V_i, the goal in the latter is, roughly speaking, to find a stable subset of $[n]$, of an arbitrary size, that includes *at least* half of the elements of each V_i. The specification of the size k in the inputs of UNFAIR-IS-CYCLE makes the problem non-trivial and allows us to study it for various settings of the quantities n and k.

The following result shows that the complexity of the UNFAIR-IS-CYCLE problem is perfectly captured by the class PPA. This is established using the SCHRIJVER and FAIR-IS-CYCLE problems which are PPA-complete [15].

▶ **Theorem 4.** *The UNFAIR-IS-CYCLE problem is* PPA*-complete.*

We next consider some restrictions of the UNFAIR-IS-CYCLE problem to instances in which the integer n is somewhat larger than $2k$. On the one hand, the restriction of the problem to instances with $n = 3k$ is at least as hard as the CYCLE-PLUS-TRIANGLES problem, for which no efficient algorithm is known (see Proposition 15). On the other hand, we prove that on instances whose ratio between n and k is above some absolute constant, the problem can be solved in polynomial time.

▶ **Theorem 5.** *There exists a constant $c > 0$, such that there exists a polynomial-time algorithm for the restriction of the UNFAIR-IS-CYCLE problem to instances with $n \geq c \cdot k$.*

The proof of Theorem 5 is based on a probabilistic argument with alterations, which is derandomized into a deterministic algorithm using the method of conditional expectations (see, e.g., [5, Chapters 3 and 16.1]). The approach is inspired by a probabilistic argument of Kiselev and Kupavskii [19], who proved that for $n \geq (2 + o(1)) \cdot k^2$, every proper coloring of the Kneser graph $K(n, k)$ with $n - 2k + 2$ colors has a trivial color class (all of whose members share a common element).

1.2 Outline

The rest of the paper is organized as follows. In Section 2, we collect some definitions and results that will be used throughout the paper. In Section 3, we study the generalized Schrijver problem and prove Theorems 1, 2, and 3. In Section 4, we study the Unfair-IS-Cycle problem and prove Theorems 4 and 5. Finally, in Section 5, we consider the family of unstable k-subsets of $[n]$ and study the chromatic and independence numbers of the graph $U(n, k)$. Some proofs are omitted and can be found in the full version of this paper.

2 Preliminaries

2.1 Kneser and Schrijver Graphs

For integers n and k, let $\binom{[n]}{k}$ denote the family of all k-subsets of $[n]$. A subset of $[n]$ is called *stable* if it does not include two consecutive elements nor both 1 and n, equivalently, it forms an independent set in the cycle on the vertex set $[n]$ with the natural order along the cycle. Otherwise, the set is called *unstable*. The family of stable k-subsets of $[n]$ is denoted by $\binom{[n]}{k}_{\text{stab}}$. The Kneser graph and the Schrijver graph are defined as follows.

▶ **Definition 6.** *For integers n and k with $n \geq 2k$, the* Kneser graph $K(n, k)$ *is the graph on the vertex set $\binom{[n]}{k}$, where two sets $A, B \in \binom{[n]}{k}$ are adjacent if they satisfy $A \cap B = \emptyset$. The* Schrijver graph $S(n, k)$ *is the subgraph of $K(n, k)$ induced by the vertices of $\binom{[n]}{k}_{\text{stab}}$.*

Obviously, the number of vertices in $K(n, k)$ is $\binom{n}{k}$. The number of vertices in $S(n, k)$ is given by the following lemma (see, e.g., [16, Fact 4.1]).

▶ **Lemma 7.** *For all integers n and k with $n \geq 2k$, the number of stable k-subsets of $[n]$ is $\frac{n}{k} \cdot \binom{n-k-1}{k-1}$.*

As usual, we denote the independence number of a graph G by $\alpha(G)$, and its chromatic number by $\chi(G)$. The chromatic numbers of $K(n, k)$ and $S(n, k)$ were determined, respectively, by Lovász [22] and by Schrijver [28], as stated below.

▶ **Theorem 8** ([22, 28])**.** *For all integers n and k with $n \geq 2k$,*

$$\chi(K(n, k)) = \chi(S(n, k)) = n - 2k + 2.$$

2.2 Intersecting Families

A family $\mathcal{F}$ of sets is called *intersecting* if for every two sets $A, B \in \mathcal{F}$ it holds that $A \cap B \neq \emptyset$. Note that a family of k-subsets of $[n]$ is intersecting if and only if it forms an independent set in the graph $K(n, k)$. An intersecting family $\mathcal{F}$ is said to be *trivial* if there exists an element that belongs to all members of $\mathcal{F}$. Otherwise, the family $\mathcal{F}$ is *non-trivial*. The famous Erdös-Ko-Rado theorem [11] asserts that the largest size of an intersecting family

of k-subsets of $[n]$ is $\binom{n-1}{k-1}$, which is attained by the maximal trivial intersecting families. The following result of Hilton and Milner [18] determines the largest size of a non-trivial intersecting family in this setting and characterizes the extremal families attaining it.

▶ **Theorem 9** (Hilton–Milner Theorem [18]). *For integers $k \geq 3$ and $n \geq 2k$, let $\mathcal{F} \subseteq \binom{[n]}{k}$ be a non-trivial intersecting family. Then,*

$$|\mathcal{F}| \leq \binom{n-1}{k-1} - \binom{n-k-1}{k-1} + 1.$$

Moreover, if $n > 2k$ then equality holds if and only if there exist an element $i \in [n]$ and a k-subset A of $[n]$ with $i \notin A$ such that $\mathcal{F} = \left\{F \in \binom{[n]}{k} \;\middle|\; i \in F,\ F \cap A \neq \emptyset\right\} \cup \{A\}$, or $k = 3$ and there exists a 3-subset A of $[n]$ such that $\mathcal{F} = \left\{F \in \binom{[n]}{3} \;\middle|\; |F \cap A| \geq 2\right\}$.

2.3 Complexity Classes

The complexity class TFNP consists of the total search problems in NP, i.e., the search problems in which every input has a solution, where a solution can be verified in polynomial time. The complexity class PPA (Polynomial Parity Argument [26]) consists of the problems in TFNP that can be reduced in polynomial time to a problem called Leaf. The definition of the Leaf problem is not needed in this paper, but we mention it briefly below for completeness.

The Leaf problem asks, given a graph with maximum degree 2 and a leaf (i.e., a vertex of degree 1), to find another leaf in the graph. The input graph, though, is not given explicitly. Instead, the vertex set of the graph is defined to be $\{0,1\}^n$ for some integer n, and the graph is succinctly represented by a Boolean circuit that for a vertex of the graph computes its (at most two) neighbors. Note that the size of the graph might be exponential in the size of its description.

2.4 Computational Problems

We gather here several computational problems that will be studied and used throughout the paper. We start with a computational search problem associated with Schrijver graphs. This problem is studied in Section 3.

▶ **Definition 10** (Generalized Schrijver Problem). *For $m = m(n,k)$, the Schrijver(n,k,m) problem is defined as follows. The input is a coloring $c : \binom{[n]}{k}_{\text{stab}} \to [m]$ of the vertices of the graph $S(n,k)$ with m colors, and the goal is to find a monochromatic edge, i.e., two vertices $A, B \in \binom{[n]}{k}_{\text{stab}}$ such that $A \cap B = \emptyset$ and $c(A) = c(B)$. In the black-box input model, the coloring c is given as an oracle access that given a vertex A outputs its color $c(A)$. In the white-box input model, the coloring c is given by a Boolean circuit that for a vertex A computes its color $c(A)$. For $m = n - 2k + 1$, the problem Schrijver(n,k,m) is denoted by Schrijver.*

The Kneser problem is defined similarly to the Schrijver problem. Here, the input coloring $c : \binom{[n]}{k} \to [n-2k+1]$ is defined on the entire vertex set of $K(n,k)$. By Theorem 8, every input of the Schrijver and Kneser problems is guaranteed to have a solution. Moreover, whenever $m = m(n,k) \leq n - 2k + 1$, every input of the Schrijver(n,k,m) problem has a solution as well.

We remark that algorithms for the Schrijver(n,k,m) problem are considered in this paper with respect to the black-box input model. The running time of such an algorithm is referred to as polynomial if it is polynomial in n. Observe that a polynomial-time algorithm

for the SCHRIJVER(n, k, m) problem in the black-box input model yields an algorithm for the analogue problem in the white-box input model, whose running time is polynomial as well (in the input size). For computational complexity results, like reductions and PPA-completeness, we adopt the more suitable white-box input model. For example, the SCHRIJVER problem in the white-box input model was shown in [15] to be PPA-complete.

Another search problem studied in [15] is the following.

▶ **Definition 11** (Fair Independent Set in Cycle Problem). *In the FAIR-IS-CYCLE problem, the input consists of integers n and m along with a partition $V_1, \ldots, V_m$ of $[n]$ into m sets. The goal is to find a stable subset S of $[n]$ satisfying $|S \cap V_i| \geq \frac{1}{2} \cdot |V_i| - 1$ for all $i \in [m]$.*

The existence of a solution for every input of the FAIR-IS-CYCLE problem was proved in [1]. It was shown in [15] that the FAIR-IS-CYCLE problem is PPA-complete, even restricted to instances in which each part V_i of the given partition has an odd size larger than 2.

We next define the UNFAIR-IS-CYCLE problem, studied in Section 4.

▶ **Definition 12.** *The input of the UNFAIR-IS-CYCLE problem consists of two integers n and k with $n \geq 2k$ and ℓ subsets $V_1, \ldots, V_\ell$ of $[n]$, where $\ell \leq n - 2k + 1$ and $|V_i| \geq 2$ for all $i \in [\ell]$. The goal is to find a stable k-subset S of $[n]$ that satisfies the constraints $|S \cap V_i| \leq |V_i|/2$ for $i \in [\ell]$.*

Note that Definition 12 requires the sets $V_1, \ldots, V_\ell$ of an instance of the UNFAIR-IS-CYCLE problem to satisfy $|V_i| \geq 2$ for all $i \in [\ell]$. This requirement is justified by the observation that if $|V_i| = 1$ for some $i \in [\ell]$, then any solution for the instance does not include the single element of V_i. Hence, by removing this element from the given sets and from the ground set, such an instance can be reduced to an instance with ground set of size smaller by one. By repeatedly applying this reduction, one can get a "core" instance that fits Definition 12.

We observe that the UNFAIR-IS-CYCLE problem is total. The argument relies on the chromatic number of the graph $S(n, k)$.

▶ **Lemma 13.** *Every instance of the UNFAIR-IS-CYCLE problem has a solution.*

Proof. Consider an instance of the UNFAIR-IS-CYCLE problem, i.e., integers n and k with $n \geq 2k$ and ℓ subsets $V_1, \ldots, V_\ell$ of $[n]$, where $\ell \leq n - 2k + 1$ and $|V_i| \geq 2$ for all $i \in [\ell]$. For every $i \in [\ell]$, let

$$\mathcal{F}_i = \left\{ S \in \binom{[n]}{k}_{\text{stab}} \;\middle|\; |S \cap V_i| > |V_i|/2 \right\},$$

and notice that every two sets of $\mathcal{F}_i$ have a common element of V_i, hence $\mathcal{F}_i$ is an intersecting family. However, by Theorem 8, the chromatic number of $S(n, k)$ is $n - 2k + 2$, hence the family of stable k-subsets of $[n]$ cannot be covered by fewer than $n - 2k + 2$ intersecting families. By $\ell \leq n - 2k + 1$, this implies that there exists a set $S \in \binom{[n]}{k}_{\text{stab}}$ that does not belong to any of the families $\mathcal{F}_i$, hence it satisfies $|S \cap V_i| \leq |V_i|/2$ for all $i \in [\ell]$. This implies that S is a valid solution for the given instance, and we are done. ◀

We end this section with the definition of the CYCLE-PLUS-TRIANGLES problem.

▶ **Definition 14** (Cycle plus Triangles Problem). *In the CYCLE-PLUS-TRIANGLES problem, the input consists of an integer k and a graph G on $3k$ vertices, whose edge set is the disjoint union of a Hamilton cycle and k pairwise vertex-disjoint triangles. The goal is to find an independent set in G of size k.*

The existence of a solution for every input of the CYCLE-PLUS-TRIANGLES problem follows from a result of [13] (see also [27, 1]).

3 The Generalized Schrijver Problem

In this section, we prove our results on the SCHRIJVER(n, k, m) problem (see Definition 10). We start with Theorem 1.

Proof of Theorem 1. Fix some integer $d \geq 2$. For integers n and k with $n \geq 2k$, put $t = \lfloor \frac{n}{2k+d-2} \rfloor$ and $m = d \cdot t - 1$, and consider an instance of the SCHRIJVER(n, k, m) problem, i.e., a coloring $c : \binom{n}{k}_{\text{stab}} \to [m]$ of the vertices of $S(n, k)$. The definition of t allows us to consider t pairwise disjoint subsets $J_1, \ldots, J_t$ of $[n]$, where each of the subsets includes $2k + d - 2$ consecutive elements. For each $i \in [t]$, let $\mathcal{S}_i$ denote the family of all stable k-subsets of J_i with respect to the natural cyclic order of J_i (where the largest element precedes the smallest one), and notice that $\mathcal{S}_i \subseteq \binom{n}{k}_{\text{stab}}$. Consider the algorithm that given an oracle access to a coloring c as above, queries the oracle for the colors of all the sets of $\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_t$, and returns a pair of disjoint sets from this collection that are assigned the same color by c.

For correctness, we show that the collection of sets $\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_t$ necessarily includes two vertices that form a monochromatic edge. Indeed, since the number of colors used by the coloring c does not exceed $d \cdot t - 1$, it follows that either there exist distinct $i, j \in [t]$ for which a vertex of $\mathcal{S}_i$ and a vertex of $\mathcal{S}_j$ have the same color, or there exists an $i \in [t]$ for which the vertices of $\mathcal{S}_i$ are colored using fewer than d colors. For the former case, notice that for distinct i and j, every vertex of $\mathcal{S}_i$ is disjoint from every vertex of $\mathcal{S}_j$, hence the collection includes two vertices that form a monochromatic edge. For the latter case, let $i \in [t]$ be an index for which the vertices of $\mathcal{S}_i$ are colored using fewer than d colors. Observe that the subgraph of $S(n, k)$ induced by $\mathcal{S}_i$ is isomorphic to the graph $S(2k + d - 2, k)$, hence by Theorem 8, its chromatic number is $(2k + d - 2) - 2k + 2 = d$. Since the vertices of $\mathcal{S}_i$ are colored using fewer than d colors, it follows that they include two vertices that form a monochromatic edge, and we are done.

We finally analyze the running time of the algorithm. By Lemma 7, the number of vertices in the graph $S(n, k)$ is

$$\frac{n}{k} \cdot \binom{n-k-1}{k-1} = \frac{n}{k} \cdot \binom{n-k-1}{n-2k} \leq n \cdot (n-k-1)^{n-2k} \leq n^{n-2k+1}.$$

Since the subgraph of $S(n, k)$ induced by each $\mathcal{S}_i$ is isomorphic to $S(2k+d-2, k)$, it follows that the total number of queries that the algorithm makes does not exceed $t \cdot (2k+d-2)^{d-1} \leq n^{O(d)}$. This implies that in running time $n^{O(d)}$, it is possible to enumerate all the sets of $\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_t$, to query the oracle for their colors, and to find the desired monochromatic edge. This completes the proof. ◀

We consider now the SCHRIJVER(n, k, m) problem with $m = \lfloor n/2 \rfloor - 2k + 1$. We first prove Theorem 2 that says that the problem is efficiently reducible to the KNESER problem (whose definition is given in Section 2.4).

Proof of Theorem 2. Put $m = \lfloor n/2 \rfloor - 2k + 1$, and let $c : \binom{[n]}{k}_{\text{stab}} \to [m]$ be an instance of the SCHRIJVER(n, k, m) problem. Consider the reduction that maps such a coloring c to a coloring $c' : \binom{[n]}{k} \to [n - 2k + 1]$ of the vertices of $K(n, k)$ defined as follows. For every set $A \in \binom{[n]}{k}$, if A is unstable then it includes an odd element, so denote its smallest odd element by $2i - 1$, and define $c'(A) = i$. Notice that this i satisfies $1 \leq i \leq \lceil n/2 \rceil$. Otherwise, A is a stable k-subset of $[n]$, and we define $c'(A) = c(A) + \lceil n/2 \rceil$. Notice that $m + \lceil n/2 \rceil = n - 2k + 1$, hence the colors used by c' are all in $[n - 2k + 1]$, as needed for an instance of the KNESER problem. Notice further that given a Boolean circuit that computes the coloring c, it is possible to efficiently produce a Boolean circuit that computes the coloring c'.

For correctness, we simply show that any solution for the produced instance of the Kneser problem is also a solution for the given instance of the Schrijver(n, k, m) problem. To see this, consider a solution for the former, i.e., two disjoint k-subsets A and B of $[n]$ with $c'(A) = c'(B)$. By the definition of c', the color assigned by c' to A and B cannot be some $i \leq \lceil n/2 \rceil$ because this would imply that the element $2i - 1$ belongs to both A and B, which are disjoint. It thus follows that A and B are stable k-subsets of $[n]$ satisfying $c'(A) = c(A)+\lceil n/2 \rceil$ and $c'(B) = c(B)+\lceil n/2 \rceil$. By $c'(A) = c'(B)$, it follows that $c(A) = c(B)$, hence A and B form a monochromatic edge in $S(n, k)$ and thus a solution for the given instance of the Schrijver(n, k, m) problem. This completes the proof. ◀

The reduction presented in the proof of Theorem 2 extends a given coloring of $S(n, k)$ to a coloring of the entire graph $K(n, k)$. To do so, it uses a proper coloring with $\lceil n/2 \rceil$ colors of the subgraph $U(n, k)$ of $K(n, k)$ induced by the *unstable* k-subsets of $[n]$. However, in order to obtain a coloring of $K(n, k)$ with $n-2k+1$ colors, as required for instances of the Kneser problem, one has to reduce from the Schrijver(n, k, m) problem with $m = \lfloor n/2 \rfloor - 2k + 1$. This suggests the question of whether $U(n, k)$ can be properly colored using fewer colors. Motivated by this question, we study some properties of this graph in Section 5, where we essentially answer this question in the negative (see Corollary 19 and the discussion that follows it).

We next show that the Schrijver(n, k, m) problem with $m = \lfloor n/2 \rfloor - 2k + 1$ is not harder than the restriction of the standard Schrijver problem to colorings of $S(n, k)$ with $n = 4k$. This confirms Theorem 3.

Proof of Theorem 3. Suppose that there exists a polynomial-time algorithm, called Algo, for the restriction of the Schrijver problem to colorings of $S(n, k)$ with $n = 4k$. Such an algorithm is able to efficiently find a monochromatic edge in the graph $S(4k, k)$ given an access to a coloring of its vertices with fewer than $\chi(S(4k, k))$ colors. By Theorem 8, it holds that $\chi(S(4k, k)) = 2k+2$. Suppose without loss of generality that the algorithm Algo queries the oracle for the colors of the two vertices of the monochromatic edge that it returns.

For integers n and k with $n \geq 4k$, put $m = \lfloor n/2 \rfloor - 2k + 1$, and let $c : \binom{n}{k}_{\text{stab}} \to [m]$ be an instance of the Schrijver(n, k, m) problem, i.e., a coloring of the vertices of $S(n, k)$ with m colors. We present an algorithm that finds a monochromatic edge in $S(n, k)$. It may be assumed that $n > 8k$. Indeed, otherwise it holds that $m \leq 2k + 1 < \chi(S(4k, k))$, hence a monochromatic edge can be found by running the given algorithm Algo on the restriction of the coloring c to the subgraph of $S(n, k)$ induced by the stable k-subsets of $[4k]$. Since this graph is isomorphic to $S(4k, k)$, Algo is guaranteed to find a monochromatic edge in this subgraph, which also forms a monochromatic edge in the entire graph $S(n, k)$.

Now, put $t = \lfloor \frac{n}{4k} \rfloor$, and let $J_1, \ldots, J_t$ be t pairwise disjoint subsets of $[n]$, where each of the subsets includes $4k$ consecutive elements. For each $i \in [t]$, let $\mathcal{S}_i$ denote the family of all stable k-subsets of J_i with respect to the natural cyclic order of J_i (where the largest element precedes the smallest one). Observe that the subgraph of $S(n, k)$ induced by the vertices of each $\mathcal{S}_i$ is isomorphic to $S(4k, k)$. Observe further that

$$t \cdot (2k + 2) > \left(\frac{n}{4k} - 1\right) \cdot (2k + 2) = \frac{n}{2} + \frac{n}{2k} - 2k - 2 > \left\lfloor \frac{n}{2} \right\rfloor - 2k + 1 = m, \tag{1}$$

where the last inequality holds because $n > 8k$.

Consider the algorithm that given an oracle access to a coloring c as above, for each $i \in [t]$, simulates the algorithm Algo on the restriction of the coloring c to the subgraph of $S(n, k)$ induced by the vertices of $\mathcal{S}_i$. If all the vertices queried throughout the ith simulation have at most $2k + 1$ distinct colors, then the algorithm returns the monochromatic edge

returned by Algo. Otherwise, for each $i \in [t]$, the algorithm uses the queries made in the ith simulation of Algo to produce a set $\mathcal{F}_i \subseteq \mathcal{S}_i$ of $2k+2$ vertices with distinct colors. Then, the algorithm finds a monochromatic edge that involves two vertices of $\mathcal{F}_1 \cup \cdots \cup \mathcal{F}_t$ and returns it. This completes the description of the algorithm. Since the running time of Algo is polynomial, the described algorithm can be implemented in polynomial time.

Let us prove the correctness of the algorithm. Suppose first that for some $i \in [t]$, all the vertices queried throughout the ith simulation of Algo have at most $2k+1$ distinct colors (including the two vertices of the returned edge). In this case, the answers of the oracle in the ith simulation is consistent with some coloring with at most $2k+1$ colors of the subgraph of $S(n,k)$ induced by $\mathcal{S}_i$. Since this graph is isomorphic to $S(4k,k)$, whose chromatic number is $2k+2$, Algo is guaranteed to find in the graph a monochromatic edge, which is also a monochromatic edge in $S(n,k)$, and thus a valid output of the algorithm. Otherwise, for each $i \in [t]$, the attempt to simulate Algo on the subgraph of $S(n,k)$ induced by $\mathcal{S}_i$ provides a set $\mathcal{F}_i$ of $2k+2$ vertices of $\mathcal{S}_i$ with distinct colors. By (1), the total number m of colors used by the coloring c is smaller than $t \cdot (2k+2)$. This implies that there exist distinct indices $i, j \in [t]$ for which a vertex of $\mathcal{F}_i$ and a vertex of $\mathcal{F}_j$ have the same color. Since the vertices of $\mathcal{S}_i$ are disjoint from those of $\mathcal{S}_j$, these two vertices form a monochromatic edge in $S(n,k)$ and form a valid output of the algorithm. ◀

We end this section with the observation that there exists an efficient randomized algorithm for the Schrijver(n,k,m) problem with $m = \lfloor n/2 \rfloor - 2k + 1$ on instances with $n = \Omega(k^4)$. This follows from the paper [17], which yields that for such n and k, the Schrijver(n,k,m) problem is essentially reducible to the Schrijver$(n-1,k,m-1)$ problem in randomized polynomial time (with exponentially small failure probability). By applying this reduction $m-1$ times, it follows that the Schrijver(n,k,m) problem with $m = \lfloor n/2 \rfloor - 2k + 1$ where $n = \Omega(k^4)$, is efficiently reducible to the Schrijver$(\lceil n/2 \rceil + 2k, k, 1)$ problem, which can obviously be solved efficiently.

4 The Unfair Independent Set in Cycle Problem

In this section, we study the Unfair-IS-Cycle problem (see Definition 12).

4.1 Hardness

We prove now Theorem 4, which asserts that Unfair-IS-Cycle is PPA-complete.

Proof of Theorem 4. We first show that the Unfair-IS-Cycle problem belongs to PPA. To do so, we show a polynomial-time reduction to the Schrijver problem in the white-box input model, which lies in PPA [15] (see Definition 10).

Consider an instance of the Unfair-IS-Cycle problem, i.e., integers n and k with $n \geq 2k$ and ℓ subsets $V_1, \ldots, V_\ell$ of $[n]$, where $\ell \leq n - 2k + 1$ and $|V_i| \geq 2$ for all $i \in [\ell]$. For such an instance, the reduction produces a Boolean circuit that given a stable k-subset A of $[n]$, outputs the smallest index $i \in [\ell]$ such that $|A \cap V_i| > |V_i|/2$ if such an i exists, and outputs ℓ otherwise. Note that this circuit represents a coloring $c : \binom{[n]}{k}_{\text{stab}} \to [\ell]$ of the vertices of the graph $S(n,k)$ with $\ell \leq n - 2k + 1$ colors, hence it is an appropriate instance of the Schrijver problem. Clearly, the Boolean circuit that computes c can be constructed in polynomial time.

For correctness, we show that a solution for the constructed Schrijver instance can be used to efficiently find a solution for the given Unfair-IS-Cycle instance. Consider a monochromatic edge of $S(n,k)$, i.e., two disjoint sets $A, B \in \binom{[n]}{k}_{\text{stab}}$ with $c(A) = c(B)$.

Since A and B are disjoint, it is impossible that $|A \cap V_i| > |V_i|/2$ and $|B \cap V_i| > |V_i|/2$ for some $i \in [\ell]$. By the definition of the coloring c, it follows that $c(A) = c(B) = \ell$, hence $|A \cap V_i| \leq |V_i|/2$ and $|B \cap V_i| \leq |V_i|/2$ for all $i \in [\ell - 1]$. Moreover, at least one of A and B intersects V_ℓ at no more than $|V_\ell|/2$ elements, and thus forms a valid solution for the given Unfair-IS-Cycle instance. Since it is possible to check in polynomial time which of the sets A and B satisfies this requirement, the proof of the membership of Unfair-IS-Cycle in PPA is completed.

We next prove that the Unfair-IS-Cycle problem is PPA-hard. To do so, we reduce from the Fair-IS-Cycle problem (see Definition 11). We use here the fact, proved in [15], that this problem is PPA-hard even when it is restricted to the instances in which the parts of the given partition have odd sizes larger than 2. Consider such an instance of the Fair-IS-Cycle problem, i.e., integers n and m along with a partition $V_1, \ldots, V_m$ of $[n]$ such that $|V_i|$ is odd and satisfies $|V_i| \geq 3$ for all $i \in [m]$. Notice that n and m have the same parity, and define $k = \frac{n-m}{2}$. Our reduction simply returns the integers n and k, which clearly satisfy $n \geq 2k$, and the sets $V_1, \ldots, V_m$. Note that $|V_i| \geq 2$ for all $i \in [m]$ and that the number m of sets is $n - 2k$. Since the latter does not exceed $n - 2k + 1$, this is a valid instance of the Unfair-IS-Cycle problem.

For correctness, we show that a solution for the constructed Unfair-IS-Cycle instance is also a solution for the given Fair-IS-Cycle instance. Let S be a solution for the Unfair-IS-Cycle instance, i.e., a stable k-subset of $[n]$ such that for all $i \in [m]$ it holds that $|S \cap V_i| \leq |V_i|/2$. Since the sizes of the sets $V_1, \ldots, V_m$ are odd, it follows that $|S \cap V_i| \leq \frac{|V_i|-1}{2}$ for all $i \in [m]$. Since the sets $V_1, \ldots, V_m$ form a partition of $[n]$, it further follows that

$$|S| = \sum_{i \in [m]} |S \cap V_i| \leq \sum_{i \in [m]} \tfrac{|V_i|-1}{2} = \frac{n-m}{2} = k. \tag{2}$$

However, by $|S| = k$, we derive from (2) that $|S \cap V_i| = \frac{|V_i|-1}{2}$ for all $i \in [m]$. This implies that S is a stable k-subset of $[n]$ satisfying $|S \cap V_i| \geq |V_i|/2 - 1$ for all $i \in [m]$, hence it forms a valid solution for the given Fair-IS-Cycle instance. This completes the proof. ◀

Given the PPA-hardness of the Unfair-IS-Cycle problem, it is interesting to identify the range of the parameters n and k for which the hardness holds. One can verify, using properties of the hard instances constructed in [15], that the hardness given in Theorem 4 holds for instances with $n = (2 + o(1)) \cdot k$, where the $o(1)$ term tends to 0 as n and k tend to infinity. The following simple result shows that for $n = 3k$ the problem is at least as hard as the Cycle-Plus-Triangles problem, whose tractability is an open question (see Definition 14). The proof can be found in the full version of this paper.

▶ **Proposition 15.** *The Cycle-Plus-Triangles problem is polynomial-time reducible to the restriction of the Unfair-IS-Cycle problem to instances that consist of k sets of size 3 that form a partition of $[n]$ where $n = 3k$.*

4.2 Algorithms

We next prove Theorem 5, which states that the Unfair-IS-Cycle problem can be solved efficiently on instances with $n \geq c \cdot k$ for some absolute constant c.

Proof of Theorem 5. We start by presenting a randomized algorithm, based on a probabilistic argument with alterations, and then derandomize it using the method of conditional expectations.

Consider an instance of the Unfair-IS-Cycle problem, i.e., integers n and k with $n \geq 2k$ and ℓ subsets $V_1, \ldots, V_\ell$ of $[n]$, where $\ell \leq n - 2k + 1$ and $|V_i| \geq 2$ for all $i \in [\ell]$. Put $r_i = |V_i| \geq 2$ for each $i \in [\ell]$. Suppose further that $n \geq c \cdot k$ for a sufficiently large constant c to be determined later. Let $p = 2k/n \leq 2/c$, and consider the following randomized algorithm.

1. Pick a random subset A of $[n]$ by including in A every element of $[n]$ independently with probability p.
2. Remove from A every element $j \in [n]$ that satisfies $\{j, j+1\} \subseteq A$ (where for $j = n$, the element $j + 1$ is considered as 1). Let A' denote the obtained set.
3. For every $i \in [\ell]$ that satisfies $|A' \cap V_i| > r_i/2$, remove from A' arbitrary $|A' \cap V_i| - \lfloor r_i/2 \rfloor$ elements of V_i. Let A'' denote the obtained set.
4. If $|A''| \geq k$, then return an arbitrary k-subset of A''. Otherwise, return "failure".

We first claim that unless the algorithm returns "failure", it returns a valid output. Indeed, Item 2 of the algorithm guarantees that the set A' is stable. Further, Item 3 guarantees that its subset A'' satisfies $|A'' \cap V_i| \leq \lfloor r_i/2 \rfloor$ for all $i \in [\ell]$. Therefore, in the case where $|A''| \geq k$, any k-subset of A'' returned in Item 4 of the algorithm is a valid solution for the given Unfair-IS-Cycle instance.

We next estimate the expected size of the set A'' produced by the algorithm. The set A chosen in Item 1 of the algorithm includes every element of $[n]$ with probability p. Hence, its expected size satisfies $\mathbf{E}[|A|] = p \cdot n$. In Item 2 of the algorithm, the probability of every element of $[n]$ to be removed from A is equal to the probability that both the element and its successor modulo n belong to A, which is p^2. By linearity of expectation, this implies that the expected size of the set A' satisfies $\mathbf{E}[|A'|] = (p - p^2) \cdot n$. It remains to estimate the expected number of elements removed from A' in Item 3 of the algorithm. Observe that for each $i \in [\ell]$, the algorithm removes from A' the smallest possible number of elements of V_i ensuring that the obtained set A'' includes at most $\lfloor r_i/2 \rfloor$ of them. Therefore, the number of removed elements of V_i does not exceed the number of subsets of V_i of size $\lfloor r_i/2 \rfloor + 1$ that are contained in A (because it suffices to remove one element from each of them). It thus follows that the expected number of elements of V_i that are removed from A' in Item 3 of the algorithm is at most

$$\binom{r_i}{\lfloor r_i/2 \rfloor + 1} \cdot p^{\lfloor r_i/2 \rfloor + 1} \leq 2^{r_i} \cdot p^{\lfloor r_i/2 \rfloor + 1} \leq (4p)^{\lfloor r_i/2 \rfloor + 1} \leq (4p)^2,$$

where in the last inequality we use the assumption $r_i \geq 2$ and the fact that $p \leq 1/4$ (which holds for any sufficiently large choice of the constant c). It therefore follows, using again the linearity of expectation, that the expected size of A'' satisfies

$$\mathbf{E}[|A''|] \geq (p - p^2) \cdot n - \ell \cdot (4p)^2 \geq (p - 17p^2) \cdot n \geq k,$$

where the second inequality holds by $\ell \leq n$, and the last inequality by the definition of $p = 2k/n$, assuming again that $n \geq c \cdot k$ for a sufficiently large constant c (say, $c = 68$). This implies that there exists a random choice for the presented randomized algorithm for which it returns a valid solution.

We next apply the method of conditional expectations to derandomize the above algorithm. Let us start with a few notations. For a set $S \subseteq [n]$, define

$$f(S) = |S| - |\{j \in [n] \mid \{j, j+1\} \subseteq S\}| - \sum_{i \in [\ell]} \left| \left\{ B \subseteq S \cap V_i \;\middle|\; |B| = \lfloor r_i/2 \rfloor + 1 \right\} \right|. \tag{3}$$

In words, $f(S)$ is determined by subtracting from the size of S the number of pairs of consecutive elements in S (modulo n) as well as the number of subsets of $S \cap V_i$ of size $\lfloor r_i/2 \rfloor + 1$ for each $i \in [\ell]$. For a vector $x \in \{0,1,*\}^n$, let S_x denote a random subset of $[n]$ such that for every $i \in [n]$, if $x_i = 1$ then $i \in S_x$, if $x_i = 0$ then $i \notin S_x$, and if $x_i = *$ then i is chosen to be included in S_x independently with probability $p = 2k/n$. We refer to the vector x as a *partial choice* of a subset of $[n]$. We further define a potential function $\phi : \{0,1,*\}^n \to \mathbb{R}$ that maps every vector $x \in \{0,1,*\}^n$ to the expected value of $f(S)$ where S is chosen according to the distribution S_x, that is, $\phi(x) = \mathbf{E}[f(S_x)]$.

We observe that given a partial choice $x \in \{0,1,*\}^n$, the value of $\phi(x)$ can be calculated efficiently, in time polynomial in n. Indeed, to calculate the expected value of $f(S_x)$, it suffices, by linearity of expectation, to calculate the expected value of each of the three terms in (3) evaluated at the set S_x. It is easy to see that the expected value of the first term is

$$|\{j \in [n] \mid x_j = 1\}| + p \cdot |\{j \in [n] \mid x_j = *\}|,$$

and that the expected value of the second term is

$$\begin{aligned} \left|\left\{j \in [n] \middle| x_j = x_{j+1} = 1\right\}\right| &+ p \cdot \left|\left\{j \in [n] \middle| \{x_j, x_{j+1}\} = \{1,*\}\right\}\right| \\ &+ p^2 \cdot \left|\left\{j \in [n] \middle| x_j = x_{j+1} = *\right\}\right|. \end{aligned}$$

As for the third term, by linearity of expectation, it suffices to determine the expected value of

$$\left|\left\{B \subseteq S_x \cap V_i \;\middle|\; |B| = \lfloor r_i/2 \rfloor + 1\right\}\right|$$

for $i \in [\ell]$. Letting $s_i = |\{j \in V_i \mid x_j = *\}|$ and $t_i = |\{j \in V_i \mid x_j = 1\}|$, one can check that the required expectation is precisely

$$\sum_{m=0}^{\lfloor r_i/2 \rfloor + 1} \binom{s_i}{m} \cdot \binom{t_i}{\lfloor r_i/2 \rfloor + 1 - m} \cdot p^m.$$

Since all the terms can be calculated in time polynomial in n, so can $\phi(x)$.

We describe a deterministic algorithm that finds a set $S \subseteq [n]$ satisfying $f(S) \geq k$. Given such a set, the algorithm is completed by applying Items 2, 3, and 4 of the algorithm presented above. Indeed, by applying Items 2 and 3 we obtain a stable set S'' such that $|S'' \cap V_i| \leq r_i/2$ for all $i \in [\ell]$. The fact that $f(S) \geq k$ guarantees that this set S'' satisfies $|S''| \geq k$, hence Item 4 returns a valid solution.

To obtain the desired set $S \subseteq [n]$ with $f(S) \geq k$, our algorithm maintains a partial choice $x \in \{0,1,*\}^n$ satisfying $\phi(x) \geq k$. We start with $x = (*, \ldots, *)$, for which the analysis of the randomized algorithm guarantees that $\phi(x) \geq k$, provided that $n \geq c \cdot k$ for a sufficiently large constant c. We then choose the entries of x, one by one, to be either 0 or 1. In the ith iteration, in which $x_1, \ldots, x_{i-1} \in \{0,1\}$, the algorithm evaluates ϕ at the two partial choices $x_{i \leftarrow 0} = (x_1, \ldots, x_{i-1}, 0, *, \ldots, *)$ and $x_{i \leftarrow 1} = (x_1, \ldots, x_{i-1}, 1, *, \ldots, *)$, and continues to the next iteration with one of them which maximizes the value of ϕ. By the law of total expectation, it holds that $\phi(x) = p \cdot \phi(x_{i \leftarrow 1}) + (1-p) \cdot \phi(x_{i \leftarrow 0})$, implying that the choice of the algorithm preserves the inequality $\phi(x) \geq k$. At the end of the process, we get a vector $x \in \{0,1\}^n$ with $\phi(x) \geq k$, which fully determines the desired set S with $f(S) \geq k$. Since the evaluations of ϕ can be calculated in time polynomial in n, the algorithm can be implemented in polynomial time. This completes the proof. ◀

Given the above result, it would be interesting to determine the smallest constant c for which the Unfair-IS-Cycle problem can be solved efficiently on instances with $n \geq c \cdot k$. Of particular interest is the restriction of the problem to instances with $n = 3k$ and with pairwise disjoint sets of size 3, because as follows from Proposition 15, an efficient algorithm for this restriction would imply an efficient algorithm for the Cycle-Plus-Triangles problem. Interestingly, it turns out that the restriction of the Unfair-IS-Cycle problem to instances with $n = 4k$ and with pairwise disjoint sets of size 4 does admit an efficient algorithm. This is a consequence of the following result derived from an argument of Alon [2] (see also [4]).

▶ **Proposition 16.** *There exists a polynomial-time algorithm that given an integer k and a partition of $[4k]$ into k subsets $V_1, \ldots, V_k$ with $|V_i| = 4$ for all $i \in [k]$, finds a partition of $[4k]$ into four stable k-subsets S_1, S_2, S_3, S_4 of $[4k]$ such that $|S_j \cap V_i| = 1$ for all $j \in [4]$ and $i \in [k]$.*

5 Unstable Sets

In this section, we consider two subgraphs of the Kneser graph $K(n, k)$ induced by families of unstable k-subsets of $[n]$. These subgraphs are defined as follows.

▶ **Definition 17.** *Let n and k be integers with $n \geq 2k$. Let $\widetilde{U}(n, k)$ denote the subgraph of $K(n, k)$ induced by the family of all k-subsets of $[n]$ that include a pair of consecutive elements (where the elements n and 1 are* not *considered as consecutive for $n > 2$). Let $U(n, k)$ denote the subgraph of $K(n, k)$ induced by the family of all k-subsets of $[n]$ that include a pair of consecutive elements modulo n, i.e., the family of unstable k-subsets of $[n]$.*

5.1 Chromatic Number

We study now the chromatic numbers of the graphs $U(n, k)$ and $\widetilde{U}(n, k)$. It is worth mentioning here that a result of Dol'nikov [8] generalizes the lower bound of Lovász [22] on the chromatic number of $K(n, k)$ to general graphs, using a notion called colorability defect (see also [24, Chapter 3.4] and [21]). This generalization implies a tight lower bound of $n - 2k + 2$ on the chromatic number of $K(n, k)$ and a somewhat weaker lower bound of $n - 4k + 4$ on the chromatic number of $S(n, k)$ (see, e.g., [25]). It turns out, though, that this generalized approach of [8] does not yield any meaningful bounds on the chromatic numbers of the graphs from Definition 17.

The following theorem determines the exact chromatic number of the graph $\widetilde{U}(n, k)$.

▶ **Theorem 18.** *For all integers n and k with $n \geq 2k$,*

$$\chi(\widetilde{U}(n, k)) = \min(n - 2k + 2, \lfloor n/2 \rfloor).$$

The proof of Theorem 18 relies on a topological argument and can be found in the full version of the paper. We derive the following result on the chromatic number of $U(n, k)$.

▶ **Corollary 19.** *For all integers n and k with $n \geq 2k$,*

$$\min(n - 2k + 2, \lfloor n/2 \rfloor) \leq \chi(U(n, k)) \leq \min(n - 2k + 2, \lceil n/2 \rceil).$$

Proof. For the upper bound, apply first Theorem 8 to obtain that

$$\chi(U(n, k)) \leq \chi(K(n, k)) = n - 2k + 2.$$

Next, since every vertex of $U(n,k)$ includes two consecutive elements modulo n, it must include an odd element. By assigning to every such vertex its minimal odd element, we obtain a proper coloring of $U(n,k)$ with $\lceil n/2 \rceil$ colors, hence $\chi(U(n,k)) \leq \lceil n/2 \rceil$. This completes the proof of the upper bound. The lower bound follows by combining Theorem 18 with the fact that $\widetilde{U}(n,k)$ is an induced subgraph of $U(n,k)$. ◀

We conclude this section with a discussion on the tightness of Corollary 19. Notice that the upper and lower bounds provided in Corollary 19 coincide whenever the integer n is even or satisfies $n \leq 4k-4$. For other values of n and k the two bounds differ by 1. Yet, it turns out that the proof technique of Theorem 18 can be used to show that the upper bound in Corollary 19 is tight for all integers n that are congruent to 1 modulo 4. This leaves us with a gap of 1 between the upper and lower bounds in Corollary 19 only for those integers n and k, where n is congruent to 3 modulo 4 and satisfies $n \geq 4k-1$.

We further observe that for an odd integer n and for every proper coloring of $U(n,k)$ that includes a trivial color class (all of whose members share a common element), the number of used colors is at least the upper bound in Corollary 19. Indeed, the restriction of such a coloring to the vertices that do not include the common element of the trivial color class is a proper coloring of a graph isomorphic to $\widetilde{U}(n-1,k)$, so by Theorem 18 it uses at least $\min(n-2k+1,(n-1)/2)$ colors. Together with the additional color of the trivial color class, the total number of colors is at least $\min(n-2k+2,\lceil n/2 \rceil)$, as claimed.

5.2 Independence Number

We next determine the largest size of an independent set in the graph $U(n,k)$. The proof uses the Hilton–Milner theorem (Theorem 9) and can be found in the full version of the paper.

▶ **Theorem 20.** *For all integers $k \geq 2$ and $n \geq 2k$, it holds that*

$$\alpha(U(n,k)) = \binom{n-1}{k-1} - \binom{n-k-1}{k-1}.$$

References

1. Ron Aharoni, Noga Alon, Eli Berger, Maria Chudnovsky, Dani Kotlar, Martin Loebl, and Ran Ziv. Fair representation by independent sets. In M. Loebl, J. Nešetřil, and R. Thomas, editors, *A Journey Through Discrete Mathematics: A Tribute to Jiří Matoušek*, pages 31–58. Springer, 2017.
2. Noga Alon. The strong chromatic number of a graph. *Random Struct. Algorithms*, 3(1):1–8, 1992.
3. Noga Alon. Combinatorial Nullstellensatz. *Combinatorics, Probability and Computing*, 8(1–2):7–29, 1999.
4. Noga Alon. Fair partitions. In A. Nixon and S. Prendiville, editors, *Surveys in Combinatorics 2022*, pages 1–20. Cambridge University Press, 2022.
5. Noga Alon and Joel H. Spencer. *The Probabilistic Method*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, fourth edition, 2016.
6. Karol Borsuk. Drei Sätze über die n-dimensionale euklidische Sphäre. *Fundamenta Mathematicae*, 20(1):177–190, 1933.
7. Xiaotie Deng, Zhe Feng, and Rucha Kulkarni. Octahedral Tucker is PPA-complete. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:118, 2017.
8. Vladimir Doľnikov. Transversals of families of sets. In *Studies in the theory of functions of several real variables (Russian)*, volume 29, pages 30–36. Yaroslavl', 1981.

9 Ding-Zhu Du, D. Frank Hsu, and Frank K. Hwang. The Hamiltonian property of consecutive-d digraphs. *Math. and Computer Modelling*, 17(11):61–63, 1993.

10 Paul Erdös. On some of my favourite problems in graph theory and block designs. *Matematiche*, 45(1):61–73, 1990.

11 Paul Erdös, Chao Ko, and Richard Rado. Intersection theorems for systems of finite sets. *Quart. J. Math.*, 12(1):313–320, 1961.

12 Aris Filos-Ratsikas and Paul W. Goldberg. The complexity of splitting necklaces and bisecting ham sandwiches. In *Proc. of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC'19)*, pages 638–649, 2019.

13 Herbert Fleischner and Michael Stiebitz. A solution to a colouring problem of P. Erdös. *Discret. Math.*, 101(1–3):39–48, 1992.

14 Herbert Fleischner and Michael Stiebitz. Some remarks on the cycle plus triangles problem. In *The Mathematics of Paul Erdös II*, volume 14, pages 136–142. Springer, 1997.

15 Ishay Haviv. The complexity of finding fair independent sets in cycles. *Comput. Complex.*, 31(2):14, 2022. Preliminary version in ITCS'21.

16 Ishay Haviv. A fixed-parameter algorithm for the Kneser problem. In *Proc. of the 49th International Colloquium on Automata, Languages, and Programming (ICALP'22)*, pages 72:1–72:18, 2022.

17 Ishay Haviv. A fixed-parameter algorithm for the Schrijver problem. In *Proc. of the 17th International Symposium on Parameterized and Exact Computation (IPEC'22)*, pages 16:1–16:16, 2022.

18 Anthony J. W. Hilton and Eric Charles Milner. Some intersection theorems for systems of finite sets. *Quart. J. Math.*, 18(1):369–384, 1967.

19 Sergei Kiselev and Andrey Kupavskii. Trivial colors in colorings of Kneser graphs. *arXiv*, abs/2012.14528, 2020. `arXiv:2012.14528`.

20 Martin Kneser. Aufgabe 360. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 58(2):27, 1955.

21 Igor Kriz. Equivariant cohomology and lower bounds for chromatic numbers. *Trans. Amer. Math. Soc.*, 333(2):567–577, 1992.

22 László Lovász. Kneser's conjecture, chromatic number, and homotopy. *J. Comb. Theory, Ser. A*, 25(3):319–324, 1978.

23 Pasin Manurangsi and Warut Suksompong. Computing a small agreeable set of indivisible items. *Artif. Intell.*, 268:96–114, 2019. Preliminary versions in IJCAI'16 and IJCAI'17.

24 Jiří Matoušek. *Using the Borsuk-Ulam Theorem: Lectures on Topological Methods in Combinatorics and Geometry*. Springer Publishing Company, Incorporated, 2007.

25 Jiří Matoušek and Günter M. Ziegler. Topological lower bounds for the chromatic number: A hierarchy. *Jahresbericht der DMV*, 106(2):71–90, 2004.

26 Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994.

27 Horst Sachs. Elementary proof of the cycle-plus-triangles theorem. In *Combinatorics, Paul Erdös is eighty*, volume 1, pages 347–359. Bolyai Soc. Math. Stud., 1993.

28 Alexander Schrijver. Vertex-critical subgraphs of Kneser graphs. *Nieuw Arch. Wiskd.*, 26(3):454–461, 1978.

Faster Submodular Maximization for Several Classes of Matroids

Monika Henzinger
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Paul Liu
Stanford University, CA, USA

Jan Vondrák
Stanford University, CA, USA

Da Wei Zheng
University of Illinois Urbana-Champaign, IL, USA

Abstract

The maximization of submodular functions have found widespread application in areas such as machine learning, combinatorial optimization, and economics, where practitioners often wish to enforce various constraints; the matroid constraint has been investigated extensively due to its algorithmic properties and expressive power. Though tight approximation algorithms for general matroid constraints exist in theory, the running times of such algorithms typically scale quadratically, and are not practical for truly large scale settings. Recent progress has focused on fast algorithms for important classes of matroids given in explicit form. Currently, nearly-linear time algorithms only exist for graphic and partition matroids [12]. In this work, we develop algorithms for monotone submodular maximization constrained by graphic, transversal matroids, or laminar matroids in time near-linear in the size of their representation. Our algorithms achieve an optimal approximation of $1 - 1/e - \varepsilon$ and both generalize and accelerate the results of Ene and Nguyen [12]. In fact, the running time of our algorithm cannot be improved within the fast continuous greedy framework of Badanidiyuru and Vondrák [6].

To achieve near-linear running time, we make use of dynamic data structures that maintain bases with approximate maximum cardinality and weight under certain element updates. These data structures need to support a weight decrease operation and a novel Freeze operation that allows the algorithm to freeze elements (i.e. force to be contained) in its basis regardless of future data structure operations. For the laminar matroid, we present a new dynamic data structure using the top tree interface of Alstrup, Holm, de Lichtenberg, and Thorup [2] that maintains the maximum weight basis under insertions and deletions of elements in $O(\log n)$ time. This data structure needs to support certain subtree query and path update operations that are performed every insertion and deletion that are non-trivial to handle in conjunction. For the transversal matroid the Freeze operation corresponds to requiring the data structure to keep a certain set S of vertices matched, a property that we call S-stability. While there is a large body of work on dynamic matching algorithms, none are S-stable *and* maintain an approximate maximum weight matching under vertex updates. We give the first such algorithm for bipartite graphs with total running time linear (up to log factors) in the number of edges.

2012 ACM Subject Classification Theory of computation → Data structures design and analysis; Mathematics of computing → Submodular optimization and polymatroids; Theory of computation → Dynamic graph algorithms

Keywords and phrases submodular optimization, dynamic data structures, matching algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.74

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2305.00122

Funding *Monika Henzinger*: This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101019564 "The Design of Modern Fully Dynamic Data Structures (MoDynStruct)" and from the Austrian Science Fund (FWF) project "Static and Dynamic Hierarchical Graph Decompositions", I 5982-N, and project "Fast Algorithms for a Reactive Network Layer (ReactNet)", P 33775-N, with additional funding from the *netidee SCIENCE Stiftung*, 2020–2024.
Jan Vondrák: Supported by NSF Award 2127781.

1 Introduction

Submodular optimization is encountered in a variety of applications – combinatorial optimization, information retrieval, and machine learning, to name a few [7]. Many such applications involve constraints, which are often in the form of cardinality or weight constraints on certain subsets of elements, or combinatorial constraints such as connectivity or matching. A convenient abstraction which has been studied heavily in this context is that of a *matroid constraint*[1]. For instance, transversal matroids appear in ad placement and matching applications [4, 3], laminar and partition matroids capture capacity constraints on subsets which are widely used in recommendation settings (e.g. YouTube video recommendation algorithm [25]), and graphic matroids appear in applications for approximating Metric TSP [26]. Maximization of a submodular function subject to any of these constraints is an APX-hard problem, but a $(1 - 1/e)$-approximation is known in this setting for any matroid constraint [11], and the factor of $1 - 1/e$ is also known to be optimal [22, 13]. Considering this, it has been a long-standing quest to develop fast algorithms for the submodular maximization problem that achieve an approximation close to the optimal factor of $1 - 1/e$. In this work, we achieve this goal for several common classes of matroids.

Perhaps the first step in this direction was the *threshold-greedy technique* which gives a fast $(1/2 - \varepsilon)$-approximation [5] for the cardinality constraint. With more work, this technique can be extended to give approximations close to $1 - 1/e$ [6] and ultimately a $(1-1/e-\varepsilon)$-approximation in running time $O(n/\varepsilon)$ was found for the cardinality constraint.[2]

For general matroids, the fastest known algorithm is the *fast continuous greedy algorithm*, which uses $O(nr\varepsilon^{-4}\log^2(n/\varepsilon))$ oracle, where n is the number of elements in the matroid and r is the rank of the matroid [6]. (The exact running time would depend on the implementation of these queries, which [6] does not address.) We can assume that the rank r scales polynomially with n and hence this algorithm is not near-linear. Further work on fast submodular optimization developed in the direction of parallelized and distributed settings (see [18, 20, 21] and the references therein); we do not discuss these directions in this paper.

A recent line of work initiated by Ene and Nguyen [12] attempts to develop a "nearly-linear" continuous greedy algorithm, i.e., with a running time of $n \cdot \mathrm{poly}(1/\varepsilon, \log n)$. They achieved this goal for partition and graphic matroids. Prior to their work, the fastest known algorithm for any matroid class beyond cardinality was the work of Buchbinder, Feldman, and Schwartz [10], who showed an $O(n^{3/2})$-time algorithm for partition matroids.

This immediately leads to the question of whether such improvements are also possible for other classes of matroid constraints. As observed by Ene and Nguyen [12], even determining feasibility may take longer than linear time for certain matroids. One such example is for

[1] A matroid on N is a family of "independent sets" $\mathcal{I} \subset 2^N$ which is down-closed, and satisfies the extension axiom: For any $A, B \in \mathcal{I}$, if $|A| < |B|$ then there is an element $e \in B \setminus A$ such that $A \cup \{e\} \in \mathcal{I}$.
[2] Running time in this paper includes value queries to the objective function $f(S)$ as unit-time operations.

matroids represented by linear systems, as simply checking the independence of a linear system takes $O(\text{rank}(\mathcal{M})^\omega)$ time, where $\text{rank}(\mathcal{M})$ is the rank of the linear system and ω is the exponent for fast matrix multiplication.

1.1 Our contributions

In this paper, we generalize and significantly improve the work of Ene and Nguyen [12], to develop a $(1 - 1/e - \varepsilon)$-approximation for maximizing a monotone submodular function subject to a matroid constraint, for several important classes of matroids: namely for graphic, laminar, and transversal matroids. The technical developments behind these results are on two fronts:
(i) a refinement of the optimization framework of [12] and formulation of abstract data structures required for this framework;
(ii) implementation of such data structures for graphic, laminar and transversal matroids. (See Section 2 for definitions of these classes of matroids.)

To describe our results in more detail, the efficiency of optimizing a submodular function f can be broken down into two components: the number of oracle calls to f, and the number of additional arithmetic operations needed to support the algorithm optimizing f. The number of oracle calls to f that our framework need to achieve a $(1-1/e-\varepsilon)$-approximation is $O_\varepsilon(n \log^2(n))$ regardless of the matroid, where n is the number of elements in the matroid constraint. Thus for all results below, the running time is measured in the number of the number of arithmetic operations needed by the data structures supporting the optimization of f. Our contributions are as follows:

- We give nearly-linear time versions of continuous greedy for laminar, graphic, and transversal matroids. These algorithms are accelerated by special data structures we developed for each matroid and which might be of independent interest. For all of our matroids, it is impossible to improve our running time without improving the continuous greedy algorithm itself.
- For graphic matroids on n vertices and m edges, we improve the running time of [12] from $O_\varepsilon(n \log^5 n + m \log^2 n)$ to $O_\varepsilon(m \log^2 n)$.
- We generalize the partition matroids results of [12] to laminar matroids, and match their running time of $O_\varepsilon(n \log^2 n)$ for continuous greedy. As a by-product, we also develop the first data structure that maintains the maximum weight basis on a laminar matroid with $O(\log n)$ update time for insertions and deletions that may be of independent interest. This data structure uses the top tree interface of Alstrup, Holm, de Lichtenberg, and Thorup [2].
- For transversal matroids represented by a bipartite graph with m edges and the ground set being one side of the partition with n vertices, we give an algorithm running in $O_\varepsilon(m \log n + n \log^2 n)$ time.[3] This is the first such fast algorithm for transversal matroids. For this we develop a dynamic matching algorithm in a vertex-weighted, bipartite graph with a weighted vertex sets L and an unweighted vertex set R with the following conditions: (i) There exists a dynamically changing set $S \subseteq L$ such that every vertex in S must be matched in the current matching. (ii) The matching must give both an approximation in terms of cardinality in comparison to the maximum cardinality matching *as well as* the weight of the matching in comparison to the best matching that matches all vertices of S, where the weight of the matching is the sum of the weights of the matched vertices of L.

[3] Any transversal matroid with n elements can be represented as the family of matchable sets the left-hand side of an $n \times n^2$ bipartite graph, with degrees at most n. See Section 2 for more details.

We emphasize that our results are true running times, as opposed to black-box independence queries to the matroid. The only black box operation we need is the query to the objective function $f(S)$.

The performance of the dynamic matching data structure used in our transversal matroid algorithm is interesting as none of the earlier work on dynamic matching can maintain both a constant approximation to the weight as well as to the cardinality of the matching. Our algorithm builds on a recent fast algorithm for maximum-weight matching by Zheng and Henzinger [27]. We briefly mention a few relevant works:

- There is a conditional lower bound based on the OMv conjecture [16] that shows that maintaining a *maximum weight matching* in an edge weighted bipartite graph with only *edge weight increase operations* cannot be done in amortized time $O(n^{1-\delta})$ per edge weight increase and amortized time $O(n^{2-\delta})$ per query operation for any $\delta > 0$ [17]. The reduction from [17] can be adapted to the setting with only *edge weight decrement* operations, achieving the same lower bound. Thus, this shows that our running time bound cannot achieved if a *exact* maximum weight matching has to be maintained under edge weight decrement operations.
- Le, Milenkovic, Solomon, and Vassilevska-Williams [19] studied one-sided vertex updates (insertions and deletions) in bipartite graphs and gave a *maximal matching* algorithm whose total running time is $O(K)$, where K is the total number of inserted and deleted *edges*. Bosek el al. [8] studied one-sided vertex updates (either insertions-only or deletions-only) in bipartite graphs and gave algorithms that for any $\varepsilon > 0$ maintain a $(1-\varepsilon)$-approximate maximum cardinality matching in total time $O(m/\varepsilon)$. Gupta and Peng [15] developed the best known dynamic algorithm that allows both *edge* insertions and deletions and maintains a $(1-\varepsilon)$-approximate maximum cardinality matching (for any $\varepsilon > 0$). It requires time $O(\sqrt{m}/\varepsilon)$ amortized time per operation. For all these algorithms, they either cannot be extended to the weighted setting, or cannot maintain both a constant approximation to the weight as well as to the cardinality of the matching.

1.2 Technical Overview

The submodular optimization framework. Our framework is an adaptation and improvement over that of Ene and Nguyen [12]. The framework consists of two phases:

(1) a LazySamplingGreedy phase, which aims to build a partial solution that either provides a good approximation on its own, or reduces the problem to a residual instance with bounded marginal values of elements.

(2) a ContinuousGreedy phase, which is essentially the original fast continuous greedy algorithm [11, 6], with an improved analysis based on the fact the marginal values are bounded.

The original fast ContinuousGreedy runs in time $O_\varepsilon(nr\log^2 n)$, where the factor of $r\log^2 n$ is due to the cost in evaluating the multilinear extension of f. The multilinear extension is an average over values of f on randomly drawn subsets of the input. In ContinuousGreedy, $O(r\log^2 n)$ samples are needed to estimate the multilinear extension to sufficient accuracy.

The LazySamplingGreedy phase transforms f into a function $\tilde{f}$ such that (a) the number of samples required in ContinuousGreedy for $\tilde{f}$ is reduced by a factor of r and (b) the optimal solution of $\tilde{f}$ is within a $(1-\varepsilon)$ factor of the optimal solution f. LazySamplingGreedy does this by constructing an initial independent set S in our matroid $\mathcal{M}$ such that $\tilde{f}(T) = f(T \cup S)$ has relatively small marginal values. The final solution is obtained by running ContinuousGreedy on $\tilde{f}$ constrained by $\mathcal{M} \setminus S$ and

combining the solution with S. The construction of S relies on a fast data structure to get the maximum weight independent set of $\mathcal{M}$ at any given time, subject to weight changes on each element.

We begin by simplifying and improving the LazySamplingGreedy phase (Section 3.2). A significant part of LazySamplingGreedy in [12] is dedicated to randomly checking and refreshing estimates of marginal values for each element in the matroid. We show that (a) this random checking can be dramatically reduced, and (b) the maximum-weight independent set requirement for the data structure can be relaxed to a constant-factor approximation of the maximum-weight independent set. The relaxation to constant-factor approximations enables us to design much more efficient data structures than the previous work of [12], and also allows us to handle more classes of matroids.

In the ContinuousGreedy phase (Section 3.3), a subroutine requires an independence oracle to check if proposed candidate solutions are independent sets of the matroid. Ene and Nguyen use fully dynamic independence oracles to implement this, where independence queries require $O(\text{polylog } n)$ time. We show that only incremental independence oracles are needed. This opens the door to implementing such oracles for new classes of matroids, as sublinear fully dynamic independence oracles are provably hard in many settings (e.g. bipartite maximum-cardinality matchings are hard to maintain exactly in faster than $O(n^{2-\varepsilon})$ amortized time per update [1], which corresponds to finding the maximum independent set in a transversal matroid).

Dynamic data structures for various matroid constraints. From a data structures point of view, we give the first efficient data structure for two settings. (We refer the reader to Section 2 for definitions of laminar, graphic and transversal matroids.)

Maximum-weight basis in a laminar matroid. In a laminar matroid with n elements we are able to output the maximum-weight basis under an online sequence of insertions and deletions of weighted elements with $O(\log n)$ time per update, which we show to be worst-case optimal. The biggest challenge for laminar matroids is that each element may have as many as $O(n)$ constraints that need to be kept track of on each insertion and deletion. We leverage the tree structure induced by a laminar set system to build data structures. Specifically, we show there exists a data structure with $O(\log^2 n)$ query time using the heavy-light decomposition technique of [24]. Since the heavy-light decomposition technique decomposes the tree into paths, we store some carefully chosen auxiliary information at each vertex to transform our subtree queries into path queries. We further improve this to $O(\log n)$ using the top tree interface of Alstrup, Holm, de Lichtenberg, and Thorup [2] that more naturally support both path and subtree operations using a similar idea of carefully storing the right auxiliary information at each top tree node combined with lazily propagating path changes.

Approximate maximum-weight basis in a transversal matroid. In the case of transversal matroids, our LazySamplingGreedy+ algorithm requires what we call a (c, d)-*approximate maximum weight matching oracle*: a matching that is an c-approximation in weight and at least d of the cardinality of the maximum cardinality matching. In addition, the oracle must implement two update operations, (a) a Freeze operation that adds a vertex of L to S and (b) a Decrement operation that reduces the weight of a vertex of L to a given value. We give a novel algorithm that maintains a maximal (inclusion-wise) matching M in a weighted graph such that the weight of the matching is a $(1-\varepsilon)$-approximation of the max-weight solution in total time $O(m(1/\varepsilon + \log n))$. Thus our algorithm is a $(1-\varepsilon, 1/2)$-approximate maximum weight matching oracle. Due to standard rescaling techniques we can assume that the maximum weight $W = O(n)$.

To illustrate the challenges introduced by FREEZE operations, assume we want to maintain a $(1/2, 1/2)$-approximate maximum weight matching oracle and let $n \geq 5$ be an odd integer. Consider a graph consisting of the path $\ell_0, r_0, \ell_1, r_2, \ldots, \ell_{(n+1)/2}$ of length $n-1$ where the first and last edge have large weight $W<$ say $W = 100n$, and all other edges have weight 1. If the initial $(\Omega(1), \Omega(1))$-approximate matching has greedily picked every second edge, it achieves a weight of $W + \frac{n-1}{2} - 1$ versus an optimum weight of $2W + \frac{n-1}{2} - 2$. Now if the weight of the first edge is halved, the weight of the computed solution drops to $W/2 + \frac{n-1}{2} - 1$, which is no longer a 2-approximation of the weight of the optimum solution (for large enough W). Thus the algorithm needs to match the last edge in order to maintain a 2-approximation to the weight. This would only require two changes to the matching, namely un-matching the edge $(\ell_{(n-1)/2}, r_{(n-1)/2})$ and matching the last edge. However, if prior FREEZE operations added the vertices $\ell_1, \ell_2, \ldots \ell_{(n-1)/2}$ to S, i.e. $S = V \setminus \{\ell_0\}$, then we cannot un-match $\ell_{(n-1)/2}$. Thus, the edge $(\ell_{(n-1)/2}, r_{(n-3)/2})$ needs to be unmatched, which in turn un-matches the vertex $\ell_{(n-3)/2}$, leading to further un-matchings and matchings along the path. More specifically, all $(n-1)/2$ matched edges need to change. Next, the weight of the last edge is divided by 4 and the process along the path starts again. Thus, due to the prior FREEZE operations, each DECREMENT operation can lead to $\Theta(n) = \Theta(m)$ changes in the graph. As this can be repeated $\Theta(\log W) = \Theta(\log n)$ times it shows that work $\Omega(m \log n)$ is unavoidable if a 2-approximation to the weight is maintained with FREEZE operations. This is the running time that we achieve.

More precisely, for any $\varepsilon > 0$, we give an $(1-\varepsilon, 1/2)$-approximate maximum weight matching oracle under FREEZE and DECREMENT operations with total time $O(m(\log n + 1/\varepsilon))$. It follows that $O(m(\log n + 1/\varepsilon))$ is also an upper bound on the number of matching and un-matching operations of the algorithm. To do so we extend a recent algorithm [27] for fast $(1-\varepsilon)$-approximate maximum weight matching algorithm in bipartite graphs based on the multiplicative weight update idea. The analysis within [27] shows stability properties that makes the FREEZE operation trivial to implement. However, LAZYSAMPLINGGREEDY+ requires that the maintained matching is at least $1/2$ the size of the maximum matching. Furthermore, we need to support the DECREMENT operation. We show that both extensions are possible and obtain an algorithm with total time (for all operations) of $O(m(1/\varepsilon + \log n))$.

2 Preliminaries

Set notation shorthands. Given a set $S \subseteq \mathcal{N}$ and an element $u \in \mathcal{N}$, we denote $S \cup \{u\}$ and $S \setminus \{u\}$ by $S + u$ and $S - u$ respectively. Similarly, given sets $S, T \subseteq \mathcal{N}$, we denote $S \cup T$ and $S \setminus T$ by $S + T$ and $S - T$ respectively.

Submodular functions. Given a set $\mathcal{N}$, a set function $f \colon 2^{\mathcal{N}} \to \mathbb{R}$ is called *submodular* if for any two sets S and T we have

$$f(S) + f(T) \geq f(S + T) + f(S - T).$$

We only consider monotone submodular functions, where $f(S) \leq f(T)$ for any sets $S \subseteq T$.

Matroids. A set system is a pair $\mathcal{M} = (\mathcal{N}, \mathcal{I})$, where $\mathcal{I} \subseteq 2^{\mathcal{N}}$. We say that a set $S \subseteq \mathcal{N}$ is *independent* in $\mathcal{M}$ if $S \in \mathcal{I}$. The *rank* of the set system M is defined as the maximum size of an independent set in it. The independent sets must satisfy (i) $S \subseteq T, T \in \mathcal{I} \implies S \in \mathcal{T}$, and (ii) $S, T \in I, |S| < |T| \implies \exists e \in T \setminus S$ such that $S + e \in \mathcal{I}$.

A matroid constraint means that f is optimized over the independent sets of a matroid.

Graphic matroids. Let $G = (V, E)$ be a graph. A graphic matroid has $\mathcal{N} = E$ and $\mathcal{I}$ equal to the forests of G. The rank of $\mathcal{M}$ is $|V| - C$ where C is the number of connected components in G.

Laminar matroids. Let $\{S_1, S_2, \ldots, S_m\}$ be a collection of subsets of a set $\mathcal{N}$ such that for any two intersecting sets S_i and S_j where $i \neq j$, either $S_i \subset S_j$ or $S_j \subset S_i$. Furthermore, let there be non-negative integers $\{c_1, c_2, \ldots, c_m\}$ associated with the S_i's. Let $\mathcal{I}$ be the sets $S \subseteq \mathcal{N}$ for which $|S \cap S_i| \leq c_i$ for all i. $\mathcal{I}$ is then the collection of independent sets in a laminar matroid on $\mathcal{N}$. A laminar matroid has a natural representation as a tree on m nodes, which we describe in the full version.

Transversal matroids. Let $G = ((L, R), E)$ be a bipartite graph with a bipartition of vertices (L, R). Let $\mathcal{I}$ be the collection of sets $S \subseteq L$ such that there is a matching of all vertices in S to $|S|$ vertices in R. A transversal matroid has $\mathcal{N} = L$ and $\mathcal{I}$ as its independent sets. From the definition, it is clear that $\text{rank}(\mathcal{M})$ is the size of the maximum cardinality matching in G. It is known that every transveral matroid can be represented by a bipartite graph where L is the ground set of the matroid and $|R| = \text{rank}(\mathcal{M})$ (see [23], Volume B, equation (39.18)).

3 Improved nearly-linear submodular maximization

In what follows, f is the submodular function we want to maximize, $\mathcal{M}$ is the matroid constraint, n is the number of elements in $\mathcal{M}$, and OPT is the optimal independent set. Additionally, we can assume that $\text{rank}(\mathcal{M}) = \omega(\log n)$, as the standard ContinuousGreedy would run in $O(n \operatorname{polylog} n)$ time otherwise.

Our high-level framework adapts and improves upon the nearly-linear time framework of Ene and Nguyen [12]. We will do the following:

Algorithm 3.1: Overall Framework.

1. Run LazySamplingGreedy+ (see below), to obtain a partial solution S_0.
2. Run ContinuousGreedy on $\tilde{f}(T) = f(S_0 \cup T) - f(S_0)$ with the constraint $\mathcal{M}/S_0$, to obtain a solution S_1.
3. Return $S_0 \cup S_1$.

As previously discussed, the original ContinuousGreedy runs in time $O_\varepsilon(nr \log^2 n)$, where the $r \log^2 n$ is due to the number of samples needed to evaluate the multilinear extension of f. LazySamplingGreedy+ finds a set S_0 such that $\tilde{f}(T) := f(T \mid S_0) = f(S_0 \cup T) - f(S_0)$ has a tighter range of marginal values. This allows us to reduce the number of samples used in ContinuousGreedy by a factor of r.

The overall idea is to run LazySamplingGreedy+ until the marginals in $\tilde{f}$ are small enough to guarantee good performance in the ContinuousGreedy phase of our overall framework (Algorithm 3.1). To accelerate our algorithms, we construct specialized data structures for both LazySamplingGreedy+ and ContinuousGreedy.

3.1 Data structure requirements

We next describe the data structures needed for the two phases of our algorithm. In the LazySamplingGreedy+ phase we need a c-approximate dynamic max-weight independent set oracle. In the CountinuousGreedy phase we have two options of dynamic independence

oracles, both of them unweighted. In addition, our LazySamplingGreedy+ requires the ability to obtain a weighted sample from the approximate max-weight independent set. Since we use these data structures as subroutines in our static algorithm which uses the answers of the data structure to determine future updates, it is important that their running time bounds are valid against an *adaptive* adversary.

Dynamic (c, d)-approximate maximum weight oracle. Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid. Given an independent set $S \subseteq E$ the *independent sets relative to* S are the independent sets of $\mathcal{M}$ that contain S. Let $rank(\mathcal{M})$ denote the size of the largest independent set, which equals the size of the largest independent set containing S. The weight of an independent set is the sum of the weights of its elements. A *maximum weight basis in* $\mathcal{M}$ *relative to* S is a basis B^* that maximizes the sum of w_e over all bases of $\mathcal{M}$ that contain S.

Let $c < 1$ and $d < 1$ be constants. An independent set B is called an (c, d)-*approximate independent set relative to* S if it fulfills the following conditions: (a) its size is at least $\text{rank}(\mathcal{M}) \cdot d$ and (b) its weight is at least a c-approximation to the weight of a maximum weight basis relative to S.

We study the dynamic setting where each element $e \in E$ has a dynamically changing weight $w_e \in \mathbb{R}^+$ and where S is a dynamically growing subset of E. A (c, d)-*approximate dynamic maximum weight oracle* is a data structure which maintains a (c, d)-approximate independent set B relative to S (i.e. in the matroid $\mathcal{M}/S$) while S and the weight of elements not in S can change. Initially S is an empty set and the data structure supports the following operations:

- Freeze(e): Add to S the element e, where e must belong to the current (c, d)-approximate basis relative to (the old) S and return the changes to B.
- Decrement(e, w): Return the weight w_e of $e \notin S$ to w, which is guaranteed to be smaller than the current weight of e and return the changes to B.
- ApproxBaseWeight(): Return the weight of the (c, d)-approximate independent set maintained by this data structure.

If $c = 1$ and $d = 1$ we call such a data structure a *dynamic maximum weight oracle relative to* S.

We will use (c, d)-approximate maximum weight oracles in the LazySamplingGreedy+ phase of the algorithm.

We will also need to augment this data structure with two additional sampling operations. Whenever the independent set B maintained by the data structure changes, we need to spend an extra $O(1)$ time updating a sampling data structure. This sampling data structure can be generically and efficiently implemented to augment any (c, d)-approximate maximum weight oracle as long as the (c, d)-approximate maximum weight oracle does not change the independent set B too much amortized over all calls to the data structure. This is described in the full version of the paper.

- Sample(t): Return a subset of $B \setminus S$, where each element is included independently with probability $\min\left(1, \frac{t}{w(B \setminus S)} w_e\right)$.
- UniformSample(): Return a uniformly random element from $B \setminus S$.

$(1 - \varepsilon)$-approximate independence oracles. For the second phase of our algorithm ContinuousGreedy we have a choice between two data structures. Both of them are unweighted, i.e., elements have no associated weights. We can either use an incremental (i.e. insertions-only) exact data structure or a dynamic $(1 - \varepsilon)$-approximate data structure, for a small $\varepsilon > 0$. Next we define both in more details.

Incremental independence oracle. The incremental independence oracle data structure maintains an independent set B and supports the following operation:

- Test(e): Given an element e, decide if $B \cup \{e\}$ is independent. If so, output YES, otherwise output NO.
- Insert(e): Given an element e such that $B \cup \{e\}$ is independent, add e to B.

$(1-\varepsilon)$-approximate dynamic maximum independent set data structure. Let $\varepsilon > 0$ be a small constant and let us call an independent set B of a matroid $\mathcal{M}$ that contains at least $(1-\varepsilon)\cdot\text{rank}(\mathcal{M})$ elements an $(1-\varepsilon)$-*approximate basis* of $\mathcal{M}$. The $(1-\varepsilon)$-approximate data structure maintains an $(1-\varepsilon)$-approximate basis B for a dynamically changing matroid $\mathcal{M}$ and supports the following operations.

- Batch-Insert(E'): Given a set E' of new elements, insert all elements of E' into the matroid $\mathcal{M}$ and compute a new $(1-\varepsilon)$-approximate basis B such that all elements that were in the basis before the update belong to B. Return all new elements that were introduced to B.
- Delete(e): Given an element e of $\mathcal{M}$, delete e from $\mathcal{M}$ and update the independent set B such that it consists of at least $(1-\varepsilon)\cdot\text{rank}(\mathcal{M})$ elements of the new $\mathcal{M}$. If any new elements were added to B, return this set of new elements. Otherwise, return $\emptyset$.

Depending on which version of the algorithm we use, we will need either an exact incremental oracle or a $(1-\varepsilon)$-approximate dynamic maximum independent set data structure.

3.2 The LazySamplingGreedy+ algorithm

In this section, we describe the implementation of LazySamplingGreedy+.

The LazySamplingGreedy+ algorithm is inspired by the Random Greedy algorithm of Buchbinder, Feldman, Naor, and Schwartz [9] and the Lazy Sampling Greedy algorithm of Ene and Nguyen [12]. The algorithm begins with an initially empty solution S, and runs until the function $\tilde{f}(T) = f(T|S)$ has small enough marginals to reduce the sampling requirements of ContinuousGreedy.

We denote the weight of an element by $w_e(S) := f(S \cup \{e\}) - f(S)$, and weight$(T)$ to denote $\sum_{e\in T} w_e(S)$. The algorithm will only ever add elements to S, so by submodularity, $w_e(S)$ can only decrease as the algorithm runs (satisfying the requirements of Section 3.1). Throughout this algorithm, we use a (c,d)-approximate maximum-weight oracle (Section 3.1) that maintains a maximum-weight independent set B as the weights $w_e(S)$ are updated. For the sake of exposition, we defer proofs to the full version of the paper and assume $c \geq 1/2$, and $d \geq 1/2$.

Discretizing the marginal weights. Whenever S is changed, the weight $w_e(S)$ of all elements e can be changed. To reduce the number of weight changes, we use a standard rounding trick. Assume we have some constant-factor approximation M to $f(OPT)$ (which can be computed in $O(n)$ time via well-known algorithms [10]). Instead of maintaining $w_e(S)$ exactly, we round $w_e(S)$ to one of logarithmically many weight classes, that is, $w_e(S)$ belongs to weight class j if $w_e(S) \in ((1-\varepsilon)^{j+1}M, (1-\varepsilon)^j M]$, with the lowest class containing all weights from $[0, O(\varepsilon M/r)]$. The value of the rounded weight is then $\tilde{w}_e = (1-\varepsilon)^{j_e}$. We denote by *bucket* $\mathcal{B}^{(j)}$ all elements that belong to weight class j. Throughout the algorithm, we maintain estimates $\tilde{j_e}$ for the weight class that e is in (and thus estimates of w_e as well). An estimate is called *stale* if $w_e(S)$ is not actually in the weight class indicated by $\tilde{j_e}$. To achieve a multiplicative error of $(1-\varepsilon)$, it suffices for the number of different weight classes to be at most $O(\varepsilon^{-1}\log(r/\varepsilon))$, where r is the rank of the matroid. We denote by weight(B) the sum of current weight estimates over the set B.

The analysis of our algorithm works with any constant-factor approximation to $f(OPT)$ and any constant c-approximate maximum weight independent set data structure, albeit with slight changes in the approximation factors.

Algorithm 3.2: LazySamplingGreedy+.

$S \leftarrow \emptyset$, and set the weight estimate $\tilde{w}_e$ to $w_e(\emptyset)$ for all $e \in \mathcal{M}$.
$\mathcal{D} \leftarrow (c, d)$-approximate dynamic maximum weight oracle on $\mathcal{M}$ and $\tilde{w}$.

While $\mathcal{D}$.ApproxBaseWeight() $\geq \frac{50}{\varepsilon} f(OPT)$:

1. $B' \leftarrow \mathcal{D}$.Sample($128 \log n$)
   (a random subset of $B \setminus S$, each element included independently with probability $p_e = \min\{1, \frac{128 \log n}{\tilde{w}(B \setminus S)} \tilde{w}_e\}$)
2. Update the weights of all stale elements $e \in B'$ by computing j_e, $\tilde{w}_e = (1-\varepsilon)^{j_e}$ and then calling $\mathcal{D}$.Decrement($e, \tilde{w}_e$).
3. If less than half of the elements in B' where $p_e = 1$ were stale (i.e. needed an update), and less than half of the elements in B' where $p_e < 1$ were stale, then add $e = \mathcal{D}$.UniformSample() to S by calling $\mathcal{D}$.Freeze(e).

Note that in each iteration, we check and update only the weights of some random sample of elements. This is for efficiency; we show the estimated weight $\tilde{w}(B)$ is still correct in expectation up to a constant multiplicative factor. We begin the correctness proof by showing the following lemma.

▶ **Lemma 1.** *Assume $0 < \varepsilon < 1/3$. With high probability, if less than $\frac{1}{2}$ of the estimated weight of B' is in elements which are stale, then*

$$\sum_{e \in B \setminus S} w_e(S) \geq \frac{4}{\varepsilon} f(OPT).$$

Next, we show a bound on the computational complexity of LazySamplingGreedy+.

▶ **Lemma 2.** LazySamplingGreedy+ *uses at most $O(n\varepsilon^{-1} \log(r/\varepsilon))$ arithmetic operations, calls to f, and calls to the maximum weight data structure.*

Next we observe that S cannot have too many elements, otherwise $f(S)$ is close to $f(OPT)$ and we are done.

▶ **Observation 3.** *With high probability, S at the end of the algorithm has at most $\varepsilon r/2$ elements.*

▶ **Theorem 4.** *Let S be the set returned at the end of* LazySamplingGreedy+, *$OPT := \arg\max_{T \in \mathcal{M}} f(T)$, and $OPT^* := \arg\max_{T \in \mathcal{M}/S} f(T|S)$. The following inequality holds:*

$$\mathbf{E}[f(OPT^* \cup S)] \geq (1 - 2\varepsilon) f(OPT).$$

3.3 The ContinuousGreedy algorithm

In this section we discuss our implementation of the ContinuousGreedy algorithm. The basis of our algorithm is the fast implementation from [6], with additional speed-up due to the fact that the LazySamplingGreedy+ stage reduces the marginal values of the remaining elements. The previous section shows that our LazySamplingGreedy+ algorithm runs

with at most $O_\varepsilon(n \log r)$ arithmetic operations, calls to f, and calls to the maximum weight data structure. In this section, we describe how LazySamplingGreedy+ helps the runtime of ContinuousGreedy. The proofs are given in the full version of the paper.

At the termination of LazySamplingGreedy+ it holds that $\tilde{w}(B) \leq \frac{50}{\varepsilon} f(OPT)$. Stale weights in B have true weights lower than its weight estimate $\tilde{w}_e$. Therefore, the true weight of elements of B must be also at most $\frac{50}{\varepsilon} f(OPT)$. Furthermore, since B is a constant-factor approximation to the true maximum weight basis $B^\star$, this implies that $\text{weight}(B^\star) = O(\frac{1}{\varepsilon} f(OPT))$.

Let $\tilde{f}(T) = f(T|S)$, where S is the set output at the termination of LazySampling-Greedy+. We observe that for any set $T \in \mathcal{M}/S$, $\sum_{e \in T} \tilde{f}(e) = O(\frac{1}{\varepsilon} f(OPT))$. When this is the case, [10] (Corollary 3.2) gives the following result:

▶ **Lemma 5** ([10]). ContinuousGreedy *to obtain a* $(1 - 1/e - \varepsilon)$*-approximation uses* $O(n\varepsilon^{-2} \log(n/\varepsilon))$ *independent set data structure operations and* $O(n\varepsilon^{-5} \log^2(n/\varepsilon))$ *queries to* $\tilde{f}$.

In this section, we make two observations that improve the number of independent set queries by a log factor. The inner loop of the ContinuousGreedy algorithm is essentially a greedy algorithm which operates on a function derived from the multilinear extension of $\tilde{f}$: $g(T) = F(\mathbf{x} + \varepsilon \mathbf{1}_T)$ where $F(\mathbf{x}) = \mathbf{E}[\tilde{f}(R)]$, R sampled independently with probabilities x_e. The inner loop of ContinuousGreedy finds an increment of the current fractional solution $\mathbf{x}$ by running a greedy algorithm to approximate a maximum-weight basis with respect to the function g. Let us define $w_e(T) = g(T + e) - g(T)$ to be the marginal values of this function.

A fast implementation of this inner loop is the DescendingThreshold subroutine of Badanidiyuru and Vondrák [6], which also appears in the algorithm of [10]. This subroutine uses the marginal values $w_e(B)$ defined above; the expectation requires $O(\varepsilon^{-1} \log^2(n/\varepsilon))$ samples to estimate for the required accuracy of ContinuousGreedy. In the algorithms below, $w_e(S)$ can be thought of as a black-box that issues $O(\varepsilon^{-1} \log^2(n/\varepsilon))$ calls to the function $\tilde{f}$.

Algorithm 3.3: DescendingThreshold.

$B \leftarrow \emptyset$
$\tau \leftarrow \max_{\{e\} \in \mathcal{M}} w_e(\emptyset)$
While $\tau \geq \frac{\varepsilon}{r} f(O)$:
1. Iterate through $e \in E$ one by one. If $B \cup \{e\} \in \mathcal{I}$ and $w_e(B) \geq \tau$, add e to B. Otherwise, if $B \cup \{e\} \notin \mathcal{I}$, remove e from E.
2. $\tau \leftarrow (1 - \varepsilon)\tau$

Return B

The number of independent set queries in ContinuousGreedy is dominated by the first line of the while loop in DescendingThreshold.

We make two observations about the DescendingThreshold algorithm of Badanidiyuru and Vondrák [6], resulting in two modifications to DescendingThreshold that uses the *incremental independence oracle* and *approximate maximum independent set data structure* outlined in Section 3.1.

▶ **Observation 6.** *Only* $O(n/\varepsilon)$ *independence oracle queries are required. Furthermore, it is sufficient to use an* ***incremental*** **independence oracle.**

Thus, we can modify the DescendingThreshold of [6] by simply ignoring elements that have been previously rejected within the descending threshold greedy subprocedure (see Algorithm 3.4). This yields the following:

▶ **Lemma 7.** ContinuousGreedy *uses* $O(n/\varepsilon)$ *incremental independent set data structure operations and* $O(n\varepsilon^{-5}\log^2(n/\varepsilon))$ *queries to* $\tilde{f}$.

Algorithm 3.4: DT-Incremental.

$\mathcal{D} \leftarrow$ Incremental independence oracle maintaining a set B (Section 3.1)
$\tau \leftarrow \max_{\{e\}\in\mathcal{M}} w_e(\emptyset)$
While $\tau \geq \frac{\varepsilon}{r} f(O)$:
1. $E_\tau \leftarrow \{e \mid w_e(B) \geq \tau, e \in E \setminus B\}$
2. Iterate through $e \in E_\tau$ one by one. If $\mathcal{D}$.Test(e) returns YES and $w_e(B) \geq \tau$, call $\mathcal{D}$.Insert(e) and add e to B. Otherwise, if $B \cup \{e\} \notin \mathcal{I}$, remove e from E.
3. $\tau \leftarrow (1-\varepsilon)\tau$

Return B

An alternative observation

In the case of transversal matroids, exact incremental independence oracle with polylogarithmic update times are not known. Instead, we will make the following observation: An approximate *decremental maximal independent set* data structure can be used instead of an incremental independence oracle. This results in the modification of descending threshold described in Algorithm 3.5.

Algorithm 3.5: DT-ApproxIndepSet.

$\mathcal{D} \leftarrow$ Approximate dynamic maximum independent set data structure maintaining a set B (Section 3.1)
$\tau \leftarrow \max_{\{e\}\in\mathcal{M}} w_e(\emptyset)$
While $\tau \geq \frac{\varepsilon}{r} f(O)$:
1. $E_\tau \leftarrow \{e \mid w_e(B) \geq \tau, e \in E \setminus B\}$
2. $B^+ \leftarrow \mathcal{D}$.Batch-Insert$(E_\tau)$
3. While $B^+ \neq \emptyset$:
   a. Get any $e \in B^+$. If $w_e(B) < \tau$, $D \leftarrow \mathcal{D}$.Delete(e) and set $B^+ \leftarrow B^+ \cup D$.
   b. Remove e from B^+.
4. $\tau \leftarrow (1-\varepsilon)\tau$

Return B

▶ **Observation 8.** *Instead of an incremental independence oracle,* ContinuousGreedy *can be implemented with a decremental approximate maximum independent set data structure. Furthermore,* ContinuousGreedy *will only make* $O(\varepsilon^{-1}\log r)$ *calls to* Batch-Insert *and* $O(n\varepsilon^{-1}\log r)$ *calls to* Delete.

Rounding the fractional solutions. The ContinuousGreedy algorithm makes $O(1/\varepsilon)$ calls to Algorithm 3.3, and outputs a fractional solution that is a convex combination of the $O(1/\varepsilon)$ bases returned by these calls [6]. This fractional solution then needs to be rounded to an integral solution efficiently. In the full version, we show that the data structures we develop can speed up the rounding as well, leading to the overall cost being dominated by the LazySamplingGreedy+ and ContinuousGreedy phases.

3.4 Analysis of the overall framework

▶ **Lemma 9.** *The approximation returned by our framework has approximation ratio at least* $1 - 1/e - \varepsilon$.

Proof. Let S_0 be the set returned by LazySamplingGreedy+. Recall that $\tilde{f}(T) := f(T|S_0)$. By the results in the previous sections, there exists a set $OPT^\star$ such that $OPT^\star \cup S_0$ is independent and $\mathbf{E}[\tilde{f}(OPT^\star)] \geq (1-\varepsilon/2)f(OPT) - f(S_0)$ (by running LazySamplingGreedy+ with $\varepsilon/4$ instead of ε). Running continuous greedy on $\tilde{f}$ yields a $(1-1/e-\varepsilon/2)$-approximation S_1 to $OPT^\star$. Thus the final value of our solution $f(S_0 + S_1)$ is:

$$\begin{aligned}\mathbf{E}[f(S_0 + S_1)] &= \mathbf{E}[\tilde{f}(S_1) + f(S_0)] \\ &\geq (1 - 1/e - \varepsilon/2)\mathbf{E}[\tilde{f}(OPT^\star) + f(S_0)] \\ &\geq (1 - 1/e - \varepsilon/2)(1 - \varepsilon/2)f(OPT) \\ &\geq (1 - 1/e - \varepsilon)f(OPT).\end{aligned}$$

◀

▶ **Observation 10.** *Our framework uses at most:*

- $O(n\varepsilon^{-5}\log^2(n/\varepsilon))$ *calls to the submodular function oracle* f.
- $O(n\varepsilon^{-1}\log(r/\varepsilon))$ *calls to an approximate maximum weight oracle (Section 3.2).*
- *Either* $O(n/\varepsilon)$ *incremental oracle data structure operations or* $O(\varepsilon^{-1}\log r)$ *calls to* BatchInsert *and* $O(n\varepsilon^{-1}\log r)$ *calls to* Delete *on a decremental approximate maximum independent set data structure (Section 3.3).*

The cost of evaluating f is dominated by the ContinuousGreedy phase (see Lemma 7), as LazySamplingGreedy+ only uses $O(n\varepsilon^{-1}\log(r/\varepsilon))$ oracle calls to f, where r is the rank of the matroid (Lemma 2).

4 Data structures for various matroids

In this section, we give dynamic (c, d)-approximate maximum weight oracles and $(1-\varepsilon)$-approximate independence oracles for laminar matroids, graphic matroids, and transversal matroids.

Limitations for further improvements. For both the laminar, graphic, and transversal matroid, the total runtime of the data structure operations in LazySamplingGreedy+ and ContinuousGreedy is $O_\varepsilon(|\mathcal{M}|\log^2|\mathcal{M}|)$, where $|\mathcal{M}|$ is the number of matroid elements. Without improving the original ContinuousGreedy algorithm itself, it is impossible to improve the runtime further. This is because the ContinuousGreedy phase requires at least $O_\varepsilon(|\mathcal{M}|\log^2|\mathcal{M}|)$ oracle calls to f, which is at least $O(1)$ cost in any reasonable model of computing.

Weighted sampling on (c, d)-approximate independent sets. Our (c, d)-approximate maximum weight oracles in Section 3.1 require the ability to sample from the independent set they maintain. This sampling operation can be handled independently from the other operations of the data structure, by augmenting the Decrement and Freeze operations. As this augmentation is the same in all our data structures, we describe it in the full version.

4.1 Laminar matroids

Laminar matroids generalize uniform and partition matroids. In the full version of the paper we present a data structure $\mathcal{D}$ using top trees [2] that maintains a fully dynamic maximum weight basis for a laminar matroid under insertions and deletions of elements with

arbitrary weights in $O(\log n)$ update time. This data structure satisfies the (c, d)-approximate maximum weight oracle requirements with $c = d = 1$ and satisfies the $(1 - \varepsilon)$-approximate independence oracle requirements with $\varepsilon = 0$.

Dynamic maximum weight oracle. The data structure $\mathcal{D}$ maintains the maximum weight basis under insertion and deletions. For FREEZE(e) operations, we don't need to do anything. For DECREMENT(e, w) operations, we can simulate a decrement with the deletion of e and an insertion of e with the changed weight. As we show in the appendix of the full version, deleting and inserting an element removes at most the deleted element and adds at most one element to the maximum weight basis, and thus would never remove a frozen element from the basis whose weight never decreases.

Incremental independence oracle. This data structure can also be used to implement an incremental independence oracle as follows: Run the data structure $\mathcal{D}$ where every element has the same weight and that maintains a maximum basis B. Both TEST and INSERT can easily be handled by our data structure.

4.2 Graphic matroids

A graphic matroid can be represented with a weighted undirected graph $G = (V, E, w)$ where the weight of and edge $e \in E$ is given by $w(e)$.

Dynamic $(1/2, 1/2)$-approximate maximum weight oracle. To obtain an approximate maximum spanning tree of a graph $G = (V, E)$, take the largest edge incident to every vertex, with ties broken according to the edge numbering. For every vertex $v \in V$, let E_v denote the set of edges incident to v. We can store the weights of edges in E_v in a heap H_v and maintain that the maximum element of H_v is part of our approximate maximum spanning tree. It is easy to show that the set of edges maintained, F, is a forest with at least $1/2$ the weights and $1/2$ the number of edges of the optimal maximum spanning tree T.

For the correctness of the algorithm we show first that there cannot be any cycle in F. Assume by contradiction that there is a cycle C in F. Direct each edge in C towards the vertex where it was the maximum weight edge, breaking ties according to the vertex number. If C is a cycle, then C must give a directed cycle, where each edge is larger than the next edge in the directed cycle in the lexicographic order induced by the edge weight and the vertex number. This is a contradiction.

Approximation factor. Root T at an arbitrary vertex and consider the vertices of T starting at the leaves. We will use a simple charging argument to show that F has at least $1/2$ the weight of T and that $|F| \geq |T|/2$. The edge of a vertex v going to its parent u in the tree T can be charged to the largest weight edge leaving v, which is in F. Since each edge of $e \in F$ can be charged at most twice from the two endpoints of e by edges of lesser or the same weight, F has at least half the weight of T and at least half the edges as well.

DECREMENT(e, w): When the weight of an edge $e = (u, v) \in E$ changes to w, we update H_u and H_v accordingly. This may change the maximum weight edge incident to u or v, but we can lookup and accordingly modify our approximate maximum spanning tree with the new maximum weight edge of T_u and T_v in $O(\log n)$ time and report these changes.

FREEZE(e): When we freeze an edge $e = (u, v) \in F$, we can contract the graph along the edge. To do so, we can merge the heaps H_u and H_v and associate the merged heap with the new merged vertex. This can be done in $O(\log n)$ time with binomial heaps or $O(1)$ time using the Fibonacci heaps of Fredman and Tarjan [14]. When we merge two vertices, the maximum weight edge incident to the new merged vertex may be added to the approximate maximum spanning tree.

Incremental independence oracle. Unweighted incremental maximum spanning tree involves checking if inserting any edge increases the size of the spanning tree. This can be done in $O(\alpha(n))$ update and query time with the disjoint set union data structure of Tarjan [24].

4.3 Transversal matroids

Representation of transversal matroids. As stated in Section 2, we assume that our transversal matroids are given as minimal representations. This means that the matroid $\mathcal{M}$ is represented by a bipartite graph $G = ((L, R), E)$ where $|R| = \text{rank}(\mathcal{M})$. For sake of notation let $n = |L|$ and $m = |E|$. As a reminder, each element of the matroid corresponds to a node in L, and an independent set I is a subset of L such that there exists a matching in G that matches every element of I. We will let $N(v)$ denote the neighbors of v in G, that is $N(v) = \{u \mid (u, v) \in E\}$. If $m > n^2$ we can remove neighbours from each vertex in L until their degree is at most n. This doesn't affect whether a vertex belongs to an independent set, as it can always be matched. This reduces m to at most $O(n^2)$.

Dynamic $(1 - \varepsilon, 1/2)$-approximate maximum weight oracles. Recall that in the case of transversal matroids, the weighted setting of LazySamplingGreedy+ leads to a dynamic matching problem on a *vertex-weighted* bipartite graph $G = ((L, R), E)$, where each vertex $\ell \in L$ has a non-negative weight $w(\ell)$ and all edges incident to L have weight $w(\ell)$. We assume that each vertex in L has a value $w_{min} \geq O(w_{max}\varepsilon/n)$ such that we may ignore the weight of any vertex that drops below w_{min}. For the purposes of LazySamplingGreedy+, we stop if the maximum weight basis decreases below $O(f(OPT)) \geq w_{max}$, and so even if we discard all items with weight less than $O(w_{max}\varepsilon/n)$, we can discard at most an ε fraction of $f(OPT)$. Thus after appropriate multiplicative rescaling, we may assume that $w_{min} = 1$ and $w_{max} = (1 + \varepsilon)^k$ for $k = O(\log_{1+\varepsilon} n)$. Furthermore we may assume that the weight of any $\ell \in L$ is $(1+\varepsilon)^j$ for some $j \geq 0$ as we can round all weights in the range of $[(1+\varepsilon)^j, (1+\varepsilon)^{j+1}]$ down to the nearest $(1 + \varepsilon)^j$ and lose only a $(1 + \varepsilon)^{-1}$ factor in the value of the solution.

We will design a data structure that maintains a matching M such that whenever a Decrement(ℓ, w) operation is performed on $\ell \in L$, then ℓ will be the only node of L that may potentially become unmatched in M. We will call a data structure that has this property *L-stable*. The basis we output will be the set of nodes of L matched in M. Note that L-stable data structures can handle the Freeze operation by not doing anything and always returning an empty set. No frozen element will be removed from the basis because frozen elements are never decremented.

The high level idea of our algorithm is as follows: We want to maintain a maximal matching according to some weights, as this guarantees that at least half as many nodes of L are matched as in the optimum solution. The question is just which weights to choose and which algorithm to use to guarantee maximality while fulfilling L-stability. Note that L-stability allows edges in the matching to change, just un-matching a matched vertex of L is forbidden. For this reason we chose an algorithm that is greedy for the vertices in R, i.e., each vertex in R is matched with a neighbor of largest weight for a suitable choice of weight. In order to maintain the invariant at every vertex r of R our greedy algorithm allows r to "steal" the matched neighbor l of another vertex r' of R. This maintains L-stability as l remains matched. However, the newly un-matched vertex r' might want to steal l right back from r. To avoid this, we do not use the original weights in the greedy algorithm, but instead we use "virtual weights" that are initialized by the original weights and that decrease by a factor of $(1 + \varepsilon)$ whenever l is (re-)matched. This makes l less attractive for r' and, as l

is never re-matched when its weight is below 1, it also guarantees that l is only re-matched $\tilde{O}_\varepsilon(1)$ times in total over all decrement operations. For formal details and the proof, see the full version of the paper.

▶ **Theorem 11.** *Given a bipartite graph $G = ((L, R), E)$ and a value w_{min}, there exists a L-stable data structure that handles* Decrement *operations and maintains a $(1-\varepsilon, 1/2)$-approximate maximum weighted matching provided that the maximum weighted matching has cost at least w_{min}. The total running time for preprocessing and all operations as well as the total number of changes to the set of matched vertices is $O(|E|(1/\varepsilon + \log|L|))$. Furthermore, the matching maintained is maximal.*

$(1-\varepsilon)$-approximate dynamic maximum independent set data structure. The fastest known algorithm for incremental maximum bipartite matching takes $O(m\sqrt{n})$ total time [8]. However, given a bipartite graph $G = (L, R)$ there is a $(1-\varepsilon)$-approximate maximum matching data structure $\mathcal{D}_M$ for deletions of vertices in L [8]. It has three properties that are crucial for our algorithm: (1) It does not unmatch a previously matched vertex of L as long as it is not deleted, (2) it maintains an explicit integral matching, i.e., it stores at each vertex whether and if so, along which edge it is matched, and (3) the total time for computing the initial matching and all vertex deletions is $O((m + |L|)/\varepsilon)$, where m is the number of edges in the initial graph.

Given an initial graph G_0 and a partial matching B of G_0 this algorithm can be modified to guarantee that the initial $(1-\varepsilon)$-approximate matching computed for G_0 matches all vertices of $B \cap L$. See teh full version of the paper for details. We use this data structure $\mathcal{D}_M$ to implement a $(1-\varepsilon)$-approximate dynamic maximum independent set data structure for the transversal matroid as follows:

Batch-Insert(E'): Let B be the $(1-\varepsilon)$-approximate matching before the update. Initialize a new data $\mathcal{D}_M$ with all current elements and compute an initial $(1-\varepsilon)$-approximate matching computed for G_0 matching all vertices in $B \cap L$. This is possible by the discussion above.

Delete(e): Execute a vertex deletion of vertex e in $\mathcal{D}_M$.

Test(e): Return YES if e is matched and NO otherwise.

▶ **Lemma 12.** *Given a transversal matroid there exists a $(1-\varepsilon)$-approximate dynamic maximum independent set data structure such that each* Batch-Insert*(B, E_1, E_2) and all* Delete *operations until the next* Batch-Insert *take $O((m' + |E_1| + |E_2|)/\varepsilon)$ total worst-case time and each* Test *operation takes $O(1)$ worst-case time.*

References

1. Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 434–443. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.53.
2. Stephen Alstrup, Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup. Maintaining information in fully dynamic trees with top trees. *ACM Trans. Algorithms*, 1(2):243–264, 2005. doi:10.1145/1103963.1103966.
3. Moshe Babaioff, Jason Hartline, and Robert Kleinberg. Selling banner ads: Online algorithms with buyback. In *Fourth workshop on ad auctions*, 2008.

4 Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 434–443. SIAM, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283429.

5 Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: massive data summarization on the fly. In Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani, editors, *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA – August 24 – 27, 2014*, pages 671–680. ACM, 2014. doi:10.1145/2623330.2623637.

6 Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1497–1514. SIAM, 2014. doi:10.1137/1.9781611973402.110.

7 Jeff A. Bilmes. Submodularity in machine learning and artificial intelligence. *CoRR*, abs/2202.00132, 2022. arXiv:2202.00132.

8 Bartlomiej Bosek, Dariusz Leniowski, Piotr Sankowski, and Anna Zych. Online bipartite matching in offline time. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 384–393. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.48.

9 Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1433–1452. SIAM, 2014. doi:10.1137/1.9781611973402.106.

10 Niv Buchbinder, Moran Feldman, and Roy Schwartz. Comparing apples and oranges: Query trade-off in submodular maximization. *Math. Oper. Res.*, 42(2):308–329, 2017. doi:10.1287/moor.2016.0809.

11 Gruia Călinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.*, 40(6):1740–1766, 2011. doi:10.1137/080733991.

12 Alina Ene and Huy L. Nguyen. Towards nearly-linear time algorithms for submodular maximization with a matroid constraint. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 54:1–54:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.54.

13 Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998. doi:10.1145/285055.285059.

14 Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987. doi:10.1145/28869.28874.

15 Manoj Gupta and Richard Peng. Fully dynamic $(1+\varepsilon)$-approximate matchings. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 548–557. IEEE Computer Society, 2013. doi:10.1109/FOCS.2013.65.

16 Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 21–30. ACM, 2015. doi:10.1145/2746539.2746609.

17 Monika Henzinger, Ami Paz, and Stefan Schmid. On the complexity of weight-dynamic network algorithms. In Zheng Yan, Gareth Tyson, and Dimitrios Koutsonikolas, editors, *IFIP Networking Conference, IFIP Networking 2021, Espoo and Helsinki, Finland, June 21-24, 2021*, pages 1–9. IEEE, 2021. doi:10.23919/IFIPNetworking52078.2021.9472803.

18 Ehsan Kazemi, Marko Mitrovic, Morteza Zadimoghaddam, Silvio Lattanzi, and Amin Karbasi. Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3311–3320. PMLR, 2019. URL: http://proceedings.mlr.press/v97/kazemi19a.html.

19 Hung Le, Lazar Milenkovic, Shay Solomon, and Virginia Vassilevska Williams. Dynamic matching algorithms under vertex updates. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 – February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 96:1–96:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.96.

20 Wenxin Li, Moran Feldman, Ehsan Kazemi, and Amin Karbasi. Submodular maximization in clean linear time. *CoRR*, abs/2006.09327, 2022. arXiv:2006.09327.

21 Paul Liu and Jan Vondrák. Submodular optimization in the mapreduce model. In Jeremy T. Fineman and Michael Mitzenmacher, editors, *2nd Symposium on Simplicity in Algorithms, SOSA 2019, January 8-9, 2019, San Diego, CA, USA*, volume 69 of *OASIcs*, pages 18:1–18:10. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/OASIcs.SOSA.2019.18.

22 George L. Nemhauser and Laurence A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Math. Oper. Res.*, 3(3):177–188, 1978.

23 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

24 Robert Endre Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM*, 22(2):215–225, 1975. doi:10.1145/321879.321884.

25 Mark Wilhelm, Ajith Ramanathan, Alexander Bonomo, Sagar Jain, Ed H. Chi, and Jennifer Gillenwater. Practical diversified recommendations on youtube with determinantal point processes. In Alfredo Cuzzocrea, James Allan, Norman W. Paton, Divesh Srivastava, Rakesh Agrawal, Andrei Z. Broder, Mohammed J. Zaki, K. Selçuk Candan, Alexandros Labrinidis, Assaf Schuster, and Haixun Wang, editors, *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pages 2165–2173. ACM, 2018. doi:10.1145/3269206.3272018.

26 Zhou Xu and Brian Rodrigues. A 3/2-approximation algorithm for the multiple tsp with a fixed number of depots. *INFORMS Journal on Computing*, 27(4):636–645, 2015.

27 Da Wei Zheng and Monika Henzinger. Multiplicative auction algorithm for approximate maximum weight bipartite matching. *CoRR*, abs/2301.09217, 2023. doi:10.48550/arXiv.2301.09217.

Twin-Width of Planar Graphs Is at Most 8, and at Most 6 When Bipartite Planar

Petr Hliněný
Masaryk University, Brno, Czech republic

Jan Jedelský
Masaryk University, Brno, Czech republic

Abstract

Twin-width is a structural width parameter introduced by Bonnet, Kim, Thomassé and Watrigant [FOCS 2020]. Very briefly, its essence is a gradual reduction (a contraction sequence) of the given graph down to a single vertex while maintaining limited difference of neighbourhoods of the vertices, and it can be seen as widely generalizing several other traditional structural parameters. Having such a sequence at hand allows us to solve many otherwise hard problems efficiently. Graph classes of bounded twin-width, in which appropriate contraction sequences are efficiently constructible, are thus of interest in combinatorics and in computer science. However, we currently do not know in general how to obtain a witnessing contraction sequence of low width efficiently, and published upper bounds on the twin-width in non-trivial cases are often "astronomically large".

We focus on planar graphs, which are known to have bounded twin-width (already since the introduction of twin-width), but the first explicit "non-astronomical" upper bounds on the twin-width of planar graphs appeared just a year ago; namely the bound of at most 183 by Jacob and Pilipczuk [arXiv, January 2022], and 583 by Bonnet, Kwon and Wood [arXiv, February 2022]. Subsequent arXiv manuscripts in 2022 improved the bound down to 37 (Bekos et al.), 11 and 9 (both by Hliněný). We further elaborate on the approach used in the latter manuscripts, proving that the twin-width of every planar graph is at most 8, and construct a witnessing contraction sequence in linear time. Note that the currently best lower-bound planar example is of twin-width 7, by Král' and Lamaison [arXiv, September 2022]. We also prove that the twin-width of every bipartite planar graph is at most 6, and again construct a witnessing contraction sequence in linear time.

2012 ACM Subject Classification Mathematics of computing → Graph theory

Keywords and phrases twin-width, planar graph

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.75

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2210.08620` [17]

Funding Jan Jedelský was partially supported by Masaryk University project MUNI/I/1677/2018.

1 Introduction

Twin-width is a relatively new structural width measure of graphs and relational structures introduced in 2020 by Bonnet, Kim, Thomassé and Watrigant [10]. Informally, twin-width of a graph measures how diverse the neighbourhoods of the graph vertices are. E.g., *cographs* – the graphs which can be built from singleton vertices by repeated operations of a disjoint union and taking the complement, have the lowest possible value of twin-width, 0, which means that the graph can be brought down to a single vertex by successively identifying twin vertices. (Two vertices x and y are called *twins* in a graph G if they have the same neighbours in $V(G) \setminus \{x, y\}$.) Hence the name, *twin-width*, for the parameter.

Importance of this new concept is clearly witnessed by numerous recent papers on the topic, such as the follow-up series [5–9, 11] and more related research papers represented by, e.g., [1, 2, 4, 13, 15, 20].

Twin-width definition. In general, the concept of twin-width can be considered over arbitrary binary relational structures of a finite signature, but here we will define it and deal with it for only finite *simple graphs*, i.e., graphs without loops and multiple edges. A *trigraph* is a simple graph G in which some edges are marked as *red*, and with respect to the red edges only, we naturally speak about *red neighbours* and *red degree* in G. However, when speaking about edges, neighbours and/or subgraphs without further specification, we count both ordinary and red edges together as one edge set denoted by $E(G)$. The edges of G which are not red are sometimes called (and depicted) black for distinction. For a pair of (possibly not adjacent) vertices $x_1, x_2 \in V(G)$, we define a *contraction* of the pair x_1, x_2 as the operation creating a trigraph G' which is the same as G except that x_1, x_2 are replaced with a new vertex x_0 (said to *stem from* x_1, x_2) such that:

- the (full) neighbourhood of x_0 in G' (i.e., including the red neighbours), denoted by $N_{G'}(x_0)$, equals the union of the neighbourhoods $N_G(x_1)$ of x_1 and $N_G(x_2)$ of x_2 in G except x_1, x_2 themselves, that is, $N_{G'}(x_0) = (N_G(x_1) \cup N_G(x_2)) \setminus \{x_1, x_2\}$, and
- the red neighbours of x_0, denoted here by $N^r_{G'}(x_0)$, inherit all red neighbours of x_1 and of x_2 and add those in $N_G(x_1) \Delta N_G(x_2)$, that is, $N^r_{G'}(x_0) = \big(N^r_G(x_1) \cup N^r_G(x_2) \cup (N_G(x_1) \Delta N_G(x_2))\big) \setminus \{x_1, x_2\}$, where Δ denotes the symmetric set difference.

A *contraction sequence* of a trigraph G is a sequence of successive contractions turning G into a single vertex, and its *width* d is the maximum red degree of any vertex in any trigraph of the sequence. We also then say that it is a d-contraction sequence of G. The *twin-width* of a trigraph G is the minimum width over all possible contraction sequences of G. In other words, a graph has twin-width at most d, if and only if it admits a d-contraction sequence.

To define the twin-width of an ordinary (simple) graph G, we consider G as a trigraph with no red edges.

Algorithmic aspects. Twin-width, as a structural width parameter, has naturally many algorithmic applications in the FPT area. Among the most important ones we mention that the first order (FO) model checking problem – that is, deciding whether a fixed first-order sentence holds in an input graph – can be solved in linear FPT-time [11]. This and other algorithmic applications assume that a contraction sequence of bounded width is given alongside with the input graph. Deciding the exact value of twin-width (in particular, twin-width 4) is in general NP-hard [4], but for many natural graph classes we know that they are of bounded twin-width. However, published upper bounds on the twin-width in non-trivial cases are often non-explicit or "astronomically large", and it is not usual that we could, alongside such a bound, compute a contraction sequence of provably "reasonably small" width efficiently and practically. We pay attention to this particular aspect; and we will accompany our fine mathematical upper bounds on the twin-width with rather simple linear-time algorithms for computing contraction sequences of the claimed widths.

Twin-width of planar graphs. The fact that the class of planar graphs is of bounded twin-width was mentioned already in the pioneering paper [10], but without giving any explicit upper bound on the twin-width. The first explicit (numeric) upper bounds on the twin-width of planar graphs have been published only quite recently; chronologically on arXiv, the bound of 183 by Jacob and Pilipczuk [18], of 583 by Bonnet, Kwon and Wood [12]

(this paper more generally bounds the twin-width of k-planar graphs by asymptotic $2^{\mathcal{O}(k)}$), and of 37 by Bekos, Da Lozzo, Hliněný, and Kaufmann [3] (this paper more generally bounds the twin-width of so-called h-framed graphs by $\mathcal{O}(h)$).

It is worth to mention that all three papers [3, 12, 18], more or less explicitly, use the product structure machinery of planar graphs (cf. [14]). We have then developed an alternative decomposition-based approach, leading to a single-digit upper bound of 9 for all planar graphs in [16], followed by an upper bound of 6 on the twin-width of *bipartite* planar graphs thereafter. However, the approach of [16] seems to be stuck right at 9, and new ideas were needed to obtain further improvements, even by 1.

Here we give the following strengthened upper bound, which uses an improved approach over previous [16] and also simplifies some cumbersome technical details of the former:

▶ **Theorem 1.** *The twin-width of any simple planar graph is at most* 8*, and a corresponding contraction sequence can be found in linear time.*

It is worth to note that, recently, Král' and Lamaison [19] have found a construction (with a proof) of a planar graph with twin-width 7. Hence, the lower bound is by just one off our upper bound, but the right maximum value (7 or 8?) is still an open question.

In an addition, we also prove an upper bound for bipartite planar graphs, which follows from an adaptation of our new techniques specially to the bipartite case. While the bipartite-case bound stays the same as in previous [16], its proof is significantly simpler now.

▶ **Theorem 2.** *The twin-width of any simple* bipartite *planar graph is at most* 6*, and a corresponding contraction sequence can be found in linear time.*

Due to space restrictions, proofs of the *-marked statements are left for the full paper [17].

2 Notation and Tools

We start with a few technical definitions and claims needed for the proofs.

BFS layering and contractions. Let G be a *connected* graph and $r \in V(G)$ a fixed vertex. The *BFS layering of G determined by r* is the vertex partition $\mathcal{L} = (L_0 = \{r\}, L_1, L_2, \ldots)$ of G such that L_i contains all vertices of G at distance exactly i from r. A path $P \subseteq G$ is *r-geodesic* if P is a subpath of some shortest path from r to any vertex of G (in particular, P intersects every layer of $\mathcal{L}$ in at most one vertex). Let T be a *BFS tree* of G rooted at the vertex r as above (that is, for every vertex $v \in V(G)$, the distance from v to r is the same in G as in T). A path $P \subseteq G$ is *T-vertical*, or shortly *vertical* with respect to implicit T, if P is a subpath of some root-to-leaf path of T. Notice that a T-vertical path is r-geodesic, but the converse may not be true. Analogously, an edge $e \in E(G)$ is *horizontal* (with respect to implicit $\mathcal{L}$) if both ends of G are in the same $\mathcal{L}$-layer.

Observe the following trivial claim (cf. also Claim 5):

▷ Claim 3. For every edge $\{v, w\}$ of G with $v \in L_i$ and $w \in L_j$, we have $|i - j| \leq 1$, and so a contraction of a pair of vertices from L_i may create new red edges only to the remaining vertices of $L_{i-1} \cup L_i \cup L_{i+1}$.

Plane graphs; Left-aligned BFS trees. We will deal with *plane graphs*, which are planar graphs with a given (combinatorial) embedding in the plane, and one marked *outer face* (the remaining faces are then *bounded*). A plane graph is a *plane triangulation* if every face of its embedding is a triangle. Likewise, a plane graph is a *plane quadrangulation* if every face of

its embedding is of length 4. It is easy to turn an embedding of any simple planar graph into a simple plane triangulation by adding vertices and incident edges into each non-triangular face. Furthermore, twin-width is non-increasing when taking induced subgraphs, and so it suffices to focus on plane triangulations in the proof of Theorem 1, and to similarly deal with plane quadrangulations in the proof of Theorem 2.

For algorithmic purposes, we represent a plane graph G in the standard combinatorial way – as a graph (the vertices and their adjacencies) with the counter-clockwise cyclic orders of the incident edges of each vertex, and we additionally mark the outer face of G.

In this planar setting, consider now a plane graph G, and a BFS tree T spanning G and rooted in a vertex r of the outer face of G, and picture (for clarity) the embedding G such that r is the vertex of G most at the top. For two adjacent vertices $u, v \in V(G)$, $\{u, v\} \in E(G)$, we say that u *is to the left of* v (wrt. T) if neither of u, v lies on the vertical path from r to the other, and the following holds; if r' is the least common ancestor of u and v in T and $P_{r',u}$ (resp., $P_{r',v}$) denote the vertical path from r' to u (resp., v), then the cycle $(P_{r',u} \cup P_{r',v}) + uv$ has the triple (r', u, v) in this counter-clockwise cyclic order.

A BFS tree T of G with the BFS layering $\mathcal{L} = (L_0, L_1, \ldots)$ is called *left-aligned* if there is no edge $f = uv$ of G such that, for some index i, $u \in L_{i-1}$ and $v \in L_i$, and u is to the left of v (an informal meaning is that one cannot choose another BFS tree of G which is "more to the left" of T in the geometric picture of G and T, such as by picking the edge uv instead of the parental edge of v in T).

▶ **Lemma 4.** *Given a simple plane graph G, and a vertex r on the outer face, there exists a left-aligned BFS tree of G and it can be found in linear time.*

Proof. For this proof, we have to extend the above relation of "being left of" to edges emanating from a common vertex of G. So, for an arbitrary BFS tree T of G and edges $f_1, f_2 \in E(G)$ incident to $v \in V(G)$, such that neither of f_1, f_2 is the parental edge of v in T, we write $f_1 \leq_l f_2$ if there exist adjacent vertices $u_1, u_2 \in V(G)$ such that u_1 is to the left of u_2, the least common ancestor of u_1 and u_2 in T is v and, for $i = 1, 2$, the edge f_i lies on the vertical path from u_i to v. Observe the following; if f_0 is the parental edge of v in T (or, in case of $v = r$, f_0 is a "dummy edge" pointing straight up from r), then $f_1 \leq_l f_2$ implies that the counter-clockwise cyclic order around v is (f_0, f_1, f_2). In particular, $\leq_l$ can be extended into a linear order on its domain.

We first run a basic linear-time BFS search from r to determine the BFS layering $\mathcal{L}$ of G. Then we start the construction of a left-aligned BFS tree $T \subseteq G$ from $T := \{r\}$, and we recursively (now in a "DFS manner") proceed as follows:

- Having reached a vertex $v \in V(T) \subseteq V(G)$ such that $v \in L_i$, and denoting by $X := (N_G(v) \cap L_{i+1}) \setminus V(T)$ all neighbours of v in L_{i+1} which are not in T yet, we add to T the nodes X and the edges from v to X.
- We order the vertices in X using the cyclic order of edges emanating from v to have it compatible with $\leq_l$ at v, and in this increasing order we recursively (depth-first, to be precise) call the procedure for them.

The result T is clearly a BFS tree of G. Assume, for a contradiction, that T is not left-aligned, and let $u_1 \in L_{i-1}$ and $u_2 \in L_i$ be a witness pair of it, where $\{u_1, u_2\} \in E(G)$ and u_1 is to the left of u_2. Let v be the least common ancestor of u_1 and u_2 in T, and let v_1 and v_2 be the children of v on the T-paths from v to u_1 and u_2, respectively. So, by the definition, $vv_1 \leq_l vv_2$ at v, and hence when v has been reached in the construction of T, its child v_1 has been taken for processing before the child v_2. Consequently, possibly deeper in the recursion, u_1 has been processed before the parent of u_2 and, in particular, the procedure has added the edge u_1u_2 into T, a contradiction to u_1 being to the left of u_2.

This recursive computation is finished in linear time, since every vertex of G is processed only in one branch of the recursion, and one recursive call takes time linear in the number of incident edges (to v). ◀

Notice that we have not assumed G to be a triangulation in the previous definition and in Lemma 4, which will be useful for the case of bipartite planar graphs.

Vertex levels in contraction sequences. We are going to work with contraction sequences which, preferably, preserve the BFS layers of $\mathcal{L}$ of connected G. However, we do *not always* preserve the layers, and so we need a notion which is related to the layers of $\mathcal{L}$, but it can differ from these layers when needed – informally, when this "causes no harm at all". For the graph G itself, we define $\lambda[G](v) = i$ if and only if $v \in L_i \in \mathcal{L}$. If G' is a trigraph along a contraction sequence of G, and a vertex $v' \in V(G')$ stems from a set $X \subseteq V(G)$ by (possible) contractions, then $\lambda[G'](v')$ equals the minimum i such that $L_i \cap X \neq \emptyset$. We say that $\lambda[G'](v')$ is the *level of* v' *in* G' along the considered contraction sequence of G, or simply the *level of* v' when the particular graph of a sequence is implicit. In other words, we can inductively say that if v'' of G'' results by the contraction of u' and v' of G', then $\lambda[G''](v'') := \min(\lambda[G'](u'), \lambda[G'](v'))$. If w' does not participate in a contraction along the subsequence from G' to G'', then $\lambda[G''](w') := \lambda[G'](w')$.

A *partial contraction sequence* of G is defined in the same way as a contraction sequence of G, except that it does not have to end with a single-vertex graph. A partial contraction sequence of G is *level-respecting* if every step contracts, in a trigraph G' along the sequence, only a pair $x, y \in V(G')$ such that the following inductively holds; the levels of x and y are the same, i.e. $\lambda[G'](y) = \lambda[G'](x)$, or all neighbours of y (red or black) in G' are on the same level as x is on, i.e. $\lambda[G'](z) = \lambda[G'](x)$ for all z such that $\{y, z\} \in E(G')$. (The conditions in the latter case, in particular, imply that $\lambda[G'](y) = \lambda[G'](x) + 1$; cf. Claim 5.)

Usefulness of level-respecting contraction sequences lies in the subsequent claim. Informally, we may say that our levels in G' behave analogously to the BFS layers of G; the levels form a layering (in the usual sense), albeit not ncessarily a BFS layering.

▷ Claim 5. Let a trigraph G' result from a level-respecting partial contraction sequence of a connected graph G. Then any vertex $z \in V(G')$ may have neighbours (red or black) only on the levels $\lambda[G](z) - 1$, $\lambda[G](z)$ and $\lambda[G](z) + 1$. Moreover, z must have some neighbour on the level $\lambda[G](z) - 1$.

Proof. We proceed easily by induction. The claim is trivial from the definition of a BFS layering when $G' = G$ and G is connected. Assume that $z \in V(G'')$ results from a contraction of a pair $x, y \in V(G')$, where $\lambda[G''](z) = \lambda[G'](x)$. Then $\lambda[G'](y) \in \{\lambda[G'](x), \lambda[G'](x) + 1\}$ by the definition of a level-respecting contraction and connectivity of G. So, there cannot be any neighbour of z on the levels lower than $\lambda[G''](z) - 1 = \lambda[G'](x) - 1$ from the induction. Regarding levels higher than $\lambda[G''](z) + 1$, they cannot host any neighbour of x in G' by the induction, and no neighbour of y as well by the definition of a level-respecting contraction (if $\lambda[G'](y) = \lambda[G'](x) + 1$). Lastly, since x has a neighbour on the level $\lambda[G''](z) - 1$ in G', so does z in G''. ◁

3 Proof of Theorem 1

3.1 Induction setup for a bounded region of the graph

Our main proof proceeds by induction on suitably defined subregions of the assumed plane triangulation G. In this subsection, we define the setup of this induction in Lemma 6, and show how it will imply the main result.

For a plane graph G and its cycle C, the *subgraph of G bounded by C*, denoted by G_C, is the subgraph of G formed by the vertices and edges of C and the vertices and edges of G drawn inside C – formally, in the region of the plane bounded by C and not containing the outer face. Let the vertices in the set $U := V(G_C) \setminus V(C)$ be called the *interior vertices* of C. We call a set $U_0 \subseteq U$ an *interior section of C* in G if all neighbours of vertices of U_0 belong to $U_0 \cup V(C)$ (in other words, U_0 is a collection of connected components of $G[U]$).

Consider a now fixed BFS tree T of G. Assume that a cycle C of G is formed as $C = (P_1 \cup P_2) + f$, where P_1 and P_2 are two T-vertical paths of length at least 1 with a common end $u \in V(P_1) \cap V(P_2)$ and $f \in E(G)$ is an edge joining the other ends v_1 of P_1 and v_2 of P_2. Observe that u is the (unique) vertex of G_C closest to the root r of T. Then we say that C is a *V-separator* in G with respect to implicit T ('V' as vertical), and we call u the *sink of C* and f the *lid of C*. If the vertices u, v_1, v_2 lie on C in this counter-clockwise order (equivalently, if v_1 is to the left of v_2 with respect to T), then we say that P_1 is the *left path* of C and P_2 is the *right path* of C (picture the sink at the top).

▶ **Lemma 6.** *Let G be a simple plane triangulation, and T be a left-aligned BFS tree of G rooted at a vertex $r \in V(G)$ of the outer triangular face and defining the initial levels $\lambda[G](\cdot)$. Assume that a cycle C of G is a V-separator of G, that G_C is the subgraph of G bounded by C, and u is the sink of C. Let the distance of u from the root r be ℓ, so $\lambda[G](u) = \ell$, and the maximum distance from a vertex of C to r be $m \geq \ell + 1$. Let $U \subseteq V(G_C)$ be an interior section of C in G, and denote by $W := V(G) \setminus (V(C) \cup U)$ the set of the "remaining" vertices.*

Then there exists a level-respecting partial contraction sequence of G which contracts only pairs of vertices that are in or stem from U, results in a trigraph G^*, and satisfies the following conditions for every trigraph G' along this sequence from G to G^*:

- **(I)** *For $U' := V(G') \setminus (V(C) \cup W)$ (which are the vertices that are in or stem from U in G'), every vertex of U' in G' has red degree at most 8,*
- **(II)** *every vertex of the left path of C has at most 5 red neighbours and every vertex of the right path of C has at most 3 red neighbours in U',*
- **(III)** *the sink u of C has no red neighbour in U', and if the least level of a vertex of U in G is $k \geq \ell + 2$, then the vertices of C on levels up to $k - 2$ in G have no red neighbours in U' as well and each of the (two) vertices of C on the level $k - 1$ in G has at most 1 red neighbour in U', and*
- **(IV)** *at the end of the partial contraction sequence, for the set $U^* := V(G^*) \setminus (V(C) \cup W)$ that stems from U in G^*, we have that if $z \in U^*$ is of level i, then $\ell < i \leq \max(m, \ell+2)$ and z is the only vertex in U^* of level i.*

Before proceeding further, we comment on two important things. First, we remark that, in Lemma 6, all vertices of U have the distance from r greater than ℓ, but on the other hand the distance from r to some vertices in U may be much larger than m (and our coming proof is aware of this possibility). Second, we observe that all vertices of U on level $\ell + 1$ must be adjacent to the sink u, since all other potential neighbours of them have the distance from r greater than ℓ. Consequently, contracting U on level $\ell + 1$ into one vertex within the claimed sequence indeed does not create a red edge to u, as long as we do not contract into it from higher levels (which we will explicitly avoid in the proof). We illustrate Lemma 6 in Figure 1.

We also observe that the assumptions and conditions of Lemma 6 directly imply some other properties useful for the coming proofs.

▷ Claim 7. *Respecting the notation and assumptions of Lemma 6, we also have that:*

- **(V)** *Every red edge in G' has one end in U' and the other end in $U' \cup V(C)$,*
- **(VI)** *if P_1 and P_2 are the left and right paths of C, respectively, and $v \in V(P_2)$ is of level j in G', then there is no edge of G' (red or black) from v to a vertex of $U' \cup (V(P_1) \setminus \{u\})$ of level $j - 1$ in G',*

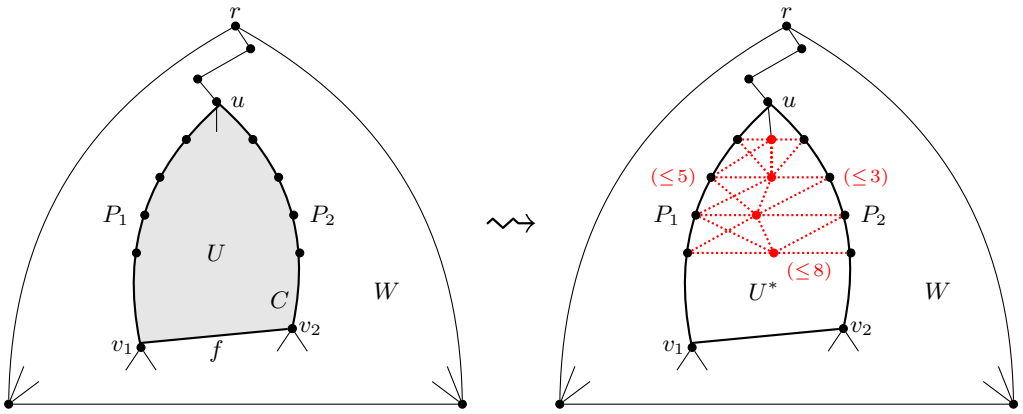


Figure 1 (left) The setup of Lemma 6, where P_1 and P_2 are the left and right paths of the chosen V-separator C. (right) The outcome of the claimed partial contraction sequence which contracts only vertices of U inside the shaded region from the left, and which maintains bounded red degrees in the region and on its boundary C. No other vertex than the sketched ones is affected by the contraction sequence. Not all depicted red edges do exist, and some of them may actually be black.

(VII) *at the end, that is, in G^*, every vertex of the left path of C has at most 3 red neighbours and every vertex of the right path of C has at most 2 red neighbours in U^*.*

Proof. Regarding (V), observe that since only vertices that stem from U participate in contractions, every red edge of G' must have an end in U'. Furthermore, since U is an interior section of C, no vertex of U is adjacent to a vertex of W in G, and hence no vertex of W is ever adjacent to a vertex being contracted in our sequence from G to G^*.

Concerning (VI), if v were adjacent to $x \in V(P_1)$ of level $j-1$, then this was already true in G; $\{x,v\} \in E(G)$. If v were adjacent to $x' \in U'$ of level $j-1$ in G', then, among the vertices of U contracted into x', there had to be $x \in U$ such that $\{x,v\} \in E(G)$. By the definition of a level-respecting sequence, possible vertices of level higher than $j-1$ contracted into x' cannot be adjacent to v of level j, and so $\lambda[G](x) = j-1$, too. Since, in both cases, such x would be to the left of v in G, this contradicts the assumption that T is left-aligned.

Finally, (VII) directly follows from Claim 5 and (IV) for the left path of C. For the right path we additionally apply (VI), which for $x \in V(P_2)$ of level j says that potential red neighbours of x are only on levels j and $j+1$. ◁

We also show how Lemma 6 implies the first part of our main result:

Proof of Theorem 1 (the upper bound). We start with a given simple planar graph H, and extend any plane embedding of H into a simple plane triangulation G such that H is an induced subgraph of G. Then we choose a root r on the outer face of G and, for some left-aligned BFS tree of G rooted in r which exists by Lemma 4, the facial cycle C of the outer face incident to r, and $u = r$, we apply Lemma 6.

This way we get a partial contraction sequence from G to a trigraph G^* of maximum red degree 8 (along the sequence). Observe by (IV) that the set $U^* = V(G^*) \setminus V(C)$ contains only two vertices, on levels 1 and 2. In the final phase, we may hence pairwise contract the remaining vertices in an arbitrary order. The restriction of this whole contraction sequence of G to only $V(H)$ then certifies that the twin-width of H is at most 8. ◀

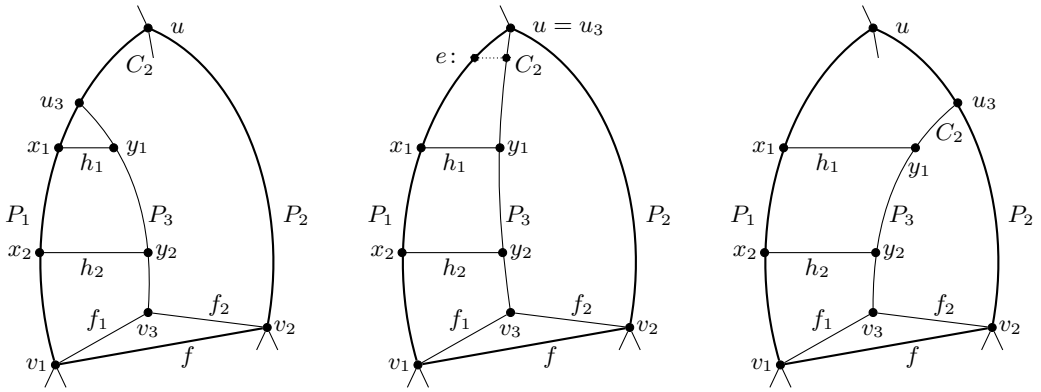


Figure 2 Three schematic cases of the vertical-horizontal division of G_C discussed in Section 3.2. The (possible) dotted horizontal edge e in the middle case is not counted as h_1 since no vertex of U is on a level above it in the picture.

3.2 Vertical-horizontal division into subregions

In order to apply induction in the proof of Lemma 6, we need to decompose the considered subgraph into suitable subregions, based on the plane drawing. Here we formulate the general decomposition step, while possible degenerate cases will be handled later in Section 3.3.

Let C be a V-separator in the plane triangulation G, formed by the left path P_1, the right path P_2 and the lid edge $f = \{v_1, v_2\}$ where v_i is an end of P_i. Let G_C be the subgraph bounded by C and $U \subseteq V(G_C)$ be an interior section of C in G. Moreover, assume that there exists a triangular face in G incident to f with the vertices v_1, v_2, v_3 where $v_3 \in U$, and that the T-vertical path from v_3 to the root r contains neither v_1 nor v_2. In particular, since T is left-aligned, we have that $\lambda[G](v_2) \leq \lambda[G](v_1)$ and $\lambda[G](v_3) \leq \lambda[G](v_1)$. Under these assumptions, we are going to define the *vertical-horizontal division* of G_C as follows.

Let $P \subseteq T$ be the vertical path connecting v_3 to the root r where, by $r \notin U$ and planarity of G, we have that P contains the sink u. Let $P_3 \subseteq P$ be the subpath of P from v_3 to the first vertex $u_3 \in V(P) \cap V(C)$ shared with the cycle C. We have $u_3 \neq v_3$. (It may be that $u_3 \in V(P_1)$ or $u_3 \in V(P_2)$ or even $u_3 = u$; see in Figure 2.) Let P_{31} denote the subpath of P from v_3 to the first intersection x with P_1 ($x \in \{u_3, u\}$), and P_{11} the subpath of P_1 from v_1 till x. Similarly, let P_{32} denote the subpath of P from v_3 to the first intersection y with P_2, and P_{22} be the subpath of P_2 from v_2 till y. Let $f_1 = \{v_1, v_3\}$ and $f_2 = \{v_2, v_3\}$. Observe that $P_{31}, P_{32} \subseteq G_C$, and that $C_1 := (P_{11} \cup P_{31}) + f_1$ and $C_2 := (P_{22} \cup P_{32}) + f_2$ are again V-separators in G, such that P_{31} is the right path of C_1 and P_{32} is the left path of C_2.

Furthermore, let $h_1, \ldots, h_a$, $a \geq 0$, be the collection of all horizontal edges of G_C such that, for $h_i = \{x_i, y_i\}$, we have $x_i \in V(P_1)$, $y_i \in V(P_3) \setminus V(C)$ and $\lambda[G](x_i) = \lambda[G](y_i)$, and that $\lambda[G](z) \leq \lambda[G](x_i) - 1$ holds for some $z \in U$ (the reason for this strange-looking restriction is in property (III) of Lemma 6). These edges $h_1, \ldots, h_a$ are ordered by their increasing level $\lambda[G](x_i)$. This is illustrated in Figure 2 (where the ordering of h_i's is top-down). For $i = 1, \ldots, a$, let $C_{1,i-1}$ denote the cycle passing through the sink of C_1 (which is u_3 or u) and formed by relevant subpaths of P_{11}, P_{31} and the edge h_i. Let $C_{1,a} = C_1$. Let $U_{1,0}$ denote the set of the interior vertices of $C_{1,0}$ in G, and for $i = 1, \ldots, a$, let $U_{1,i} := X \setminus U_{1,i-1}$ where X is the set of the interior vertices of $C_{1,i}$ in G. Let U_2 denote the set of the interior vertices of C_2 in G.

The system of the cycles $C_{1,0}, \ldots, C_{1,a}, C_2$ and of the sets $U_{1,0}, \ldots, U_{1,a}, U_2$ is called the *vertical-horizontal division* of G_C. The following is straightforward from the definition:

▷ Claim 8. For $i = 0, 1, \ldots, a$, the cycle $C_{1,i}$ is a V-separator in G, and each vertex of $U_{1,i}$ has neighbours only in $U_{1,i} \cup V(C_{1,i})$. Hence, $U_{1,i}$ is an interior section of $C_{1,i}$. Consequently, for every $z \in U_{1,i}$ where $i \geq 1$, we have $\lambda[G](z) \geq \lambda[G](x_i) + 1$ (where $\{x_i, y_i\} = h_i$ above).

The intended purpose of a vertical-horizontal division in the proof of Lemma 6 is to start the induction step, as precisely formulated in the next lemma with a straightforward proof:

▶ **Lemma 9.** *Assume the notation and assumptions of Lemma 6 for the graph G, cycle C and set U, and consider the vertical-horizontal division of the subgraph G_C as defined above; that is, the cycles $C_{1,0}, \ldots, C_{1,a}, C_2$ and the sets $U_{1,0}, \ldots, U_{1,a}, U_2$. Then the following hold:*
a) Each cycle $C^1 \in \{C_{1,0}, \ldots, C_{1,a}, C_2\}$ and the corresponding set $U^1 \in \{U_{1,0}, \ldots, U_{1,a}, U_2\}$ satisfy the assumptions of Lemma 6 (in the place of C and U).
b) Let $\tau_{1,i}$, $i = 0, \ldots, a$, denote the level-respecting partial contraction sequence of G claimed by Lemma 6 for the input as in (a) $C^1 := C_{1,i}$ and $U^1 := U_{1,i}$, and likewise, τ_2 be that for the input $C^1 := C_2$ and $U^1 := U_2$. Then the concatenated partial contraction sequence $\sigma_0 := \tau_2 \cdot \tau_{1,0} \cdot \ldots \tau_{1,a}$, i.e., one starting with τ_2 and ending with $\tau_{1,a}$, again satisfies the properties (I), (II) *and* (III) *of Lemma 6.*

Proof. Part (a) immediately follows from the definition and Claim 8.

In part (b), we first argue that the concatenation σ_0 is well-founded; that contractions in one of the subsequences of σ_0 have no effect on vertices being contracted in another of the subsequences. This is since, by Claim 8, neighbours of contracted pairs of one subsequence are only in the interior section of the same subsequence or on the bounding cycles which together form $V(C) \cup V(P_3)$ (and the latter set is not participating in the contractions of σ_0).

Following on the previous argument, we have that the only vertices of G_C that may potentially receive red edges from more than one of the subseqeunces forming σ_0, are those of $V(C) \cup V(P_3)$. See Figure 2. For all other vertices of G_C, we have that (I) is true for them along whole σ_0 since it has been true for them along their subsequences of σ_0.

For a vertex $z \in V(P_3) \setminus V(C)$, we see that z belongs to C_2 and to each of $C_{1,i}, \ldots, C_{1,a}$ for some $i \in \{0, \ldots, a\}$. However, even if $i \leq a - 2$, vertices of $U_{1,i+2}$ cannot be neighbours of z in G_C due to a combination of Claim 5 and Claim 8. Therefore, z may have neighbours (and so can get red edges from by contractions) only in the interior sections U_2 and $U_{1,i}$, and possibly in $U_{1,i+1}$ if z is an end of the horizontal edge h_{i+1}. Recall also that z belongs to the right path of $C_{1,i}$. Along the sequence σ_0, but before $\tau_{1,i+1}$, the vertex z has red degree at most $5 + 3 = 8$ by (II) applied to U_2 and $U_{1,i}$. After $\tau_{1,i}$ is finished, z has red degree at most $3 + 2 = 5 < 8$ by (VII) of Claim 7. So, along the rest of σ_0, (I) stays true for z with red degree at most $5 + 1 = 6$ by (III) applied possibly to $U_{1,i+1}$.

For the vertex u itself, (III) is true automatically. For $z \in V(P_1) \setminus \{u\}$ and $i \in \{0, \ldots, a\}$ being the least index such that $z \in V(C_{1,i})$, we get that the properties are true along $\tau_{1,i}$ and before by (II), and since $\tau_{1,i}$ ends, the vertex z has at most 3 red neighbours in $U_{1,i}$ by (VII). Additionally, z may get at most 1 red neighbour in $U_{1,i+1}$ (and none in $U_{1,i+2}, \ldots$) by (III), altogether at most 4, satisfying (II). In the special case of $z \in V(P_1) \setminus \{u\}$ covered by property (III) with respect to C, that is when all vertices belonging to the interior of C_1 are on levels higher than $\lambda[G](z)$, we get that this property is satisfied by (III) with respect to $C_{1,i}$, and there are no more red neighbours of z from elsewhere.

Finally, for $z \in V(P_2) \setminus \{u\}$, the conditions are simply true by (II) and possibly (III) for τ_2 and then along the whole sequence σ_0. ◀

3.3 Finishing the proof

Now we get to the core proof of Lemma 6 which will conclude our main result.

Proof of Lemma 6. We first resolve several special cases. If $U = \emptyset$, we are immediately done with the empty partial contraction sequence. So, assume $U \neq \emptyset$.

Recall the edge $f = \{v_1, v_2\} \in E(G)$ connecting the other ends of the left path P_1 and the right path P_2 of C. See again Figure 2. If v_1 has no neighbour in U, then $\{v_2, v_3\} \in E(G)$ where v_3 is the neighbour of v_1 on P_1. In such case, we simply apply Lemma 6 inductively to $P_1 - v_1$ and P_2, while the rest of the assumptions remain the same. The symmetric argument is applied when v_2 has no neighbour in U.

Otherwise, let $v_3 \in U$ be the vertex (unique in U) such that (v_1, v_2, v_3) bound a triangular face of G. Let $P \subseteq T$ be the vertical path connecting v_3 to the root r. If $v_1 \in V(P)$, then $P \supset P_1$ and we (similarly as above) apply Lemma 6 inductively to the V-separator $C^1 = P \cup P_2$ with the lid $\{v_2, v_3\}$, while the rest of the assumptions again remain the same. Note that in this case, $\lambda[G](v_1) = \lambda[G](v_3) - 1 = \lambda[G](v_2)$ since T is left-aligned. In the resulting trigraph G^1, we have the set $U^1 := V(G^1) \setminus (V(C^1) \cup W)$ that stems by contractions from the interior of C^1. There is no vertex in U^1 of level higher than $\lambda[G](v_3) \geq 2$ and at most one of level equal to $\lambda[G](v_3)$, by (IV) of Lemma 6. We contract the latter vertex with v_3, and then with the vertex of U^1 of the previous level $\lambda[G](v_1)$ unless v_1 is a neighbour of u (cf. the special case in (IV)). This clearly does not exceed red degree 8 there, and does not add new potential red neighbours to the vertices of C. Since (IV) is now satisfied, too, we are done. If $v_2 \in V(P)$, we solve the case similarly by induction applied to the V-separator $C^1 = P_1 \cup P$ with the lid $\{v_1, v_3\}$.

In all other cases, we have got a vertical-horizontal division of the subgraph G_C, with $P_3 \neq \{v_3, u\}$, with the horizontal edges $h_1, \ldots, h_a$, $h_i = \{x_i, y_i\}$, the cycles $C_{1,0}, \ldots, C_{1,a}, C_2$ and the interior sets $U_{1,0}, \ldots, U_{1,a}, U_2$, and we apply Lemma 9 to it. This way we get a level-respecting partial contraction sequence σ_0, which satisfies the properties (I), (II) and (III) of Lemma 6. Let G^0 denote the trigraph which results from G by σ_0, and let $U^0_{1,0}, \ldots, U^0_{1,a}, U^0_2$ denote the vertex sets of G^0 that stem from $U_{1,0}, \ldots, U_{1,a}, U_2$, respectively.

We first consider a subcase, that P_3 consists of a single edge $\{v_3, u_3\}$ and there is no vertex $z \in U$ in G such that $\lambda[G](z) \leq \lambda[G](u_3)$. This subcase has to be treated specially to fulfill (III) of Lemma 6. Then $a = 0$ in the vertical-horizontal division of G_C, and $\lambda[G](v_1) \leq \lambda[G](v_3) + 1 = \lambda[G](u_3) + 2$. Each of the sets $U^0_{1,0}$ and U^0_2 hence contains vertices at most on the levels $\lambda[G](v_3)$ and $\lambda[G](v_3) + 1$, by (IV) of Lemma 6.

We finish the desired partial contraction sequence from G^0 in this subcase by firstly contracting the two (if existing) vertices of $U^0_{1,0} \cup U^0_2$ on the level $\lambda[G](v_3) + 1$, and secondly by contracting each of the vertices of $U^0_{1,0} \cup U^0_2$ on the level $\lambda[G](v_3)$ with v_3. If $u_3 = u$ is the sink of C, then the only vertex on the level $\lambda[G](v_3) - 1 = \lambda[G](u)$ in G_C is u, and so every vertex of $U^0_{1,0} \cup U^0_2$ on the level $\lambda[G](v_3)$ must be adjacent to u (cf. Claim 5), and this is by a black edge due to an inductive invocation of (III). Therefore, the contractions into v_3 do not create a red edge to u. If $u_3 \neq u$, then let u'_3 denote the other vertex of $P_1 \cup P_2$ on the level $\lambda[G](u_3)$. Analogously to the previous case, one of the contractions into v_3 does not create a red edge to $\{u_3, u'_3\}$ and the other contraction can do so, but at most one red edge to each of u_3, u'_3. Therefore, (IV) is true here, and the remaining properties of Lemma 6 are fulfilled easily.

In the remaining cases, we possibly add the following bit in a sequence σ_1 after σ_0 (while this bit has not been possible in the special subcase above): If $u_3 \in V(P_1) \setminus \{u\}$ and u_3 is a neighbour of both x_1, y_1 (of h_1), then $U^0_{1,0}$ by (IV) consists of at most two vertices, which we

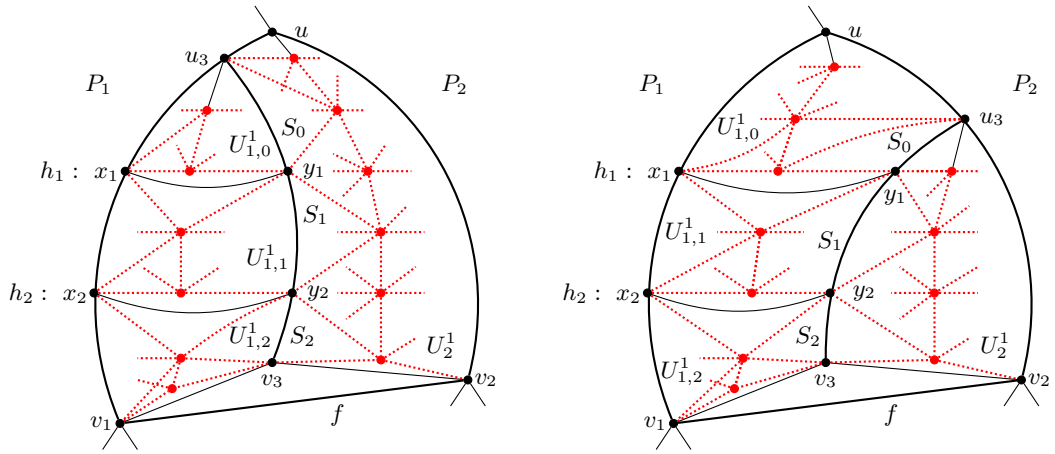


Figure 3 Proof of Lemma 6: a schematic picture of the situation after the parts of the depicted vertical-horizontal division of G_C have been recursively contracted (right before the σ_2-contractions start). The cases of u_3 on the left and right paths are not symmetric in general.

contract into one vertex in σ_1 – this move adds one red edge incident to u_3. Analogously, if $u_3 \in V(P_2) \setminus \{u\}$ and u_3 is a neighbour of both v_2, v_3 (of f_2), then we contract the at most two vertices of U_2^0 into one within σ_1. Although this contraction in σ_1 does not preserve levels, it is level-respecting by Claim 5 since $U_{1,0}$ and U_2 were interior sections of the triangles $C_{1,0}$ and C_2, respectively. In both cases, the added red edge incident to u_3 does not violate the properties of Lemma 6; this follows from the bounds in (VII) of Claim 7 which are by at least one lower than the bounds in (II) of Lemma 6, and property (III) is void for u_3 unless we have got the previous special subcase. Otherwise, we leave $\sigma_1 = \emptyset$.

After applying σ_1 to $U_{1,0}^0, \ldots, U_{1,a}^0, U_2^0$ in G^0, we get the trigraph G^1 and the sets $U_{1,0}^1, \ldots, U_{1,a}^1, U_2^1$ (which are identical to the former ones except possibly $U_{1,0}^0$ or U_2^0). See Figure 3.

In the next steps, we are going to define level-respecting partial contraction sequences $\sigma_{2,a}, \sigma_{2,a-1}, \ldots, \sigma_{2,0}$ which, when concatenated after $\sigma_0 \cdot \sigma_1$, give the desired outcome. If $a = 0$, the sequence $\sigma_{2,0}$ is going to contract the sets $U_{1,0}^1$ with $S_0 := V(P_3) \setminus \{u_3\}$ and $U_{2,0}^1 := U_2^1$. If $a > 0$, the sequence $\sigma_{2,a}$ is going to contract $U_{1,a}^1$ with the sets S_a and $U_{2,a}^1$, where $S_a \subseteq V(P_3) \setminus \{u_3\}$ and $U_{2,a}^1 \subseteq U_2$ are both the subsets of those vertices on levels greater than $\lambda[G](y_a)$. The sequence $\sigma_{2,i}$ for $0 \leq i < a$ is going to contract $U_{1,i}^1$ with the sets S_i and $U_{2,i}^1$, where $S_i \subseteq V(P_3) \setminus \{u_3\}$ and $U_{2,i}^1 \subseteq U_2^1$ are the subsets of those vertices on levels greater (if $i \geq 1$) than $\lambda[G](y_i)$ and not greater than $\lambda[G](y_{i+1})$. Of course, some of these sets may be empty, and hence some contractions may not happen.

Specifically, for $i \in \{0, \ldots, a\}$ let $p = \max_{z \in C_{1,i}} \lambda[G](z)$ and $q = 1 + \min_{z \in C_{1,i}} \lambda[G](z)$. Observe that there is no vertex in $U_{1,i}^1 \cup U_{2,i}^1$ of level lower than q or greater than p. This follows from an inductive invocation of (IV) of Lemma 6, and from the sequence σ_1. So, the union $U_1^1 := U_{1,0}^1 \cup \ldots \cup U_{1,a}^1$ has at most one vertex on each level. Likewise, each of the sets $V(P_3)$ and U_2^1 has at most one vertex on each level. The sequence $\sigma_{2,i}$ first runs over $j = p, p-1, \ldots, q$ in this order, and contracts the pair of vertices of $S_i \cup U_{2,i}^1$ of the equal level j in G^1 (or nothing if there is at most one such vertex there). In its second round, $\sigma_{2,i}$ again runs over $j = p, p-1, \ldots, q$ in this order, and contracts the vertex of level j that stems from $S_i \cup U_{2,i}^1$ in the first round, with the vertex of $U_{1,i}^1$ of equal level j in G^1.

Let σ_2 be the concatenation of the described sequences, $\sigma_2 := \sigma_{2,a} \cdot \sigma_{2,a-1} \cdot \ldots \sigma_{2,0}$ in this order, and G^2 denote the trigraph which results from G^1 by applying σ_2. Let $U^2 := V(G^2) \setminus (V(C) \cup W)$ denote the contracted vertices in the interior of C in G^2. Then G^2 and U^2 satisfy property (IV) of Lemma 6 (in the place of G^* and U^*), which is immediate from the previous definition of σ_2. It thus remains to verify the properties (I), (II) and (III) of Lemma 6 along the sequence σ_2 from G^1 to G^2, that is, for every trigraph G' along σ_2.

Denote by $U' := V(G') \setminus (V(C) \cup W)$ all interior vertices of C in G', and by $U'' := U' \setminus V(G^1)$ the (new) interior vertices that stem by σ_2-contractions from G^1 to G', and recall (from Section 3.2) that $P_{31} \supseteq P_3$ is the right path of C_1 and $P_{32} \supseteq P_3$ is the left path of C_2 in G_C.

We start with verification of (III) which has already been in parts addressed above. Regarding the sink vertex u, it has got red edges neither from the sequence σ_0 by an inductive invocation of (III), nor from the sequence σ_1. The vertices of C on levels up to $k-2$ as in (III) do not have any neighbour in U' by Claim 5. Consider the vertices $z, z' \in V(C)$ on the level $k-1$ as in (III) (if $k \geq \ell + 2$ there). If $\lambda[G](u_3) \geq k$, then no contraction on the level k happens within σ_2 (Figure 3), and so z and z' have each at most one red edge to U' by an inductive invocation of (III). Otherwise, up to symmetry, $z' = u_3$. Similarly as argued earlier in this proof, u_3 then has a black edge to $U^1_{1,0}$ (if $u_3 \in V(P_1)$) or to U^1_2 (if $u_3 \in V(P_2)$) in G^1, and so the contraction on the level k incident with this black edge does not create a new red edge to either of z, u_3. At the same time, each of z, u_3 has at most one red edge in G^1 by an inductive invocation of (III), and this stays true also (with the set U') during and after contractions on the level k within σ_2.

We move towards verification of (I). Let $z \in U'$ for the rest. If $z \in U^1_2$, then no σ_2-contraction has touched z so far. In this case z may have red edges to up to 3 vertices of $V(P_2) \cup U^1_2 \cup V(P_{32})$ of level $\lambda[G^1](z) - 1$, to 2 vertices of $V(P_2) \cup V(P_{32})$ of level $\lambda[G^1](z)$, and to 2 vertices of $V(P_{32}) \cup U^1_2 \cup U''$ of level $\lambda[G^1](z) + 1$, altogether at most 7. Note that there is no edge from z to the vertex of P_2 of level $\lambda[G^1](z) + 1$ by (VI) of Claim 7. If $z \in V(P_3) \setminus \{u_3\}$, then similarly, z may have red edges to up to $2 + 2$ vertices of $U^1_1 \cup U^1_2$ on the levels $\lambda[G^1](z) - 1$ and $\lambda[G^1](z)$, and to up to 2 vertices of $U^1_1 \cup U^1_2 \cup U''$ on the level $\lambda[G^1](z) + 1$. The case of $z \in U^1_1$ (not-yet touched by a σ_2-contraction) is similarly easy.

Assume now that $z \in U''$ has been created in G' by a contraction of $z_2 \in U^1_2$ and $z_3 \in V(P_3) \setminus \{u_3\}$ (i.e., within the first round of some $\sigma_{2,i}$ above), but z is not contracted with a vertex of U^1_1 yet. Let $t \in V(P_1) \setminus V(P_3)$ denote the possible (unless equal to u_3) vertex of P_1 of level $\lambda[G^1](z_3) - 1$. Then there is no edge in G^1 from t to z_2 by planarity, and no from t to z_3 by (VI) of Claim 7. The same applies to the possible vertex $t' \in U^1_1$ of level $\lambda[G^1](z_3) - 1$. Consequently, z may have red edges to up to 3 vertices of $V(P_2) \cup U^1_2 \cup V(P_{32})$ of level $\lambda[G'](z) - 1$, to 3 vertices of $V(P_2) \cup V(U^1_1) \cup V(P_1)$ of level $\lambda[G'](z)$, and to up to 3 vertices of $U'' \cup V(U^1_1) \cup V(P_1)$ of level $\lambda[G'](z)+1$, again using (VI). This sums to $3+3+3 = 9$, but we are going to show that this maximum of 9 cannot be achieved. Let $z_1 \in V(P_1)$ be such that $\lambda[G'](z_1) = \lambda[G'](z)$. If $\{z_1, z_3\} \notin E(G)$, then no red edge $\{z_1, z\}$ is created by the current contraction and the sum is at most 8, as needed. If $\{z_1, z_3\} = h_i \in E(G)$, then the sequence $\sigma_{2,i}$ has already contracted $S_i \cup U^1_{1,i}$ into U'', and so there are only 2 red neighbours of z in $U'' \cup V(P_1)$ on the level $\lambda[G'](z) + 1$, again summing to at most 8. If $\{z_1, z_3\} \in E(G)$, but $\{z_1, z_3\}$ has not been chosen as any h_i in the vertical-horizontal division above, then there are no vertices in $U^1_2 \cup (V(P_3) \setminus \{u_3\})$ on the level $\lambda[G'](z) - 1$, and so the sum is at most 8.

Assume that $z \in U''$ has already been created in G' by a contraction of all vertices in $U^1_2 \cup V(P_3) \cup U^1_1$ of the same level. Let this contraction be part of $\sigma_{2,i}$ for some $0 \leq i \leq a$. Then z may have red edges to up to 2 vertices of $V(P_2) \cup V(P_1)$ of level $\lambda[G'](z)$, to 2

vertices of $U'' \cup V(P_1)$ of level $\lambda[G'](z)+1$ (but not to $V(P_2)$ due to (VI)), and to vertices of $V(P_2) \cup U_2^1 \cup V(P_3) \cup V(U_1^1) \cup U'' \cup V(P_1) =: Y$ of level $\lambda[G'](z)-1$. However, at most 4 of the vertices of Y are potential red neighbours of z (so summing to at most 8), as we now show. If contractions on the level $\lambda[G'](z)-1$ are part of $\sigma_{2,i}$, too, then red neighbours of z in Y of level $\lambda[G'](z)-1$ actually belong to $V(P_2) \cup U'' \cup V(U_1^1) \cup V(P_1)$ with an upper bound of 4. Otherwise, if contractions on the level $\lambda[G'](z)-1$ are part of $\sigma_{2,i-1}$, then there is no edge from z to a vertex of U_{i-1}^1, or U_{i-1}^1 on the level $\lambda[G'](z)-1$ has already been contracted into U'', too. Then red neighbours of z in Y of level $\lambda[G'](z)-1$ belong to $V(P_2) \cup U_2^1 \cup V(P_3) \cup V(P_1)$ or to $V(P_2) \cup U'' \cup V(P_1)$, and we again get a bound of 4.

Finally, we want to verify (II) of Lemma 6. Consider $z \in V(P_2) \setminus \{u\}$. By the definition of σ_2, the vertex z may have at most one red neighbour of each level in U' (at any moment of σ_2). Then the bound of at most 3 red edges from z to U' follows immediately in view of Claim 5. Consider now $z \in V(P_1) \setminus \{u\}$, which is a bit more complicated case. On each of the levels $\lambda[G^1](z)-1$, $\lambda[G^1](z)$ and $\lambda[G^1](z)+1$ of U', there are clearly at most 2 red neighbours of z. Although, we now show that the maximum sum of 6 cannot be achieved. If $z = u_3$, then there is actually at most red neighbour of z on the level $\lambda[G^1](z)-1$. Otherwise, we denote the following vertices of G^1 of level $\lambda[G^1](z)+1$ by z_1, z_2, z_3 such that $z_1 \in U_1^1$, $z_2 \in U_2^1$, $z_3 \in V(P_3)$, and $\lambda[G^1](z_1) = \lambda[G^1](z_2) = \lambda[G^1](z_3) = \lambda[G^1](z)+1$. Then z_3 has no edge to $z \neq u_3$ by (VI) of Claim 7, and z_2 has no edge to z by planarity. If $z_3 \in U'$ (i.e., not contracted yet), then only z_1 may be a red neighbour of z. If z_2 and z_3 have already been contracted in G', but $z_1 \in U'$, then the new vertex again has no edge to z. Finally, if all of z_1, z_2, z_3 have been contracted in G', then U' has only (this) one vertex of level $\lambda[G^1](z)+1$. In any case, z has at most 5 red neighbours in U'.

We have verified all conditions of Lemma 6 for the partial contraction sequence $\sigma_0 \cdot \sigma_1 \cdot \sigma_2$, and so we can set $G^* := G^2$ and the proof is done. ◀

Proof of Theorem 1 (the algorithmic part). We can construct a simple plane triangulation $G \supseteq H$ in linear time using standard planarity algorithms, and then construct a left-aligned BFS tree $T \subseteq G$ again in linear time by Lemma 4. In the rest, we straightforwardly implement the recursive vertical-horizontal division of G as used in the proof of Lemma 6, and construct the contraction sequence of H on return from the recursive calls as defined in the proof. Note that we do not need at all to construct the intermediate trigraphs along the constructed contraction sequence, and so the construction of the sequence is very easy – each recursive call returns just a simple list of the vertices which stem from the recursive contractions, indexed by the levels. Then these (up to) two lists are easily in linear time "merged" together with the dividing path P_3, as specified by the proof of Lemma 6, into the resulting list of this call.

We may account total runtime in the "division part" of the algorithm to the edge(s) of v_3 into v_1 or v_2 and the edges of the path P_3 starting in v_3 in each call of the recursion, and these edges are not counted multiple times in different branches of the recursion. Likewise, runtime of the "merging" part of each recursive call can be counted to the individual steps of the resulting contraction sequence, which is of linear length. Hence, altogether, the algorithm runs in linear time. ◀

4 Proof of Theorem 2

On a high level, the proof will still proceed in the same way as in [16], and will prove the same bound. However, there are significant changes in the technical details, in which ideas from the previous section can save a lot of difficulties of the cumbersome proof from [16].

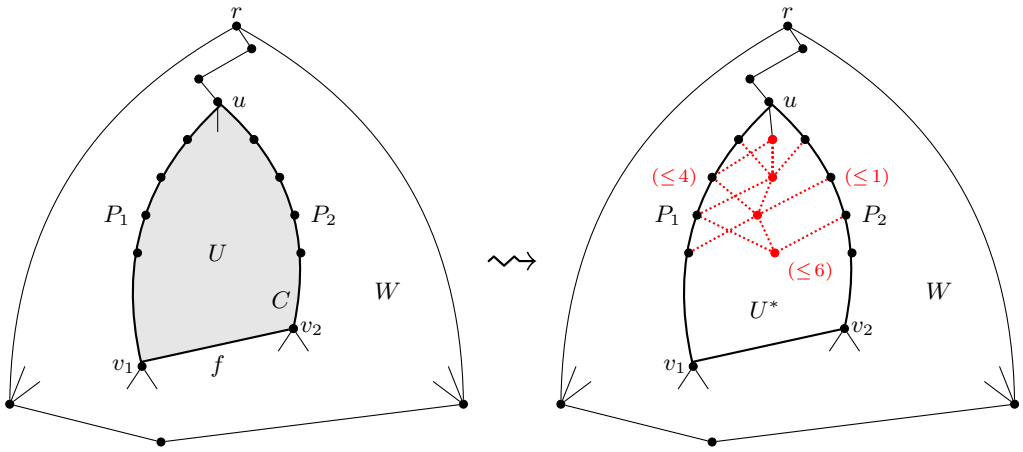


Figure 4 (left) The setup of Lemma 11, where P_1 and P_2 are the left and right paths of the chosen V-separator C. (right) The outcome of the claimed partial contraction sequence which contracts only vertices of U inside the shaded region from the left, and which maintains bounded red degrees in the region and on its boundary C.

In a nutshell, the bipartite case carries two major differences from the proof of the general planar case in Section 3:

- Since our graph is now bipartite, we will work with a plane quadrangulation (instead of a triangulation). However, with a suitable detailed analysis, it does not bring any significant new challenges to the proof.
- Since, again, our graph is bipartite, we immediately get that in any BFS layering, each layer is an independent set, and so we will never create a red edge inside the same layer. This is the crucial saving which allows us to derive a better upper bound on the red degree along the constructed sequence.

Before proceeding further, we need to adjust the concept of a level-respecting contraction sequence (beacuse of the fact that the definition from Section 2 possibly allowed to create new (red) edges inside the same level).

A partial contraction sequence of G is *bi-level-respecting* if every step contracts, in a trigraph G' along the sequence, only a pair $x, y \in V(G')$ such that the following inductively holds; the levels of x and y are the same, i.e. $\lambda[G'](y) = \lambda[G'](x)$, or all neighbours of y (red or black) in G' are on the level $\lambda[G'](x) + 1$, i.e. $\lambda[G'](z) = \lambda[G'](x) + 1$ is true for all z such that $\{y, z\} \in E(G')$. Again, we easily get by induction as in Claim 5:

▷ Claim 10. Let a trigraph G' result from a bi-level-respecting partial contraction sequence of a bipartite connected graph G. Then any vertex $z \in V(G')$ may have neighbours (red or black) only on the levels $\lambda[G](z) - 1$ and $\lambda[G](z) + 1$. In particular, the trigraph G' is again bipartite. Moreover, z must have some neighbour on the level $\lambda[G](z) - 1$.

Our proof is again by induction, precisely as set up in the following lemma. We illustrate this lemma in Figure 4. Before starting, note that a plane quadrangulation is always bipartite, and so it has all cycles (not only the faces) of length at least 4.

▶ **Lemma 11.*** *Let G be a simple plane quadrangulation, and T be a left-aligned BFS tree of G rooted at a vertex $r \in V(G)$ of the outer face and defining the initial levels $\lambda[G](\cdot)$. Assume that a cycle C of G is a V-separator of G, that G_C is the subgraph of G bounded by C, and u*

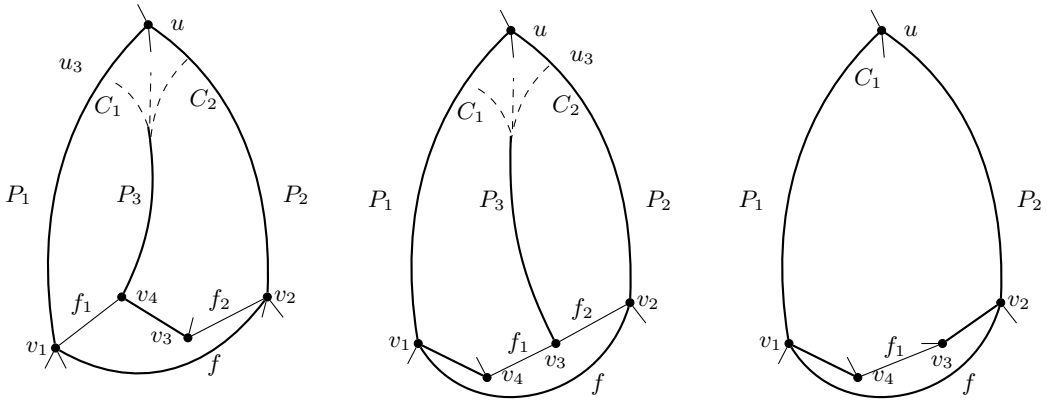


Figure 5 Three schematic cases of decomposing the drawing of G_C into subregions (bounded by C_1 and C_2) discussed in Lemma 13. They generally cover all possibilities in which $v_3, v_4 \notin V(P_1 \cup P_2)$. Vertical positions of the vertices of the 4-cycle $A = (v_1, v_2, v_3, v_4)$ outline their levels in G. Note that, in the right-most case, we have only one "nested" V-separator C_1, which is however not equal to $C = (P_1 \cup P_2) + f$, but instead C_1 passes through v_2, v_3, v_4, v_1 and has the lid edge $f_1 = \{v_3, v_4\}$.

is the sink of C. Let the distance of u from the root r be ℓ, so $\lambda[G](u) = \ell$, and the maximum distance from a vertex of C to r be $m \geq \ell + 2$. Let $U := V(G_C) \setminus V(C)$ be the interior vertices of C, and denote by $W := V(G) \setminus (V(C) \cup U)$ the set of the "remaining" vertices.

Then there exists a bi-level-respecting partial contraction sequence of G which contracts only pairs of vertices that are in or stem from U, results in a trigraph G^*, and satisfies the following conditions for every trigraph G' along this sequence from G to G^*:

- **(I)'** *For $U' := V(G') \setminus (V(C) \cup W)$ (which are the vertices that are in or stem from U in G'), every vertex of U' in G' has red degree at most 6,*
- **(II)'** *every vertex of the left path of C has at most 4 red neighbours and every vertex of the right path of C has at most 1 red neighbour in U',*
- **(III)'** *the sink u of C has no red neighbour in U',*
- **(IV)'** *if the next step of the sequence is going to contract a pair $x, y \in U'$ such that $\lambda[G'](y) > \lambda[G'](x)$, then y has no neighbour in the right path of C, and*
- **(V)'** *at the end of the partial contraction sequence, for the set $U^* := V(G^*) \setminus (V(C) \cup W)$ that stems from U in G^*, we have that if $z \in U^*$ is of level i, then $\ell < i \leq m + 1$ and z is the only vertex in U^* of level i.*

Lemma 11 already easily implies the first combinatorial part of Theorem 2, as we have seen with Theorem 1 in Section 3.1. Details of the algorithmic part again tightly follow the detailed proof steps which are present in the full paper.

In the proof of Lemma 11, we proceed analogously to Section 3. Namely, we start with a decomposition step analogous to Lemma 9 and illustrated in Figure 5. With a bit of technical work, we prove:

▶ **Lemma 13.*** *Assume the setting of Lemma 11, and with respect to it, let $U \neq \emptyset$ and $A \subseteq G_C$, $A = (v_1, v_2, v_3, v_4)$, denote the cycle bounding the 4-face incident to the lid edge f of the V-separator C and drawn in the closed disk of C.*

a) *There exists a vertical path $P_3 \subseteq G_C$ (internally disjoint from $C \cup A$ and possibly empty) such that the plane subgraph $C \cup A \cup P_3$ has two (if $P_3 = \emptyset$) or three distinct bounded faces, one of them being the face of A. The one or two bounded faces of $C \cup A \cup P_3$ other than that of A are bounded by cycles C_1 and C_2, where $v_1 \in V(C_1)$, and each of C_1 and C_2 (or just C_1 if $P_3 = \emptyset$) is again a V-separator whose lid edge is from $E(A) \setminus \{f\}$.*

b) *Assume now that $C \cup A \cup P_3$ has three bounded faces. Then $C_1 \cap C_2 = P_3$, and the sinks of C_1 and C_2 are in some order the sink u of C and the end u_3 of P_3 not in A (which may be the same vertex). If τ_i, $i = 1, 2$, is the bi-level-respecting partial contraction sequence of G obtained by inductively applying Lemma 11 to the cycle C_i, then the concatenation $\sigma_0 := \tau_2 \cdot \tau_1$ of these two sequences is a bi-level-respecting partial contraction sequence of G which satisfies the properties* (I)*' to* (IV)*' of Lemma 11.*

Then, we follow on the partial contraction sequence of Lemma 13 analogously to the proof in Section 3.3, albeit with slightly simpler arguments thanks to a simpler decomposition step with only at most two subregions. In this way we finish both Lemma 11 and Theorem 2, and the details are now left for the full paper [17].

5 Concluding Remarks

We have further improved by one the previous best upper bound [16] on the twin-width of planar graphs. This seemingly small improvement has required a careful reconsideration of the previous method and several new ideas, and although our new approach has simplified some cumbersome technical details in [16], new technical difficulties emerged which makes some parts of the proof again quite technical. This is probably to be expected since we are now very close to the currently best lower bound of 7 on the twin-width of planar graphs [19].

To recapitulate the fine improvements leading to the upper bound of 8 on the twin-width compared to previous larger bounds in [12, 18] and [3]; we communicate that the biggest (numerical) jump comes from the use of a specially tailored BFS-based decomposition formulated in Section 3.2, but we regard as the most important contribution in the quest the use of a left-aligned BFS tree (Section 2), which essentially "slashes down" additional up to three possible red neighbours from the analysis in the proof of Lemma 6. While both previous improvements have been introduced already in [16], the use of the "horizontal items" in the vertical-horizontal division of Section 3.2 then gives a final touch improving the bound to 8 (while 9 seemed to be unbeatable without this final trick).

Related to the twin-width is the notion of reduced bandwidth [12] which, informally stating, requires the subgraph induced by the red edges (along the sequence) to not only have bounded degrees, but also bounded bandwidth. Strictly speaking, as our construction of the contraction sequence creates arbitrarily large "red grids" in some cases, it does not directly imply any constant upper bound on the reduced bandwidth of planar graphs. However, a simple modification of the construction (informally, delaying contractions that would create red edges to the vertices of P_2 as in Figure 4) can easily bring a reasonable two-digit upper bound on the reduced bandwidth of planar graphs, which can possibly be further tightened with a specialized refined argument.

Besides the core question of the maximum twin-width of planar graphs, one may also reconsider the fact that our proof method is (distantly) based on the proof of the product structure of planar graphs [14] and ask whether we could possibly improve the product structure over the currently best variant in [21]. Unfortunately, our recursive decomposition of planar graphs is very tailored to the purpose of proving a good upper bound on the twin-width and it currently does not seem to yield an improvement in the planar product struture, or in the maximum queue number of planar graphs. This direction, however, is the subject of our ongoing research.

In the end we would like to dwell on the very idea of left-aligned BFS trees from Section 2. This seems like a quite general idea about planar graphs, related to other specialized BFS- and DFS-search routines in the algorithmic world, but we have not found this exact idea

anywhere in the published literature (the existing related concepts we are aware of do not feature the BFS property). We believe that this new idea could possibly find its use in other problems regarding planar graphs and drawings.

To finally conclude, the problem to determine the exact maximum value of the twin-width over all planar graphs is still open, but our continuing research suggests that the value of 7 is much more likely (than 8) to be the right answer. Likewise, the problem to determine the exact maximum value of the twin-width over bipartite planar graphs is open, and we cannot now decide whether the value of 6 is the right maximum value over bipartite planar graphs, or whether the upper bound may possibly be 5 (while an upper bound lower than 5 is not likely since a bipartite construction analogous to [19] seems to exclude it, but we are not aware of this claim being written up as a formal statement).

References

1 Jungho Ahn, Kevin Hendrey, Donggyu Kim, and Sang-il Oum. Bounds for the twin-width of graphs. *CoRR*, abs/2110.03957, 2021. `arXiv:2110.03957`.

2 Jakub Balabán and Petr Hliněný. Twin-width is linear in the poset width. In *IPEC*, volume 214 of *LIPIcs*, pages 6:1–6:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

3 Michael A. Bekos, Giordano Da Lozzo, Petr Hliněný, and Michael Kaufmann. Graph product structure for h-framed graphs. *CoRR*, abs/2204.11495v1, 2022. `arXiv:2204.11495v1`.

4 Pierre Bergé, Édouard Bonnet, and Hugues Déprés. Deciding twin-width at most 4 is NP-complete. In *ICALP*, volume 229 of *LIPIcs*, pages 18:1–18:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

5 Édouard Bonnet, Dibyayan Chakraborty, Eun Jung Kim, Noleen Köhler, Raul Lopes, and Stéphan Thomassé. Twin-width VIII: delineation and win-wins. In *IPEC*, volume 249 of *LIPIcs*, pages 9:1–9:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

6 Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width II: small classes. In *SODA*, pages 1977–1996. SIAM, 2021.

7 Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width III: max independent set, min dominating set, and coloring. In *ICALP*, volume 198 of *LIPIcs*, pages 35:1–35:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

8 Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, Pierre Simon, Stéphan Thomassé, and Szymon Torunczyk. Twin-width IV: ordered graphs and matrices. In *STOC*, pages 924–937. ACM, 2022.

9 Édouard Bonnet, Eun Jung Kim, Amadeus Reinald, and Stéphan Thomassé. Twin-width VI: the lens of contraction sequences. In *SODA*, pages 1036–1056. SIAM, 2022.

10 Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width I: tractable FO model checking. In *FOCS*, pages 601–612. IEEE, 2020.

11 Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width I: tractable FO model checking. *J. ACM*, 69(1):3:1–3:46, 2022.

12 Édouard Bonnet, O-joung Kwon, and David R. Wood. Reduced bandwidth: a qualitative strengthening of twin-width in minor-closed classes (and beyond). *CoRR*, abs/2202.11858, 2022. `arXiv:2202.11858`.

13 Édouard Bonnet, Jaroslav Nesetril, Patrice Ossona de Mendez, Sebastian Siebertz, and Stéphan Thomassé. Twin-width and permutations. *CoRR*, abs/2102.06880, 2021. `arXiv:2102.06880`.

14 Vida Dujmovic, Gwenaël Joret, Piotr Micek, Pat Morin, Torsten Ueckerdt, and David R. Wood. Planar graphs have bounded queue-number. *J. ACM*, 67(4):22:1–22:38, 2020. `doi:10.1145/3385731`.

15 Jakub Gajarský, Michal Pilipczuk, Wojciech Przybyszewski, and Szymon Torunczyk. Twin-width and types. In *ICALP*, volume 229 of *LIPIcs*, pages 123:1–123:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

16 Petr Hliněný. Twin-width of planar graphs is at most 9, and at most 6 when bipartite planar. *CoRR*, abs/2205.05378, 2022. arXiv:2205.05378.

17 Petr Hliněný and Jan Jedelský. Twin-width of planar graphs is at most 8, and at most 6 when bipartite planar. *CoRR*, abs/2210.08620, 2022. arXiv:2210.08620.

18 Hugo Jacob and Marcin Pilipczuk. Bounding twin-width for bounded-treewidth graphs, planar graphs, and bipartite graphs. In *WG*, volume 13453 of *Lecture Notes in Computer Science*, pages 287–299. Springer, 2022.

19 Daniel Král and Ander Lamaison. Planar graph with twin-width seven. *CoRR*, abs/2209.11537, 2022. arXiv:2209.11537.

20 Michal Pilipczuk, Marek Sokolowski, and Anna Zych-Pawlewicz. Compact representation for matrices of bounded twin-width. In *STACS*, volume 219 of *LIPIcs*, pages 52:1–52:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

21 Torsten Ueckerdt, David R. Wood, and Wendy Yi. An improved planar graph product structure theorem. *CoRR*, abs/2108.00198, 2021. arXiv:2108.00198.

A Sparse Johnson-Lindenstrauss Transform Using Fast Hashing

Jakob Bæk Tejs Houen
BARC, Department of Computer Science, University of Copenhagen, Denmark

Mikkel Thorup
BARC, Department of Computer Science, University of Copenhagen, Denmark

Abstract

The *Sparse Johnson-Lindenstrauss Transform* of Kane and Nelson (SODA 2012) provides a linear dimensionality-reducing map $A \in \mathbb{R}^{m \times u}$ in ℓ_2 that preserves distances up to distortion of $1 + \varepsilon$ with probability $1 - \delta$, where $m = O(\varepsilon^{-2} \log 1/\delta)$ and each column of A has $O(\varepsilon m)$ non-zero entries. The previous analyses of the Sparse Johnson-Lindenstrauss Transform all assumed access to a $\Omega(\log 1/\delta)$-wise independent hash function. The main contribution of this paper is a more general analysis of the Sparse Johnson-Lindenstrauss Transform with less assumptions on the hash function. We also show that the *Mixed Tabulation hash function* of Dahlgaard, Knudsen, Rotenberg, and Thorup (FOCS 2015) satisfies the conditions of our analysis, thus giving us the first analysis of a Sparse Johnson-Lindenstrauss Transform that works with a practical hash function.

2012 ACM Subject Classification Theory of computation → Random projections and metric embeddings; Theory of computation → Pseudorandomness and derandomization

Keywords and phrases dimensionality reduction, hashing, concentration bounds, moment bounds

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.76

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2305.03110`

Funding Research supported by Investigator Grant 16582, Basic Algorithms Research Copenhagen (BARC), from the VILLUM Foundation.

1 Introduction

Dimensionality reduction is an often applied technique to obtain a speedup when working with high dimensional data. The basic idea is to map a set of points $X \subseteq \mathbb{R}^u$ to a lower dimension while approximately preserving the geometry. The Johnson-Lindenstrauss lemma [24] is a foundational result in that regard.

▶ **Lemma 1** ([24]). *For any $0 < \varepsilon < 1$, integers n, u, and $X \subseteq \mathbb{R}^u$ with $|X| = n$, there exists a map $f \colon X \to \mathbb{R}^m$ with $m = O(\varepsilon^{-2} \log n)$ such that*

$$\forall w, w' \in X,\ \left| \|f(w) - f(w')\|_2 - \|w - w'\|_2 \right| \leq \varepsilon \|w - w'\|_2 .$$

It has been shown in [6, 30] that the target dimension m is optimal for nearly the entire range of n, u, ε. More precisely, for any n, u, ε there exists a set of points $X \subseteq \mathbb{R}^u$ with $|X| = n$ such that for any map $f \colon X \to \mathbb{R}^m$ where the Euclidean norm is distorted by at most $(1 \pm \varepsilon)$ must have $m = \Omega(\min\{u, n, \varepsilon^{-2} \log(\varepsilon^2 n)\})$.

All known proofs of the Johnson-Lindenstrauss lemma constructs a linear map f. The original proof of Johnson and Lindenstrauss [24] chose $f(x) = \Pi x$ where $\Pi \in \mathbb{R}^{m \times u}$ is an appropriately scaled orthogonal projection into a random m-dimensional subspace. Another simple construction is to set $f(x) = \frac{1}{\sqrt{m}} Ax$ where $A \in \mathbb{R}^{m \times u}$ and each entry is an independent Rademacher variable.[1] In both cases, it can be shown that as long as $m = \Omega(\varepsilon^{-2} \log 1/\delta)$ then

$$\forall w \in \mathbb{R}^u, \ \Pr\left[\left|\|f(w)\|_2^2 - \|w\|_2^2\right| \geq \varepsilon \|w\|_2^2\right] \leq \delta. \tag{1}$$

The Johnson-Lindenstrauss lemma follows by setting $\delta < 1/\binom{n}{2}$ and taking $w = z - z'$ for all pairs $z, z' \in X$ together with a union bound. (1) is also known as the distributional Johnson-Lindenstrauss lemma and it has been shown that the target dimension m is tight, more precisely, m must be at least $\Omega(\min\left\{u, \varepsilon^{-2} \log 1/\delta\right\})$ [23, 26].

Sparse Johnson-Lindenstrauss Transform

One way to speed up the embedding time is replacing the dense A of the above construction by a sparse matrix. The first progress in that regard came by Achlioptas in [3] who showed that A can be chosen with i.i.d. entries where $A_{ij} = 0$ with probability $2/3$ and otherwise A_{ij} is chosen uniformly in $\pm\sqrt{\frac{3}{m}}$. He showed that this construction can achieve the same m as the best analyses of the Johnson-Lindenstrauss lemma. Hence this achieves essentially a 3x speedup, but the asymptotic embedding time is still $O(m \|x\|_0)$ where $\|x\|_0$ is number of non-zeros of x.

Motivated by improving the asymptotic embedding time, Kane and Nelson in [28], following the work in [14, 27, 8], introduced the Sparse Johnson-Lindenstrauss Transform which maps down to essentially optimal dimension $m = O(\varepsilon^{-2} \log n)$ and only has $s = O(\varepsilon^{-1} \log n)$ non-zeros entries per column. This speeds up the embedding time to $O(\varepsilon^{-1} \log n \|x\|_0) = O(\varepsilon m \|x\|_0)$ thus improving the embedding time by a factor of ε^{-1}. It nearly matches a sparsity lower bound by Nelson and Nguyen [31] who showed that any sparse matrix needs at least $s = \Omega(\varepsilon^{-1} \log(n)/\log(1/\varepsilon))$ non-zeros per column.

Using Hashing

When the input dimension, u, is large it is not feasible to store the matrix A explicitly. Instead, we use a hash function to calculate the non-zero entries of A. Unfortunately, the previous analyses of the Sparse Johnson-Lindenstrauss Transform [28, 10] assume access to a $\Omega(\log 1/\delta)$-wise independent hash function which is inefficient. This motivates the natural question:

What are the sufficient properties we need of the hash function for a Sparse Johnson-Lindenstrauss Transform to work?

The goal of this work is to make progress on this question. In particular, we provide a new analysis of a Sparse Johnson-Lindenstrauss Transform with fewer assumptions on the hash function. This improved analysis allows us to conclude that there exists a Sparse Johnson-Lindenstrauss Transform that uses Mixed Tabulation hashing which is efficient.

[1] A Rademacher variables, X, is a random variable that is chosen uniformly in ± 1, i.e., $\Pr[X = 1] = \Pr[X = -1] = \frac{1}{2}$.

Mixed Tabulation Hashing

Before introducing Mixed Tabulation hashing, we will first discuss *Simple Tabulation hashing* which was introduced by Zobrist [39]. Simple Tabulation hashing takes an integer parameter $c > 1$, and we view a key $x \in [u] = \{0, \dots, u-1\}$ as a vector of c characters, $x_0, \dots, x_{c-1} \in \Sigma = [u^{1/c}]$. For each character, we initialize a fully random table $T_i \colon \Sigma \to [2^r]$ and the hash value of x is then calculated as

$$h(x) = T_0[x_0] \oplus \ldots \oplus T_{c-1}[x_{c-1}],$$

where $\oplus$ is the bitwise XOR-operation. We say that h is a Simple Tabulation hash function with c characters.

We can now define *Mixed Tabulation hashing* which is a variant of Simple Tabulation hashing that was introduced in [11]. As with Simple Tabulation hashing, Mixed Tabulation hashing takes $c > 1$ as a parameter, and it takes a further integer parameter $d \geq 1$. Again, we view a key $x \in [u]$ as vector of c characters, $x_0, \dots, x_{c-1} \in \Sigma = [u^{1/c}]$. We then let $h_1 \colon \Sigma^c \to [2^r]$, $h_2 \colon \Sigma^c \to \Sigma^d$, and $h_3 \colon \Sigma^d \to [2^r]$ be independent Simple Tabulation hashing. Mixed Tabulation hashing is then defined as follows

$$h(x) = h_1(x) \oplus h_3(h_2(x)).$$

We say that h a mixed tabulation hash function with c characters and d derived characters. We call $h_2(x) \in \Sigma^d$ the *derived* characters. Mixed Tabulation hashing can be efficiently implemented by storing h_1 and h_2 as a single table with entries in $[2^r] \times \Sigma^d$, so the whole hash function can be computed with just $c + d$ lookups.

Our Contributions

Our main contribution is a new analysis of a Sparse Johnson-Lindenstrauss Transform that does not rely on the high independence of the hash function. Instead we show that it suffices that the hash function supports a decoupling-decomposition combined with strong concentration bounds.

We show that Mixed Tabulation hashing satisfies these conditions. This gives the first instance of a practical hash function that can support a Sparse Johnson-Lindenstrauss Transform.

1.1 Sparse Johnson-Lindenstrauss Transform

As mentioned earlier, the Sparse Johnson-Lindenstrauss Transform was introduced by Kane and Nelson [28] and they provided two different constructions with the same sparsity. Later a simpler analysis was given in [10] which also generalized the result to a more general class of constructions. In this paper, we will only focus on one of the constructions which is described below.

Before we discuss the construction of the Sparse Johnson-Lindenstrauss Transform, we will first consider the related CountSketch which was introduced in [9] and was analyzed for dimensionality reduction in [36]. In CountSketch, we construct the matrix A as follows: We pick a pairwise independent hash function, $h \colon [u] \to [m]$, and a 4-wise independent sign function $\sigma \colon [u] \to \{-1, 1\}$. For each $x \in [u]$, we set $A_{h(x),x} = \sigma(x)$ and the rest of the x'th column to 0. Clearly, this construction has exactly 1 non-zero entry per column. It was shown in [36] that if $m = \Omega(\varepsilon^{-2}\delta^{-1})$ then it satisfies the distributional Johnson-Lindenstrauss lemma, Equation (1). The result follows by bounding the second moment of $\|Ax\|_2^2 - \|x\|_2^2$ for any $x \in \mathbb{R}^d$ and then apply Chebyshev's inequality.

The bad dependence in the target dimension, m, on the failure probability, δ, is because we only use the second moment. So one might hope that you can improve the dependence by looking at higher moments instead. Unfortunately, it is not possible to improve the dependence for general $x \in \mathbb{R}^d$, and it is only possible to improve the dependence if $\|x\|_\infty^2 / \|x\|_2^2$ is small. Precisely, how small $\|x\|_\infty^2 / \|x\|_2^2$ has to be, has been shown in [17]. So to improve the dependence on δ, we need to increase the number of non-zero entries per column.

We are now ready to describe the construction of the Sparse Johnson-Lindenstrauss Transform. The construction is to concatenate s CountSketch matrices and scale the resulting matrix by $\frac{1}{\sqrt{s}}$. This clearly gives a construction that has s non-zero entries per column and as it has been shown in [28, 10] if $s = \Omega(\varepsilon^{-1} \log(1/\delta))$ then we can obtain the optimal target dimension $m = O(\varepsilon^{-2} \log(1/\delta))$. More formally, we construct the matrix A as follows:

1. We pick a hash function, $h\colon [s] \times [u] \to [m/s]$ and a sign function $\sigma\colon [s] \times [u] \to \{-1, 1\}$.
2. For each $x \in [u]$, we set $A_{i \cdot m/s + h(i,x), x} = \frac{\sigma(i,x)}{\sqrt{s}}$ for every $i \in [s]$ and the rest of the x'th column to 0.

In the previous analyses [28, 10], it was shown that if h and σ are $\Omega(\log 1/\delta)$-wise independent then the construction works. Unfortunately, it is not practical to use a $\Omega(\log 1/\delta)$-wise independent hash function so the goal of this work is to obtain an analysis of a Sparse Johnson-Lindenstrauss Transform with fewer assumptions about the hash function. In particular, we relax the assumptions of the hash function, h, and the sign function, σ, to just satisfying a decoupling-decomposition and a strong concentration property. The formal theorem is stated in Section 3.

We also show that Mixed Tabulation satisfies these properties and thus that the Sparse Johnson-Lindenstrauss Transform can be implemented using Mixed Tabulation. Let us describe more formally, what we mean by saying that Mixed Tabulation can implement the Sparse Johnson-Lindenstrauss Transform. We let $h_1\colon \Sigma^c = [u] \to [m/s]$, $h_2\colon \Sigma^c \to \Sigma^d$, and $h_3\colon \Sigma^d \to [m/s]$ be the independent Simple Tabulation hash functions that implement the Mixed Tabulation hash function, $h_1(x) \oplus h_3(h_2(x))$. We then extend it to the domain $[s] \times [u]$ as follows:

1. Let $h_2'\colon [s] \times \Sigma^c \to \Sigma^d$ be defined by $h_2'(i,x) = h_2(x) \oplus \underbrace{(i, \ldots, i)}_{d \text{ times}}$, i.e., each derived character gets xor'ed by i.
2. We then define $h\colon [s] \times [u] \to [m/s]$ and $\sigma\colon [s] \times [u] \to \{-1, 1\}$ by $h(i,x) = h_1(x) \oplus h_3(h_2'(i,x))$ and $\sigma(i,x) = \sigma_1(x) \cdot \sigma_3(h_2'(i,x))$, where h_1 and h_3 are the Simple Tabulation hash functions described above, and $\sigma_1\colon \Sigma^c \to \{-1, 1\}$ and $\sigma_3\colon \Sigma^d \to \{-1, 1\}$ are independent Simple Tabulation functions.

1.2 Hashing Speed

When we use tabulation schemes, it is often as a fast alternative to $\Omega(\log n)$-independent hashing. Typically, we implement a q-independent hash function using a degree $q-1$ polynomial in $O(q)$ time, and Siegel [34] has proved that this is best possible unless we use large space. More precisely, for some key domain $[u]$, if we want to do $t < q$ memory accesses, then we need space at least $u^{1/t}$. Thus, if we want higher than constant independence but still constant evaluation times, then we do need space $u^{\Omega(1)}$. In our application, we have to compute many hash values simultaneously, so an alternative strategy would be to evaluate the polynomial using multi-point evaluation. This would reduce the time per hash value to $O(\log^2 q)$ but this is still super constant time.

With tabulation hashing, we use tables of size $O(|\Sigma|)$ where $|\Sigma| = u^{1/c}$ and $c = O(1)$. The table lookups are fast if the tables fit in cache, which is easily the case for 8-bit characters. In connection with each lookup, we do a small number of very fast AC^0 operations: a cast, a bit-wise xor, and a shift. This is incomparable to polynomial in the sense of fast cache versus multiplications, but the experiments from [1, Table 1] found Simple Tabulation hashing to be faster than evaluation a 2-wise independent polynomial hashing.

Tabulation schemes are most easily compared by the number of lookups. Storing h_1 and h_2 in the same table, Mixed Tabulation hashing uses $c + d$ lookups. With $d = c$, the experiments from [1] found Mixed Tabulation hashing to be slightly more than twice as slow as Simple Tabulation hashing, and the experiments from [12] found Mixed Tabulation hashing to be about as fast as 3-wise independent polynomial hashing. This motivates our claim that Mixed Tabulation hashing is practical.

In theory, we could also use a highly independent hash function that uses large space, but we don't know of any efficient construction. Siegel states about his construction, it is "far too slow for any practical application" [34], and while Thorup [35] has presented a simpler construction than Siegel's, it is still not efficient. The experiments in [1] found it to be more than an order magnitude slower than Mixed Tabulation hashing.

2 Related Work

Even Sparser Johnson-Lindenstrauss Transforms

As touched upon earlier, there is a lower bound by Nelson and Nguyen [31] that rules out significant improvements, but never the less there has been research into sparser embedding. In the extreme, Feature Hashing of [38] considers the case of $s = 1$. The lower bound excludes Feature Hashing from working for all vectors, but in [17] they gave tight bounds for which vectors it works in terms of the measure $\|w\|_\infty^2 / \|w\|_2^2$. This was later generalized in [21] to a complete understanding between the tradeoff between s and the measure $\|w\|_\infty^2 / \|w\|_2^2$. In this paper, we will only focus on the case $s = \Theta(\varepsilon^{-1} \log 1/\delta)$ and $m = \Theta(\varepsilon^{-2} \log 1/\delta)$

Fast Johnson-Lindenstrauss Transform

Another direction to speed-up the evaluation of Johnson-Lindenstrauss transforms is to exploit dense matrices with fast matrix-vector multiplication. This was first done by Ailon and Chazelle [4] who introduced the Fast Johnson-Lindenstrauss Transform. Their original construction was recently [16] shown to give an embedding time $O(u \log u + m(\log 1/\delta + \varepsilon \log^2(1/\delta)/\log(1/\varepsilon)))$.

This has generated a lot follow-up work that has tried to improve the running to a clean $O(u \log u)$. Some of the work sacrifice the optimal target dimension, $m = O(\varepsilon^{-2} \log 1/\delta)$, in order to speed-up the construction, and are satisfied with sub-optimal $m = O(\varepsilon^{-2} \log n \log^4 u)$ [29], $m = O(\varepsilon^{-2} \log^3 n)$ [15], $m = O(\varepsilon^{-1} \log^{3/2} n \log^{3/2} u + \varepsilon^{-2} \log n \log^4 u)$ [29], $m = O(\varepsilon^{-2} \log^2 n)$ [19, 37, 18], and $m = O(\varepsilon^{-2} \log n \log^2(\log n) \log^3 u)$ [22]. Another line of progress is to assume that the target dimension, m, is substantially smaller then the starting dimension, u. Under the assumption that $m = o(u^{1/2})$ the work in [5, 7] achieves embedding time $O(u \log m)$. The only construction that for some regimes improves on the original Fast Johnson-Lindenstrauss Transform is the recent analysis [22] of the Kac Johnson-Lindenstrauss Transform, which uses the Kac random walk [25]. They show that it can achieve an embedding time of $O(u \log u + \min\{u \log n, m \log n \log^2(\log n) \log^3 u\})$.

Previous Work on Tabulation Hashing

The work by Patrascu and Thorup [33] initiated the study of tabulation based hashing that goes further than what 3-wise independence of constructions would suggest. A long line of papers have shown tabulation based hashing to work for min-wise hashing [32, 13], hashing for k-statistics [11], and the number of non-empty-bins [2]. Furthermore, multiple papers have been concerned with showing strong concentration results for tabulation based hashing [33, 32, 1, 20]. Tabulation based hashing has also been studied experimentally where they have been shown to exhibit great performance [12, 1].

Preliminaries

In this section, we will introduce the notation which will be used throughout the paper. First we introduce p-norms.

▶ **Definition 2** (p-norm). *Let $p \geq 1$ and X be a random variable with $\mathrm{E}[|X|^p] < \infty$. We then define the p-norm of X by $\|X\|_p = \mathrm{E}[|X|^p]^{1/p}$.*

Throughout the paper, we will repeatedly work with value functions $v\colon U \times [m] \to \mathbb{R}$. We will allow ourself to sometime view them as vectors, and in particular, we will write

$$\|v\|_2 = \sqrt{\sum_{x\in U}\sum_{j\in[m/s]} v(x,j)^2},$$
$$\|v\|_\infty = \max_{x\in U, j\in[m/s]} |v(x,j)|\,.$$

We will also use the Ψ_p-function introduced in [20].

▶ **Definition 3.** *For $p \geq 2$ we define the function $\Psi_p\colon \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}_+$ as follows,*

$$\Psi_p(M,\sigma^2) = \begin{cases} \left(\frac{\sigma^2}{pM^2}\right)^{1/p} M & \text{if } p < \log\frac{pM^2}{\sigma^2} \\ \frac{1}{2}\sqrt{p}\sigma & \text{if } p < e^2\frac{\sigma^2}{M^2} \\ \frac{p}{e\log\frac{pM^2}{\sigma^2}} M & \text{if } \max\left\{\log\frac{pM^2}{\sigma^2}, e^2\frac{\sigma^2}{M^2}\right\} \leq p \end{cases}.$$

It was shown in [20] that $\Psi_p(1,\lambda)$ is within a constant factor of the p-norm of a Poisson distributed random variable with parameter λ. They also showed that $\Psi_p(M,\sigma^2)$ can be used to upper bound expressions involving a fully random hash function $h\colon U \to [m]$. Let $v\colon U \times [m] \to \mathbb{R}$ be a value function then they showed that

$$\left\|\sum_{x\in U} v(x,h(x))\right\|_{\leq} C\Psi_p(\|v\|_\infty, \|v\|_2^2/m)\,,$$

where C is a universal constant.

3 Overview of the New Analysis

Our main technical contribution is a new analysis of the Sparse Johnson-Lindenstrauss Transform that relaxes the assumptions on the hash function, h. We show that if h satisfies a decoupling decomposition property and a strong concentration property then we obtain

the same bounds for the Sparse Johnson-Lindenstrauss Transform. Both of these properties are satisfied by h if h is $\Omega(\log 1/\delta)$-wise independent so our assumptions are weaker than those of the previous analyses.

In this section, we will give an informal overview of new approach. The technical details and the formal statement of the result will be in Section 4.

In order to describe our approach, we look at the random variable

$$Z = \|Aw\|_2^2 - 1 = \frac{1}{s}\sum_{i\in[s]}\sum_{x\neq y\in[u]} \sigma(i,x)\sigma(i,y)\left[h(i,x)=h(i,y)\right] w_x w_y. \tag{2}$$

Here $w \in \mathbb{R}^u$ is a unit vector. With this notation the goal becomes to bound $\Pr[|Z| \geq \varepsilon]$.

The first step in our analysis is that we want to decouple Equation (2). Decoupling was also used in one of the proofs in [10], but since we want to prove the result for more general hash functions, we cannot directly use the standard decoupling inequalities. We will instead assume that our hash function allows a *decoupling-decomposition*. This will formally be defined in Section 4 and we will for now assume that our hash function allows for the standard decoupling inequality. If we apply Markov's inequality and a standard decoupling inequality for fully random hashing we obtain the expression.

$$\begin{aligned}\Pr[|Z|\geq\varepsilon] &\leq \varepsilon^{-p}\,\mathrm{E}[|Z|^p] \\ &\leq \left(\varepsilon^{-1}\frac{4}{s}\right)^p \mathrm{E}\left[\left|\sum_{i\in[s]}\sum_{x,y\in[u]}\sigma(i,x)\sigma'(i,y)\left[h(i,x)=h'(i,y)\right]w_x w_y\right|^p\right] \end{aligned}\tag{3}$$

where (h',σ') are independent copies of (h,σ) and $p \geq 2$. The power of decoupling stems from the fact that it breaks up some of the dependencies and allows for a simpler analysis.

The goal is now to analyse $\left\|\sum_{i\in[s]}\sum_{x,y\in[u]}\sigma(i,x)\sigma'(i,y)\left[h(i,x)=h'(i,y)\right]w_x w_y\right\|_p$. This is done by first fixing (h',σ') and bounding $\left\|\sum_{i\in[s],j\in[m/s]}\sum_{x\in[u]}\sigma(i,x)\left[h(i,x)=j\right]w_x a_{ij}\right\|_p$ using the randomness of (h,σ) where $a_{ij} = \sum_{y\in[u]}\sigma'(i,y)\left[h'(i,y)=j\right]w_y$. In order to do this, we will assume that the pair (h,σ) is *strongly concentrated*. Again the formal definition of this is postponed to Section 4, but informally, we say that the pair is strongly concentrated if it has concentration results similar to those of fully random hashing.

We now take the view that $|a_{ij}|$ is the load of the bin $(i,j) \in [s]\times[m/s]$. The idea is then to split $[s]\times[m/s]$ into heavy and light bins and handle each separately. We choose a parameter k and let I be the heaviest k bins. Using the triangle inequality, we then get that

$$\begin{aligned}\left\|\sum_{i\in[s],j\in[m/s]}\sum_{x\in[u]}\sigma(i,x)\left[h(i,x)=j\right]w_x a_{ij}\right\|_p &\leq \left\|\sum_{(i,j)\in I}\sum_{x\in[u]}\sigma(i,x)\left[h(i,x)=j\right]w_x a_{ij}\right\|_p \\ &+ \left\|\sum_{(i,j)\in[s]\times[m/s]\setminus I}\sum_{x\in[u]}\sigma(i,x)\left[h(i,x)=j\right]w_x a_{ij}\right\|_p.\end{aligned}$$

We show that the contribution from the light bins is as if the collisions are independent. This should be somewhat intuitive since if we only have few collisions in each bin then the collisions behave as if they were independent. In contrast, we show that the contribution from the heavy bins is dominated by the heaviest bin. This turns out to be exactly what we need to finish the analysis.

4 Technical Results

In this section, we will expand on the description from Section 3 and formalize the ideas.

Decoupling

Ideally, we would like to use the standard decoupling inequality, Equation (3). Unfortunately, we cannot expect more general hash functions to support such a clean decoupling. We therefore introduce the notion of a decoupling-decomposition.

▶ **Definition 4** (Decoupling-decomposition). *Let $p \geq 2$, $L \geq 1$, and $0 \leq \gamma \leq 1$. We say that a collection of possibly randomized sets, (U_α), is a* (p, L, γ)-decoupling-decomposition *for a property P of a pair (h, σ), if there exist hash functions $h_\alpha \colon [s] \times U_\alpha \to [m/s]$ and sign functions $s \colon [s] \times U_\alpha \to \{-1, 1\}$ for all α such that*

$$\Pr[|Z| \geq \varepsilon] \leq \left(\varepsilon^{-1} \sum_\alpha \frac{L}{s} \left\| \sum_{i \in [s]} \sum_{x,y \in U_\alpha} \sigma_\alpha(i,x)\sigma'_\alpha(i,y) \left[h_\alpha(i,x) = h'_\alpha(i,y)\right] w_x w_y \right\|_p \right)^p + \gamma \tag{4}$$

where $(h_\alpha, \sigma_\alpha)$ and $(h'_\alpha, \sigma'_\alpha)$ has the same distribution, and $(h_\alpha, \sigma_\alpha)$ satisfies the property P when conditioned on $(h'_\alpha, \sigma'_\alpha)$ and U_α.

The reader should compare Equation (3) for fully random hashing with Equation (4). There are 3 main differences between the expressions.

1. The first thing to notice is that, in the decoupling-decomposition we sum over different sets (U_α), where this is not needed for fully random hashing. We allow the decoupling-decomposition to use a different decoupling on each of the sets U_α. This is very powerful since general hash functions are not necessarily uniform over the input domain.
2. For the decoupling-decomposition, we allow an additive error probability γ. This is useful if the hash function allows for decoupling most of the time except when some unprobable event is happening.
3. The last difference is that a much larger loss-factor is allowed by the decoupling-decomposition than Equation (3). In the case of fully random hashing, we only lose a factor of 4 but for more general hash functions this loss might be bigger.

Finally, we note that Equation (3) implies if (h, σ) is $2p$-wise independent for an integer $p \geq 2$ then $[u]$ is a decoupling-decomposition of (h, σ) for any property P that is satisfied by (h, σ).

Strong Concentration

The second property we need is that the hash function is strongly concentrated.

▶ **Definition 5** (Strong concentration). *Let $h \colon [s] \times U \to [m/s]$ be a hash function and $\sigma \colon [s] \times U \to \{-1, 1\}$ be a sign function. We say that the pair (h, σ) is* (p, L)-strongly-concentrated *if*

1. *For all value functions,* $v\colon [s]\times[m/s]\to\mathbb{R}$*, and all vectors,* $w\in\mathbb{R}^U$*,*

$$\left\|\sum_{i\in[s]}\sum_{x\in U}\sigma(i,x)v(i,h(i,x))w_x\right\|_p \leq \Psi_p\left(L\left\|v\right\|_\infty\left\|w\right\|_\infty, L\frac{s}{m}\left\|v\right\|_2^2\left\|w\right\|_2^2\right)\,, \tag{5}$$

$$\left\|\sum_{i\in[s]}\sum_{x\in U}\sigma(i,x)v(i,h(i,x))w_x\right\|_p \leq \sqrt{L\frac{p}{\log(m/s)}\left\|v\right\|_2^2\left\|w\right\|_2^2}\,. \tag{6}$$

2. *For all vectors,* $w\in\mathbb{R}^U$*,*

$$\left\|\sum_{i\in[s]}\sum_{j\in[m/s]}\left(\sum_{x\in U}\sigma(i,x)\left[h(i,x)=j\right]w_x\right)^2\right\|_{p/2} \leq L\max\left\{s\left\|w\right\|_2^2, \frac{p}{\log m/s}\left\|w\right\|_2^2\right\}. \tag{7}$$

3. *If* $p\leq\log m$*,*

$$\left\|\max_{i\in[s],j\in[m/s]}\left|\sum_{x\in U}\sigma(i,x)\left[h(i,x)=j\right]w_x\right|\right\|_p \leq e\sqrt{L\frac{\log m}{\log m/s}}\left\|w\right\|_2. \tag{8}$$

We need essentially 3 different properties of our hash function to say that it is strongly concentrated.

1. The first property is a concentration result on the random variable

$$\sum_{i\in[s]}\sum_{x\in U}\sigma(i,x)v(i,h(i,x))w_x.$$

   Here we need two different concentration results: The first concentration result, Equation (5), roughly corresponds to a p-norm version of what you would obtain by applying Bennett's inequality to a fully random hash function, while the second concentration result, Equation (5), corresponds to the best hypercontractive result you can obtain for weighted sums of independent Bernoulli-Rademacher variables with parameter s/m.[2]
2. The second property bounds the sum of squares

$$W=\sum_{i\in[s]}\sum_{j\in[m/s]}\left(\sum_{x\in U}\sigma(i,x)\left[h(i,x)=j\right]w_x\right)^2.$$

   The condition, Equation (7), bounds $\left\|W\right\|_{p/2}$ by the maximum of two cases. The first case corresponds to $\mathrm{E}[W]$, and the second case is motivated by applying Equation (6) to

$$\sup_{\substack{z\in\mathbb{R}^{[s]\times[m/s]},\\ \left\|z\right\|_2=1}}\left\|\sum_{i\in[s]}\sum_{j\in[m]}\sum_{x\in U}\sigma(i,x)z_{i,h(i,x)}w_x\right\|_p^2.$$

   While this at first glance might seem odd, it is roughly the best you can do, since one can show that

$$\max\left\{\mathrm{E}[W]\,, \sup_{\substack{z\in\mathbb{R}^{[s]\times[m/s]},\\ \left\|z\right\|_2=1}}\left\|\sum_{i\in[s]}\sum_{j\in[m]}\sum_{x\in U}\sigma(i,x)z_{i,h(i,x)}w_x\right\|_p^2\right\} \leq \left\|W\right\|_{p/2}.$$

[2] A Bernoulli-Rademacher variable with parameter α is random variable, $X\in\{-1,0,1\}$, with $\Pr[X=1]=\Pr[X=-1]=\alpha/2$ and $\Pr[X=0]=1-\alpha$.

3. The final property is a bound on the largest coordinate, $\max_{i\in[s],j\in[m/s]} \left|\sum_{x\in U} \sigma(i,x)\,[h(i,x)=j]\,w_x\right|$. The bound is a natural consequence of Equation (6) for fully random hashing. Namely, for fully random hashing we get that

$$\begin{aligned}
&\left\| \max_{i\in[s],j\in[m/s]} \left|\sum_{x\in U} \sigma(i,x)\,[h(i,x)=j]\,w_x\right| \right\|_p \\
&\qquad \le \left\| \max_{i\in[s],j\in[m/s]} \left|\sum_{x\in U} \sigma(i,x)\,[h(i,x)=j]\,w_x\right| \right\|_{\log m} \\
&\qquad \le e \max_{i\in[s],j\in[m/s]} \left\| \left|\sum_{x\in U} \sigma(i,x)\,[h(i,x)=j]\,w_x\right| \right\|_{\log m} \\
&\qquad \le e\sqrt{L\frac{\log m}{\log m/s}}\,\|w\|_2\,.
\end{aligned}$$

   This derivation is not true for general hash function, but the hash function can still satisfy Equation (8).

The results of [20] show that if the hash function $h\colon [s]\times U \to [m/s]$ and the sign function $\sigma\colon [s]\times U \to [m/s]$ is p-wise independent for an integer $p \ge 2$ then the pair (h,σ) is (p,K)-strongly-concentrated where K is a universal constant.

The Main Result

We are now ready to state our main result which is a new analysis of a Sparse Johnson-Lindenstrauss Transform that only assumes that the hash function has a decoupling-decomposition for the strong concentration property.

▶ **Theorem 6.** *Let $h\colon [s]\times[u]\to[m/s]$ be a hash function and $\sigma\colon [s]\times[u]\to\{-1,1\}$ be a sign function. Furthermore, let $0<\varepsilon<1$ and $0<\delta<1$ be given, and define $p=\log 1/\delta$.*

Assume that there exists constants L_1, L_2, L_3, and $0\le\gamma<1$, that only depends on (h,σ) and p, such that

1. *There exists a (p,L_1,γ)-decoupling-decomposition, (U_α), for the (p,L_2)-strong-concentration property of (h,σ)*
2. *For all vectors $w\in\mathbb{R}^u$, $\sum_\alpha \sum_{x\in U_\alpha} w_x^2 \le L_3\,\|w\|_2^2$.*
3. *$m \ge \left(16e^7 L_1^2 L_2^3 L_3^2\right)\cdot\varepsilon^{-2}\log(1/\delta)$.*
4. *$s \ge \left(64e^3 L_1 L_2^{3/2} L_3\right)\cdot\varepsilon^{-1}\log(1/\delta)$.*

Then the following is true

$$\Pr[|Z|\ge\varepsilon] \le \delta+\gamma.$$

As discussed earlier, a fully random hash function satisfies all the property needed of the theorem and thus gives a new analysis of the Sparse Johnson-Lindenstrauss Transform for fully random hashing. We will also later show that Mixed Tabulation satisfies the assumption of the theorem hence giving the first analysis of a Sparse Johnson-Lindenstrauss Transform with a practical hash function that works.

The main difficulty in the analysis of Theorem 6 is contained in the following technical lemma. The idea in the proof of Theorem 6 is to use the decoupling-decomposition and apply the following lemma to each part.

▶ **Lemma 7.** *Let $h, \overline{h}\colon [s] \times U \to [m/s]$ be hash functions and $\sigma, \overline{\sigma}\colon [s] \times U \to \{-1, 1\}$ be sign functions. Let $p \geq 2$ and assume that there exists a constant L such that (h, σ) is (p, L)-strongly concentrated when conditioning on $(\overline{h}, \overline{\sigma})$, and similarly, $(\overline{h}, \overline{\sigma})$ is (p, L)-strongly concentrated when conditioning on (h, σ). Then for all vectors $w \in \mathbb{R}^U$,*

$$\left\| \sum_{i \in [s]} \sum_{x,y \in U} \sigma(i,x)\overline{\sigma}(i,y) \left[h(i,x) = \overline{h}(i,y)\right] w_x w_y \right\|_p \leq \Psi_p \left(32e^3 L^{3/2} \|w\|_2^2 , 32e^6 L^3 \frac{s^2}{m} \|w\|_2^4 \right) + 36e^3 L \frac{p}{\log m/s} \|w\|_2^2 .$$

The lemma shows that the expression has two different regimes. The first regime, $\Psi_p \left(32e^3 L^{3/2} \|w\|_2^2 , 32e^6 L^3 \frac{s^2}{m} \|w\|_2^4 \right)$, is essentially what we would expect if each of the collisions, $\left[h(i,x) = \overline{h}(i,y)\right]$, are independent of each other. The other regime, $36e^3 L \frac{p}{\log m/s} \|w\|_2^2$, is essentially what you expect the largest coordinate to contribute.

Our analysis is inspired by these two regimes and tries to exploit them explicitly. We start by fixing (h, σ) and divide the coordinates into heavy and light coordinates. We then show that contribution of the light coordinates is $\Psi_p \left(32e^3 L^{3/2} \|w\|_2^2 , 32e^6 L^3 \frac{s^2}{m} \|w\|_2^4 \right)$ which matches the intuition that if we only have few collisions on each coordinate then the collisions behave as if they were independent. Similarly, we show that the contribution of the heavy coordinates is dominated by the heaviest coordinate, namely, the contribution is $36e^3 L \frac{p}{\log m/s} \|w\|_2^2$.

Mixed Tabulation Hashing

Our main result for Mixed Tabulation hashing is the following.

▶ **Theorem 8.** *Let $h\colon [s] \times [u] \to [m/s]$ and $\sigma\colon [s] \times [u] \to \{-1, 1\}$ be Mixed Tabulation functions as described in Section 1.1. Furthermore, let $0 < \varepsilon < 1$ and $0 < \delta < 1$ be given, and define $p = \log 1/\delta$.*

If $m \geq \gamma_p^{3c} \varepsilon^{-2} \log(1/\delta)$ and $s \geq \gamma_p^{3/2c} \varepsilon^{-1} \log(1/\delta)$ where $\gamma_p = Kc \max\left\{1, \frac{p}{\log|\Sigma|}\right\}$ for a universal constant K.

Then the following is true

$$\Pr[|Z| \geq \varepsilon] \leq \delta + \varepsilon 3^c |\Sigma|^{-d} .$$

The result follows by proving that Mixed Tabulation hashing has a $(p, 4^{c+2}, 4\varepsilon^{-2} 3^c \frac{s}{m} |\Sigma|^{-d})$-decoupling-decomposition and that Mixed Tabulation has the strong concentration property. The main new part is in showing the decoupling-decomposition while the analysis of the strong concentration property is modification of the analysis in [20].

Due to space constraints, the proof is deferred to the full version.

5 Analysis of the Sparse Johnson-Lindenstrauss Transform

Lets us start by showing how Lemma 7 implies our main result, Theorem 6.

Proof of Theorem 6. We start by using Equation (4) of the decoupling decomposition to get that

$$\Pr[|Z| \geq \varepsilon] \leq \left(\varepsilon^{-1} \sum_{\alpha} \frac{L_1}{s} \left\| \sum_{i \in [s]} \sum_{x,y \in U_\alpha} \sigma_\alpha(i,x)\sigma'_\alpha(i,y) \left[h_\alpha(i,x) = h'_\alpha(i,y) \right] v_x v_y \right\|_p \right)^p + \gamma$$

Now we fix α and apply Lemma 7 while fixing U_α

$$\left\| \sum_{i \in [s]} \sum_{x,y \in U_\alpha} \sigma_\alpha(i,x)\sigma'_\alpha(i,y) \left[h_\alpha(i,x) = h'_\alpha(i,y) \right] v_x v_y \right\|_p \leq \Psi_p \left(32e^3 L_2^{3/2}, 32e^6 L^3 \frac{s^2}{m} \right) \sum_{x \in U_\alpha} w_x^2 + 36e^3 L_2 \frac{p}{\log m/s} \sum_{x \in U_\alpha} w_x^2$$

Using this we get that

$$\sum_{\alpha} \frac{L_1}{s} \left\| \sum_{i \in [s]} \sum_{x,y \in U_\alpha} \sigma_\alpha(i,x)\sigma'_\alpha(i,y) \left[h_\alpha(i,x) = h'_\alpha(i,y) \right] v_x v_y \right\|_p \leq \sum_{\alpha} \frac{L_1}{s} \left(\Psi_p \left(32e^3 L_2^{3/2}, 32e^6 L^3 \frac{s^2}{m} \right) \sum_{x \in U_\alpha} w_x^2 + 36e^3 L_2 \frac{p}{\log m/s} \sum_{x \in U_\alpha} w_x^2 \right)$$

We now use that $\sum_\alpha \sum_{x \in U_\alpha} w_x^2 \leq L_3 \|w\|_2^2$ to get that

$$\sum_{\alpha} \frac{L_1}{s} \left(\Psi_p \left(32e^3 L_2^{3/2}, 32e^6 L^3 \frac{s^2}{m} \right) \sum_{x \in U_\alpha} w_x^2 + 36e^3 L_2 \frac{p}{\log m/s} \sum_{x \in U_\alpha} w_x^2 \right) \leq \frac{L_3 L_1}{s} \left(\Psi_p \left(32e^3 L_2^{3/2}, 32e^6 L^3 \frac{s^2}{m} \right) + 36e^3 L_2 \frac{p}{\log m/s} \right) \|w\|_2^2$$

It can now be checked that if m and s satisfies the stated assumptions then

$$\frac{L_3 L_1}{s} \left(\Psi_p \left(32e^3 L_2^{3/2}, 32e^6 L^3 \frac{s^2}{m} \right) + 36e^3 L_2 \frac{p}{\log m/s} \right) \|w\|_2^2 \leq e^{-1}\varepsilon$$

Combining all the facts, we get that

$$\Pr[|Z| \geq \varepsilon] \leq \left(\varepsilon^{-1}(e^{-1}\varepsilon) \right)^p + \gamma = \delta + \gamma.$$

This finishes the proof. ◀

The rest of the section is concerned with proving our main technical lemma, Lemma 7. First we need the following two lemmas from [20].

▶ **Lemma 9.** *Let $f\colon \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0}$ be a non-negative function which is monotonically increasing in every argument, and assume that there exists positive reals $(\alpha_i)_{i\in[n]}$ and $(t_i)_{i\in[n]}$ such that for all $\lambda \geq 0$,*

$$f(\lambda^{\alpha_0} t_0, \ldots, \lambda^{\alpha_{n-1}} t_{n-1}) \leq \lambda f(t_0, \ldots, t_{n-1}) \,.$$

Let $(X_i)_{i\in[n]}$ be non-negative random variables. Then for all $p \geq 1$ we have that

$$\|f(X_0, \ldots, X_{n-1})\|_p \leq n^{1/p} \max_{i\in[n]} \left(\frac{\|X_i\|_{p/\alpha_i}}{t_i} \right)^{1/\alpha_i} f(t_0, \ldots, t_{n-1}) \,.$$

▶ **Lemma 10.** *Let $p \geq 2$, $M > 0$, and $\sigma^2 > 0$ then*

$$\frac{1}{2}\sqrt{p}\sigma \leq \Psi_p(M, \sigma^2) \leq \max\left\{ \frac{1}{2}\sqrt{p}\sigma, \frac{1}{2e} pM \right\} .$$

We are now ready to prove Lemma 7.

Proof of Lemma 7. We start by defining $v_h, v_{\bar{h}}\colon [s] \times [m/s] \to \mathbb{R}$ by,

$$v_h(i,j) = \sum_{x\in U} \sigma(i,x) w_x \left[h(i,x) = j\right] \,,$$
$$v_{\bar{h}}(i,j) = \sum_{y\in U} \overline{\sigma}(i,y) w_y \left[\overline{h}(i,y) = j\right] \,.$$

We then want to prove that

$$\left\| \sum_{i\in[s], j\in[m/s]} v_h(i,j) v_{\bar{h}}(i,j) \right\|_p \leq \Psi_p\left(32e^3 L^{3/2} \|w\|_2^2 , 32e^6 L^3 \|w\|_2^4 \right) + 4e^3 L \frac{p}{\log m/s} \|w\|_2^2 \,.$$

First we consider the case where $\frac{p}{\log m/s} \|w\|_2^2 \geq s \|w\|_2^2$. By Cauchy-Schwartz and Equation (7) we get that

$$\left\| \sum_{i\in[s], j\in[m/s]} v_h(i,j) v_{\bar{h}}(i,j) \right\|_p \leq \left\| \sum_{i\in[s], j\in[m/s]} v_h(i,j)^2 \right\|_p \leq L \frac{p}{\log m/s} \|w\|_2^2 \,.$$

We now focus on the case where $\frac{p}{\log m/s} \|w\|_2^2 < s \|w\|_2^2$. We define $\pi\colon [m] \to [s] \times [m/s]$ to be a bijection which satisfies that

$$|v_h(\pi(0))| \geq |v_h(\pi(1))| \geq \ldots \geq |v_h(\pi(m-1))| \,.$$

We note that π is a random function but we can define π such that it only depends on the randomness of h and σ. We define $k = \lfloor p/\log(m/p) \rfloor$, $I = \{\pi(i) \,|\, i \in [k]\}$, and the random functions $v'_h, v''_h\colon [s] \times [m/s] \to \mathbb{R}$ by

$$v'_h(i,j) = v_h(i,j) \left[(i,j) \in I\right] ,$$
$$v''_h(i,j) = v_h(i,j) \left[(i,j) \notin I\right] .$$

Again we note that v'_h and v''_h only depends on the randomness of h and σ. We can then write our expression as

$$\left\| \sum_{i\in[s],j\in[m/s]} v_h(i,j)v_{\bar{h}}(i,j) \right\|_p = \left\| \sum_{i\in[s]} \sum_{y\in U} \overline{\sigma}(i,y)v_h(i,\overline{h}(i,y))w_y \right\|_p$$
$$\leq \left\| \sum_{i\in[s]} \sum_{y\in U} \overline{\sigma}(i,y)v'_h(i,\overline{h}(i,y))w_y \right\|_p + \left\| \sum_{i\in[s]} \sum_{y\in U} \overline{\sigma}(i,y)v''_h(i,\overline{h}(i,y))w_y \right\|_p .$$

We will bound each of the term separately. We start by bounding $\left\|\sum_{i\in[s]} \sum_{y\in U} \overline{\sigma}(i,y)v'_h(i,\overline{h}(i,y))w_y\right\|_p$. We fix h and σ and use Equation (6) to get that

$$\left\| \sum_{i\in[s]} \sum_{y\in U} \overline{\sigma}(i,y)v'_h(i,\overline{h}(i,y))w_y \right\|_p \leq \left\| \sqrt{L\frac{p}{\log m/s}\left\|w\right\|_2^2 \left\|v'_h\right\|_2^2} \right\|_p$$
$$= \sqrt{L\frac{p}{\log m/s}}\left\|w\right\|_2 \left\| \sqrt{\sum_{(i,j)\in I} v'_h(i,j)^2} \right\|_p .$$

We note that $\sum_{(i,j)\in I} v'_h(i,j)^2 = \max_{J\subseteq[s]\times[m/s],|J|=k} \sum_{(i,j)\in J} v_h(i,j)^2$. We then get that

$$\left\| \sqrt{\sum_{(i,j)\in I} v'_h(i,j)^2} \right\|_p = \left\| \sqrt{\max_{J\subseteq[s]\times[m/s],|J|=k} \sum_{(i,j)\in J} v_h(i,j)^2} \right\|_p$$
$$\leq \left(\sum_{J\subseteq[s]\times[m/s],|J|=k} \left\| \sqrt{\sum_{(i,j)\in J} v_h(i,j)^2} \right\|_p^p \right)^{1/p}$$
$$\leq \binom{ms}{k}^{1/p} \max_{J\subseteq[s]\times[m/s],|J|=k} \left\| \sqrt{\sum_{(i,j)\in J} v_h(i,j)^2} \right\|_p$$

We use Sterling's bound and get that $\binom{ms}{k}^{1/p} \leq \left(\frac{ems}{k}\right)^{k/p} \leq \left(\frac{ems\log(ms/p)}{p}\right)^{1/\log(ms/p)} \leq e^3$. So we get that

$$\left\| \sqrt{\sum_{(i,j)\in I} v'_h(i,j)^2} \right\|_p \leq e^3 \max_{J\subseteq[s]\times[m/s],|J|=k} \left\| \sqrt{\sum_{(i,j)\in J} v_h(i,j)^2} \right\|_p$$

A standard volumetric argument gives that there exists a 1/4-net, $Z \subseteq \mathbb{R}^J$, with $|Z| \leq 9^k$, such that

$$\begin{aligned}
\left\|\sqrt{\sum_{(i,j)\in J} v_h(i,j)^2}\right\|_p &= \left\|\sup_{z\in\mathbb{R}^J,\|z\|_2=1}\sum_{(i,j)\in J} z_{i,j}v_h(i,j)\right\|_p \\
&\leq \left\|\sup_{z\in Z}\sum_{(i,j)\in J} z_{i,j}v_h(i,j)\right\|_p \\
&\quad + \left\|\sup_{z\in\mathbb{R}^J,\|z\|_2=1}\sum_{(i,j)\in J} (z_{i,j}-z'_{i,j})v_h(i,j)\right\|_p \\
&\leq \left\|\sup_{z\in Z}\sum_{(i,j)\in J} z_{i,j}v_h(i,j)\right\|_p \\
&\quad + \sup_{z\in\mathbb{R}^J,\|z\|_2=1}\|z-z'\|_2\left\|\sqrt{\sum_{(i,j)\in J} v_h(i,j)^2}\right\|_p
\end{aligned}$$

where $z' \in Z$ is the closest element to z, and as such $\|z-z'\|_2 \leq 1/4$. Since there are at most 9^k elements in Z then $\left\|\sup_{z\in Z}\sum_{(i,j)\in J} z_{i,j}v_h(i,j)\right\|_p \leq 9\sup_{z\in Z}\left\|\sum_{(i,j)\in J} z_{i,j}v_h(i,j)\right\|_p$, where we used that $k \leq p$. Collecting the fact we get that

$$\left\|\sqrt{\sum_{(i,j)\in J} v_h(i,j)^2}\right\|_p \leq 36\sup_{z\in Z}\left\|\sum_{(i,j)\in J} z_{i,j}v_h(i,j)\right\|_p$$

Using this we get that

$$\begin{aligned}
&e^3\max_{J\subseteq[s]\times[m/s],|J|=k}\left\|\sqrt{\sum_{(i,j)\in J} v_h(i,j)^2}\right\|_p \\
&\quad\leq 36e^3\max_{J\subseteq[s]\times[m/s],|J|=k}\max_{z\in Z}\left\|\sum_{(i,j)\in J} z_{i,j}v_h(i,j)\right\|_p \\
&\quad= 36e^3\max_{J\subseteq[s]\times[m/s],|J|=k}\max_{\substack{z\in\mathbb{R}^{s\times m/s},\\ \|z\|_2=1}}\left\|\sum_{i\in[s]}\sum_{x\in U}\sigma(i,x)z_{i,h(i,x)}\left[(i,h(i,x))\in J\right]w_x\right\|_p
\end{aligned}$$

We can then use Equation (6) to get that

$$\begin{aligned}
&36e^3\max_{J\subseteq[s]\times[m/s],|J|=k}\max_{\substack{z\in\mathbb{R}^{s\times m/s},\\ \|z\|_2=1}}\left\|\sum_{i\in[s]}\sum_{x\in U}\sigma(i,x)z_{i,h(i,x)}\left[(i,h(i,x))\in J\right]w_x\right\|_p \\
&\quad\leq 36e^3\max_{J\subseteq[s]\times[m/s],|J|=k}\max_{\substack{z\in\mathbb{R}^{s\times m/s},\\ \|z\|_2=1}}\sqrt{L\frac{p}{\log m/s}}\,\|w\|_2\,\|z\|_2 \\
&\quad= 36e^3\sqrt{L\frac{p}{\log m/s}}\,\|w\|_2
\end{aligned}$$

Combining the facts, we get that $\left\|\sum_{i\in[s]}\sum_{y\in U}\overline{\sigma}(i,y)v'_h(i,\overline{h}(i,y))w_y\right\|_p \leq 36e^3L\frac{p}{\log m/s}\,\|w\|_2^2$.

We will now bound $\left\|\sum_{i\in[s]}\sum_{y\in U}\overline{\sigma}(i,y)v_h''(i,\overline{h}(i,y))w_y\right\|_p$. We fix h and ε and use Equation (5) to get that

$$\begin{aligned}\left\|\sum_{i\in[s]}\sum_{y\in U}\overline{\sigma}(i,y)v_h''(i,\overline{h}(i,y))w_y\right\|_p &\leq \left\|\Psi_p\left(L\left\|w\right\|_\infty\left\|v_h'\right\|_\infty, L\frac{s}{m}\left\|w\right\|_2^2\left\|v_h'\right\|_2^2\right)\right\|_p \\ &\leq \left\|\Psi_p\left(L\left\|w\right\|_\infty\left|v_h'(\pi(k+1))\right|, L\frac{s}{m}\left\|w\right\|_2^2\left\|v_h\right\|_2^2\right)\right\|_p .\end{aligned}$$

Now we use Lemma 9 to get that,

$$\begin{aligned}&\left\|\Psi_p\left(L\left\|w\right\|_\infty\left|v_h'(\pi(k+1))\right|, L\frac{s}{m}\left\|w\right\|_2^2\left\|v_h\right\|_2^2\right)\right\|_p \\ &\qquad\leq \sqrt{2}\Psi_p\left(L\left\|w\right\|_\infty\left\|\left|v_h'(\pi(k+1))\right|\right\|_p, L\frac{s}{m}\left\|w\right\|_2^2\left\|\left\|v_h\right\|_2^2\right\|_{p/2}\right).\end{aligned}$$

Since we assume that $\frac{p}{\log m}\left\|w\right\|_2^2 < s\left\|w\right\|_2^2$ then Equation (7) give us that $\left\|\left\|v_h\right\|_2^2\right\|_{p/2} \leq Ls\left\|w\right\|_2^2$.

We will now bound $\left\|v_h'(\pi(k+1))\right\|_p$. For this, we will distinguish between two cases: Either $p \geq \log m$ or $p < \log m$. Let us first case where $p \geq \log m$. We will use that $|v_h'(\pi(k+1))| \leq \frac{\sum_{i\in[k+1]}|v_h'(\pi(i))|}{k+1}$. We then get that

$$\begin{aligned}&\left\|v_h'(\pi(k+1))\right\|_p \\ &\leq \left\|\frac{\sum_{i\in[k+1]}|v_h'(\pi(i))|}{k+1}\right\|_p \\ &\leq \left(\binom{m}{k+1}2^{k+1}\max_{\substack{J\subseteq[s]\times[m/s],\,(\sigma_{i,j})_{(i,j)\in J}\in\{-1,1\}^J\\|J|=k+1}}\left(\frac{\left\|\sum_{(i,j)\in J}\sigma_{i,j}v_h(i,j)\right\|_p}{k+1}\right)^p\right)^{1/p} \\ &\leq \max_{\substack{J\subseteq[s]\times[m/s],\,(s_{i,j})_{(i,j)\in J}\in\{-1,1\}^J\\|J|=k+1}} 2\binom{m}{k+1}^{1/p}\frac{\left\|\sum_{(i,j)\in J}\sigma_{i,j}v_h(i,j)\right\|_p}{k+1}\end{aligned}$$

We note that $\left\|\sum_{(i,j)\in J}\sigma_{i,j}v_h(i,j)\right\|_p = \left\|\sum_{x\in U}\sum_{(i,j)\in J}\sigma(i,x)s_{i,j}\left[h(i,x)=j\right]w_x\right\|_p$. Since we have that $p \geq \log m$ then $k \geq 1$ which implies that $k+1 \leq 2\frac{p}{\log(m/p)}$. We then get that $\binom{m}{k+1}^{1/p} \leq \left(\frac{em}{2p/\log(m/p)}\right)^{2/\log(m/p)} \leq 2e^3$. We now use Equation (6) to get that,

$$\begin{aligned}\left\|\sum_{x\in U}\sum_{(i,j)\in J}\sigma(i,x)s_{i,j}\left[h(i,x)=j\right]w_x\right\|_p &= \left\|\sum_{i\in[s]}\sum_{x\in U}\sigma(i,x)\left[(i,h(i,x))\in J\right]s_{i,h(i,x)}w_x\right\|_p \\ &\leq \sqrt{L\frac{p}{\log m/s}}\sqrt{|J|}\left\|w\right\|_2 \\ &= \sqrt{L\frac{p}{\log m/s}}\sqrt{k+1}\left\|w\right\|_2\end{aligned}$$

Combining this we get that $\|v'_h(\pi(k+1))\|_p \leq 4e^3\sqrt{L\frac{p}{\log m/s}}\frac{\|w\|_2}{\sqrt{k+1}}$. We then obtain that,

$$\begin{aligned}\left\|\sum_{i\in[s]}\sum_{y\in U}\overline{\sigma}(i,y)v''_h(i,\overline{h}(i,y))w_y\right\|_p &\\ \leq \sqrt{2}\Psi_p\left(4e^3L\sqrt{L\frac{p}{(k+1)\log m/s}}\|w\|_\infty\|w\|_2, L^2\frac{s^2}{m}\|w\|_2^4\right)&\\ \leq \sqrt{2}\Psi_p\left(4e^3L\sqrt{L\frac{\log m/p}{\log m/s}}\|w\|_2^2, L^2\frac{s^2}{m}\|w\|_2^4\right)&\end{aligned}$$

If $\log m/p \leq 4\log m/s$ then we get that,

$$\begin{aligned}\left\|\sum_{i\in[s]}\sum_{y\in U}\overline{\sigma}(i,y)v''_h(i,\overline{h}(i,y))w_y\right\|_p &\leq \sqrt{2}\Psi_p\left(16e^3L^{3/2}\|w\|_2^2, L^2\frac{s^2}{m}\|w\|_2^4\right)\\ &\leq \Psi_p\left(32e^3L^{3/2}\|w\|_2^2, 2L^2\frac{s^2}{m}\|w\|_2^4\right)\end{aligned}$$

If $\log m/p > 4\log m/s$ then $m/p > (m/s)^4$ which implies that $\frac{pm}{s^2} \leq \sqrt{p}m$. Using this we get that $\frac{p16e^6L^3\frac{\log m/p}{\log m/s}\|w\|_2^4}{L^2\frac{s^2}{m}\|w\|_2^4} \leq 16e^6L\sqrt{p}m\log m/p \leq 16e^6L$. Where we have used that $\sqrt{s}\log 1/x \leq 1$. Now we use Lemma 10 to get that

$$\begin{aligned}\left\|\sum_{i\in[s]}\sum_{y\in U}\overline{\sigma}(i,y)v''_h(i,\overline{h}(i,y))w_y\right\|_p &\leq \sqrt{2}\Psi_p\left(32e^3L^{3/2}\|w\|_2^2, L^2\frac{s^2}{m}\|w\|_2^4\right)\\ &\leq \sqrt{2}\sqrt{pL^3 16e^6\frac{s^2}{m}\|w\|_2^4}\\ &\leq \Psi_p\left(8e^3L^{3/2}\|w\|_2, 32e^6L^3\frac{s^2}{m}\|w\|_2^4\right)\end{aligned}$$

Now let us consider the case where $p < \log m$. By Equation (8), we get that

$$\|v'_h(\pi(k+1))\|_p \leq \left\|\max_{i\in[s],j\in[m/s]}\left|\sum_{x\in U}\sigma(i,x)\left[h(i,x)=j\right]w_x\right|\right\|_p \leq e\sqrt{L\frac{\log m}{\log m/s}}\|w\|_2$$

We then obtain that,

$$\begin{aligned}\left\|\sum_{i\in[s]}\sum_{y\in U}\overline{\sigma}(i,y)v''_h(i,\overline{h}(i,y))w_y\right\|_p &\leq \sqrt{2}\Psi_p\left(eL\sqrt{L\frac{\log m}{\log m/s}}\|w\|_\infty\|w\|_2, L\frac{s^2}{m}\|w\|_2^4\right)\\ &\leq \sqrt{2}\Psi_p\left(eL\sqrt{L\frac{\log m}{\log m/s}}\|w\|_2^2, L\frac{s^2}{m}\|w\|_2^4\right)\end{aligned}$$

If $s \leq m^{3/4}$ then we get that $\log m \leq 4\log m/s$ and

$$\sqrt{2}\Psi_p\left(eL\sqrt{L\frac{\log m}{\log m/s}}\|w\|_2^2, L\frac{s^2}{m}\|w\|_2^4\right) \leq \sqrt{2}\Psi_p\left(4eL^{3/2}\|w\|_2^2, L\frac{s^2}{m}\|w\|_2^4\right)$$

as wanted. If $s \geq m^{3/4}$ then we get that $\frac{peL^3 \log m}{L^2 s^2/m} \leq \frac{eLm \log^2 m}{s^2} \leq \frac{eL \log^2 m}{m^{1/2}} \leq 16/eL$, where we have used that $\sqrt{x} \log^2 1/x \leq 16/e^2$. Again we use Lemma 10 to get that

$$\begin{aligned}\sqrt{2}\Psi_p\left(4eL^{3/2}\left\|w\right\|_2^2, L\frac{s^2}{m}\left\|w\right\|_2^4\right) &\leq \sqrt{p32/eL^3\frac{s^2}{m}}\left\|w\right\|_2^2 \\ &\leq \Psi_p\left(8L^{3/2}\left\|w\right\|_2^2, 32L^3\frac{s^2}{m}\left\|w\right\|_2^4\right).\end{aligned}$$

Combining everything we get that

$$\begin{aligned}\left\|\sum_{i\in[s]}\sum_{y\in U}\overline{\sigma}(i,y)v''_h(i,\overline{h}(i,y))w_y\right\|_p &\leq \Psi_p\left(32e^3L^{3/2}\left\|w\right\|_2, 32e^6L^3\frac{s^2}{m}\left\|w\right\|_2^4\right), \\ \left\|\sum_{i\in[s]}\sum_{y\in U}\overline{\sigma}(i,y)v'_h(i,\overline{h}(i,y))w_y\right\|_p &\leq 4e^3L\frac{p}{\log m/s}\left\|w\right\|_2^2.\end{aligned}$$

Now we conclude that

$$\begin{aligned}&\left\|\sum_{i\in[s],j\in[m/s]} v_h(i,j)v_{\overline{h}}(i,j)\right\|_p \\ &\quad\leq \left\|\sum_{i\in[s]}\sum_{y\in U}\overline{\sigma}(i,y)v'_h(i,\overline{h}(i,y))w_y\right\|_p + \left\|\sum_{i\in[s]}\sum_{y\in U}\overline{\sigma}(i,y)v''_h(i,\overline{h}(i,y))w_y\right\|_p \\ &\quad\leq \Psi_p\left(32e^3L^{3/2}\left\|w\right\|_2^2, 32e^6L^3\left\|w\right\|_2^4\right) + 4e^3L\frac{p}{\log m/s}\left\|w\right\|_2^2.\end{aligned}$$

Thus finishing the proof. ◀

References

1. Anders Aamand, Jakob Bæk Tejs Knudsen, Mathias Bæk Tejs Knudsen, Peter Michael Reichstein Rasmussen, and Mikkel Thorup. Fast hashing with strong concentration bounds. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 1265–1278, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3357713.3384259.
2. Anders Aamand and Mikkel Thorup. Non-empty bins with simple tabulation hashing. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2498–2512. SIAM, 2019. doi:10.1137/1.9781611975482.153.
3. Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671–687, 2003. Special Issue on PODS 2001. doi:10.1016/S0022-0000(03)00025-4.
4. Nir Ailon and Bernard Chazelle. The fast johnson–lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on Computing*, 39(1):302–322, 2009. doi:10.1137/060673096.
5. Nir Ailon and Edo Liberty. Fast dimension reduction using rademacher series on dual BCH codes. In Shang-Hua Teng, editor, *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 1–9. SIAM, 2008. URL: http://dl.acm.org/citation.cfm?id=1347082.1347083.

6 Noga Alon and Bo'az Klartag. Optimal compression of approximate inner products and dimension reduction. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 639–650, October 2017. doi:10.1109/FOCS.2017.65.

7 Stefan Bamberger and Felix Krahmer. Optimal fast johnson-lindenstrauss embeddings for large data sets. *Sampling Theory, Signal Processing, and Data Analysis*, 19, June 2021. doi:10.1007/s43670-021-00003-5.

8 Vladimir Braverman, Rafail Ostrovsky, and Yuval Rabani. Rademacher chaos, random eulerian graphs and the sparse johnson-lindenstrauss transform. *CoRR*, abs/1011.2590, 2010. arXiv:1011.2590.

9 Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15, 2004. Automata, Languages and Programming. doi:10.1016/S0304-3975(03)00400-6.

10 Michael B. Cohen, T. S. Jayram, and Jelani Nelson. Simple analyses of the sparse johnson-lindenstrauss transform. In Raimund Seidel, editor, *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, volume 61 of *OASIcs*, pages 15:1–15:9. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/OASIcs.SOSA.2018.15.

11 S. Dahlgaard, M. B. T. Knudsen, E. Rotenberg, and M. Thorup. Hashing for statistics over k-partitions. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1292–1310, 2015. doi:10.1109/FOCS.2015.83.

12 Søren Dahlgaard, Mathias Bæk Tejs Knudsen, and Mikkel Thorup. Practical hash functions for similarity estimation and dimensionality reduction. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6618–6628, USA, 2017. Curran Associates Inc. URL: http://dl.acm.org/citation.cfm?id=3295222.3295407.

13 Søren Dahlgaard and Mikkel Thorup. Approximately minwise independence with twisted tabulation. In R. Ravi and Inge Li Gørtz, editors, *Algorithm Theory – SWAT 2014*, pages 134–145, Cham, 2014. Springer International Publishing.

14 Anirban Dasgupta, Ravi Kumar, and Tamás Sarlos. A sparse johnson: Lindenstrauss transform. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, pages 341–350, New York, NY, USA, 2010. Association for Computing Machinery. doi:10.1145/1806689.1806737.

15 Thong T. Do, Lu Gan, Yi Chen, Nam Nguyen, and Trac D. Tran. Fast and efficient dimensionality reduction using structurally random matrices. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1821–1824, 2009. doi:10.1109/ICASSP.2009.4959960.

16 Ora Nova Fandina, Mikael Møller Høgsgaard, and Kasper Green Larsen. Barriers for faster dimensionality reduction, 2022. doi:10.48550/arXiv.2207.03304.

17 Casper Freksen, Lior Kamma, and Kasper Green Larsen. Fully understanding the hashing trick. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 5394–5404, Red Hook, NY, USA, 2018. Curran Associates Inc.

18 Casper Benjamin Freksen and Kasper Green Larsen. On using toeplitz and circulant matrices for johnson-lindenstrauss transforms. *Algorithmica*, 82(2):338–354, 2020. doi:10.1007/s00453-019-00644-y.

19 Aicke Hinrichs and Jan Vybíral. Johnson-lindenstrauss lemma for circulant matrices. *Random Structures & Algorithms*, 39(3):391–398, 2011. doi:10.1002/rsa.20360.

20 Jakob Bæk Tejs Houen and Mikkel Thorup. Understanding the moments of tabulation hashing via chaoses. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 74:1–74:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.74.

21 Meena Jagadeesan. Understanding sparse jl for feature hashing. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, NeurIPS'19, Red Hook, NY, USA, 2019. Curran Associates Inc.

22 Vishesh Jain, Natesh S. Pillai, Ashwin Sah, Mehtaab Sawhney, and Aaron Smith. Fast and memory-optimal dimension reduction using Kac's walk. *The Annals of Applied Probability*, 32(5):4038–4064, 2022. doi:10.1214/22-AAP1784.

23 T. S. Jayram and David P. Woodruff. Optimal bounds for johnson-lindenstrauss transforms and streaming problems with subconstant error. *ACM Trans. Algorithms*, 9(3), June 2013. doi:10.1145/2483699.2483706.

24 William Johnson and Joram Lindenstrauss. Extensions of lipschitz maps into a hilbert space. *Contemporary Mathematics*, 26:189–206, January 1984. doi:10.1090/conm/026/737400.

25 Mark Kac. Foundations of kinetic theory. In *Proceedings of The third Berkeley symposium on mathematical statistics and probability*, volume 3, pages 171–197, 1956.

26 Daniel Kane, Raghu Meka, and Jelani Nelson. Almost optimal explicit johnson-lindenstrauss families. In Leslie Ann Goldberg, Klaus Jansen, R. Ravi, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 628–639, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

27 Daniel M. Kane and Jelani Nelson. A derandomized sparse johnson-lindenstrauss transform, 2010. doi:10.48550/arXiv.1006.3585.

28 Daniel M. Kane and Jelani Nelson. Sparser johnson-lindenstrauss transforms. *J. ACM*, 61(1), January 2014. doi:10.1145/2559902.

29 Felix Krahmer and Rachel Ward. New and improved johnson–lindenstrauss embeddings via the restricted isometry property. *SIAM Journal on Mathematical Analysis*, 43(3):1269–1281, 2011. doi:10.1137/100810447.

30 Kasper Green Larsen and Jelani Nelson. Optimality of the johnson-lindenstrauss lemma. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 633–638, 2017. doi:10.1109/FOCS.2017.64.

31 Jelani Nelson and Huy L. NguyÅn. Sparsity lower bounds for dimensionality reducing maps. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 101–110, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2488608.2488622.

32 Mihai Patrascu and Mikkel Thorup. Twisted tabulation hashing. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 209–228. SIAM, 2013. doi:10.1137/1.9781611973105.16.

33 Mihai Pătraşcu and Mikkel Thorup. The power of simple tabulation hashing. *J. ACM*, 59(3), June 2012. doi:10.1145/2220357.2220361.

34 Alan Siegel. On universal classes of extremely random constant-time hash functions. *SIAM Journal on Computing*, 33(3):505–543, 2004. Announced at FOCS'89.

35 Mikkel Thorup. Simple tabulation, fast expanders, double tabulation, and high independence. In *54th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 90–99, 2013.

36 Mikkel Thorup and Yin Zhang. Tabulation-based 5-independent hashing with applications to linear probing and second moment estimation. *SIAM Journal on Computing*, 41(2):293–331, 2012. doi:10.1137/100800774.

37 Jan Vybíral. A variant of the johnson–lindenstrauss lemma for circulant matrices. *Journal of Functional Analysis*, 260(4):1096–1105, 2011. doi:10.1016/j.jfa.2010.11.014.

38 Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1113–1120, New York, NY, USA, 2009. Association for Computing Machinery. doi:10.1145/1553374.1553516.

39 Albert Lindsey Zobrist. A new hashing method with application for game playing. Technical Report 88, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 1970.

Approximating Max-Cut on Bounded Degree Graphs: Tighter Analysis of the FKL Algorithm

Jun-Ting Hsieh
Carnegie Mellon University, Pittsburgh, PA, USA

Pravesh K. Kothari
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

In this note, we describe a $\alpha_{GW} + \widetilde{\Omega}(1/d^2)$-factor approximation algorithm for Max-Cut on weighted graphs of degree $\leqslant d$. Here, $\alpha_{GW} \approx 0.878$ is the worst-case approximation ratio of the Goemans-Williamson rounding for Max-Cut. This improves on previous results for unweighted graphs by Feige, Karpinski, and Langberg [1] and Florén [3]. Our guarantee is obtained by a tighter analysis of the solution obtained by applying a natural local improvement procedure to the Goemans-Williamson rounding of the basic SDP strengthened with triangle inequalities.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Max-Cut, approximation algorithm, semidefinite programming

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.77

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2206.09204

Funding *Jun-Ting Hsieh*: Supported by NSF CAREER Award #2047933.
Pravesh K. Kothari: Supported by NSF CAREER Award #2047933, Alfred P. Sloan Fellowship and a Google Research Scholar Award.

Acknowledgements We thank Prasad Raghavendra for his Simons Institute talk[1] on approximation algorithms for bounded degree constraint satisfaction problems and various related discussions that directly motivated this work. We thank Simons Institute Berkeley for hosting us in the Fall 2021 research program on Computational Complexity of Statistical Inference.

1 Introduction

In 1994, Goemans and Williamson [4] described a polynomial time algorithm based on rounding the natural semidefinite relaxation for the Max-Cut problem to obtain an approximation ratio of $\alpha_{\mathrm{GW}} \approx 0.878$. Assuming the Unique Games Conjecture [5], this rounding algorithm is in fact worst-case optimal [6].

When the underlying graph has bounded degree, the complexity landscape of Max-Cut is much less clear. In particular, the Goemans-Williamson (GW) rounding algorithm can be improved by an elementary local-search based post-processing step applied to the cut obtained by GW rounding of the SDP relaxation strengthened by adding triangle inequalities. The first such result was shown by Feige, Karpinski, and Langberg [1] (FKL) to obtain an approximation ratio of $\alpha_{\mathrm{GW}} + \epsilon(d)$ for $\epsilon(d) = \Omega(1/d^4)$ for unweighted degree d graphs. Florén [3] later improved the FKL analysis to obtain $\epsilon(d) = \Omega(1/d^3)$ in the same setting.

In this note, we refine the local search scheme of FKL and give a tighter analysis to show that $\epsilon(d) = \widetilde{\Omega}(1/d^2)$. Moreover, we extend the result to weighted instances of Max-2LIN, which generalizes Max-Cut.

▶ **Theorem 1.** *There is a polynomial time algorithm that takes input a (weighted) Max-2LIN instance φ on a graph with n vertices and degree $\leqslant d$ and outputs an assignment that satisfies a number of constraints that is within an $\alpha_{\mathrm{GW}} + \Omega\left(\frac{1}{d^2 \log d}\right)$ of the optimum.*

[1] https://simons.berkeley.edu/talks/title-tba-5

As in FKL, the high-level plan is to show that when the "edgewise" analysis of the Goemans-Williamson rounded solution is "tight" (otherwise, we already obtain an improvement on the worst-case GW guarantee), the rounded solution is locally suboptimal. That is, there is a c_d-fraction of vertices, for some constant c_d depending only on d, such that switching them to the other side should cut more of the edges to their neighbors and thus increase the cut by at least one edge. But flipping a vertex may kill the increase from others. So we may not be able to "win" from every one of the candidates. The analysis then involves managing this dependency and lower bounding c_d. Specifically, FKL and Florén accomplish this by analyzing the chance that more than half of a vertex's neighbors lie on the same side of the partition generated by GW rounding. In such a plan, they have to handle vertices of odd and even degrees slightly differently and generalizations to weighted graphs seem to lose additional factors in the approximation ratio.

Instead of trying to gain only a single edge from a vertex flip, our analysis directly focuses on lower bounding the quantitative gain in the weighted cut by such an operation. This strategy also allows us to easily generalize our results to Max-2LIN and to arbitrary weighted graphs.

There is still a wide gap between approximation algorithms and hardness for Max-Cut in the bounded degree setting. In particular, for every $\epsilon > 0$, the best known hardness result by Trevisan [7] rules out polynomial time algorithms with an approximation ratio $> \alpha_{GW} + 5/\sqrt{d} + \epsilon$ for graphs of large enough constant degree d. Improving the approximation ratio to get closer to this bound (this will likely need a scheme different from FKL's) or obtaining a better inapproximability results are outstanding open questions.

2 The Algorithm and Analysis

▶ **Algorithm 2** (Max-2LIN on bounded degree graphs).

Given: A graph $G = (V, E)$ on n vertices, m edges and maximum degree d, signs $b_{ij} \in \{\pm 1\}$, and non-negative weights $w_{ij} > 0$ for each $\{i, j\} \in E$.

Operation:

1. Find unit vectors[a] $v_1, \ldots, v_n \in \mathbb{R}^n$ that maximize $\frac{1}{m} \sum_{\{i,j\} \in E} w_{ij} \frac{1}{2}(1 + b_{ij} \langle v_i, v_j \rangle)$, and, satisfy $\|a_i v_i - a_j v_j\|^2 + \|a_j v_j - a_k v_k\|^2 \geqslant \|a_i v_i - a_k v_k\|^2$ for every triple $\{i, j, k\}$ and $(a_i, a_j, a_k) \in \{\pm 1\}^3$.
2. Sample $g \sim \mathcal{N}(0, \mathbb{I}_n)$[b] , and set $x_i = \operatorname{sgn}(\langle g, v_i \rangle) \in \{\pm 1\}$.
3. Set $\varepsilon = \frac{1}{Cd\sqrt{\log d}}$ for some large enough constant $C > 0$. Define $S \coloneqq \{i \in V : \langle g, v_i \rangle \in (-\varepsilon, \varepsilon)\}$ to be the *candidate set* of suboptimal vertices. For each $i \in S$, partition its neighbors $N(i)$ into 3 disjoint sets:
   (a) $A_i = N(i) \cap S = \{j \in N(i) : \langle g, v_j \rangle \in (-\varepsilon, \varepsilon)\}$,
   (b) $B_i = \{j \in N(i) : b_{ij} x_i \langle g, v_j \rangle \leqslant -\varepsilon\}$,
   (c) $C_i = \{j \in N(i) : b_{ij} x_i \langle g, v_j \rangle \geqslant \varepsilon\}$.
4. Output x' obtained by flipping x_i for every $i \in S$ such that $\sum_{j \in B_i} w_{ij} > \sum_{j \in A_i \cup C_i} w_{ij}$.

[a] Approximately solving an SDP in poly(n) time produces such vectors with value at most $2^{-\operatorname{poly}(n)}$ smaller.

[b] As in standard implementations of the GW rounding scheme, truncating Gaussian samples to rationals of poly(n)-bits suffices to recover the stated guarantees up to a loss of an additive $2^{-\operatorname{poly}(n)}$ in the approximation ratio. We will omit a detailed discussion of issues of numerical precision in this note.

Analysis. We will appeal to the following three elementary facts.

▶ **Fact 3** (Gaussian Measure in the Core). *For any $\varepsilon > 0$,*

$$\frac{\varepsilon}{\sqrt{2\pi}} e^{-\varepsilon^2/2} \leqslant \Pr_{g \sim \mathcal{N}(0,1)} [g \in (0, \varepsilon)] \leqslant \frac{\varepsilon}{\sqrt{2\pi}} .$$

▶ **Fact 4** (Sheppard's Lemma [2]). *Let g_1, g_2 be standard Gaussian variables with covariance $\sigma = \mathbb{E}[g_1 g_2] \in [-1, 1]$. Then, $\mathbf{Pr}[g_1 \geqslant 0 \wedge g_2 \geqslant 0] = \frac{1}{2} - \frac{1}{2\pi} \arccos(\sigma)$.*

▶ **Fact 5** (Corollary of Schur Product Theorem). *Let $A \in \mathbb{R}^{n \times n}$ be positive semidefinite. For any $t \in \mathbb{N}$, the matrix $B \in \mathbb{R}^{n \times n}$ with entries $B_{ij} = A_{ij}^t$ is positive semidefinite.*

Our analysis crucially relies on the following basic fact about the arcsin function.

▶ **Lemma 6** (Basic Facts about Taylor Approximation for arcsin). *Let the Taylor expansion of the* arcsin *function be* $\arcsin(x) = \sum_{k=0}^{\infty} c_k x^{2k+1}$ *for* $|x| \leqslant 1$ *where* $c_k = \frac{(2k)!}{2^{2k}(k!)^2(2k+1)}$. *For any $\tau \in \mathbb{N}$, $\tau > \tau_0$ where τ_0 is a universal constant,*
(1) *for $x > 0$,* $\arcsin(x) \geqslant \sum_{k=0}^{\tau} c_k x^{2k+1}$,
(2) *for $|x| \leqslant 1/2$,* $\arcsin(x) \geqslant \sum_{k=0}^{\tau} c_k x^{2k+1} - O(\tau^{-1/2} \cdot 2^{-2\tau})$,
(3) *for $x = 1$,* $\arcsin(1) = \frac{\pi}{2} = \sum_{k=0}^{\tau} c_k + \Theta(\tau^{-1/2})$.

Proof. We first show that $c_k = \Theta(k^{-3/2})$ by Stirling's approximation:

$$c_k = (1 + O(1/k)) \cdot \frac{\sqrt{4\pi k}(2k/e)^{2k}}{2^{2k} \cdot 2\pi k (k/e)^{2k} \cdot (2k+1)} = \frac{1}{2\sqrt{\pi}} k^{-3/2} (1 + O(1/k)) .$$

Therefore, for any $\tau > \tau_0$ where τ_0 is a universal constant,

$$\sum_{k=\tau}^{\infty} c_k = \Theta(1) \sum_{k=\tau}^{\infty} k^{-3/2} = \Theta(1) \int_{\tau}^{\infty} x^{-3/2} \, dx = \Theta(\tau^{-1/2}) .$$

Now we prove the lemma. (1) is straightforward because $c_k > 0$ for all k. (2) holds since

$$\left| \sum_{k=\tau+1}^{\infty} c_k x^{2k+1} \right| \leqslant \sum_{k=\tau+1}^{\infty} c_k |x|^{2k+1} \leqslant 2^{-2\tau} \sum_{k=\tau+1}^{\infty} c_k \leqslant O(\tau^{-1/2} \cdot 2^{-2\tau}) .$$

Finally, (3) follows directly from $\sum_{k=\tau+1}^{\infty} c_k = \Theta(\tau^{-1/2})$. ◀

Lemma 6 states that for some large threshold τ, the Taylor approximation error for $|x| \leqslant 1/2$ is a factor $2^{-2\tau}$ smaller than the error for $x = 1$. This gap allows us to prove the key lemma below:

▶ **Lemma 7.** *Let $d \in \mathbb{N}$, $d \geqslant 2$. Let $A \in \mathbb{R}^{d \times d}$ be a positive semidefinite matrix such that $A_{ii} = 1$ and $A_{ij} \geqslant -1/2$ for all $i, j \in [d]$. Let $w \in \mathbb{R}^d_{\geqslant 0}$ be a vector with non-negative entries. Then,*

$$\sum_{i,j=1}^{d} w_i w_j \arcsin(A_{ij}) \geqslant \Omega\left(\frac{\|w\|_1^2}{d\sqrt{\log d}} \right) .$$

Proof. Pick a threshold $\tau = C \log_2 d$ for a large enough constant C. Since we have the assumption that $A_{ij} \geqslant -1/2$ and $w_i \geqslant 0$ for all i, j, we can bound the off-diagonal entries using (1) and (2) of Lemma 6,

$$\sum_{i\neq j}^{d} w_i w_j \arcsin(A_{ij}) \geqslant \sum_{i\neq j}^{d} w_i w_j \left(\sum_{k=0}^{\tau} c_k A_{ij}^{2k+1} - O(\tau^{-1/2}2^{-2\tau})\right)$$
$$= \sum_{k=0}^{\tau} c_k \sum_{i\neq j}^{d} w_i w_j A_{ij}^{2k+1} - \widetilde{O}\left(\frac{\|w\|_1^2}{d^{2C}}\right).$$

The diagonal entries are $\arcsin(1) = \sum_{k=0}^{\tau} c_k + \Theta(\tau^{-1/2})$ by (3) of Lemma 6. Thus, we get

$$\sum_{i,j=1}^{d} w_i w_j \arcsin(A_{ij}) \geqslant \sum_{k=0}^{\tau} c_k \sum_{i,j=1}^{d} w_i w_j A_{ij}^{2k+1} + \sum_{i=1}^{d} w_i^2 \cdot \Omega(\tau^{-1/2}) - \widetilde{O}\left(\frac{\|w\|_1^2}{d^{2C}}\right)$$
$$= \sum_{k=0}^{\tau} c_k \sum_{i,j=1}^{d} w_i w_j A_{ij}^{2k+1} + \Omega\left(\frac{\|w\|_2^2}{\sqrt{\log d}}\right) - \widetilde{O}\left(\frac{\|w\|_1^2}{d^{2C}}\right).$$

Since $A \succeq 0$, by Fact 5 we know that $\sum_{ij=1}^{d} w_i w_j A_{ij}^{2k+1} \geqslant 0$ for all $k \geqslant 0$. Finally, any $w \in \mathbb{R}^d$ satisfies $\|w\|_2^2 \geqslant \frac{1}{d}\|w\|_1^2$. This completes the proof. ◀

2.1 Proof of Theorem 1

Let $\rho_* = \arg\min_{\rho\in[-1,1]} \frac{1+\frac{2}{\pi}\arcsin(\rho)}{1+\rho} \approx 0.689$ such that $\frac{1+\frac{2}{\pi}\arcsin(\rho_*)}{1+\rho_*} = \alpha_{\mathrm{GW}}$. Following [1], we can assume without loss of generality that for all $(i,j) \in E$ with sign b_{ij}, the SDP solution satisfies $b_{ij}\langle v_i, v_j\rangle \in [\rho_* - 0.01, \rho_* + 0.01]$.

Observe that after Item 2 of Algorithm 2, for every i in the candidate set S, by definition all edges between i and B_i are violated, while all edges between i and C_i are satisfied. Moreover, it is crucial that B_i and C_i are *disjoint* from S, so their assignments will not be flipped in Item 4. For edges between i and A_i, in the worst case all of them are violated after flipping. Thus, if $\sum_{j\in B_i} w_{ij} > \sum_{j\in A_i\cup C_i} w_{ij}$, then flipping x_i will increase the Max-2LIN value.

We will prove that the expected gains from such local updates are large. For a vertex $i \in S$, let $W_i \coloneqq \sum_{j\in N(i)} w_{ij}$ be the total weight of edges incident to i. Define the *local gain* from i to be

$$\Delta_i \coloneqq \left(\sum_{j\in B_i} w_{ij} - \sum_{j\in A_i\cup C_i} w_{ij}\right)_+ = \left(2\sum_{j\in B_i} w_{ij} - W_i\right)_+,$$

where we denote $(z)_+ = \max(0, z)$. The following is the key lemma of our analysis showing that conditioned on $i \in S$, the expected local gain from vertex i is at least $\widetilde{\Omega}(\frac{W_i}{d})$.

▶ **Lemma 8** (Expected local gain). *Let $d \in \mathbb{N}, d \geqslant 2$ and $\varepsilon = \frac{1}{Cd\sqrt{\log d}}$ for a large enough constant C. For a vertex $i \in V$ with degree d, the expected local gain $\mathbb{E}[\Delta_i | i \in S] \geqslant \Omega\left(\frac{W_i}{d\sqrt{\log d}}\right)$.*

Proof. Consider vertex $i \in S$ and its d neighbors, denoted $[d]$. We first introduce some notations for the analysis.

- We can assume that $v_i = (1, 0, \ldots, 0)$ without loss of generality due to rotational symmetry.
- For every neighbor $j \in [d]$, let $v_j = (\rho_j, v_j')$ where the first coordinate is $\rho_j \in b_{ij}\cdot[\rho_* \pm 0.01]$ and $v_j' \in \mathbb{R}^{n-1}$ since we assume that $b_{ij}\langle v_i, v_j\rangle \approx \rho_*$ for all $(i,j) \in E$.

- Denote the unit vector $\widehat{v}_j = v'_j/\|v'_j\|_2$.
- Let $g = (g_1, \ldots, g_n) \sim \mathcal{N}(0, \mathbb{I}_n)$ be the sampled Gaussian vector, and let $g' = (g_2, \ldots, g_n)$.
- Let $h_j \coloneqq b_{ij} x_i \langle \widehat{v}_j, g' \rangle$. The random vector $h = (h_1, \ldots, h_d)$ is a multivariate Gaussian variable with covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$ where $\Sigma_{jj} = 1$ and $\Sigma_{jk} = b_{ij} b_{ik} \langle \widehat{v}_j, \widehat{v}_k \rangle$.

Since $\|v_j\|_2 = 1$, we have $\|v'_j\|_2 = \sqrt{1-\rho_j^2} \in [0.726 \pm 0.01]$, and $\langle v_j, g \rangle = \rho_j g_1 + \sqrt{1-\rho_j^2}\langle \widehat{v}_j, g' \rangle$. Recall that $S = \{i \in V : \langle g, v_i \rangle \in (-\varepsilon, \varepsilon)\}$. Thus, $i \in S$ means that $\langle v_i, g \rangle = g_1 \in (-\varepsilon, \varepsilon)$. Next, we define the following random variable

$$Z \coloneqq \sum_{j=1}^{d} w_{ij} \cdot \mathbf{1}\left(h_j \leqslant -3\varepsilon\right) .$$

Conditioned on the event that $|g_1| < \varepsilon$,

$$h_j \leqslant -3\varepsilon \implies b_{ij} x_i \langle v_j, g \rangle \leqslant |\rho_j g_1| + \sqrt{1-\rho_j^2} \cdot h_j \leqslant -\varepsilon .$$

Therefore, we have $Z \leqslant \sum_{j \in B_i} w_{ij}$ (recall that $B_i = \{j \in N(i) : b_{ij} x_i \langle v_j, g \rangle \leqslant -\varepsilon\}$). Then,

$$\Delta_i = \left(2 \sum_{j \in B_i} w_{ij} - W_i \right)_+ \geqslant (2Z - W_i)_+ .$$

Thus, it suffices to lower bound $\mathbb{E}[(2Z - W_i)_+]$.

First, every h_j is a standard Gaussian, so let $p \coloneqq \mathbf{Pr}[h_j \leqslant -3\varepsilon] = \frac{1}{2} - c\varepsilon$ for some $c \leqslant \frac{3}{\sqrt{2\pi}}$ by Fact 3. Then, $\mathbb{E}[Z] = pW_i$. Next, we lower bound $\mathbb{E}[(Z - W_i/2)^2]$.

$$\begin{aligned}
\mathbb{E}\left[(Z - W_i/2)^2\right] &= \mathbb{E}\left(\sum_{j=1}^{d} w_{ij} \left(\mathbf{1}(h_j \leqslant -3\varepsilon) - \frac{1}{2} \right) \right)^2 \\
&= \frac{1}{4} \sum_{j=1}^{d} w_{ij}^2 + \sum_{j \neq k}^{d} w_{ij} w_{ik} \left(\mathbf{Pr}\left[h_j \leqslant -3\varepsilon \wedge h_k \leqslant -3\varepsilon\right] - p + \frac{1}{4} \right) \\
&\geqslant \frac{1}{4} \sum_{j=1}^{d} w_{ij}^2 + \sum_{j \neq k}^{d} w_{ij} w_{ik} \left(\frac{1}{2} - \frac{1}{2\pi} \arccos(\Sigma_{jk}) - 2c\varepsilon - \left(\frac{1}{4} - c\varepsilon \right) \right) \\
&\geqslant \frac{1}{2\pi} \sum_{j,k=1}^{d} w_{ij} w_{ik} \arcsin(\Sigma_{jk}) - c\varepsilon W_i^2 . \qquad (1)
\end{aligned}$$

The third line follows from $\mathbf{Pr}[h_j \leqslant -3\varepsilon \wedge h_k \leqslant -3\varepsilon] \geqslant \mathbf{Pr}\left[h_j \leqslant 0 \wedge h_k \leqslant 0\right] - 2 \cdot \mathbf{Pr}[-3\varepsilon \leqslant h_j \leqslant 0]$ and applying Fact 4. The final inequality is because $\frac{\pi}{2} - \arccos(\theta) = \arcsin(\theta)$ and $\arcsin(\Sigma_{jj}) = \arcsin(1) = \frac{\pi}{2}$.

By the triangle inequality of the SDP solution, for any $j \neq k$, $\|v_i - b_{ij} v_j\|^2 + \|v_i - b_{ik} v_k\|^2 \geqslant \|b_{ij} v_j - b_{ik} v_k\|^2$. Expanding this, we get

$$b_{ij} b_{ik} \left(\rho_j \rho_k + \sqrt{1-\rho_j^2}\sqrt{1-\rho_k^2} \langle \widehat{v}_j, \widehat{v}_k \rangle \right) \geqslant b_{ij}\rho_j + b_{ik}\rho_k - 1 .$$

Since $b_{ij}\rho_j \in [\rho_* \pm 0.01]$ for all $j \in [d]$, we have

$$\Sigma_{jk} = b_{ij} b_{ik} \langle \widehat{v}_j, \widehat{v}_k \rangle \geqslant -0.2 .$$

This is crucial since we can now apply Lemma 7 to Equation (1) and get

$$\mathbb{E}\left[(Z - W_i/2)^2\right] \geqslant \Omega\left(\frac{W_i^2}{d\sqrt{\log d}}\right) - O(\varepsilon W_i^2)\,.$$

Finally, we lower bound $\mathbb{E}[(2Z - W_i)_+]$. Let $\overline{Z} = Z - W_i/2$, and let $\overline{Z}_+ = \max(0, \overline{Z})$ and $\overline{Z}_- = \max(0, -\overline{Z})$. Thus, $\overline{Z} = \overline{Z}_+ - \overline{Z}_-$ and $\overline{Z}^2 = \overline{Z}_+^2 + \overline{Z}_-^2$ by definition, and both $\overline{Z}_+$ and $\overline{Z}_-$ lie in $[0, W_i/2]$. Furthermore, $\mathbb{E}[\overline{Z}] = \mathbb{E}[\overline{Z}_+] - \mathbb{E}[\overline{Z}_-] = pW_i - W_i/2 = -c\varepsilon W_i$. Then,

$$\mathbb{E}\left[\overline{Z}^2\right] = \mathbb{E}\left[\overline{Z}_+^2 + \overline{Z}_-^2\right] \leqslant \frac{W_i}{2} \cdot \mathbb{E}\left[\overline{Z}_+ + \overline{Z}_-\right] = W_i \cdot \mathbb{E}[\overline{Z}_+] + \frac{c}{2}\varepsilon W_i^2\,.$$

Setting $\varepsilon \leqslant \frac{1}{Cd\sqrt{\log d}}$ for a large enough C, we have $\mathbb{E}[\overline{Z}_+] \geqslant \Omega\left(\frac{W_i}{d\sqrt{\log d}}\right)$. This completes the proof. ◀

We can now prove Theorem 1.

Proof of Theorem 1. We first assume that for all $(i,j) \in E$ with sign b_{ij}, the SDP solution satisfies $b_{ij}\langle v_i, v_j\rangle \in [\rho_* - 0.01, \rho_* + 0.01]$ where $\rho_* \approx 0.689$. Recall that in Algorithm 2, for every $i \in S$, all edges between i and B_i are violated, and all edges between i and C_i are satisfied. Further, since B_i and C_i are disjoint from S, the vertices in B_i and C_i will not be flipped. For edges between i and A_i, in the worst case all of them are violated after flipping. Thus, we have a local gain of at least $\Delta_i = (\sum_{j\in B_i} w_{ij} - \sum_{j\in A_i\cup C_i} w_{ij})_+$.

The expected total gain from the local updates (over the random sample of g) is

$$\mathbb{E}[\Delta] = \mathbb{E}\sum_{i\in S}\Delta_i = \sum_{i\in V}\mathbb{E}\left[\mathbf{1}(i\in S)\Delta_i\right] = \sum_{i\in V}\mathbf{Pr}[i\in S]\cdot\mathbb{E}[\Delta_i | i\in S] \geqslant \Omega(\varepsilon)\sum_{i\in V}\mathbb{E}[\Delta_i | i\in S]\,,$$

where $\mathbf{Pr}[i \in S] = \Omega(\varepsilon)$ is due to Fact 3. Then, setting $\varepsilon = \frac{1}{Cd\sqrt{\log d}}$ for a large enough constant C, by Lemma 8 and the fact that the total weight $W = \frac{1}{2}\sum_{i\in V} W_i$, we have

$$\mathbb{E}[\Delta] \geqslant \Omega(\varepsilon)\cdot\sum_{i\in V}\Omega\left(\frac{W_i}{d\sqrt{\log d}}\right) = W\cdot\Omega\left(\frac{1}{d^2\log d}\right)\,.$$

Therefore, if the Max-2LIN instance φ has optimum $\mathsf{OPT} \leqslant W$, then in expectation we can find an assignment that satisfies

$$\varphi(x) \geqslant \alpha_{\mathrm{GW}}\cdot\mathsf{OPT} + W\cdot\Omega\left(\frac{1}{d^2\log d}\right) \geqslant \left(\alpha_{\mathrm{GW}} + \Omega\left(\frac{1}{d^2\log d}\right)\right)\cdot\mathsf{OPT}\,.$$

From here on, we follow the same argument as [1]. Let $\delta = \Omega(\frac{1}{d^2\log d})$ be the improvement above. If more than $\delta/2$ fraction of the (weighted) edges satisfy $b_{ij}\langle v_i, v_j\rangle \notin [\rho_* \pm 0.01]$, then hyperplane rounding already gives us $\alpha_{\mathrm{GW}} + \Omega(\delta)$ approximation ratio. If at most $\delta/2$ fraction of the edges have $b_{ij}\langle v_i, v_j\rangle \notin [\rho_* \pm 0.01]$, then we can simply ignore those edges and get an approximation of $(1 - \delta/2)(\alpha_{\mathrm{GW}} + \delta) \geqslant \alpha_{\mathrm{GW}} + \Omega(\delta)$. This completes the proof. ◀

References

1 Uriel Feige, Marek Karpinski, and Michael Langberg. Improved approximation of max-cut on graphs of bounded degree. *Journal of Algorithms*, 43(2):201–219, 2002.

2 Noah Fleming, Pravesh Kothari, and Toniann Pitassi. Semialgebraic Proofs and Efficient Algorithm Design. *Foundations and Trends in Theoretical Computer Science*, 14(1-2):1–221, 2019.

3 Mikael Florén. Approximation of max-cut on graphs of bounded degree, 2016.

4 Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

5 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 767–775, 2002.

6 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM Journal on Computing*, 37(1):319–357, 2007.

7 Luca Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 453–461, 2001.

Ellipsoid Fitting up to a Constant

Jun-Ting Hsieh
Carnegie Mellon University, Pittsburgh, PA, USA

Pravesh K. Kothari
Carnegie Mellon University, Pittsburgh, PA, USA

Aaron Potechin
University of Chicago, IL, USA

Jeff Xu
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

In [11, 13], Saunderson, Parrilo, and Willsky asked the following elegant geometric question: what is the largest $m = m(d)$ such that there is an ellipsoid in $\mathbb{R}^d$ that passes through $v_1, v_2, \ldots, v_m$ with high probability when the v_is are chosen independently from the standard Gaussian distribution $N(0, I_d)$? The existence of such an ellipsoid is equivalent to the existence of a positive semidefinite matrix X such that $v_i^\top X v_i = 1$ for every $1 \leqslant i \leqslant m$ – a natural example of a *random* semidefinite program. SPW conjectured that $m = (1 - o(1))d^2/4$ with high probability. Very recently, Potechin, Turner, Venkat and Wein [10] and Kane and Diakonikolas [8] proved that $m \gtrsim d^2/\log^{O(1)}(d)$ via a certain natural, explicit construction.

In this work, we give a substantially tighter analysis of their construction to prove that $m \gtrsim d^2/C$ for an absolute constant $C > 0$. This resolves one direction of the SPW conjecture up to a constant. Our analysis proceeds via the method of *Graphical Matrix Decomposition* that has recently been used to analyze correlated random matrices arising in various areas [3, 2]. Our key new technical tool is a refined method to prove singular value upper bounds on certain correlated random matrices that are tight up to absolute dimension-independent constants. In contrast, all previous methods that analyze such matrices lose logarithmic factors in the dimension.

2012 ACM Subject Classification Theory of computation → Semidefinite programming

Keywords and phrases Semidefinite programming, random matrices, average-case complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.78

Category Track A: Algorithms, Complexity and Games

Funding *Jun-Ting Hsieh*: Supported by NSF CAREER Award #2047933.
Pravesh K. Kothari: Supported by NSF CAREER Award #2047933, Alfred P. Sloan Fellowship and a Google Research Scholar Award.
Aaron Potechin: Supported in part by NSF grant CCF-2008920.
Jeff Xu: Supported in part by NSF CAREER Award #2047933, and a Cylab Presidential Fellowship.

1 Introduction

Given vectors $v_1, \ldots, v_m \in \mathbb{R}^d$, we say that these vectors satisfy the *ellipsoid fitting property* if there exists an origin-centered ellipsoid that passes through all these points, i.e., if there exists a matrix Λ such that

1. $v_i^T \Lambda v_i = 1$ for all $i \in [m]$,
2. $\Lambda \succeq 0$.

In this work, we study vectors sampled i.i.d. from the standard Gaussian distribution. It is known that when $m \leqslant d+1$, the vectors satisfy the ellipsoid fitting property with probability 1 [12]. On the other hand, when $m > \binom{d+1}{2}$, by a simple dimension argument, the vectors

don't satisfy the ellipsoid fitting property with probability 1. This prompts the question: *what is the largest $m = m(d)$ such that $v_1, \ldots, v_m \sim \mathcal{N}(0, I_d)$ satisfy the ellipsoid fitting property with probability at least $1 - o_d(1)$ (taking $d \to \infty$)?*

In a series of work, Saunderson et. al. [11, 12, 13] studied this problem in the context of diagonal and low-rank matrix decomposition. Motivated by numerical experiments, they conjectured that the ellipsoid fitting property for Gaussian random v_is exhibits a phase transition at $m \sim \frac{d^2}{4}$ (see also the experiments presented in [10]).

▶ **Conjecture 1** (SCPW conjecture). *Let $\varepsilon > 0$ be a constant and $v_1, \ldots, v_m \sim \mathcal{N}(0, I_d)$ be i.i.d. standard Gaussian vectors in $\mathbb{R}^d$. Then,*

1. *If $m \leqslant (1 - \varepsilon)\frac{d^2}{4}$, then $v_1, \ldots, v_m$ have the ellipsoid fitting property with probability $1 - o_d(1)$.*
2. *If $m \geqslant (1 + \varepsilon)\frac{d^2}{4}$, then $v_1, \ldots, v_m$ have the ellipsoid fitting property with probability $o_d(1)$.*

Prior works have focused on establishing the positive result – that is, part (1) of the above conjecture. Early works [11, 13] established that the ellipsoid fitting property holds for $m \leqslant O(d^{6/5-\varepsilon})$ independent Gaussian vector whp. In the context of proving Sum-of-Squares lower bounds for the Sherringtin-Kirkpatrick model, the work [4] obtains a result that, as an immediate corollary, improves the above bound to $O(d^{3/2-\varepsilon})$. In fact, their work gives an implicit bound of $m \leqslant O(d^2/\operatorname{polylog}(d))$ for ellipsoid fitting when restricted to degree-2 Sum-of-Squares.

Very recently, two independent works of Potechin et. al. [10] and Kane and Diakonikolas [8] proposed new constructions of Λ (that differ from the constructions obtained by the method of pseudo-calibration in [4]) and recovered the bound of $m \leqslant O(d^2/\operatorname{polylog}(d))$. In their works [10, 8], the authors ask the question of analyzing their construction (or a different one) to obtain an improved and almost optimal estimate of $m = d^2/C$ for some absolute constant $C > 0$. The main result of this paper accomplishes this goal. Specifically, we prove:

▶ **Theorem 2** (Main result). *There is a universal constant $c > 0$ such that if $m \leqslant cd^2$, then $v_1, \ldots, v_m \sim \mathcal{N}(0, I_d)$ have the ellipsoid fitting property with probability $1 - o_d(1)$.*

We establish Theorem 2 by analyzing the construction of Kane and Diakonikolas [8] (which is a variant of the construction proposed in [10]). Our key idea is to depart from the analysis conducted by [8] and instead rely on the *graphical matrix decomposition* method. This method decomposes a random matrix with correlated entries into a sum of structured random matrices called *graph matrices*. Graph matrices can be thought of as an analog of the *Fourier basis* in the analysis of functions over product spaces. This method was first employed in the works establishing tight sum-of-squares lower bound on the planted clique problem [5, 1, 3, 7] and has since then been employed in several follow-up works on proving sum-of-squares lower bounds and more recently in analyzing well-conditionedness of linear algebraic algorithms for generalizations of tensor decomposition [2]).

The key technical work in the analysis then becomes understanding the smallest and largest singular values of graph matrices. All prior works rely on arguments that establish bounds on the largest singular values that are accurate up to polylogarithmic factors in the underlying dimension of the matrices. The work of [2] recently showed how to use such bounds to also obtain estimates of the smallest singular values of graph matrices (which, otherwise are significantly more challenging to prove). Nevertheless, the slack in such bounds does not allow us to obtain any improvement on the previous estimates [8] in our application.

Our main technical contribution is a new technique to establish bounds on the largest singular values of graph matrices that are tight up to dimension-independent absolute constants. This allows us to obtain substantially improved estimates for the SCPW conjecture. Given the host of previous applications of such bounds, we expect that our results will have many more applications down the line.

Table 1 Comparison of our result with prior work.

Construction	Bound on m
Conjectured	$d^2/4$
[11, 13]	$O(d^{6/5-\varepsilon})$
[4]	$O(d^{3/2-\varepsilon})$ *
[10]	$O(d^2/\operatorname{polylog}(d))$
[8]	$O(d^2/\log^4(d))$
this paper	$O(d^2)$

*The bound $O(d^2/\operatorname{polylog}(d))$ is implicit in their work.

1.1 Technical overview

Following the convention of [8], for the rest of the paper we will assume that $v_1, \ldots, v_m \sim \mathcal{N}(0, \frac{1}{d} I_d)$ such that each vector has expected norm 1. Note that this does not change the problem as we can simply scale Λ.

Our construction of Λ is the "identity perturbation construction", which is the same one analyzed in [8] and was proposed in [10]. As an intuition, observe that $\Lambda = I_d$ almost works: $v_i^T I_d v_i = \|v_i\|_2^2 \approx 1$. Thus, the idea is to define Λ as a perturbation of I_d: $\Lambda = I_d - \sum_{i=1}^m w_i v_i v_i^T$, where $w = (w_1, \ldots, w_m) \in \mathbb{R}^m$. To determine w, observe that the constraints $v_i^T \Lambda v_i = 1$ give m linear constraints on w, and this can be written as a linear system represented by a matrix $M \in \mathbb{R}^{m \times m}$ with entries $M[i,j] = \langle v_i, v_j \rangle^2$. Thus, given that M is full rank, w is uniquely determined by $w = M^{-1}\eta$ for some vector η (see Eq. (2)). This construction satisfies $v_i^T \Lambda v_i = 1$ automatically, so the next thing is to prove that $\Lambda \succeq 0$. Therefore, we have two high-level goals:

1. Prove that M is full rank and analyze M^{-1}.
2. Prove that $R \coloneqq \sum_{i=1}^n w_i v_i v_i^T$ has spectral norm bounded by 1.

Proving the second statement immediately implies that Λ is a valid construction.

To achieve the first goal, we *decompose* M into several components. Roughly, we write $M = A + B$ where A is a perturbed identity matrix $A = I_m - T$ and B is a rank-2 matrix (see Section 2.2). We first show that $\|T\|_{\mathrm{op}} \leqslant O(\frac{\sqrt{m}}{d}) < 0.5$ with $m \leqslant O(d^2)$ (Lemma 9), hence A is well-conditioned. Then, using the fact that B has rank 2, we can apply the Woodbury matrix identity (Fact 7 and Fact 8) – a statement on the inverse of low-rank corrections of matrices – to conclude that M is invertible and obtain an expression for M^{-1}. This is carried out in Section 2.3.

Next, for the second goal, we need to further expand A^{-1}. Since $\|T\|_{\mathrm{op}} < 1$, we can apply the Neumann series and write $A^{-1} = (I_m - T)^{-1} = \sum_{k=0}^{\infty} T^k$. For the analysis, we select certain thresholds to truncate this series such that the truncation error is small. Then, we write M^{-1} in terms of the truncated series plus a small error, which will be useful later for the analysis of R. This is carried out in the full version.

Finally, given the expression of M^{-1}, R naturally decomposes into 4 matrices. Then, all we need to do is to bound the spectral norm of each of these matrices (see the full version). Bounding $\|R\|_{\mathrm{op}} \leqslant 1$ implies that $\Lambda \succeq 0$, completing the proof.

Requiring tight norm bounds. Our main technical lemmas are the spectral norm bounds of T (Lemma 9) and the matrices in the decomposition of R. Clearly, we need our norm bound $\|T\|_{\mathrm{op}} \leqslant O(\frac{\sqrt{m}}{d})$ to be tight without polylog factors so that $m \leqslant O(d^2)$ suffices, and similarly for matrices from R.

The standard starting point is the *trace moment method*: for any symmetric matrix $M \in \mathbb{R}^{n\times n}$ and $q \in \mathbb{N}$ (usually taking $q = \mathrm{polylog}(n)$ suffices),

$$\|M\|_{\mathrm{op}}^{2q} \leqslant \mathrm{tr}(M^{2q}) = \sum_{i_1,i_2,\ldots,i_{2q}\in[n]} M[i_1,i_2]M[i_2,i_3]\cdots M[i_{2q},i_1]\,.$$

We view the summand as a closed walk $i_1 \to i_2 \to \cdots \to i_{2q} \to i_1$ on n vertices. For a random matrix, we study the expected trace $\mathbb{E}\,\mathrm{tr}(M^{2q})$. In the simple case when M is a Gaussian matrix (GOE), we see that after taking the expectation, the non-vanishing terms are closed walks where each edge (u, v) is traversed even number of times. This is in fact true for any symmetric M with independent random entries as long as the odd moments of the entries are zero. Thus, a precise upper bound on $\mathbb{E}\,\mathrm{tr}(M^{2q})$ can be obtained by carefully counting such closed walks (see [14]).

Our matrices are more complicated; each entry is a mean-zero *polynomial* of Gaussian random variables. To carry out the trace method, we represent the matrices as graphs, hence the term *graph matrices*. The framework of graph matrices was first introduced by [3], and over the years, off-the-shelf norm bounds (e.g. [1]) for graph matrices have been developed and successfully used in several works [9, 4, 6, 7, 2]. However, the currently known norm bounds are only tight up to polylog factors, hence not sufficient for us. Therefore, the bulk of our paper is to prove norm bounds for these matrices that are tight up to constant factors. In fact, some of our bounds on graph matrices are even tight in the constant factor. However, we do not pursue the exact constants for two reasons. First, obtaining bounds which are tight in the constant factor would require additional technical work. Second, numerical experiments from [10] show that the identity perturbation construction we analyze has a threshold of $\frac{d^2}{C_{IP}}$ where $C_{IP} \approx 10$, so it falls short of the $\frac{d^2}{4}$ threshold and we would need a different construction to reach this threshold.

Key idea towards tight norm bounds. Here, we briefly discuss the high-level ideas for proving tight norm bounds. To illustrate our techniques, in Section 3 we will give a full proof for a matrix that arises in our analysis as an example, and also discuss key ideas that allow us to analyze more complicated matrices.

The key to counting walks is to specify an *encoding*, which we view as *information* required for a *walker* to complete a walk. If we can show that such an encoding uniquely identifies a walk, then we can bound the walks by bounding the number of possible encodings. Thus, it suffices to come up with an (efficient) encoding scheme and prove that the walker is able to complete a walk. Using standard encoding schemes, we quickly realize that the walker may be *confused* during the walk, i.e., the walker does not have enough information to perform the next step. Thus, we need to *pay* for additional information in the encoding to resolve confusions. So far, this is the same high-level strategy that was used in prior work [14, 1, 7], and this extra pay is often the source of extra log factors in the norm bounds.

Our key innovation is to pay for the extra information during steps that require much less information than normal. Roughly speaking, we label each step of the walk as either (1) visiting a new vertex, (2) visiting an old vertex via a new edge, (3) using an old edge but not the last time, (4) using an old edge the last time (see Definition 20). The high level idea is that the dominating walks in the trace are the ones that use only the 1st and 4th

types, while the 2nd and 3rd types require less information (which we call *gaps*). The main observation is that the walker will be confused only when there are steps of the 2nd and 3rd type involved, but we can pay extra information during these steps to resolve potential (future) confusions. This is illustrated in Section 3.5.

1.2 Comparison to prior work

Comparison to Kane and Diakonikolas [8]. Our candidate matrix Λ is the same as theirs. A slight difference is that they write $\Lambda = I_d + \sum_{i=1}^m \delta_i \overline{v}_i \overline{v}_i^T$ where $\overline{v}_1, \ldots, \overline{v}_m$ are the vectors normalized to the unit sphere. Then, same as our w vector, $(\delta_1, \ldots, \delta_m)$ must satisfy a linear system represented by a matrix $\overline{M} \in \mathbb{R}^{m \times m}$ where $\overline{M}[i,j] = \langle \overline{v}_i, \overline{v}_j \rangle^2$. This is closely related to our M matrix, and to prove that $\overline{M}$ is invertible, they also decompose $\overline{M}$ into several components and bound their spectral norms. However, they were only able to bound the spectral norm by $O(\frac{\sqrt{m}\log^2 d}{d})$, which requires $m \leqslant O(d^2/\log^4(d))$. We also point out that they explicitly emphasize the gap from spectral norm bound poses a significant hurdle in their analysis, which is indeed a major contribution of our work.

Next, to bound the spectral norm of $\overline{R} := \sum_{i=1}^m \delta_i \overline{v}_i \overline{v}_i^T$, they use an elegant cover (or ε-net) argument which is significantly different than ours. They show that for any fixed unit vector $u \in \mathcal{S}^{d-1}$, $|u^T \overline{R} u| = |\sum_{i=1}^m \delta_i \langle \overline{v}_i, u \rangle^2| \leqslant 1/2$ with *exponentially* small failure probability. This allows then to take a union bound over all $2^{O(d)}$ unit vectors in an ε-net. To do this, they use the elegant trick that $\overline{v}_i$ and $\|v_i\|_2$ are independent random variables, so $u^T \overline{R} u$ can be written as a sum of independent variables: $u^T \overline{R} u = \langle \varepsilon, \gamma \rangle$, where ε_i only depends on $\|v_i\|_2$ and γ is a function of u and the $\overline{v}_i$'s. By Hoeffding's inequality, they get a tail probability of $\exp\Big(-\Omega\Big(\frac{d^3}{m\log^2(d)}\Big)\Big)$. In order to union bound over $2^{O(d)}$ vectors, this also requires that $m \leqslant O(d^2/\log^2(d))$. Thus, while the main source of their polylog gap is their matrix norm bound, another source is the epsilon-net argument. This is partially why we adopt the proof strategy of using graph matrix decompositions which is seemingly more complicated.

Comparison to Potechin, Turner, Venkat and Wein [10]. They study a construction of "least-square minimization" proposed by [11], which is equivalent to projecting out the identity mass onto the subspace of matrices satisfying the constraints. In particular, their matrix analysis proceeding via Woodbury expansion and Neumann series using graph matrices serves as a road-map for our current work, and gives rise to a motivating question in the beginning for our work: can a more careful analysis get us all the way to a constant factor gap, or is the polylog gap inherent in the analysis? A priori, it is not clear whether this kind of matrix analysis, forsaking the underlying geometric insight, might get us anywhere beyond a single polylog factor, as it is conceivable that some polylog factor is inherent for matrices that may arise in the analysis. In this work, we answer this question affirmatively and en-route we develop a more refined understanding of the structured random matrices that we believe would be useful in further and more fine-grained investigations of problems in average-case complexity.

Comparison to Ghosh et. al. [4]. In the context of the Planted Affine Plane problem, and its downstream application for the Sherrington-Kirkpatrick Hamiltonian, Ghosh et. al. reaches the threshold of $\widetilde{O}(d^{3/2-\varepsilon})$ for $n^{O(\varepsilon)}$-degree Sum-of-Squares. They adopt the framework of pseudo-calibration [3] to obtain a candidate matrix, and follow a similar recipe as ours via graph matrix decompositions and spectral analysis. Even though their stated result falls

short of the $\widetilde{O}(d^2)$ threshold for fitting ellipsoid, it is folklore among the SoS lower bounds community that their proof implicitly extends to $\widetilde{O}(d^2)$ when restricted to degree-2 SoS. That said, it is an interesting question whether solutions coming from a pseudo-calibration type of construction might give us some extra mileage in ultimately closing the constant gap. A natural idea is to analyze the planted distribution pioneered in [9, 4]: unfortunately, it can be easily verified that the low-degree polynomial hardness for the particular planted distribution actually falls apart even if we assume an arbitrary constant gap. Since the low-degree hardness is usually deemed as a precursor for SoS lower bounds, an analysis based on pseudo-calibration that gets us the right constant (or in fact, any constant) lands one on a pursuit for a "quieter" planting.

2 Proof of main result

Given $v_1, v_2, \ldots, v_m$ that are i.i.d. samples from $\mathcal{N}(0, \frac{1}{d} I_d)$, recall that we must construct a matrix Λ such that (1) $v_i^T \Lambda v_i = 1$ for any $i \in [m]$, and (2) $\Lambda \succeq 0$.

In this section, we describe our candidate matrix (Definition 3). To prove that it satisfies the two conditions above, we need to analyze certain random matrices (and their inverses) that arise in the construction, which involves decomposing the matrices into simpler components. We will state our key spectral norm bounds (Lemma 9 and Lemma 13) whose proofs are deferred to later sections, and complete the proof of Theorem 2 in Section 2.4.

2.1 Candidate construction

The following is our candidate matrix Λ, which is the same as the one used in [8].

▶ **Definition 3** (Candidate matrix). *Given $v_1, \ldots, v_m \sim \mathcal{N}(0, \frac{1}{d} I_d)$, we define the matrix $\Lambda \in \mathbb{R}^{d \times d}$ to be*

$$\Lambda := I_d - \sum_{i=1}^{m} w_i v_i v_i^T \tag{1}$$

where we take $w = (w_1, w_2, \ldots, w_m)$ to be the solution to the linear system $Mw = \eta$ for $\eta \in \mathbb{R}^m$ given by

$$\eta_i := \|v_i\|_2^2 - 1, \quad \forall i \in [m]\,, \tag{2}$$

and $M \in \mathbb{R}^{m \times m}$ with entries given by

$$M[i,j] := \langle v_i, v_j \rangle^2, \quad \forall i, j \in [m]\,. \tag{3}$$

We first make the following simple observation.

▶ **Observation 4.** *For any $i \in [m]$, the constraint $v_i^T \Lambda v_i = 1$ is satisfied.*

Proof. For any $i \in [m]$,

$$v_i^T \Lambda v_i = v_i^T I_d v_i - \sum_{j \in [m]} w_j \langle v_i, v_j \rangle^2 = \|v_i\|_2^2 - \langle M[i], w \rangle = \|v_i\|_2^2 - \eta_i = 1\,.$$

Here $M[i]$ is the i-th row of M, and the equality above follows from $Mw = \eta$ and $\eta_i = \|v_i\|_2^2 - 1$ from Eq. (2). ◀

Structure of subsequent sections. For Λ to be well-defined, we require that M is full rank (hence invertible). Note that it is easy to see that M is positive semidefinite, since M is a Gram matrix with $M[i,j] = \langle v_i^{\otimes 2}, v_j^{\otimes 2} \rangle$. To analyze M, we will show a decomposition of M in Section 2.2 that allows us to more easily analyze its inverse. In Section 2.3, we will prove that M is in fact positive definite (Lemma 12).

Next, to prove that $\Lambda \succeq 0$, we will write $\Lambda = I_d - R$ where

$$R \coloneqq \sum_{i=1}^{m} w_i v_i v_i^T = \sum_{i=1}^{m} \left(M^{-1}\eta\right)[i] \cdot v_i v_i^T \,, \tag{4}$$

and prove that $\|R\|_{\mathrm{op}}$ is bounded by 1. Finally, combining the analyses, we finish the proof of Theorem 2 in Section 2.4.

2.2 Decomposition of M

The proof of Theorem 2 requires careful analysis of the matrix M from Eq. (3) and its inverse. To this end, we first decompose M as $M = A + B$ such that intuitively, A is perturbation of a (scaled) identity matrix and B has rank 2. We will later see how this decomposition allows us to analyze M^{-1} more conveniently.

▶ **Proposition 5** (Decomposition of M)**.**

$$M = \underbrace{M_\alpha + M_\beta + M_D + \left(1 + \frac{1}{d}\right) I_m}_{\coloneqq A} + \underbrace{\frac{1}{d} J_m + \frac{1}{d}\left(1_m \cdot \eta^T + \eta \cdot 1_m^T\right)}_{\coloneqq B} \tag{5}$$

where J_m is the all-ones matrix, M_α, M_β are matrices with zeros on the diagonal and M_D is a diagonal matrix, defined as follows:

- $M_\alpha[i,j] \coloneqq \sum_{a \neq b \in [d]} v_i[a] \cdot v_i[b] \cdot v_j[a] \cdot v_j[b]$ *for* $i \neq j \in [m]$,
- $M_\beta[i,j] \coloneqq \sum_{a \in [d]} \left(v_i[a]^2 - \frac{1}{d}\right)\left(v_j[a]^2 - \frac{1}{d}\right)$ *for* $i \neq j \in [m]$,
- $M_D[i,i] \coloneqq \|v_i\|_2^4 - \frac{2}{d}\|v_i\|_2^2 - 1$ *for* $i \in [m]$.

Proof. For any off-diagonal entry $i \neq j \in [m]$, on the right-hand side we have

$$\begin{aligned} M[i,j] = \langle v_i, v_j \rangle^2 &= \left(\sum_{a \in [d]} v_i[a] v_j[a]\right)^2 \\ &= \sum_{a \neq b \in [d]} v_i[a] \cdot v_i[b] \cdot v_j[a] \cdot v_j[b] + \sum_{a \in [d]} v_i[a]^2 \cdot v_j[a]^2 \,. \end{aligned}$$

The first term is exactly $M_\alpha[i,j]$. For the second term,

$$\begin{aligned} \sum_{a \in [d]} v_i[a]^2 \cdot v_j[a]^2 &= \sum_{a \in [d]} \left(v_i[a]^2 - \frac{1}{d}\right)\left(v_j[a]^2 - \frac{1}{d}\right) + \frac{1}{d}\left(\|v_i\|_2^2 + \|v_j\|_2^2\right) - \frac{1}{d} \\ &= \underbrace{\sum_{a \in [d]} \left(v_i[a]^2 - \frac{1}{d}\right)\left(v_j[a]^2 - \frac{1}{d}\right)}_{M_\beta[i,j]} + \underbrace{\frac{\|v_i\|_2^2 - 1}{d}}_{\frac{1}{d}\eta_i} + \underbrace{\frac{\|v_j\|_2^2 - 1}{d}}_{\frac{1}{d}\eta_j} + \frac{1}{d} \,. \end{aligned}$$

Thus, $M[i,j] = M_\alpha[i,j] + M_\beta[i,j] + \frac{1}{d} + \frac{1}{d}\left(1_m \cdot \eta^T + \eta \cdot 1_m^T\right)[i,j]$.

For the diagonal entries, the right-hand side of the (i,i) entry is

$$M_D[i,i] + \left(1+\frac{1}{d}\right) + \frac{1}{d} + \frac{2}{d}\eta_i = \left(\|v_i\|_2^4 - \frac{2}{d}\|v_i\|_2^2 - 1\right) + 1 + \frac{2}{d} + \frac{2}{d}(\|v_i\|_2^2 - 1)$$
$$= \|v_i\|_2^4 = M[i,i]\,.$$

This completes the proof. ◀

▶ Remark 6. The intention behind this decomposition is that for $v_i \sim \mathcal{N}(0, \frac{1}{d}I_d)$, M_α, M_β, M_D are all mean 0 (though their variances are not the same) since $\mathbb{E}\|v_i\|_2^2 = 1$ and $\mathbb{E}\|v_i\|_2^4 = 1 + \frac{2}{d}$. Therefore, we expect $\|M_\alpha + M_\beta + M_D\|_{\text{op}}$ to be small, which implies that A is positive definite and well-conditioned. Furthermore, observe that B has rank 2:

$$B = \frac{1}{d}J_m + \frac{1}{d}\left(1_m \cdot \eta^T + \eta \cdot 1_m^T\right) = \frac{1}{d}\begin{bmatrix}1_m & \eta\end{bmatrix} \cdot \begin{bmatrix}1 & 1\\ 1 & 0\end{bmatrix} \cdot \begin{bmatrix}1_m\\ \eta\end{bmatrix}. \tag{6}$$

2.3 Inverse of M

The decomposition of M into A and a rank-2 matrix B (Eq. (5)) allows us to apply the Woodbury matrix identity about the inverse of low-rank corrections of invertible matrices.

▶ **Fact 7** (Matrix Invertibility). *Suppose $A \in \mathbb{R}^{n_1\times n_1}$ and $C \in \mathbb{R}^{n_2\times n_2}$ are both invertible matrices, and $U \in \mathbb{R}^{n_1\times n_2}$ and $V \in \mathbb{R}^{n_2\times n_1}$ are arbitrary. Then, $A+UCV$ is invertible if and only if $C^{-1}+VA^{-1}U$ is invertible.*

▶ **Fact 8** (Woodbury matrix identity [15]). *Suppose $A \in \mathbb{R}^{n_1\times n_1}$ and $C \in \mathbb{R}^{n_2\times n_2}$ are both invertible matrices, and $U \in \mathbb{R}^{n_1\times n_2}$ and $V \in \mathbb{R}^{n_2\times n_1}$ are arbitrary. Then*

$$(A+UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1}+VA^{-1}U\right)^{-1}VA^{-1}\,.$$

In light of Fact 8, we can write B in Eq. (6) as $B = UCU^T$ where $U = V^T = \frac{1}{\sqrt{d}}\begin{bmatrix}1_m & \eta\end{bmatrix} \in \mathbb{R}^{m\times 2}$ and $C = \begin{bmatrix}1 & 1\\ 1 & 0\end{bmatrix}$, and $M = A + UCU^T$. Note that $C^{-1} = \begin{bmatrix}0 & 1\\ 1 & -1\end{bmatrix}$, and we have

$$C^{-1} + U^TA^{-1}U = \begin{bmatrix}\frac{1_m^TA^{-1}1_m}{d} & 1+\frac{\eta^TA^{-1}1_m}{d}\\ 1+\frac{\eta^TA^{-1}1_m}{d} & -1+\frac{\eta^TA^{-1}\eta}{d}\end{bmatrix} \coloneqq \begin{bmatrix}r & s\\ s & u\end{bmatrix}. \tag{7}$$

We first need to show that A is invertible. Recall from Eq. (5) that $A = (1+\frac{1}{d})I_m + M_\alpha + M_\beta + M_D$. We will prove the following lemma, whose proof is deferred to the full version.

▶ **Lemma 9** (M_α, M_β, M_D are bounded). *Suppose $m \leqslant cd^2$ for a small enough constant c. With probability $1-o_d(1)$, we have*

1. $\|M_\alpha\|_{\text{op}} \leqslant 0.1$,
2. $\|M_\beta\|_{\text{op}} \leqslant 0.1$,
3. $\|M_D\|_{\text{op}} \leqslant O\left(\sqrt{\frac{\log d}{d}}\right)$.

As an immediate consequence, we get the following:

▶ **Lemma 10** (A is well-conditioned). *With probability $1-o_d(1)$, the matrix A from Eq. (5) is positive definite (hence full rank), and*

$$0.5I_m \preceq A \preceq 1.5I_m\,.$$

Proof. Since $A = (1+\frac{1}{d})I_m + M_\alpha + M_\beta + M_D$, by Lemma 9 the eigenvalues of A must lie within $1 \pm 0.2 \pm \widetilde{O}(1/\sqrt{d}) \in (0.5, 1.5)$ (we assume d is large). ◀

Next, from Fact 7, we can prove that M is invertible (Lemma 12) by showing that the 2×2 matrix $C^{-1} + U^T A^{-1} U$ is invertible, which is in fact equivalent to $ru - s^2 \neq 0$. We first need the following bound on the norm of η, whose proof is deferred to the full version.

▷ Claim 11. With probability at least $1 - o_d(1)$, $\|\eta\|_2^2 \leqslant (1 + o_d(1))\frac{2m}{d}$.

▶ **Lemma 12** (Bounds on r, s, u; M is invertible). *Suppose $m \leqslant cd^2$ for a small enough constant c. Let $r, s, u \in \mathbb{R}$ be defined as in Eq. (7). With probability at least $1 - o_d(1)$, we have*

1. $r \in \frac{m}{d} \cdot [2/3, 2]$,
2. $|s| \leqslant O(\sqrt{d})$,
3. $u \in [-1, -1/2]$.

Thus, we have $s^2 - ru \geqslant \Omega\left(\frac{m}{d}\right)$. As a consequence, M is invertible.

Proof. By Lemma 10, we know that $\frac{2}{3}I_m \preceq A^{-1} \preceq 2I_m$. Thus, $r = \frac{1}{d}1_m^T A^{-1} 1_m \in \frac{1}{d}\|1_m\|_2^2 \cdot [2/3, 2]$, hence $r \in \frac{m}{d} \cdot [2/3, 2]$.

For s, we know that $\|\eta\|_2^2 \leqslant (1 + o(1))\frac{2m}{d}$ by Claim 11. Thus,

$$\frac{1}{d}\left|\eta^T A^{-1} 1_m\right| \leqslant \frac{1}{d}\|A^{-1}\|_{\mathrm{op}} \cdot \|\eta\|_2 \cdot \|1_m\|_2 < (1 + o_d(1)) \cdot 2\sqrt{\frac{2m^2}{d^3}} \leqslant O(\sqrt{d})\,.$$

Thus, $|s| = \left|1 + \frac{\eta^T A^{-1} 1_m}{d}\right| \leqslant O(\sqrt{d})$.

For u, we have

$$\frac{1}{d}\left|\eta^T A^{-1} \eta\right| \leqslant \frac{1}{d}\|A^{-1}\|_{\mathrm{op}} \cdot \|\eta\|_2^2 < (1 + o_d(1)) \cdot \frac{4m}{d^2} < \frac{1}{2}\,,$$

where the last inequality follows for some $m < cd^2$ for small enough c. Thus, $u = -1 + \frac{\eta^T A^{-1} \eta}{d} \in [-1, -1/2]$.

With the bounds on r, s and u, we immediately get $s^2 - ru \geqslant \Omega(\frac{m}{d})$.

To prove that M is invertible, let us first recall that we write $M = A + UCU^T$ where A is defined in Eq. (5) and $U = V^T = \frac{1}{\sqrt{d}}\begin{bmatrix} 1_m & \eta \end{bmatrix} \in \mathbb{R}^{m \times 2}$ and $C = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$.

By Lemma 10, A is invertible. Then by Fact 7, we know that M is invertible if and only if $C^{-1} + U^T A^{-1} U := \begin{bmatrix} r & s \\ s & u \end{bmatrix}$ (see Eq. (7)) is invertible, which is equivalent to $ru - s^2 \neq 0$. Thus, $s^2 - ru \geqslant \Omega(\frac{m}{d})$ suffices to conclude that M is invertible. ◀

2.4 Finishing the proof of Theorem 2

The final piece of proving Theorem 2 is to show that $R = \sum_{i=1}^m w_i v_i v_i^T$ has spectral norm bounded by 1, which immediately implies that the candidate matrix $\Lambda = I_d - R \succeq 0$.

▶ **Lemma 13** (R is bounded). *There exists some absolute constant c_R s.t.for $m \leqslant \frac{d^2}{c_R}$, whp*

$$\|R\|_{\mathrm{op}} \leqslant \frac{1}{2}\,.$$

The proof is deferred to the full version. In particular, we will write an expanded expression of M^{-1} and obtain a decomposition of R. Then, we prove tight spectral norm bounds for matrices in the decomposition, which then completes the proof of Lemma 13.

Combining Lemma 12 and Lemma 13 we can finish the proof of Theorem 2.

Proof of Theorem 2. The matrix M (recall Eq. (3)) is invertible due to Lemma 12, thus our candidate matrix $\Lambda = I_d - R$ matrix defined in Definition 3 is well-defined. Furthermore, by the norm bound in Lemma 13, we have $\|R\|_{\mathrm{op}} < 1$. This proves that $\Lambda \succ 0$. ◀

3 Machinery for tight norm bounds of graph matrices

One of the main technical contributions of this paper is providing tight spectral norm bounds (up to constants per vertex/edge) for *structured random matrices with correlated entries* (a.k.a. graph matrices). We note that prior to this work, most known norm bounds for such matrices are only tight up to some logarithmic factors [1], while not much is known in terms of precise bounds without log factors except for several specific cases (see e.g. [14]).

3.1 Preliminaries

We first give a lightweight introduction to the theory of graph matrices. For interested readers who seek a thorough introduction or a more formal treatment, we refer them to its origin in a sequence of works in Sum-of-Squares lower bounds [3, 1]. We will follow the notations used in [1]. Throughout this section, we assume that there is an underlying (random) input matrix G and a Fourier basis $\{\chi_t\}_{t\in\mathbb{N}}$.

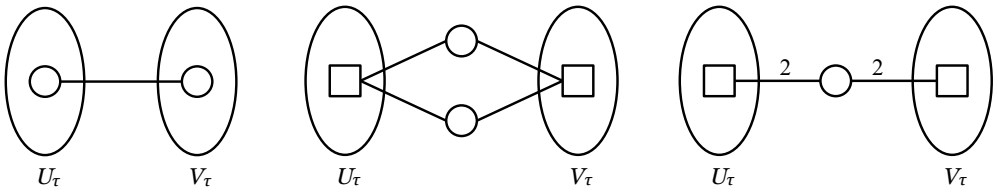
We first define *shapes*, which are representations of structured matrices whose entries depend on G.

▶ **Definition 14** (Shape). *A shape τ is a tuple $(V(\tau), U_\tau, V_\tau, E(\tau))$ associated with a (multi) graph $(V(\tau), E(\tau))$. Each vertex in $V(\tau)$ is associated with a vertex-type that indicates the range of the labels for the particular vertex. Each edge $e \in E(\tau)$ is also associated with a Fourier index $t(e) \in \mathbb{N}$. Moreover, we have $U_\tau, V_\tau \subseteq V(\tau)$ as the left and right boundary of the shape.*

We remind the reader that V_τ should be distinguished from $V(\tau)$, where V_τ is the right boundary set, while $V(\tau)$ is the set of all vertices in the graph.

Figure 1 show the shapes for matrices M_α and M_β defined in Proposition 5. For these shapes, there are two vertex-types (square and circle). The two ovals in each shape indicate the left and right boundaries U_τ and V_τ.

We next describe how to associate a shape to a matrix (given the underlying matrix G).



(a) GOE, zero diagonal. **(b)** M_α. **(c)** M_β.

Figure 1 Graph matrix representation of a $d \times d$ GOE matrix with zero diagonal, and the $m \times m$ matrices M_α and M_β as defined in Proposition 5. Square vertices take labels in $[m]$ and circle vertices take labels in $[d]$. The two ovals indicate the left and right boundaries of the shapes. If an edge e is not labeled with an index, then $t(e) = 1$ by default.

▶ **Definition 15** (Mapping of a shape). *Given a shape τ, we call a function $\sigma : V(\tau) \to \mathbb{N}$ a mapping of the shape if*

1. *σ assigns a label for each vertex according to its specified vertex-type;*
2. *σ is an injective mapping for vertices of the same type.*

▶ **Definition 16** (Graph matrix for shape). *Given a shape τ, we define its graphical matrix M_τ to be the matrix indexed by all possible boundary labelings of S, T, and for each of its entry, we define*

$$M_\tau[S,T] = \sum_{\substack{\sigma: V(\tau)\to\mathbb{N} \\ \sigma(U_\tau)=S,\ \sigma(V_\tau)=T}} \prod_{e\in E(\tau)} \chi_{t(e)}(G[\sigma(e)])\,.$$

Observe that for each entry $M_\tau[S,T]$, since σ must map U_τ and V_τ to S and T, $M_\tau[S,T]$ is simply a sum over labelings of the "middle" vertices $V(\tau)\setminus(U_\tau\cup V_\tau)$. Take Figure 1 for example. Suppose $G\in\mathbb{R}^{m\times d}$ and square and circle vertices take labels in $[m]$ and $[d]$ respectively, then we can write out the entries of the matrix: for $i\neq j\in[m]$,

$$M_\alpha[i,j] = \sum_{a\neq b\in[d]} \chi_1(G[i,a])\cdot\chi_1(G[i,b])\cdot\chi_1(G[j,a])\cdot\chi_1(G[j,b])\,,$$
$$M_\beta[i,j] = \sum_{a\in[d]} \chi_2(G[i,a])\cdot\chi_2(G[j,a])\,.$$

Note also that since σ must be injective for vertices of the same type and $U_\tau\neq V_\tau$ in both examples, there is no mapping such that $\sigma(U_\tau)=\sigma(V_\tau)$. Thus, by Definition 16, both matrices have zeros on the diagonal.

Adaptation to our setting. The above is a general introduction for graph matrices. In this work, we specialize to the following setting:

- $G\in\mathbb{R}^{m\times d}$ is a random Gaussian matrix whose rows are $v_1,\dots,v_m\sim\mathcal{N}(0,\frac{1}{d}I_d)$.
- The Fourier characters $\{\chi_t\}_{t\in\mathbb{N}}$ are the (scaled) Hermite polynomials.
- For all graph matrices that arise in our analysis,
  - $|S|=|T|=1$,
  - There are two vertex-types: square vertices take labels in $[m]$ and circle vertices take labels in $[d]$.

▶ Remark 17. For our technical analysis, we also employ our techniques on a generalization of graph matrices where we relax the injectivity condition. That said, for the purpose of illustrating our techniques, it suffices to consider ordinary graph matrices.

▶ **Definition 18** (D_V size constraint). *Let D_V be a size constraint such that for each graph matrix τ considered in this work, $|V(\tau)|\leqslant D_V$.*

For concreteness, we will take $D_V=\mathrm{polylog}(d)$ throughout this work.

Trace moment method. For all our norm bounds, we will use the trace moment method: for any graph matrix M_τ with underlying random matrix G and any $q\in\mathbb{N}$,

$$\mathbb{E}\|M_\tau\|_{\mathrm{op}}^{2q}\leqslant\mathbb{E}\,\mathrm{tr}\left((M_\tau M_\tau^T)^q\right)=\mathbb{E}\sum_{\substack{S_1,T_1,S_2,T_2,\dots S_{q-1},T_{q-1}: \\ \text{boundaries}}} M_\tau[S_1,T_1]M_\tau^T[T_1,S_2]\cdots M_\tau^T[T_{q-1},S_1]\,.$$

where the expectation is taken over G.

Notice that the summation is over *closed walks* across the boundaries: $S_1\to T_1\to S_2\to T_2\to\cdots\to S_1$, where $S_1,T_1,\dots$ are boundary labelings of M_τ. In particular, the walk is consist of $2q$-steps of a "block walk", with the $(2t-1)$-th step across a block described by M_τ and the $(2t)$-th step across a block described by M_τ^T.

The crucial observation is that after taking expectation, all closed walks must walk on each labeled edge (i.e., Fourier character) an *even* number of times, since all odd moments of the Fourier characters are zero. Therefore, bounding the matrix norm is reduced to bounding the contribution of all such walks.

$$\mathbb{E}\|M_\tau\|_{\mathrm{op}}^{2q} \leqslant \sum_{\mathcal{P}\text{: closed walk}} \prod_{e\in E(\mathcal{P})} \mathbb{E}\left[\chi_{t(e)}(G[e])^{\mathrm{mul}_{\mathcal{P}}(e)}\right], \tag{8}$$

where $E(\mathcal{P})$ denotes the set of labeled edges used by the walk $\mathcal{P}$, $\mathrm{mul}_{\mathcal{P}}(e)$ denotes the number of times e appears in the walk, and $t(e)$ denotes the Fourier index (with slight abuse of notation).

▶ Remark 19. We remind the reader not to confuse vertices/edges in the walk with vertices/edges in the shape. The vertices in a walk are "labeled" by elements in $[m]$ or $[d]$ (depending on the vertex-type). Similarly, each edge $e \in E(\mathcal{P})$ in a walk is labeled by an element in $[m] \times [d]$. We will use the terms "labeled vertex" and "labeled edge" unless it is clear from context.

3.2 Global bounds via a local analysis

Observe that Eq. (8) is a weighted sum of closed walks of length $2q$. To obtain an upper bound, the standard approach is to specify an efficient *encoding scheme* that uniquely identifies each closed walk, and then upper bound the total number of such encodings.

We begin by defining a step-labeling – a categorization of each step in the closed walk.

▶ **Definition 20** (Step-labeling). *For each step throughout the walk, we assign it the following label,*

1. *F (a fresh step): it uses a new labeled edge making the first appearance and leads to a destination not seen before;*
2. *S (a surprise step): it uses a new labeled edge to arrive at a vertex previously visited in the walk;*
3. *H (a high-mul step): it uses a labeled edge that appears before, and the edge is making a middle appearance (i.e., it will appear again in the subsequent walk);*
4. *R (a return step): it uses a labeled edge that appears before, and the edge is making its last appearance.*

Analogously, for any shape τ , we call $\mathcal{L}_\tau : E(\tau) \to \{F, R, S, H\}$ a step-labeling *of the block. The subscript τ is ignored when it is clear.*

We note that the terms "fresh", "high-mul" and "return" are adopted from the GOE matrix analysis in [14]. Next, to obtain a final bound for Eq. (8), we consider two *factors* for each step (which depend on the step-label):

1. **Vertex factor**: a combinatorial factor that specifies the destination of the step;
2. **Edge factor**: an analytical factor from the edge which accounts for the $\mathbb{E}[\chi_{t(e)}(G[e])^{\mathrm{mul}(e)}]$ term in Eq. (8).

For example, a vertex factor for an F step to a circle vertex can be d, an upper bound on the number of possible destinations. One can think of vertex factors as the information needed for a *decoder* to complete a closed walk. Essentially, the step-labeling and appropriate vertex factors should uniquely identify a closed walk, and combined with edge factors, we can obtain an upper bound for Eq. (8).

We note that the approach stated above is a *global* encoding scheme. One may proceed via a global analysis – carefully bounding the number of step-labelings allowed (e.g., using the fact that the F and R steps must form a Dyck word [14]), and then combining all vertex and edge factors to obtain a final bound. However, to get tight norm bounds for complicated graph matrices (like M_α), the global analysis becomes unwieldy.

Local analysis. One of our main insights is to use a *local* analysis. We now give a high-level overview of our strategy while deferring the specific details of our vertex/edge factor assignment scheme to subsequent sections. Recall that a closed walk consists of "block-steps" described by the shape τ. Thus, we treat each walk as a "block walk" and bound the contributions of a walk block by block. This prompts us to bound the contribution of the walk at a given block-step to the final trace in Eq. (8) by

$$\mathsf{vtxcost} \cdot \mathsf{edgeval} \leqslant B_q(\tau)$$

where $B_q(\tau)$ is some desired upper bound that depends on the vertex/edge factor assignment scheme. We define it formally in the following.

▶ **Definition 21** (Block value function). *Fix $q \in \mathbb{N}$ and a shape τ. For any vertex/edge factor assignment scheme, we call $B_q(\tau)$ a valid block-value function for τ of the given scheme if*

$$\mathbb{E}\left[\mathrm{tr}\left((M_\tau M_\tau^T)^q\right)\right] \leqslant (\textit{matrix dimension}) \cdot B_q(\tau)^{2q}\,,$$

and for each block-step $\mathsf{BlockStep}_i$ throughout the walk,

$$\mathsf{vtxcost}(\mathsf{BlockStep}_i) \cdot \mathsf{edgeval}(\mathsf{BlockStep}_i) \leqslant B_q(\tau)\,.$$

We point out that the block-value function B should be considered as a function of both the shape τ and the length of the walk q (we will drop the subscript when it is clear throughout this work), and it also depends on the assignment scheme. Thus, our task is to find a vertex/edge factor assignment scheme such that $B_q(\tau)$ is as small as possible. Moreover, the matrix dimension, which is at most $\mathrm{poly}(d)$ in our case, is the factor that comes up in the start of the walk to specify the original vertex, and can be ignored as it is ultimately an $1 + o(1)$ factor once we take a long enough walk.

Given Definition 21, the norm bound follows immediately.

▶ **Proposition 22.** *Let M_τ be a graph matrix with dimension $\mathrm{poly}(d)$, and let $q = \Omega(\log^2 d)$. Suppose $B_q(\tau)$ is a valid block-value function. Then, with probability $1 - \frac{1}{\mathrm{poly}(d)}$,*

$$\|M_\tau\|_{\mathrm{op}} \leqslant (1 + o_d(1)) \cdot B_q(\tau)\,.$$

Proof. We apply Markov's inequality: for any $\varepsilon > 0$,

$$\begin{aligned}\mathbf{Pr}\left[\|M_\tau\|_{\mathrm{op}} > (1+\varepsilon)B_q(\tau)\right] &\leqslant \mathbf{Pr}\left[\mathrm{tr}\left((M_\tau M_\tau^T)^q\right) > (1+\varepsilon)^{2q} B_q(\tau)^{2q}\right] \\ &\leqslant (1+\varepsilon)^{-2q}\,\mathrm{poly}(d) \\ &\leqslant \frac{1}{\mathrm{poly}(d)}\end{aligned}$$

for $q = \Omega(\frac{1}{\varepsilon}\log d)$. Setting $\varepsilon = \frac{1}{\log d}$, we can conclude that $\|M_\tau\|_{\mathrm{op}} \leqslant (1 + o_d(1)) \cdot B_q(\tau)$ with high probability. ◀

The next proposition shows that we can easily obtain a valid $B_q(\tau)$ once we have an appropriate factor assignment scheme.

▶ **Proposition 23.** *For any graph matrix M_τ and any valid factor assignment scheme,*

$$B_q(\tau) = \sum_{\mathcal{L}:\text{step-labelings for } E(\tau)} \mathsf{vtxcost}(\mathcal{L}) \cdot \mathsf{edgeval}(\mathcal{L})$$

is a valid block-value function for τ.

Proof. The second requirement in Definition 21 is clear. For the first requirement, observe that the trace can be bounded by the matrix dimension (specifying the start of the walk) times

$$\sum_{\substack{\mathcal{L}_1,\ldots,\mathcal{L}_{2q}:\\ \text{step-labelings for } E(\tau)}} \prod_{i=1}^{2q} \mathsf{vtxcost}(\mathcal{L}_i) \cdot \mathsf{edgeval}(\mathcal{L}_i) \leqslant \left(\sum_{\mathcal{L}:\text{step-labelings for } E(\tau)} \mathsf{vtxcost}(\mathcal{L}) \cdot \mathsf{edgeval}(\mathcal{L}) \right)^{2q}. \quad \blacktriangleleft$$

With this set-up, the main task is then to find an appropriate vertex/edge factor assignment scheme and obtain a good upper bound on $B_q(\tau)$.

3.3 Vertex factor assignment scheme

We now proceed to bound the vertex factors for each step-label. We note that in this section, "vertices" refer to "labeled vertices" in the walk (having labels in $[m]$ or $[d]$; recall Remark 19). First, we define the weight of a square (resp. circle) vertex to be m (resp. d), since we need an element in $[m]$ (resp. $[d]$) to specify which vertex to go to in the walk.

We first show a "naive" vertex factor assignment scheme. In the following scheme, we use a *potential unforced return* factor, denoted Pur, to specify the destination of any R step. We will defer the specific details of Pur to Section 3.5.

Vanilla vertex factor assignment scheme.

1. For each vertex i that first appears via an F step, a label in $\mathsf{weight}(i)$ is required;
2. For each vertex i that appears beyond the first time:
   - If it is arrived via an R step, the destination may need to be specified, and this is captured by the Pur factor.
   - If it is *not* arrived via an R step, then it must be an S or H step. A vertex cost in $2q \cdot D_V$ is sufficient to identify the destination, where we recall $2q$ is the length of our walk, and D_V the size upper bound of each block.

The first thing to check is that this scheme combined with an step-labeling uniquely identifies a closed walk (given the start of the walk). This is immediate for F and R steps by definition. For S and H steps, since the destination is visited before in the walk, $2q \cdot D_V$ is sufficient as it is an upper bound on the number of vertices in the walk.

A potential complication with analyzing the above assignment scheme directly is that it exhibits a significant difference in the vertex factors. For example, consider a vertex that appears only twice in the walk on a tree. Its first appearance requires a label in $[n]$, while its subsequent appearance does not require any cost if it is reached using an R step because backtracking from a tree is fixed (since there is only one parent). This disparity can result in a very loose upper bound for the trace when applying Proposition 23; in fact, the norm bound for M_τ obtained in this manner is equivalent to using the naive row-sum bound.

Redistribution. One of our main technical insights is to split the factors such that both first and last appearance contributes a factor of comparable magnitude; we call this *redistribution*.

We first formally define "appearance" in a block-step to clarify our terminology,

▶ **Definition 24** (Vertex appearance in block-step). *Each labeled vertex appearance can be "first", "middle" and "last". Moreover, each vertex on the block-step boundary (U_τ or V_τ) appears in both adjacent blocks.*

For example, suppose a vertex first appears in the right-boundary of block i and last appears in the left-boundary of block j, then it will make middle appearances in the left-boundary of block $i+1$ and right-boundary of block $j-1$ as well.

We are now ready to introduce the following vertex-factor assignment scheme with redistribution that assigns vertex-factor to each vertex's appearance to handle the disparity.

Vertex factor assignment scheme with redistribution.

1. For each vertex i that makes its first appearance, assign a cost of $\sqrt{\mathsf{weight}(i)}$;
2. For any vertex's middle appearance, if it is not arrived at via an R step, assign a cost of $2q \cdot D_V$ (where we recall $2q$ is the length of our walk, and D_V the size constraint of each block);
3. For any vertex's middle appearance, if it is at arrived via an R step, its cost is captured by Pur;
4. For each vertex i that makes its last appearance, assign a cost of $\sqrt{\mathsf{weight}(i)}$ that serves as a *backpay*.

Deducing vertex factor from local step-labeling. As presented, the vertex factor assignment scheme requires knowing which vertex is making first/middle/last appearance. We further show that the vertex appearances, or more accurately, an upper bound of the vertex factors, can be deduced by a given step-labeling of the block. Fix traversal direction from U to V,

Localized vertex factor assignment from step-labeling.

1. For any vertex v that is on the left-boundary U, it cannot be making the first appearance since it necessarily appears in the previous block;
2. For any vertex v that is on the right-boundary V, it cannot be making the last appearance since it necessarily appears in the subsequent block;
3. For any vertex v reached via some $S/R/H$ step, it cannot be making its first appearance;
4. For any vertex v that incident to some $F/S/H$ step, it cannot be making its last appearance since the edge necessarily appears again.

The first two points are due to Definition 24. The last point is because each labeled edge (i.e., Fourier character) must be traversed by an R step to close it.

3.4 Bounding edge-factors

To bound the contribution of the walks, we need to consider factors coming from the edges traversed by the walk. Recall from Eq. (8) that each edge e in a closed walk P gets a factor $\mathbb{E}[\chi_{t(e)}^{\mathrm{mul}_P(e)}]$, where $t(e)$ is the Fourier index associated with the edge.

In our case, the Fourier characters are the scaled Hermite polynomials. Recall that we assume that our vectors are sampled as $v_i \sim \mathcal{N}(0, \frac{1}{d} I_d)$. Thus, we define the polynomials $\{H_t\}_{t\in\mathbb{N}}$ such that they are orthogonal and $\mathbb{E}_{x\sim\mathcal{N}(0,1/d)}[H_t(x)^2] = t! \cdot d^{-t}$. Specifically,

1. $H_1(x) = x$,
2. $H_2(x) = x^2 - \frac{1}{d}$.

We first state the following bound on the moments of H_t, which follows directly from standard bounds on the moments of Hermite polynomials:

▶ **Fact 25** (Moments of Hermite polynomials). *Let $d \in \mathbb{N}$. For any $t \in \mathbb{N}$ and even $k \in \mathbb{N}$,*

$$\mathbb{E}_{x\sim\mathcal{N}(0,1/d)}\left[H_t(x)^k\right] \leqslant \frac{1}{d^{kt/2}}(k-1)^{kt/2}(t!)^{k/2} \leqslant (t!)^{k/2}\left(\frac{k}{d}\right)^{kt/2}.$$

For matrices that arise in our analysis, we only have H_1 and H_2 edges. The following is our edge-factor assignment scheme to account for contributions from the Fourier characters.

Edge-factor assignment scheme.

For an H_1 edge,

1. F/S: assign a factor of $\frac{1}{\sqrt{d}}$ for its first appearance;
2. H: assign a factor of $\frac{2q}{\sqrt{d}}$ for its middle appearance;
3. R: assign a factor of $\frac{1}{\sqrt{d}}$ for its last appearance.

For an H_2 edge,

1. F/S: assign a factor of $\frac{\sqrt{2}}{d}$ for its first appearance (equivalently, we can view a single H_2 edge as two edge-copies of H_1 and assign each a factor of $\frac{\sqrt{2}}{\sqrt{d}}$ which is a valid upper bound);
2. H: assign a factor of $\frac{8q^2}{d}$ for its middle appearance;
3. R: assign a factor of $\frac{\sqrt{2}}{d}$ for its last appearance (equivalently, we can view a single H_2 edge as two edge-copies of H_1 and assign each a factor of $\frac{\sqrt{2}}{\sqrt{d}}$ which is a valid upper bound).

▶ **Proposition 26.** *The above scheme correctly accounts for the edge factors from H_1 and H_2 edges.*

Proof. If an edge has multiplicity 2, then it must be traversed by one F/S step and one R step.

- If it is an H_1 edge, then the scheme assigns a factor $\frac{1}{d}$, which equals $\mathbb{E}_{x\sim\mathcal{N}(0,1/d)}[H_1(x)^2]$.
- If it is an H_2 edge, then the scheme assigns a factor $\frac{2}{d^2}$, which equals $\mathbb{E}_{x\sim\mathcal{N}(0,1/d)}[H_2(x)^2]$.

For an edge with multiplicity $k > 2$, it must be traversed by one F/S step, one R step and $k-2$ H steps. Moreover, since k is even and $2q$ is the length of the walk, we have $4 \leqslant k \leqslant 2q$.

- If it is an H_1 edge, then the scheme assigns a factor $\frac{1}{d}\cdot(\frac{2q}{\sqrt{d}})^{k-2} \geqslant d^{-k/2}(2q)^{k/2} \geqslant (\frac{k}{d})^{k/2}$. By Fact 25, it is an upper bound on $\mathbb{E}_{x\sim\mathcal{N}(0,1/d)}[H_1(x)^k]$.
- If it is an H_2 edge, then the scheme assigns a factor $\frac{2}{d^2}\cdot(\frac{8q^2}{d})^{k-2} \geqslant d^{-k}2^{k/2}(2q)^k \geqslant 2^{k/2}(\frac{k}{d})^k$. By Fact 25, it is an upper bound on $\mathbb{E}_{x\sim\mathcal{N}(0,1/d)}[H_2(x)^k]$.

This shows that the edge factor assignment scheme above is correct. ◀

3.5 Bounding return cost (Pur factors)

In our vertex factor assignment scheme described in Section 3.3, we use a *potential unforced return* factor, denoted Pur, to specify the destination of any return (R) step. Note that the term "unforced return" is adopted from [14] as well. In this section, we complete the bound of vertex factors by bounding the Pur factor.

For starters, we will define a potential function for each vertex at time t, which measures the number of returns R pushed out from the particular vertex by time t that may require a label in $2q \cdot D_V$. Notice that a label in $2q \cdot D_V$ is sufficient for any destination vertex arrived via an R step because the vertex appears before; however, this may be a loose bound.

We observe the following: a label in $2q \cdot D_V$ may be spared if the vertex is incident to only one un-closed F/S edge; we call this a *forced* return. Formally, we define a return step as *unforced* if it does not fall into the above categories,

▶ **Definition 27** (Unforced return). *We call a return (R) step an* unforced return *if the source vertex is incident to more than* 1 *(or* 2 *in the case of a square vertex) unclosed edge.*

We now proceed to formalize the above two observations by introducing a potential function to help us bound the number of unforced returns from any given vertex throughout the walk. The number of unforced returns throughout the walk would then be immediately given once we sum over all vertices in the walk.

▶ **Definition 28** (Potential-unforced-return factor Pur). *For any time t and vertex v, let* $\mathsf{Pur}_t(v)$ *be defined as the number of potential unforced return from v throughout the walk until time t.*

3.5.1 Pur bound for circle vertices

In our setting, each circle vertex pushes out at most 1 edge during the walk, analogous to the case of typical adjacency matrix. This serves as a starting point for our Pur bound for circle vertices.

▶ **Lemma 29** (Bounding Pur_t for circle vertices). *For any time t, suppose the walker is currently at a circle vertex v, then*

$$\begin{aligned}\mathsf{Pur}_t(v) &\leqslant \#(R \text{ steps closed from } v) + \#(\text{unclosed edges incident to } v \text{ at time } t) - 1\\ &\leqslant 2 \cdot s_t(v) + h_t(v)\,,\end{aligned}$$

where we define the following counter functions:

1. *$s_t(v)$ is the number of S steps arriving at v by time t;*
2. *$h_t(v)$ is the number of H steps arriving at v by time t.*

Proof. We first prove the first inequality. The R steps closed from v may all be unforced returns, and the unclosed edges incident to v may be closed by unforced returns in the future. Note that we have a -1 in the above bound because for each vertex we may by default assume the return is using a particular edge, hence at each time we know there is an edge presumed-to-be forced.

We prove the second inequality by induction. Define $P_t(v) \coloneqq \#(\text{R steps closed from } v) + \#(\text{unclosed edges incident to } v \text{ at time } t) - 1$ for convenience. At the time when v is first created by an F step, $P_t(v) = 0$ (1 open edge minus 1) and $s_t(v) = h_t(v) = 0$.

At time t, suppose the last time v was visited was at time $t' < t$, and suppose that the inequality holds true for t'. Note that at time $t' + 1$, $P_{t'+1}(v) = P_{t'}(v) + 1$ if a new edge was created by an F or N step leaving v, otherwise $P_{t'+1}(v) = P_{t'}(v)$ (for R step it adds 1 to the number of closed edges closed from v, but decreases 1 open edge). On the other hand, $s_{t'}(v)$ and $h_{t'}(v)$ remain the same (we don't count out-going steps for $s_t(v), h_t(v)$).

When we reach v at time t, we case on the type of steps:

- Arriving by an R step: the edge is now closed, but the R step was not from v. So $P_t(v) = P_{t'+1}(v) - 1 \leqslant P_{t'}(v)$, while $s_t(v) = s_{t'}(v)$ and $h_t(v) = h_{t'}(v)$.
- Arriving by an S step: the edge is new, so $P_t(v) = P_{t'+1}(v) + 1 \leqslant P_{t'}(v) + 2$, and we have $s_t(v) = s_{t'}(v) + 1$.
- Arriving by an H step: $P_t(v) = P_{t'+1}(v) \leqslant P_{t'}(v) + 1$, and $h_t(v) = h_{t'}(v) + 1$.

In all three cases, assuming $P_{t'}(v) \leqslant 2 \cdot s_{t'}(v) + h_{t'}(v)$, we have $P_t(v) \leqslant 2 \cdot s_t(v) + h_t(v)$, completing the induction. ◀

3.5.2 Pur bound for square vertices

The argument of Lemma 29 does not apply well for vertices incident to multiple edges in a single step. In particular, this may happen for square vertices in M_α as each is arrived via 2 edges and each pushes out 2 edges (recall Figure 1). This is not an issue for M_β, but we will treat square vertices in M_β the same way to unify the analysis; in the context of Pur for square vertices, one may think of M_β as collapsing the two circle vertices in M_α.

To handle this issue, we observe that it suffices for us to pay an extra cost of [2] for each square vertex, which would allow us to further presume 2 edges being forced. We then generalize the prior argument to capture this change.

▶ **Lemma 30** (Bounding Pur_t for square vertices)**.** *For any time t, suppose the walker is currently at a square vertex v, then*

$$\begin{aligned}\mathsf{Pur}_t(v) &\leqslant \#(R \text{ steps closed from } v) + \#(\text{unclosed edges incident to } v \text{ at time } t) - 2 \\ &\leqslant 2(s_t(v) + h_t(v))\,.\end{aligned}$$

where $s_t(v)$ and $h_t(v)$ are the number of S and H steps arriving at v by time t, respectively.

Proof. We prove this by induction. Note that this is immediate for the base case when v first appears since a square vertex is incident to 2 edges. Define $P_t(v) := \#(\text{R steps closed from } v) + \#(\text{unclosed edges incident to } v \text{ at time } t) - 2$ for convenience. Suppose the inequality is true at time t', and assume vertex v appears again at time t. The departure at time $t' + 1$ from v may open up at most 2 edges, hence $P_{t'+1}(v) \leqslant P_{t'}(v) + 2$.

When we reach v at time t (via 2 edges), we case on the type of steps:

- Arriving by two R steps: the two edges closed by the R steps are not closed from v. So $P_t(v) = P_{t'+1} - 2 \leqslant P_{t'}(v)$, while $s_t(v) = s_{t'}(v)$ and $h_t(v) = h_{t'}(v)$.
- Arriving by one S/H and one R step: in this case, $P_t(v) = P_{t'+1}(v) \leqslant P_{t'}(v) + 2$ and $s_t(v) + h_t(v) = s_{t'}(v) + h_{t'}(v) + 1$.
- Arriving by two S/H steps: in this case, $P_t(v) = P_{t'+1}(v) + 2 \leqslant P_{t'}(v) + 4$, whereas $s_t(v) + h_t(v) = s_{t'}(v) + h_{t'}(v) + 2$.

In all three cases, we have $P_t(v) \leqslant 2(s_t(v) + h_t(v))$, completing the induction. ◀

▶ **Corollary 31.** *For each surprise/high-mul visit, it suffices for us to assign a* Pur *factor of 2, which is a cost of $(2q \cdot D_V)^2$ so that each* Pur *factor throughout the walk is assigned.*

3.6 Wrapping up with a toy example

Recall Proposition 23 that for a graph matrix of shape τ,

$$B_q(\tau) = \sum_{\mathcal{L}\text{: step-labelings for } E(\tau)} \mathsf{vtxcost}(\mathcal{L}) \cdot \mathsf{edgeval}(\mathcal{L}) \tag{9}$$

is a valid block-value function for τ (Definition 21). Moreover, by Proposition 22, we can take $q = \text{polylog}(d)$ and conclude that with probability $1 - o(1)$,

$$\|M_\tau\|_{\text{op}} \leqslant (1 + o(1)) \cdot B_q(\tau)\,.$$

For each given shape, it suffices for us to bound the block-value for each edge-labeling. We demonstrate how this may be readily done given the above bounds using the GOE example, and defer the analysis of the specific matrices that show up in our setting to the full version of the paper.

3.6.1 Tight bound for GOE

We now show how the above framework allows us to readily deduce a tight norm bound for $G \sim \text{GOE}(0, \frac{1}{d})$, where G is a $d \times d$ symmetric matrix with each (off-diagonal) entry sampled from $\mathcal{N}(0, \frac{1}{d})$. It is well-known that the correct norm of G is $2 + o_d(1)$ [14]. Figure 1a shows the shape τ associated with G, which simply consists of one edge. We now proceed to bound Eq. (9).

Edge factor. According to our edge factor scheme described in Section 3.4 (for H_1 edges), an $F/R/S$ step-label gets a factor of $\frac{1}{\sqrt{d}}$ while an H step-label gets $\frac{2q}{\sqrt{d}}$.

Pur factor. By Lemma 29, there is no Pur factor for F/R, while S and H get 2 and 1 Pur factors respectively.

Vertex factor. The weight of a circle vertex is d, thus any vertex making a first or last appearance gets a factor of $\sqrt{d}$. We now case on the step-label and apply the vertex factor assignment scheme described in Section 3.3.

- F: the vertex in U_τ must be making a middle appearance; it is not first due to Definition 24, and it is not last as otherwise the edge appears only once throughout the walk. The vertex in V_τ is making a first appearance, so it gets a factor of $\sqrt{d}$;
- R: the vertex in V_τ is making a middle appearance, since it is incident to an R edge (hence not first appearance), and it is on the boundary hence bound to appear again the next block. The vertex in U_τ may be making its last appearance, so it gets a factor of $\sqrt{d}$;
- S: the vertex in U_τ is making a middle appearance (same as F), and the vertex in V_τ is making a middle appearance since it cannot be first and must appear again. In addition, it gets 2 factors of Pur, which gives a bound of $(2q \cdot D_V)^2$;
- H: analogous to the above, both vertices are making middle appearance, and it gets 1 factor of Pur, giving a bound of $2q \cdot D_V$.

Combining the vertex and edge factors, we can bound Eq. (9):

$$B_q(\tau) = \sqrt{d} \cdot \frac{1}{\sqrt{d}} + \sqrt{d} \cdot \frac{1}{\sqrt{d}} + (2q \cdot D_V)^2 \cdot \frac{1}{\sqrt{d}} + (2q \cdot D_V) \cdot \frac{2q}{\sqrt{d}} \leqslant 2 + o_d(1)\,,$$

since q and D_V are both $\text{polylog}(d)$. Therefore, by Proposition 22, we can conclude that $\|G\|_{\text{op}} \leqslant 2 + o_d(1)$ with high probability, which is the correct bound.

References

1 Kwangjun Ahn, Dhruv Medarametla, and Aaron Potechin. Graph matrices: Norm bounds and applications. *arXiv preprint*, 2016. arXiv:1604.03423.

2 Mitali Bafna, Jun-Ting Hsieh, Pravesh K Kothari, and Jeff Xu. Polynomial-Time Power-Sum Decomposition of Polynomials. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 956–967. IEEE, 2022.

3 Boaz Barak, Samuel Hopkins, Jonathan Kelner, Pravesh K Kothari, Ankur Moitra, and Aaron Potechin. A Nearly Tight Sum-of-Squares Lower Bound for the Planted Clique Problem. *SIAM Journal on Computing*, 48(2):687–735, 2019.

4 Mrinalkanti Ghosh, Fernando Granha Jeronimo, Chris Jones, Aaron Potechin, and Goutham Rajendran. Sum-of-squares lower bounds for Sherrington-Kirkpatrick via planted affine planes. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 954–965. IEEE, 2020.

5 Christopher Hoffman, Matthew Kahle, and Elliot Paquette. Spectral gaps of random graphs and applications. *International Mathematics Research Notices*, 2019.

6 Jun-Ting Hsieh and Pravesh K Kothari. Algorithmic Thresholds for Refuting Random Polynomial Systems. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1154–1203. SIAM, 2022.

7 Chris Jones, Aaron Potechin, Goutham Rajendran, Madhur Tulsiani, and Jeff Xu. Sum-of-squares lower bounds for sparse independent set. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 406–416. IEEE, 2022.

8 Daniel M Kane and Ilias Diakonikolas. A Nearly Tight Bound for Fitting an Ellipsoid to Gaussian Random Points. *arXiv preprint*, 2022. `arXiv:2212.11221`.

9 Sidhanth Mohanty, Prasad Raghavendra, and Jeff Xu. Lifting sum-of-squares lower bounds: degree-2 to degree-4. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 840–853, 2020.

10 Aaron Potechin, Paxton Turner, Prayaag Venkat, and Alexander S Wein. Near-optimal fitting of ellipsoids to random points. *arXiv preprint*, 2022. `arXiv:2208.09493`.

11 James Saunderson. *Subspace identification via convex optimization*. PhD thesis, Massachusetts Institute of Technology, 2011.

12 James Saunderson, Venkat Chandrasekaran, Pablo A Parrilo, and Alan S Willsky. Diagonal and low-rank matrix decompositions, correlation matrices, and ellipsoid fitting. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1395–1416, 2012.

13 James Saunderson, Pablo A Parrilo, and Alan S Willsky. Diagonal and low-rank decompositions and fitting ellipsoids to random points. In *52nd IEEE Conference on Decision and Control*, pages 6031–6036. IEEE, 2013.

14 Terence Tao. *Topics in random matrix theory*, volume 132. American Mathematical Soc., 2012.

15 Max A Woodbury. Inverting modified matrices. *Memorandum Rept. 42, Statistical Research Group*, 1950.

Finding Almost Tight Witness Trees

Dylan Hyatt-Denesik
Eindhoven University of Technology, The Netherlands

Afrouz Jabal Ameli
Eindhoven University of Technology, The Netherlands

Laura Sanità
Bocconi University, Milano, Italy

Abstract

This paper addresses a graph optimization problem, called the Witness Tree problem, which seeks a spanning tree of a graph minimizing a certain non-linear objective function. This problem is of interest because it plays a crucial role in the analysis of the best approximation algorithms for two fundamental network design problems: Steiner Tree and Node-Tree Augmentation. We will show how a wiser choice of witness trees leads to an improved approximation for Node-Tree Augmentation, and for Steiner Tree in special classes of graphs.

2012 ACM Subject Classification Theory of computation → Routing and network design problems

Keywords and phrases Algorithms, Network Design, Approximation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.79

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2211.12431` [15]

Funding The second and the third authors are grateful for the support received from the NWO-VIDI grant VI.Vidi.193.087.

Acknowledgements The authors would like to thank Haris Angelidakis for his valuable discussion on this project. Furthermore, the authors would like to acknowledge the 2021 Hausdorff trimester program "Discrete Optimization", during which this work was started.

1 Introduction

Network connectivity problems play a central role in combinatorial optimization. As a general goal, one would like to design a cheap network able to satisfy some connectivity requirements among its nodes. Two of the most fundamental problems in this area are *Steiner Tree* and *Connectivity Augmentation*.

Given a network $G = (V, E)$ with edge costs, and a subset of terminals $R \subseteq V$, Steiner Tree asks to compute a minimum-cost tree T of G connecting the terminals in R. In Connectivity Augmentation, we are instead given a k-edge-connected graph $G = (V, E)$ and an additional set of edges $L \subseteq V \times V$ (called *links*). The goal is to add a minimum-cardinality subset of links to G to make it $(k + 1)$-edge-connected. It is well-known that the problem for odd k reduces to $k = 1$ (called Tree Augmentation), and for even k reduces to $k = 2$ (called Cactus Augmentation) (see [9]). All these problems are NP-hard, but admit a constant factor approximation. In the past 10 years, there have been several exciting breakthrough results in the approximation community on these fundamental problems (see [5, 13, 4, 16, 17, 6, 19, 14, 1, 7, 8, 11, 2, 18, 20]).

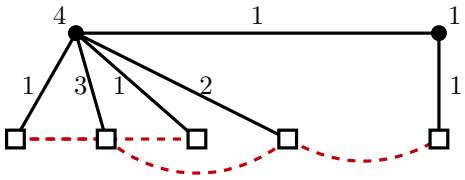


Figure 1 In black, the tree $T = (R \cup S, E)$. The dashed edges represent a witness tree W. The labels on edges of E and vertices of S indicate $\bar{w}(e)$ and $w(v)$, respectively. We have $\nu_T(W) = (H_4 + H_1)/2 = 1.541\bar{6}$. Assuming unit cost on the edges of E, we have $\bar{\nu}_T(W) = (4H_1 + H_2 + H_3)/6 = 1.\bar{2}$.

Several of these works highlight a deep relation between Steiner Tree and Connectivity Augmentation: the approximation techniques used for Steiner Tree have been proven to be useful for Connectivity Augmentation and vice versa. This fruitful exchange of tools and ideas has often lead to novel results and analyses. This paper continues bringing new ingredients in this active and evolving line of work.

Specifically, we focus on a graph optimization problem which plays a crucial role in the analysis of some approximation results mentioned before. This problem, both in its edge- and node-variant, is centered around the concept of *witness trees*. We now define this formally (see Figure 1 for an example).

Edge Witness Tree (EWT) problem. Given is a tree $T = (V, E)$ with edge costs $c : E \to \mathbb{R}_{\geq 0}$. We denote by R the set of leaves of T. The goal is to find a tree $W = (R, E_W)$, where $E_W \subseteq R \times R$, which minimizes the non-linear objective function $\bar{\nu}_T(W) = \frac{1}{c(E)} \sum_{e \in E} c(e) H_{\bar{w}(e)}$, where $c(E) = \sum_{e \in E} c(e)$, the function $\bar{w} : E \to \mathbb{Z}_{\geq 0}$ is defined as

$$\bar{w}(e) \coloneqq |\{pq \in E_W : e \text{ is an internal edge of the } p\text{-}q \text{ path in } T\}|$$

and H_ℓ denotes the ℓ^{th} harmonic number ($H_\ell = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{\ell}$).

Node Witness Tree (NWT) problem. Given is a tree $T = (V, E)$. We denote by R the set of leaves of T, and $S = V \setminus R$. The goal is to find a tree $W = (R, E_W)$, where $E_W \subseteq R \times R$, which minimizes the non-linear objective function $\nu_T(W) = \frac{1}{|S|} \sum_{v \in S} H_{w(v)}$, where $w : S \to \mathbb{Z}_{\geq 0}$ is defined as

$$w(v) \coloneqq |\{pq \in E_W : v \text{ is an internal node of the } p\text{-}q \text{ path in } T\}|$$

and again H_ℓ denotes the ℓ^{th} harmonic number.

We refer to a feasible solution W to either of the above problems as a *witness tree*. We call $\bar{w}$ (resp. w) the *vector imposed on* E (resp. S) by W. We now explain how these problems relate to Steiner Tree and Connectivity Augmentation.

EWT and relation to Steiner Tree

Currently, the best approximation factor for Steiner Tree is $(\ln(4) + \varepsilon)$, which can be achieved by three different algorithms [13] [5] [20]. These algorithms yield the same approximation because in all three of them, the analysis at some point relies on constructing witness trees.

More in detail, suppose we are given a Steiner Tree instance $(G = (V, E), R, c)$ where $c : E \to \mathbb{R}_{\geq 0}$ gives the edge costs. We can define the following:

$$\gamma_{(G,R,c)} \coloneqq \min_{\substack{T^* = (R \cup S^*, E^*):\ T^* \text{ is} \\ \text{optimal Steiner tree of } (G,R,c)}} \ \min_{\substack{W:\ W \text{ is a} \\ \text{witness tree} \\ \text{of } T^*}} \bar{\nu}_{T^*}(W)$$

We also define the following constant γ:

$$\gamma \coloneqq \sup\{\gamma_{(G,R,c)} : (G,R,c) \text{ is an instance of Steiner Tree}\}.$$

Byrka et al. [5] were the first to essentially prove the following.

▶ **Theorem 1.** *For any $\varepsilon > 0$, there is a $(\gamma+\varepsilon)$-approximation algorithm for Steiner Tree.*

Furthermore, the authors in [5] showed that $\gamma \leq \ln(4)$, and hence they obtained the previously mentioned $(\ln(4)+\varepsilon)$-approximation for Steiner Tree.

NWT and relation to Connectivity Augmentation

Basavaraju et al [3] introduced an approximation-preserving reduction from Cactus Augmentation (which is the hardest case of Connectivity Augmentation)[1] to special instances of Node-Steiner Tree, named *CA-Node-Steiner-Tree* instances in [2]: the goal here is to connect a given set R of terminals of a graph G via a tree that minimizes the number of non-terminal nodes (Steiner nodes) in it. The special instances have the crucial property that each Steiner node is adjacent to at most 2 terminals.

Byrka et al. [4] built upon this reduction to prove a 1.91-approximation for CA-Node-Steiner-Tree instances. This way, they were the first to obtain a better-than-2 approximation factor for Cactus Augmentation (and hence, for Connectivity Augmentation). Interestingly, Nutov [16] realized that a similar reduction also captures a fundamental node-connectivity augmentation problem: the *Node-Tree Augmentation* (defined exactly like Tree Augmentation, but replacing edge-connectivity with node-connectivity). This way, he could improve over an easy 2-approximation for Node-Tree Augmentation that was also standing for 40 years [12]. Angelidakis et al. [2] subsequently explicitly formalized the problem at the heart of the approximation analysis: namely, the NWT problem.

More in detail, given a CA-Node-Steiner-Tree instance $(G=(V,E),R)$, we can define the following:

$$\psi_{(G,R)} \coloneqq \min_{\substack{T^*=(R\cup S^*,E^*)\text{: } T^* \text{ is} \\ \text{optimal Steiner tree of } (G,R)}} \; \min_{\substack{W\text{: } W \text{ is a} \\ \text{witness tree} \\ \text{of } T^*}} \nu_{T^*}(W),$$

We also define the constant ψ:

$$\psi \coloneqq \sup\{\psi_{(G,R)} : (G,R) \text{ is an instance of CA-Node-Steiner-Tree}\}.$$

Angelidakis et al. [2] proved the following.

▶ **Theorem 2.** *For any $\varepsilon > 0$, there is a $(\psi+\varepsilon)$-approximation algorithm for CA-Node-Steiner Tree.*

Furthermore, the authors of [2] proved that $\psi < 1.892$, and hence obtained a 1.892-approximation algorithm for Cactus Augmentation and Node-Tree Augmentation. This is currently the best approximation factor known for Node-Tree Augmentation (for Cactus Augmentation there is a better algorithm [6]).

[1] Tree Augmentation can be easily reduced to Cactus Augmentation by introducing a parallel copy of each initial edge.

Our results and techniques

Our main result is an improved upper bound on ψ. In particular, we are able to show $\psi < 1.8596$. Combining this with Theorem 2, we obtain a 1.8596-approximation algorithm for CA-Node-Steiner-Tree. Hence, due to the above mentioned reduction, we improve the state-of-the-art approximation for Node-Tree Augmentation.

▶ **Theorem 3.** *There is a 1.8596-approximation algorithm for CA-Node-Steiner-Tree (and hence, for Node-Tree Augmentation).*

Our result is based on a better construction of witness trees for the NWT problem. At a very high level, the witness tree constructions used previously in the literature use a *marking-and-contraction* approach, that can be summarized as follows. First, root the given tree T at some internal Steiner node. Then, every Steiner node v chooses (*marks*) an edge which connects to one of its children: this identifies a path from v to a terminal. Contracting the edges along this path yields a witness tree W. The way this marking choice is made varies: it is random in [5], it is biased depending on the nature of the children in [4], it is deterministic and taking into account the structure of T in [2]. However, all such constructions share the fact that decisions can be thought of as being taken "in one shot", at the same time for all Steiner nodes. Instead, here we consider a bottom-up approach for the construction of our witness tree, where a node takes a marking decision only after the decisions of its children have been made. A sequential approach of this kind allows a node to have a more precise estimate on the impact of its own decision to the overall non-linear objective function cost, but it becomes more challenging to analyze. Overcoming this challenge is the main technical contribution of this work, and the insight behind our improved upper-bound on ψ.

We complement this result with an almost-tight lower-bound on ψ, which improves over a previous lower bound given in [2].

▶ **Theorem 4.** *For any $\varepsilon > 0$, there exists a CA-Node-Steiner-Tree instance $(G_\varepsilon, R_\varepsilon)$ such that $\psi_{(G_\varepsilon,R_\varepsilon)} > 1.841\bar{6} - \varepsilon$.*

The above theorem implies that, in order to significantly improve the approximation for Node-Tree Augmentation, very different techniques need to be used. To show our lower-bound we prove a structural property on optimal witness trees, called *laminarity*, which in fact holds for optimal solutions of both the NWT problem and the EWT problem.

As an additional result, we also improve the approximation bound for Steiner Tree in the special case of *Steiner-claw free* instances. A Steiner-Claw Free instance is a Steiner-Tree instance where the subgraph $G[V \setminus R]$ induced by the Steiner nodes is claw-free (i.e., every node has degree at most 2). These instances were introduced in [10] in the context of studying the integrality gap of a famous LP relaxation for Steiner Tree, called the *bidirected cut relaxation*, that is long-conjectured to have integrality gap strictly smaller than 2.

▶ **Theorem 5.** *There is a $(\frac{991}{732} + \varepsilon < 1.354)$-approximation for Steiner Tree on Steiner-claw free instances.*

We prove the theorem by showing that, for any Steiner-Claw Free instance (G, R, c), $\gamma_{(G,R,c)} \leq \frac{991}{732}$. The observation we use here is that an optimal Steiner Tree solution T in this case is the union of components that are caterpillar graphs[2]: this knowledge can be

[2] A caterpillar graph is defined as a tree in which every leaf is of distance 1 from a central path.

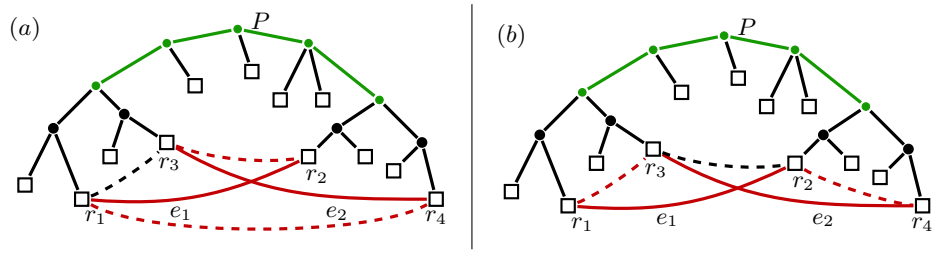


Figure 2 In both figures we have a tree, T, shown with black edges and green edges, with leaves, R, denoted by squares. Crossing edges e_1 and e_2 are shown with solid red edges. The green edges denote the path P. Figure (a): In this case, r_1 and r_3 are in the same component of $W \backslash \{e_1, e_2\}$, represented by the dashed black edge. We can replace e_1 with r_2r_3 or replace e_2 with r_1r_4 (red dashed edges). Figure (b): In this case, r_3 and r_2 are in the same component, denoted by the black dashed edge. We can replace e_1 and e_2 with r_1r_3 and r_2r_4 (red dashed edges).

exploited to design ad-hoc witness trees. Interestingly, we can also show that this bound is tight: once again, the proof of this lower-bound result relies on showing laminarity for optimal witness trees.

▶ **Theorem 6.** *For any $\varepsilon > 0$, there exists Steiner-Claw Free instance $(G_\varepsilon, R_\varepsilon, c_\varepsilon)$ such that $\gamma_{(G_\varepsilon, R_\varepsilon, c_\varepsilon)} > \frac{991}{732} - \varepsilon$.*

As a corollary of our results, we also get an improved bound on the integrality gap of the bidirected cut relaxation for Steiner-Claw Free instances (this follows directly from combining our upper bound with the results in [10]). Though these instances are quite specialized, they serve the purpose of passing the message: exploiting the structure of optimal solutions helps in choosing better witnesses, hopefully arriving at tight (upper and lower) bounds on γ and ψ.

2 Laminarity

In this section, we prove some key structural properties of witness trees. We assume to be given a Node (Edge) Witness Tree instance $T = (V, E)$ with leaves R (and edge costs $c : E \to \mathbb{R}_{\geq 0}$), where R denotes the leaves of T, we will show that we can characterize witness trees minimizing $\nu_T(W)$ ($\bar{\nu}_T(W)$) using the following notion of *laminarity*. Given a witness tree $W = (R, E_W)$, we say edges $f_1f_2, f_3f_4 \in E_W$ *cross* if the f_1-f_2 and f_3-f_4 paths in T share an internal node but not an endpoint. We say that W is *laminar* if it has no crossing edges. For nodes $u, v \in V$, we denote by T_{uv} the path in T between the nodes u and v. Similarly, for $e \in E_W$, we denote by T_e the path in T between the endpoints of e.

The following Theorem shows that there is always a witness tree minimizing $\nu_T(W)$ that is laminar.

▶ **Theorem 7.** *Given an instance of the Node Witness Tree problem $T = (V, E)$, let $\mathcal{W}$ be the family of all witness trees for T. Then there exists a laminar witness tree W such that $\nu_T(W) = \min_{W' \in \mathcal{W}} \nu_T(W')$.*

Proof. We first show that there is a witness tree W minimizing $\nu_T(W)$ such that the induced subgraph of W on any maximal set of terminals that share a neighbour in $V \backslash R$ is a star. We assume for the sake of contradiction that there is a maximal set of terminals $S \subseteq R$ sharing a neighbour $v \in V \backslash R$, such that the induced subgraph of W on S is a set of connected components $W_1, \ldots, W_i$ for $i > 1$. Without loss of generality, suppose the shortest path

between two components is from W_1 to W_2, and let e denote the edge of this path incident to W_2. We define $W' \coloneqq W \cup \{f\} \backslash \{e\}$, where f is an arbitrary edge between W_1 and W_2. Since $\{v\} = T_f \backslash R \subsetneq T_e \backslash R$, we have $\nu_T(W') < \nu_T(W)$, contradicting the minimality of W. Therefore, the induced subgraph on S is connected. We can rearrange the edges of this subgraph to be a star as this will not affect $\nu_T(W)$, so we assume this holds on W for any such S.

For a maximal set of terminals $S \subseteq R$ that share a neighbour, by a slight abuse of notation, we denote by S the induced star subgraph of W on S, and denote its center by $s \in S$. We will assume without loss of generality that edges of W incident to S have endpoint s. To see this, as S is a connected subgraph of W, any pair of edges incident to S cannot share an endpoint outside of S, otherwise we have found a cycle in W. Furthermore, for any edge of W incident to S where s is not an endpoint, we can change the endpoint in S of that edge to be s and maintain the connectivity of W since S is connected. Edges changed in this way will have the same interior nodes between their endpoints, so this does not increase $\nu_T(W)$.

We assume for the sake of contradiction that the witness tree W minimizing $\nu_T(W)$ is not a laminar witness tree. As W is not laminar, there exist distinct leaves $r_1, r_2, r_3, r_4 \in R$ such that $e_1 = r_1r_2, e_2 = r_3r_4 \in E_W$ are crossing. We denote the path $T_{e_1} \cap T_{e_2}$ by P. We denote by P_i the (potentially empty) set of internal nodes of the shortest path from P to r_i in T.

Since e_1 and e_2 are crossing edges, one of $T_{r_1r_3}$ or $T_{r_1r_4}$ contains exactly one node of P. The same is true for r_2. Without loss of generality, let us assume that the paths $T_{r_1r_3}$ and $T_{r_2r_4}$ contain exactly one node of P. We consider by cases which component of $W \backslash \{e_1, e_2\}$ contains two nodes among r_1, r_2, r_3 and r_4. See Figure 2 for an example.

- Case: r_1 and r_3 (or similarly, r_2 and r_4) are in the same component of $W \backslash \{e_1, e_2\}$. If $P_1 = P_3 = \emptyset$, then r_1 and r_3 share a neighbour and thus, as shown above, e_1 and e_2 are assumed to share an endpoint, and are thus not crossing.
  Consider $W' \coloneqq W \cup \{r_2r_3\} \backslash \{e_1\}$ and $W'' \coloneqq W \cup \{r_1r_4\} \backslash \{e_2\}$. If $\nu_T(W) - \nu_T(W') > 0$, this contradicts the minimality of $\nu_T(W)$. Therefore, we can see

$$0 \leq |V \backslash R|(\nu_T(W') - \nu_T(W)) = \sum_{u \in P_3} \frac{1}{w(u)+1} - \sum_{u \in P_1} \frac{1}{w(u)}$$
$$< \sum_{u \in P_3} \frac{1}{w(u)} - \sum_{u \in P_1} \frac{1}{w(u)+1} = |V \backslash R|(\nu_T(W) - \nu_T(W''))$$

  Clearly, we have $\nu_T(W'') < \nu_T(W)$, contradicting minimality of $\nu_T(W)$.
- Case: r_2 and r_3 (or similarly, r_1 and r_4) are in the same component of $W \backslash \{e_1, e_2\}$. Without loss of generality we can assume that $|V(P)| > 1$, because if $|V(P)| = 1$ then we can reduce to the previous case by relabelling the nodes r_1, r_2, r_3 and r_4. In this case, consider $W' \coloneqq W \cup \{r_1r_3, r_2r_4\} \setminus \{e_1, e_2\}$. Therefore, we can see

$$|V \backslash R| \left(\nu_T(W') - \nu_T(W)\right) \leq -\sum_{u \in P} \frac{1}{w(u)} < 0$$

Thus, we have $\nu_T(W') < \nu_T(W)$, contradicting the minimality of $\nu_T(W)$. ◀

The following theorem, similar to Theorem 7, shows that there are laminar witness trees that are optimal for the EWT problem. The proof is deferred to the full version of the paper.

▶ **Theorem 8.** *Given an instance of the Edge Witness Tree problem $T = (V, E)$ with edge costs c, let $\mathcal{W}$ be the family of all witness trees for T. Then there exists a laminar witness tree W such that $\bar{\nu}_T(W) = \min_{W' \in \mathcal{W}} \bar{\nu}_T(W')$.*

We now show that laminar witness trees are precisely the set of trees that one could obtain with a marking-and-contraction approach. The proof of this Theorem can be found in the full version of the paper.

▶ **Theorem 9.** *Given a tree $T = (V, E)$ with leaves R, a witness tree $W = (R, E_W)$ for T can be found by marking-and-contraction if and only if W is laminar.*

Incidentally, this has the following side implication. The authors of [13] gave a dynamic program (that is also a bottom-up approach) to compute the best possible witness tree obtainable with a marking-and-contraction scheme. Our structural results imply that their dynamic program computes an optimal solution for the EWT problem (though for the purpose of the approximation analysis, being able to compute the best witness tree is not that relevant: being able to bound ψ and γ is what matters).

3 Improved approximation for CA-Node-Steiner Tree

The goal of this section is to prove Theorem 3. We will achieve this by showing $\psi < 1.8596$, and by using Theorem 2. From now on, we assume we are given a tree $T = (R \cup S^*, E^*)$, where each Steiner node is adjacent to at most two terminals.

3.1 Preprocessing

We first apply some preprocessing operations as in [2], that allow us to simplify our witness tree construction. The first one is to remove the terminals from T, and then decompose T into smaller components which will be held separately. We start by defining a *final* Steiner node as a Steiner node that is adjacent to at least one terminal. We let $F \subseteq S^*$ denote the set of final Steiner nodes. Since we remove the terminals from T, we will construct a spanning tree W on F with edges in $F \times F$. With a slight abuse of notation, we refer to W as a witness tree: this is because [2, Section 4.1] showed that one can easily map W to a witness tree for our initial tree T (with terminals put back), and the following can be considered the vector imposed on S^* by W:

$$w(v) \coloneqq |\{pq \in E_W : v \text{ belongs to the } p\text{-}q \text{ path in } T[S^*]\}| + \mathbb{1}[v \in F] \tag{1}$$

where $\mathbb{1}[v \in F]$ denotes the indicator of the event "$v \in F$", and $T[S^*]$ is the subtree of T induced by the Steiner nodes. See Figure 3.

So, from now on, we consider $T = T[S^*]$. The next step is to root T at an arbitrary final node $r \in F$. Following [2] we can decompose T into a collection of rooted components $T_1, \dots T_\tau$, where a component is a subtree whose leaves are final nodes and non-leaves are non-final nodes. The decomposition will have the following properties: each T_i is rooted at a final node r_i that has degree one in T_i, $r_1 \coloneqq r$ is the root of T_1, $\cup_{j<i} T_j$ is connected, and $T = \cup_{i=1}^{\tau} T_i$. We will compute a witness tree W_i for each component T_i, and then show that we can join these witness trees $\{W_i\}_{i \geq 1}$ together to get a witness tree W for T.

3.2 Computing a witness tree W_i for a component T_i

Here we deal with a component T_i rooted at r_i, and describe how to construct a witness tree W_i. If T_i is a single edge $e = r_i v$, we simply let $W_i = (\{r_i, v\}, \{r_i v\})$.

Now we assume that T_i is not a single edge. We will construct a witness tree with a bottom-up procedure. At a high level, each node $u \in T_i \backslash r_i$ looks at the subtree Q_u of T_i rooted at u, and constructs a portion of the witness tree: namely, a subtree $\overline{W}^u$ spanning

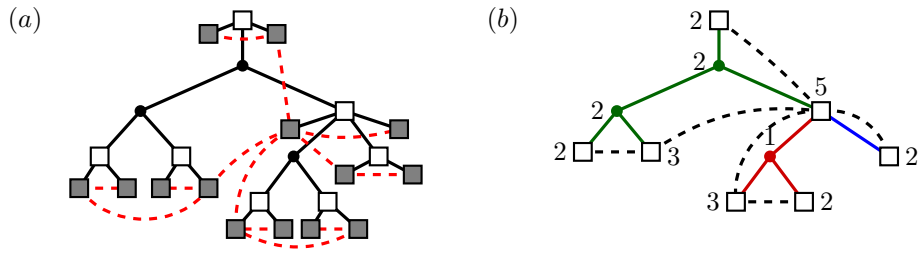


Figure 3 Figure (a): A tree T is shown by black edges. The terminals are shown by grey squares. The final Steiner nodes are shown by white squares, non-final Steiner nodes are shown by black dots. Figure (b): The tree T after the terminals have been removed. The color edges indicate the three components. A witness tree W is shown by the black dashed lines. The numbers indicate the values of w imposed on T computed according to (1). Red dashed lines in Figure (a) show how W can be mapped back.

the leaves of Q_u (note that, in case the degree of u is 1 in Q_u, we do not consider u to be a leaf of Q_u but just its root). Assume u has children $u_1, \ldots, u_k$. Because of the bottom-up procedure, each child u_j has already constructed a subtree $\overline{W}^{u_j}$. That is, u has to decide how to join these subtrees to get $\overline{W}^u$.

To describe how this is done formally, we first need to introduce some more notation. For every node $u \in T_i \backslash F$, we select one of its children as the "marked child" of u (according to some rule that we will define later). In this way, for every $u \in T_i$ there is a unique path along these marked children to a leaf. We denote this path by $P(u)$, and we let $\ell(u)$ denote the leaf descendent of this path. For final nodes $u \in F$, we define $\ell(u) := u$ and $P(u) := u$. For a subtree Q_u of T_i rooted at u and a witness tree $\overline{W}^u$ over the leaves of Q_u, let $\overline{w}^u$ be the vector imposed on the nodes of Q_u by $\overline{W}^u$ according to (1). Next, we define the following quantity (which, roughly speaking, represents the cost-increase incurred after increasing $\overline{w}^u(v)$ for each $v \in P(u)\backslash\ell(u)$ for the $(j+1)^{th}$ time):

$$C_j^u := \sum_{v \in P(u)\backslash\ell(u)} \left(H_{\overline{w}^u(v)+j+1} - H_{\overline{w}^u(v)+j}\right) = \sum_{v \in P(u)\backslash\ell(u)} \frac{1}{\overline{w}^u(v)+j+1}$$

Algorithm 1 Computing the tree $\overline{W}^u$.

u has Steiner node children $u_1, u_2, \ldots, u_k$, and $\overline{W}^{u_j}$ have been defined
if *$u_1, \ldots, u_k$ are all non-final,* **then**
  The *marked* child is u_m, minimizing $C_1^{u_m}$
else
  Assume $\{u_1, \ldots, u_{k_1}\}$, $1 \le k_1 \le k$, are final node children of u
  if *$k_1 = k$, or, for all $j \in \{k_1+1, \ldots, k\}$, $C_1^{u_j} \ge \phi - \delta - H_2$* **then**
    The *marked* child of u is u_m for $1 \le m \le k_1$ such that $C_1^{u_m}$ is minimized.
  if *There is a $j \in \{k_1+1, \ldots, k\}$ such that $C_1^{u_j} < \phi - \delta - H_2$* **then**
    The *marked* child of u is u_m for $k_1 < m \le k$ such that $C_1^{u_m}$ is minimized.
$\overline{W}^u \leftarrow \left(\bigcup_{j=1}^k V[Q_{u_j}], \bigcup_{j=1}^k \overline{W}^{u_j} \bigcup_{j \ne m} \{\ell(u_m)\ell(u_j)\}\right)$
Return $\overline{W}^u$

We can now describe the construction of the witness tree more formally. We begin by considering the leaves of T_i; for a final node (leaf) u, we define a witness tree on the (single) leaf of Q_u as $\overline{W}^u = (\{u\}, \emptyset)$. For a non-final node u, with children $u_1, \ldots, u_k$ and

corresponding witness trees $\overline{W}^{u_1}, \dots, \overline{W}^{u_k}$, we select a marked child u_m for u as outlined in Algorithm 1, setting $\phi = 1.86 - \frac{1}{2100}$ and $\delta = \frac{97}{420}$. With this choice, we compute $\overline{W}^u$ by joining the subtrees $\overline{W}^{u_1}, \dots, \overline{W}^{u_k}$ via the edges $\ell(u_m)\ell(u_j)$ for $j \neq m$. Finally, let v be the unique child of r_i. We let W_i be equal to the tree $\overline{W}^v$ plus the extra edge $\ell(v)r_i$, to account for the fact that r_i is also a final node.

3.3 Bounding the cost of W_i

It will be convenient to introduce the following definitions. For a component T_i and a node $u \in T_i \setminus r_i$, we let W^u be the tree $\overline{W}^u$ plus one extra edge e^u, defined as follows. Let $a(u)$ be the first ancestor node of u with $\ell(a(u)) \neq \ell(u)$ (recall $\ell(r_i) = r_i$). We then let the edge $e^u \coloneqq \ell(u)\ell(a(u))$. We denote by w^u the vector imposed on the nodes of Q_u by $W^u \coloneqq \overline{W}^u + e^u$. Note that, with this definition, $W_i = W^v$ for v being the unique child of r_i.

We now state two useful lemmas. The first one relates the functions w^u and w^{u_j} for a child u_j of u. The statements (a)-(c) below can be proved similarly to Lemma 4 of [2]. We defer its proof to the full version of the paper.

▶ **Lemma 10.** *Let $u \in T_i \setminus r_i$ have children $u_1, \dots, u_k$, and u_1 be its marked child. Then:*

a $w^u(u) = k$.

b *For every $j \in \{2, \dots, k\}$ and every node $v \in Q_{u_j}$, $w^u(v) = w^{u_j}(v)$.*

c *For every $v \in Q_{u_1} \setminus P(u_1)$, $w^u(v) = w^{u_1}(v)$.*

d $\sum_{v \in P(u_1) \setminus \ell(u_1)} H_{w^u(v)} = \sum_{v \in P(u_1) \setminus \ell(u_1)} H_{w^{u_1}(v)} + \sum_{j=1}^{k-1} C_j^{u_1}$.

Next lemma relates the "increase" of cost C_j^u to the *degree* of some nodes in T_i.

▶ **Lemma 11.** *Let $u \in T_i \setminus r_i$ have children $u_1, \dots, u_k$, and u_1 be its marked child. Then, $C_1^u = C_k^{u_1} + \frac{1}{k+1}$. Furthermore, if u_1 is non-final and has degree d in T_i, then:*
1) $\sum_{j=1}^{k}(C_j^{u_1} - C_1^{u_j}) \leq \sum_{j=1}^{k-1}\left(\frac{1}{d+j} - \frac{1}{d}\right)$; 2) $H_{w^u(\ell(u_1))} - H_{w^{u_1}(\ell(u_1))} \leq \sum_{j=1}^{k-1} \frac{1}{d+j}$

Proof.

1. First observe that since $C_1^{u_1} = \min_{j \in [k]} C_1^{u_j}$, we have $C_j^{u_1} - C_1^{u_j} \leq C_j^{u_1} - C_1^{u_1}$. Consider $j \geq 1$, $C_j^{u_1} - C_1^{u_1}$ is equal to

$$= \sum_{v \in P(u_1) \setminus \ell(u)} \left(H_{w^{u_1}(v)+j} - H_{w^{u_1}(v)+j-1} - H_{w^{u_1}(v)+1} + H_{w^{u_1}(v)}\right)$$

$$= \sum_{v \in P(u_1) \setminus \ell(u)} \left(\frac{1}{w^{u_1}(v)+j} - \frac{1}{w^{u_1}(v)+1}\right) \leq \frac{1}{w^{u_1}(u_1)+j} - \frac{1}{w^{u_1}(u_1)+1}$$

   Where the inequality follows since every term in the sum is negative. We know that $w^{u_1}(u_1) = d - 1$ by Lemma 10.(a), therefore, $C_j^{u_1} - C_1^{u_1} \leq \frac{1}{d+j-1} - \frac{1}{d}$, and the claim is proven by summing over $j = 1, \dots, k$.

2. To prove the second inequality, first observe that $w^u(\ell(u_1)) = w^{u_1}(\ell(u_1)) + k - 1$. This follows by recalling that W^u is equal to $\overline{W}^{u_1}, \dots, \overline{W}^{u_k}$ plus the edges $\ell(u_1)\ell(u_j)$ for $j \neq 1$, and e^u. Thus, $H_{w^u(\ell(u_1))} - H_{w^{u_1}(\ell(u_1))} = H_{w^{u_1}(\ell(u_1))+k-1} - H_{w^{u_1}(\ell(u_1))} = \sum_{i=1}^{k-1} \frac{1}{w^{u_1}(\ell(u_1))+i}$. Recall u_1 is not a final node, so $w^{u_1}(\ell(u_1)) > d$. Therefore,

$$\sum_{i=1}^{k-1} \frac{1}{w^{u_1}(\ell(u_1))+i} \leq \sum_{i=1}^{k-1} \frac{1}{d+i}.$$

◀

3.4 Key Lemma

To simplify our analysis, we define $h_{W^u}(Q_u) \coloneqq \sum_{\ell \in Q_u} H_{w^u(\ell)}$, and we let $|Q_u|$ be the number of nodes in Q_u. The next lemma is the key ingredient to prove Theorem 3.

▶ **Lemma 12.** *Let $\delta = \frac{97}{420}$ and $\phi = 1.86 - \frac{1}{2100}$. Let $u \in T_i \setminus r_i$ and k be the number of its children. Let $\beta(k)$ be equal to 0 for $k = 0, \ldots, 8$ and $\frac{1}{3} - \delta$ for $k \geq 9$. Then*

$$h_{W^u}(Q_u) + C_1^u + \delta + \beta(k) \leq \phi \cdot |Q_u|$$

Proof. The proof of Lemma 12 will be by induction on $|Q_u|$. The base case is when $|Q_u| = 1$, and hence u is a leaf of T_i. Therefore, W^u is just the edge e^u, and so by definition of w^u we have $w^u(u) = 2$. We get $h_{W^u}(Q_u) = 1.5$, $C_1^u = 0$, $\beta(k) = 0$ and the claim is clear.

For the induction step: suppose that u has children $u_1, \ldots, u_k$. We will distinguish 2 cases: (i) u has no children that are final nodes; (ii) u has some child that is a final node (which is then again broken into subcases). We report here only the proof of case (i), and defer the proof of the other case to the full version of the paper as the reasoning follows similar arguments.

Case (i): No children of u are final

According to Algorithm 1, we mark the child u_m of u that minimizes $C_1^{u_j}$. Without loss of generality, let $u_m = u_1$. Furthermore, let $\ell \coloneqq \ell(u_1)$. We note the following.

$$h_{W^u}(Q_u) = \sum_{j=1}^{k} h_{W^u}(Q_{u_j}) + H_{w^u(u)}$$

By applying Lemma 10.(a) we have $H_{w^u(u)} = H_k$. By Lemma 10.(b) we see $h_{W^u}(Q_{u_j}) = h_{W^{u_j}}(Q_{u_j})$ for $j \geq 2$. Using Lemma 10.(c) and (d) we get $h_{W^u}(Q_{u_1}) = h_{W^{u_1}}(Q_{u_1}) + \sum_{j=1}^{k-1} C_j^{u_1} + H_{w^u(\ell)} - H_{w^{u_1}(\ell)}$. Therefore:

$$h_{W^u}(Q_u) = \sum_{j=1}^{k} h_{W^{u_j}}(Q_{u_j}) + \sum_{j=1}^{k-1} C_j^{u_1} + H_k + H_{w^u(\ell)} - H_{w^{u_1}(\ell)}$$

We apply our inductive hypothesis on $Q_{u_1}, \ldots, Q_{u_k}$, and use $\beta(j) \geq 0$ for all j:

$$\begin{aligned} h_{W^u}(Q_u) &\leq \sum_{j=1}^{k} \left(\phi|Q_{u_j}| - \delta - C_1^{u_j}\right) + \sum_{j=1}^{k-1} C_j^{u_1} + H_k + H_{w^u(\ell)} - H_{w^{u_1}(\ell)} \\ &= \phi(|Q_u| - 1) - k\delta - C_k^{u_1} + \sum_{j=1}^{k} \left(C_j^{u_1} - C_1^{u_j}\right) + H_k + H_{w^u(\ell)} - H_{w^{u_1}(\ell)} \end{aligned}$$

Using Lemma 11, we get

$$\begin{aligned} &\leq \phi(|Q_u| - 1) - k\delta - C_1^u + \sum_{j=1}^{k-1} \left(\frac{1}{d+j} - \frac{1}{d}\right) + H_{k+1} + \sum_{j=1}^{k-1} \frac{1}{d+j} \\ &\leq \phi|Q_u| - \delta - C_1^u - \beta(k) \end{aligned}$$

where the last inequality follows since one checks that for any $k \geq 1$ and $d \geq 2$ we have $-\phi - (k-1)\delta + \sum_{j=1}^{k-1} \left(\frac{1}{d+j} - \frac{1}{d}\right) + H_{k+1} + \sum_{j=1}^{k-1} \frac{1}{d+j} \leq -\beta(k)$. We show this inequality the full version of the paper. ◀

3.5 Merging and bounding the cost of W

Once the $\{W_i\}_{i\geq 1}$ are computed for each component T_i, we let the final witness tree be simply the union $W = \cup_i W_i$. Our goal now is to prove the following.

▶ **Lemma 13.** $\nu_T(W) \leq \phi = 1.86 - \frac{1}{2100}$.

Proof. Recall that we decomposed T into components $\{T_i\}_{i=1}^{\tau}$, such that $\cup_{j\leq i} T_j$ is connected for all $i \in [\tau]$. For a given i, define $T' = \cup_{j<i} T_j$, $W' = \cup_{j<i} W_i$, and let w' be the vector imposed on the nodes of T' by W' (for $i = 1$, set $T' = \emptyset$, $W' = \emptyset$, and $w' = 0$). Finally, define $W'' = W_i \cup W'$ and let w'' be the vector imposed on the nodes of $T'' := T' \cup T_i$. By induction on i, we will show that $\nu_{T''}(W'') \leq \phi$. The statement will then follow by taking $i = \tau$. Recall that, for any i, r_i is adjacent to a single node v in T_i, and $W_i = W^v$.

First consider $i = 1$. Hence, $W'' = W_1 = W^v$ and $w''(r_1) = 2$. By applying Lemma 12 to the subtree Q_v we get

$$\sum_{u\in T''} H_{w''(u)} = h_{W^v}(Q_v) + H_{w''(r_i)} \leq \phi(|Q_v|) + H_2 \leq \phi(|Q_v| + 1) \Rightarrow \nu_{T''}(W'') \leq \phi$$

Now consider $i > 1$. In this case, $w''(r_i) = w'(r_i) + 1 \geq 3$. Therefore:

$$\sum_{u\in T''} H_{w''(u)} = \sum_{u\in T_i\setminus r_i} H_{w^v(u)} + \sum_{u\in T'} H_{w'(u)} - H_{w'(r_i)} + H_{w'(r_i)+1}$$
$$= \sum_{u\in T_i\setminus r_i} H_{w^v(u)} + \sum_{u\in T'} H_{w'(u)} + \frac{1}{w'(r_i)+1} \leq \sum_{u\in T_i\setminus r_i} H_{w^v(u)} + \sum_{u\in T'} H_{w'(u)} + \frac{1}{3}$$

If v is a final node, then $\sum_{u\in T_i\setminus r_i} H_{w^v(u)} = H_{w^v(v)} = H_2$ and by induction

$$\sum_{u\in T''} H_{w''(u)} \leq H_3 + \sum_{u\in T'} H_{w'(u)} \leq \phi|T''| \Rightarrow \nu_{T''}(W'') \leq \phi$$

If v is not a final node, then by induction on T' and by applying Lemma 12 to the subtree Q_v, assuming that v has k children, we can see

$$\sum_{u\in T''} H_{w''(u)} \leq \phi|T''| - C_1^v - \delta - \beta(k) + \frac{1}{3} \leq \phi|T''| - \frac{1}{k+1} - \delta - \beta(k) + \frac{1}{3}$$

If $1 \leq k \leq 8$, then $\beta(k) = 0$, but we have $\frac{1}{3} < 431/1260 = \frac{1}{9} + \delta \leq \frac{1}{k+1} + \delta$. If $k \geq 9$, $\beta(k) = \frac{1}{3} - \delta$ and $\frac{1}{3} - \delta - \beta(k) = 0$. In both cases, $\nu_{T''}(W'') \leq \phi$. ◀

Note that we did not make any assumption on T, other than being a CA-Node-Steiner-Tree. Hence, Lemma 13 yields the following corollary.

▶ **Corollary 14.** $\psi \leq 1.86 - \frac{1}{2100} < 1.8596$.

Combining Corollary 14 with Theorem 2 yields a proof of Theorem 3.

4 Improved Lower Bound on ψ

The goal of this section is to prove Theorem 4. For the sake of brevity, we will omit several details. (see the full version of the paper for a completed proof).

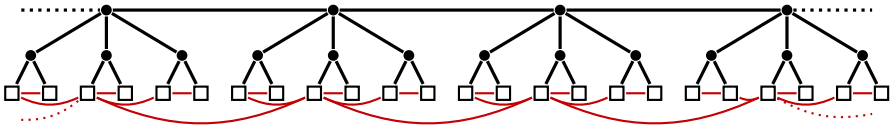


Figure 4 Lower bound instance shown in black. The white squares are terminals and black circles are Steiner nodes. Red edges form the laminar witness tree W^*.

Sketch of Proof of Theorem 4

Consider a CA-Node-Steiner-Tree instance (G, R), where G consists of a path of Steiner nodes $s_1, \ldots, s_q$ such that, for all $i \in [q]$, s_i is adjacent to Steiner nodes t_{i1}, t_{i2}, t_{i3}, and each t_{ij} is adjacent to two terminals r^1_{ij} and r^2_{ij}. See Figure 4. We will refer to B_i as the subgraph induced by $s_i, t_{ij}, r^1_{ij}, r^2_{ij}$ $(j = 1, 2, 3)$. Since G is a tree connecting the terminals, clearly the optimal Steiner tree for this instance is $T = G$.

Let W^* be a witness tree that minimizes $\nu_T(W^*)$. Recall that we can assume W^* to be laminar by Theorem 7. We arrive at an explicit characterization of W^* in three steps. First, we observe that, without loss of generality, we can assume that every pair of terminals r^1_{ij} and r^2_{ij} are adjacent in W^* and that r^2_{ij} is a leaf of W^*. Second, using the latter of these observations and laminarity, we show that for all i, the subgraph of W induced by $r^1_{i1}, r^1_{i2}, r^1_{i3}$ can only be either (a) a star, or (b) three singletons, adjacent to a unique terminal $f \notin B_i$. We say that B_i is a *center* in W^* if (a) holds. Finally, we get rid of case (b), and essentially arrive at the next lemma, whose proof can be found in the full version of the paper.

▶ **Lemma 15.** *Let $\mathcal{W}$ be the family of all laminar witness trees over T, and let W^* be a laminar witness tree such that for every $i \in [q]$, B_i is a center in W^*. Then $\nu_T(W^*) = \min_{W \in \mathcal{W}} \nu_T(W)$.*

Once we impose the condition that all B_i are centers, one notes that the tree W^* essentially must look like the one shown in Figure 4. So it only remains to compute $\nu_T(W^*)$. For every B_i, we can compute $\sum_{v \in B_i} H_{w^*(v)}$, where w^* is the vector imposed on the set S of Steiner nodes by W^*. For $i \in \{2, \ldots, q-1\}$, one notes that $\frac{1}{4}\sum_{v \in B_i} H_{w^*(v)} = \frac{1}{4}(2H_2 + H_4 + H_5) = 221/120 = 1.841\bar{6}$. Similarly, for $i = 1$ and q we have $\frac{1}{4}\sum_{v \in B_1} H_{w^*(v)} = \frac{1}{4}\sum_{v \in B_q} H_{w^*(v)} = \frac{1}{4}(2H_2 + H_3 + H_4) = \frac{83}{48} = 1.729\bar{16}$. Therefore, we can see that $\nu_T(W^*) = \sum_{v \in S} \frac{H_{w^*(v)}}{|S|} = \frac{1.841\bar{6}q - 2(1.841\bar{6} - 1.729\bar{16})}{q}$. Thus, for $q > \frac{1}{\varepsilon}$ we have $\nu_T(W^*) > 1.841\bar{6} - \frac{1}{q}$.

5 Tight bound for Steiner-Claw Free Instances

We here prove Theorem 5. Our goal is to show that for any Steiner-Claw Free instance (G, R, c), $\gamma_{(G,R,c)} \leq \frac{991}{732}$, improving over the known $\ln(4)$ bound that holds in general. From now on, we assume that we are given an optimal solution $T = (R \cup S^*, E^*)$ to (G, R, c).

Simplifying Assumptions

As standard, note that T can be decomposed into components $T_1, \ldots, T_\tau$, where each component is a maximal subtree of T whose leaves are terminals and internal nodes are Steiner nodes. Since components do not share edges of T, it is not difficult to see that one can compute a witness tree W_i for each component T_i separately, and then take the union of the $\{W_i\}_{i \geq 1}$ to get a witness tree W whose objective function $\bar{\nu}_T(W)$ will be bounded

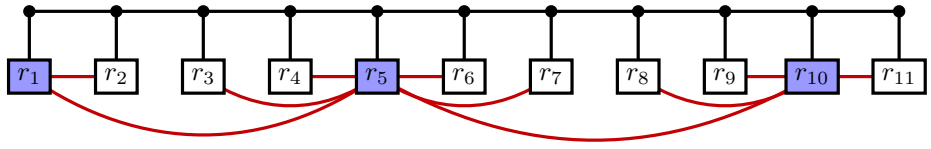


Figure 5 Edges of T are shown in black. Red edges show W. Here, $q = 11$, $t_\alpha = 5$ and $\sigma = 5$. Initially r_5 and r_{10} are picked as the centers of stars in W. Since $\sigma > \lceil \frac{t_\alpha}{2} \rceil$, r_1 is also the center of a star. Since $\sigma + t_\alpha \lfloor \frac{q-\sigma}{t_\alpha} \rfloor > q - \lceil \frac{t_\alpha}{2} \rceil$, r_q is not the center of a star.

by the maximum among $\bar{\nu}_{T_i}(W_i)$. Hence, from now on we assume that T is made by one single component. Since T is a solution to a Steiner-claw free instance, each Steiner node is adjacent to at most 2 Steiner nodes. In particular, the Steiner nodes induce a path in T, which we enumerate as $s_1, \ldots, s_q$. We will assume without loss of generality that each s_j is adjacent to exactly one terminal $r_j \in R$: this can be achieved by replacing a Steiner node incident to p terminals, with a path of length p made of 0-cost edges, if $p > 1$, and with an edge of appropriate cost connecting its 2 Steiner neighbors, if $p = 0$. We will also assume that $q > 4$. For $q \leq 4$, it is not hard to compute that $\gamma_{(G,R,c)} \leq \frac{991}{732}$. (For sake of completeness we explain this in the full version of the paper)

Witness tree computation and analysis

We denote by $L \subseteq E^*$ the edges of T incident to a terminal, and by $O = E^* \setminus L$ the edges of the path $s_1, \ldots, s_q$. Let $\alpha := c(O)/c(L)$. For a fixed value of $\alpha \geq 0$, we will fix a constant t_α as follows: If $\alpha \in [0, 32/90]$, then $t_\alpha = 5$, if $\alpha \in (32/90, 1)$, then $t_\alpha = 3$, and if $\alpha \geq 1$, then $t_\alpha = 1$. Given α (and thus t_α), we construct W using the randomized process outlined in Algorithm 2. At a high level, starting from a random offset, Algorithm 2 adds sequential stars of t_α terminals to W, connecting the centers of these stars together in this sequence. See Figure 5 for an example.

Algorithm 2 Computing the witness tree W.

Initialize $W = (R, E_W = \emptyset)$
Sample uniformly at random σ from $\{1, \ldots, t_\alpha\}$.
$E_W \leftarrow \{r_\sigma r_{\sigma+k} | 1 \leq |k| \leq \lfloor \frac{t_\alpha}{2} \rfloor, 1 \leq \sigma + k \leq q\}$
Initialize j=1
while $j \leq \frac{q-\sigma}{t_\alpha}$ **do**
  $\ell := \sigma + t_\alpha j$
  $E_W \leftarrow E_W \cup \{r_\ell r_{\ell+k} | 1 \leq |k| \leq \lfloor \frac{t_\alpha}{2} \rfloor, 1 \leq \ell + k \leq q\}$
  $E_W \leftarrow E_W \cup \{r_{\sigma+t_\alpha(j-1)} r_{\sigma+t_\alpha j}\}$
  $j \leftarrow j + t_\alpha$
if $\sigma > \lceil \frac{t_\alpha}{2} \rceil$ **then**
  $E_W \leftarrow E_W \cup \{r_1 r_k | 2 \leq k \leq \sigma - \lceil \frac{t_\alpha}{2} \rceil\} \cup \{r_1 r_\sigma\}$
$j \leftarrow \lfloor \frac{q-\sigma}{t_\alpha} \rfloor$
if $\sigma + t_\alpha j \leq q - \lceil \frac{t_\alpha}{2} \rceil$ **then**
  $E_W \leftarrow E_W \cup \{r_k r_q | \sigma + t_\alpha j + \lceil \frac{t_\alpha}{2} \rceil \leq k \leq q - 1\} \cup \{r_{\sigma+t_\alpha j} r_q\}$
Return W

Under this random scheme, we define $\lambda_L(t_\alpha) := \max_{e \in L} \mathbb{E}[H_{\bar{w}(e)}]$, and $\lambda_O(t_\alpha) := \max_{e \in O} \mathbb{E}[H_{\bar{w}(e)}]$.

▶ **Lemma 16.** *For any $\alpha \geq 0$, $\lambda_L(t_\alpha) \leq \frac{1}{t_\alpha} H_{t_\alpha+1} + \frac{t_\alpha - 1}{t_\alpha}$, and $\lambda_O(t_\alpha) \leq \frac{1}{t_\alpha} + \frac{2}{t_\alpha} \sum_{i=2}^{\lceil \frac{t_\alpha}{2} \rceil} H_i$.*

Proof. Let $W = (R, E_W)$ be a witness tree returned from running Algorithm 2 with α and $t \coloneqq t_\alpha$, and let w be the vector imposed on E^* by W. If Algorithm 2 samples $\sigma \in \{1, \ldots, t\}$, then we say that the terminals $r_{\sigma+tj}$ are *marked* by the algorithm. Moreover, if $\sigma > \lceil \frac{t_\alpha}{2} \rceil$ (resp. $\sigma + t_\alpha \lfloor \frac{q-\sigma}{t_\alpha} \rfloor \leq q - \lceil \frac{t_\alpha}{2} \rceil$) then r_1 (resp. r_q) is also considered marked.

1. Consider edge $e = s_j s_{j+1} \in O$, with $j \in \{\lceil \frac{t}{2} \rceil, \ldots, q - \lceil \frac{t}{2} \rceil\}$. Let $m \in \{j - \lfloor \frac{t}{2} \rfloor, \ldots, j + \lfloor \frac{t}{2} \rfloor\}$, such that $\sigma \mod t = m \mod t$. Observe that in this case r_m is marked. If $m = j - x$ for $x \in \{0, \ldots, \lfloor \frac{t}{2} \rfloor\}$, then $w(s_j s_{j+1}) = \lceil \frac{t}{2} \rceil - x$. Similarly if $m = j + x$ for $x \in \{1, \ldots, \lfloor \frac{t}{2} \rfloor\}$, then $w(s_j s_{j+1}) = \lceil \frac{t}{2} \rceil - x + 1$. Since $m \mod t = \sigma \mod t$ with probability $\frac{1}{t}$, we have $\mathbb{E}[H_{w(s_j s_{j+1})}] = \frac{1}{t} + \frac{2}{t} \sum_{k=2}^{\lceil \frac{t}{2} \rceil} H_k$.
   Now assume $j < \lceil \frac{t}{2} \rceil$ (the case $j > q - \lceil \frac{t}{2} \rceil$ can be handled similarly). Recalling that since t is odd it is not hard to determine the value of $w(s_j s_{j+1})$ by cases, depending on the value of σ.
   - **a.** $1 \leq \sigma \leq j$: Then $w(s_j s_{j+1}) = \lceil \frac{t}{2} \rceil + \sigma - j$.
   - **b.** $j + 1 \leq \sigma \leq \lceil \frac{t}{2} \rceil$: Then $w(s_j s_{j+1}) = j$.
   - **c.** $\lceil \frac{t}{2} \rceil + 1 \leq \sigma \leq j + \lfloor \frac{t}{2} \rfloor$: Then $w(s_j s_{j+1}) = \lceil \frac{t}{2} \rceil - \sigma + j + 1$.
   - **d.** $j + \lceil \frac{t}{2} \rceil \leq \sigma \leq t$: Then $w(s_j s_{j+1}) = \sigma - j - \lceil \frac{t}{2} \rceil + 1$.

   $$\begin{aligned}
   &\mathbb{E}[H_{w(s_j s_{j+1})}] = \\
   &= \frac{1}{t} \left(\sum_{\sigma=1}^{j} H_{\lceil \frac{t}{2} \rceil + \sigma - j} + \sum_{\sigma=j+1}^{\lceil \frac{t}{2} \rceil} H_j + \sum_{\sigma=\lceil \frac{t}{2} \rceil + 1}^{j + \lfloor \frac{t}{2} \rfloor} H_{\lceil \frac{t}{2} \rceil - \sigma + j + 1} + \sum_{\sigma=j+\lceil \frac{t}{2} \rceil}^{t} H_{\sigma - j - \lceil \frac{t}{2} \rceil + 1} \right) \\
   &= \frac{1}{t} \left(\sum_{i=\lceil \frac{t}{2} \rceil - j + 1}^{\lceil \frac{t}{2} \rceil} H_i + \left(\left\lceil \frac{t}{2} \right\rceil - j \right) H_j + \sum_{i=2}^{j} H_i + \sum_{i=1}^{\lceil \frac{t}{2} \rceil - j} H_i \right) \\
   &= \frac{1}{t} \left(\sum_{i=1}^{\lceil \frac{t}{2} \rceil} H_i + \left(\left\lceil \frac{t}{2} \right\rceil - j \right) H_j + \sum_{i=2}^{j} H_i \right) < \frac{1}{t} \left(1 + 2 \sum_{i=2}^{\lceil \frac{t}{2} \rceil} H_i \right).
   \end{aligned}$$

2. Consider edge $e = s_j r_j \in L$. We first show the bound for $j \in \{1, \ldots, q\}$. Algorithm 2 marks terminal r_i with probability $\frac{1}{t}$. If r_i is marked, then $w(e) \leq t$. If r_i is not marked, then $w(e) = 1$. Therefore, $\mathbb{E}[H_{w(e)}] \leq \frac{1}{t} H_{t+1} + \frac{t-1}{t}$
   Now consider edge $e = s_1 r_1$ (the case $e = s_q r_q$ can be handled similarly). We consider specific values of $\sigma \in \{1, \ldots, t\}$ sampled by Algorithm 2. With probability $\frac{1}{t}$, we have $\sigma = 1$, so r_1 is marked initially and $w(e) = \lceil t/2 \rceil$. For $\sigma = 2, \ldots, \lceil t/2 \rceil$, r_1 is unmarked and $w(e) = 1$. If $\sigma > \lceil t/2 \rceil$, then r_1 is marked by the algorithm and $w(e) = \sigma - \lceil t/2 \rceil$. Therefore, we can see

   $$\mathbb{E}[H_{w(r_1 s_1)}] = \frac{1}{t} \left(H_{\lceil t/2 \rceil} + \left\lfloor \frac{t}{2} \right\rfloor + \sum_{k=1}^{t - \lceil t/2 \rceil} H_k \right)$$

   We let $g(t)$ be equal to the equality above. It remains to show that $g(t) \leq \frac{1}{t} H_{t+1} + \frac{t-1}{t} \coloneqq f(t)$ for $t \in \{1, 3, 5\}$.

   $$\begin{aligned}
   g(1) &= H_1 = 1 < H_2 = f(1) \\
   g(3) &= \frac{1}{3} (H_2 + 1 + H_1) = 1.1\bar{6} < 1.36\bar{1} = \frac{1}{3}(H_4 + 2) = f(3) \\
   g(5) &= \frac{1}{5} (H_3 + 2 + H_1 + H_2) = 1.2\bar{6} < 1.29 = \frac{1}{5}(H_6 + 4) = f(5)
   \end{aligned}$$

Combining these two facts gives us the bound on $\lambda_{L_i}(t)$, for $t \in \{1, 3, 5\}$. ◀

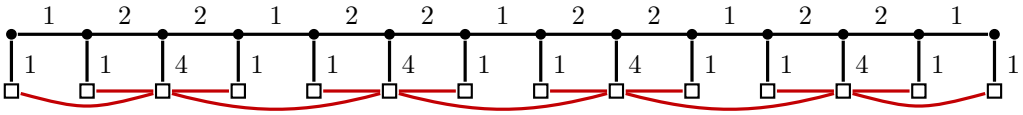


Figure 6 Lower bound instance shown in black with $c(e) = 1$ for all the edges in L and $c(e) = \alpha$ for all the edges in O, for $\alpha = \frac{32}{90}$. The white squares are terminals and black circles are Steiner nodes. Red edges form the laminar witness tree W^*, with the numbers next to each edge the value of w imposed on T.

The following Lemma is proven in the full version of the paper.

▶ **Lemma 17.** *For any $\alpha \geq 0$, the following bounds holds:*

$$\frac{1}{\alpha+1}\left(\frac{1}{t_\alpha}H_{t_\alpha+1} + \frac{t_\alpha - 1}{t_\alpha} + \alpha\left(\frac{1}{t_\alpha} + \frac{2}{t_\alpha}\sum_{i=2}^{\lceil \frac{t_\alpha}{2} \rceil} H_i\right)\right) \leq \frac{991}{732}$$

We are now ready to prove the following:

▶ **Lemma 18.** $\mathbb{E}[\bar{\nu}_T(W)] \leq \frac{991}{732}$.

Proof. One observes:

$$\sum_{e\in L\cup O} c(e)\mathbb{E}[H_{\bar{w}(e)}] \leq \sum_{e\in L} c(e)\lambda_L(t_\alpha) + \sum_{e\in O} c(e)\lambda_O(t_\alpha) = (\lambda_L(t_\alpha) + \alpha\lambda_O(t_\alpha))\sum_{e\in L} c(e)$$

Therefore $\mathbb{E}[\nu_T(W)]$ is bounded by:

$$\frac{\sum_{e\in L\cup O} c(e)\mathbb{E}[H_{\bar{w}(e)}]}{\sum_{e\in L\cup O} c(e)} \leq \frac{(\lambda_L(t_\alpha) + \alpha\lambda_O(t_\alpha))\sum_{e\in L} c(e)}{(\alpha+1)\sum_{e\in L} c(e)} = \frac{\lambda_L(t_\alpha) + \alpha\lambda_O(t_\alpha)}{\alpha+1} \leq \frac{991}{732}.$$

where the last inequality follows using Lemma 16 and 17. ◀

Now Theorem 5 follows by combining Lemma 18 with Theorem 1 in which γ is replaced by the supremum taken over all Steiner-claw free instances (rather than over all Steiner Tree instances).

Tightness of the bound

We conclude this section by spending a few words on Theorem 6. Our lower-bound instance is obtained by taking a tree T on q Steiner nodes, each adjacent to one terminal, with $c(e) = 1$ for all the edges in L and $c(e) = \alpha$ for all the edges in O, for $\alpha = \frac{32}{90}$. Similar to Section 3, a crucial ingredient for our analysis is in utilizing Theorem 8 stating that there is an optimal laminar witness tree. See Figure 6. We use this to show that there is an optimal witness tree for our tree T, whose objective value is at least $\frac{991}{732} - \varepsilon$. Details can be found in the full version of the paper.

References

1 David Adjiashvili. Beating approximation factor two for weighted tree augmentation with bounded costs. *ACM Trans. Algorithms*, 15(2):19:1–19:26, 2019.

2 Haris Angelidakis, Dylan Hyatt-Denesik, and Laura Sanità. Node connectivity augmentation via iterative randomized rounding. *Mathematical Programming*, pages 1–37, 2022.

3 Manu Basavaraju, Fedor V. Fomin, Petr A. Golovach, Pranabendu Misra, M. S. Ramanujan, and Saket Saurabh. Parameterized algorithms to preserve connectivity. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 800–811, 2014.

4 Jaroslaw Byrka, Fabrizio Grandoni, and Afrouz Jabal Ameli. Breaching the 2-approximation barrier for connectivity augmentation: a reduction to Steiner tree. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 815–825, 2020.

5 Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *J. ACM*, 60(1):6:1–6:33, 2013.

6 Federica Cecchetto, Vera Traub, and Rico Zenklusen. Bridging the gap between tree and connectivity augmentation: unified and stronger approaches. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 370–383. ACM, 2021.

7 Joseph Cheriyan and Zhihan Gao. Approximating (unweighted) tree augmentation via lift-and-project, part I: stemless TAP. *Algorithmica*, 80(2):530–559, 2018.

8 Joseph Cheriyan and Zhihan Gao. Approximating (unweighted) tree augmentation via lift-and-project, part II. *Algorithmica*, 80(2):608–651, 2018.

9 Efim A Dinitz, Alexander V Karzanov, and Michael V Lomonosov. On the structure of the system of minimum edge cuts in a graph. *Issledovaniya po Diskretnoi Optimizatsii*, pages 290–306, 1976.

10 Andreas Emil Feldmann, Jochen Könemann, Neil Olver, and Laura Sanità. On the equivalence of the bidirected and hypergraphic relaxations for steiner tree. *Mathematical programming*, 160(1):379–406, 2016.

11 Samuel Fiorini, Martin Groß, Jochen Könemann, and Laura Sanità. Approximating weighted tree augmentation via chvátal-gomory cuts. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 817–831. SIAM, 2018.

12 Greg N. Frederickson and Joseph JáJá. Approximation algorithms for several graph augmentation problems. *SIAM J. Comput.*, 10(2):270–283, 1981.

13 Michel X. Goemans, Neil Olver, Thomas Rothvoß, and Rico Zenklusen. Matroids and integrality gaps for hypergraphic steiner tree relaxations. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 1161–1176, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2213977.2214081.

14 Fabrizio Grandoni, Christos Kalaitzis, and Rico Zenklusen. Improved approximation for tree augmentation: saving by rewiring. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 632–645. ACM, 2018.

15 Dylan Hyatt-Denesik, Afrouz Jabal Ameli, and Laura Sanità. Finding almost tight witness trees, 2023. arXiv:2211.12431.

16 Zeev Nutov. 2-node-connectivity network design. In *Proceedings of the 18th International Workshop on Approximation and Online Algorithms (WAOA)*, volume 12806 of *Lecture Notes in Computer Science*, pages 220–235. Springer, 2020.

17 Zeev Nutov. Approximation algorithms for connectivity augmentation problems. In *Proceedings of the 16th International Computer Science Symposium in Russia (CSR)*, volume 12730, pages 321–338. Springer, 2021.

18 Vera Traub and Rico Zenklusen. A $(1.5+\varepsilon)$-approximation algorithm for weighted connectivity augmentation, 2022. doi:10.48550/arXiv.2209.07860.

19 Vera Traub and Rico Zenklusen. A better-than-2 approximation for weighted tree augmentation. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1–12. IEEE, 2022.

20 Vera Traub and Rico Zenklusen. Local search for weighted tree augmentation and steiner tree. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3253–3272. SIAM, 2022.

Efficient Caching with Reserves via Marking

Sharat Ibrahimpur
Department of Mathematics, London School of Economics and Political Science, UK

Manish Purohit
Google Research, USA

Zoya Svitkina
Google Research, USA

Erik Vee
Google Research, USA

Joshua R. Wang
Google Research, USA

Abstract

Online caching is among the most fundamental and well-studied problems in the area of online algorithms. Innovative algorithmic ideas and analysis – including potential functions and primal-dual techniques – give insight into this still-growing area. Here, we introduce a new analysis technique that first uses a potential function to upper bound the cost of an online algorithm and then pairs that with a new dual-fitting strategy to lower bound the cost of an offline optimal algorithm. We apply these techniques to the Caching with Reserves problem recently introduced by Ibrahimpur et al. [10] and give an $O(\log k)$-competitive fractional online algorithm via a marking strategy, where k denotes the size of the cache. We also design a new online rounding algorithm that runs in polynomial time to obtain an $O(\log k)$-competitive randomized integral algorithm. Additionally, we provide a new, simple proof for randomized marking for the classical unweighted paging problem.

2012 ACM Subject Classification Theory of computation → Caching and paging algorithms

Keywords and phrases Approximation Algorithms, Online Algorithms, Caching

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.80

Category Track A: Algorithms, Complexity and Games

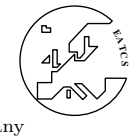
Related Version *Full Version*: `https://arxiv.org/abs/2305.02508` [11]

Funding *Sharat Ibrahimpur*: Received funding from the following sources: NSERC grant 327620-09 and an NSERC DAS Award, European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. ScaleOpt–757481), and Dutch Research Council NWO Vidi Grant 016.Vidi.189.087.

1 Introduction

Caching is a critical component in many computer systems, including computer networks, distributed systems, and web applications. The idea behind caching is simple: store frequently used data items in a cache so that subsequent requests can be served directly from the cache to reduce the resources required for data retrieval. In the classical unweighted caching problem, a sequence of page requests arrives one-by-one and an algorithm is required to maintain a small set of pages to hold in the cache so that the number of requests not served from the cache is minimized.

Traditional caching algorithms, both in theory and practice, are designed to optimize the global efficiency of the system and aim to maximize the *hit rate*, i.e., fraction of requests that are served from the cache. However, such a viewpoint is not particularly suitable for



cache management in a multi-user or multi-processor environment. Many cloud computing services allow multiple users to share the same physical workstations and thereby share the caching system. In such multi-user environments, traditional caching policies can lead to undesirable outcomes as some users may not be able to reap any benefits of the cache at all. Recently, Ibrahimpur et al. [10] introduced the *Caching with Reserves* model that ensures certain user-level fairness guarantees while still attempting to maximize the global efficiency of the system. In this formulation, a cache of size k is shared among m agents and each agent i is guaranteed a reserved cache size of k_i. An algorithm then attempts to minimize the total number of requests that are not served from the cache while guaranteeing that any time step, each agent i holds at least k_i pages in the cache. Unlike the classical paging problem, *Caching with Reserves* is NP-complete even in the offline setting when the algorithm knows the entire page request sequence ahead of time and Ibrahimpur et al. [10] gave a 2-approximation algorithm. They also gave an $O(\log k)$-competitive online fractional algorithm for *Caching with Reserves* via a primal-dual technique and then design a rounding scheme to obtain an $O(\log k)$-competitive online randomized algorithm. Unfortunately, the rounding scheme presented in [10] does not run in polynomial time and the fractional primal-dual algorithm, while simple to state, also does not yield itself to easy implementation.

Caching and its many variants have been among the most well-studied problems in theoretical computer science. It has long been a testbed for novel algorithmic and analysis techniques and it has been investigated via general techniques such as potential function analysis, primal-dual algorithms, and even learning-augmented algorithms. For the classical unweighted caching problem, a particularly simple algorithm, *randomized marking* [9], is known to yield the optimal competitive ratio (up to constant factors). At any point in time, the randomized marking algorithm partitions the set of pages in cache into *marked* and *unmarked* pages and upon a cache miss, it evicts an unmarked page chosen uniformly at random. Cache hits and pages brought into the cache are marked. When a cache miss occurs, but there are no more unmarked pages, a new *phase* begins, and all pages in the cache become unmarked. In this paper, we build upon this algorithm, adapting it to caching with reserves.

Our Contributions

We study the *Caching with Reserves* model of Ibrahimpur et al. [10] in the online setting and improve upon those results. Our first main result is a simpler fractional algorithm that is a generalization of randomized marking for classical caching.

▶ **Theorem 1.** *There is an $O(\log k)$-competitive fractional* marking *algorithm for online* Caching with Reserves*. The competitive guarantee holds even when the optimal offline algorithm is allowed to hold fractional pages in the cache.*

We remark that our algorithm in Theorem 1 and its analysis are more involved than those of the classical randomized marking algorithm. One complication is that due to the reserve constraints, a marking-style algorithm for the caching with reserves setting cannot evict an arbitrary unmarked page. Another key difficulty comes from the fact that even the notion of a *phase* is non-trivial to define in our setting. In particular, unlike in classical caching, it can happen that the cache still contains unmarked pages, but none of them can be evicted to make space for a new page, because of the reserve constraints. Thus, we need a rule to isolate agents whose reserve constraints prevent the algorithm from having a clean end of a phase, while also ensuring that the already marked pages of such isolated agents are not erased prematurely. To this end, we introduce the notion of *global* and *local* phases to effectively model the state of each agent. We elaborate on this in Section 3.1.

Our analysis of the fractional marking algorithm introduces two novel components that may be of independent interest. First, we upper-bound the total cost incurred by our fractional marking algorithm using a new potential function. This potential function, introduced in Section 3.3, depends only on the decisions of the algorithm and is independent of the optimal solution. To the best of our knowledge, all previous potential function based analyses of (variants of) caching [4, 5, 6, 10] define a potential function that depends on the optimal solution. Second, we introduce a new lower bound for the cost of the optimal solution via the dual-fitting method. Our techniques also yield a new simple proof that the classical *randomized marking* [9] for unweighted paging is $O(\log k)$-competitive (see the full version [11] for more details).

We also design a new online rounding algorithm that converts a deterministic, fractional algorithm into a randomized, integral algorithm while only incurring a constant factor loss in the competitive ratio. Via a careful discretization technique (inspired by Adamaszek et al. [2]), the new rounding algorithm runs in polynomial time and only uses limited randomization. Our fractional marking algorithm (Algorithm 1) maintains that at any point in time, a particular page p is either completely in the cache or at least $1/k$ fraction of the page has been evicted. We exploit this key property to show that the fractional solution at any time t can be discretized so that the fraction of any page that is evicted is an integral multiple of $1/k^3$. This discretization allows us to maintain a distribution over feasible integer cache states with bounded support.

▶ **Theorem 2.** *There is a polynomial-time $O(\log k)$-competitive randomized integral algorithm for online caching with reserves.*

Other Related Work

The unweighted caching (also known as paging) problem has been widely studied and its optimal competitive ratio is well-understood even up to constant factors. Tight algorithms [1, 14] are known that yield a competitive ratio of exactly H_k, where H_k is the kth harmonic number. Recently, Agrawal et al. [3] consider the *parallel paging* model where a common cache is shared among p agents – each agent is presented with a request sequence of pages and the algorithm must decide how to partition the cache among agents at any time. It allows the p processors to make progress simultaneously, i.e., incur cache hits and misses concurrently. Multi-agent paging has also been extensively studied in the systems community [7, 16, 17] often in the context of caching in multi-core systems. Closely related to the *Caching with Reserves* setting, motivated by fairness constraints in multi-agent settings, a number of recent systems [12, 13, 15, 18] aim to provide *isolation guarantees* to each user, i.e., guarantee that the cache hit rate for each user is at least as much as what it would be if each user is allocated its own isolated cache. Also motivated by fairness constraints, Chiplunkar et al. [8] consider the Min-Max paging problem where the goal is to minimize the maximum number of page faults incurred by any agent.

2 Preliminaries and Notation

Formally, an instance of the *Caching with Reserves* problem consists of the following. We are given a number of agents m and a total (integer) cache capacity k. Let $[m]$ denote the set $\{1, \ldots, m\}$. Each agent $i \in [m]$ owns a set of pages $\mathcal{P}(i)$ (referred to as i-pages) and has a *reserved cache size* $k_i \geq 0$. Pages have a unique owner, i.e. $\mathcal{P}(i) \cap \mathcal{P}(j) = \varnothing$ for all $i \neq j$, and we use $\mathcal{P} \triangleq \cup_{i \in [m]} \mathcal{P}(i)$ to refer to the universe of all pages. For any page $p \in \mathcal{P}$, let $ag(p)$ be

the unique agent that owns p. We assume without loss of generality that at least one unit of cache is not reserved: $\sum_{i\in[m]} k_i < k$.[1] At each timestep t, a page $p_t \in \mathcal{P}$ is requested. We can wrap all these into an instance tuple: $\sigma = (m, k, \{\mathcal{P}(i)\}, \{k_i\}, \{p_t\})$.

An *integral* algorithm for the *Caching with Reserves* problem maintains a set of k pages in the cache such that for each agent i, the cache always contains at least k_i pages from $\mathcal{P}(i)$. At time t, the page request p_t is revealed to the algorithm. If this page is not currently in the cache, then the algorithm is said to incur a *cache miss* and it must fetch p_t into the cache by possibly evicting another page q_t. For any (integral) algorithm $\mathcal{A}$ we write its total cache misses on instance σ as $\text{cost}_{\mathcal{A}}(\sigma)$.

A *fractional* algorithm for the *Caching with Reserves* problem maintains a fraction $x_p \in [0, 1]$ for how much each page $p \in \mathcal{P}$ is in the cache such that the total size of pages in the cache is at most k, i.e, $\sum_{p\in\mathcal{P}} x_p \le k$, and the total size of i-pages is at least k_i, i.e., $\sum_{p\in\mathcal{P}(i)} x_p \ge k_i$. At time t, the page request p_t is revealed to the algorithm, which incurs a fractional cache miss of size $1 - x_{p_t}$. The algorithm must then fully fetch p_t into cache ($x_{p_t} \leftarrow 1$) by possibly evicting other pages. For any (fractional) algorithm $\mathcal{A}$ we again write its total size of cache misses on instance σ as $\text{cost}_{\mathcal{A}}(\sigma)$.

Let $\text{cost}_{OPT}(\sigma)$ be the cost of the optimal offline algorithm on instance σ.

▶ **Definition 3** (Competitive Ratio). *An online algorithm $\mathcal{A}$ for* Caching with Reserves *is said to be c-competitive, if for any instance σ, $\mathbb{E}[\text{cost}_{\mathcal{A}}(\sigma)] \le c \cdot \text{cost}_{OPT}(\sigma) + b$, where b is a constant independent of the number of page requests in σ. The expectation is taken over all the random choices made by the algorithm (if any).*

3 Fractional $O(\log k)$-Competitive Algorithm for Caching with Reserves

For any time t and page $p \in \mathcal{P}$, the algorithm maintains a variable $y_p^t \in [0, 1]$ representing the portion of page p that is outside the cache. Then $x_p^t \triangleq 1 - y_p^t$ represents the portion of p that is in cache. Algorithm 1 ensures feasibility at all times t: the total of all y values is exactly the complementary cache size $|\mathcal{P}| - k$, i.e., $\sum_{p\in\mathcal{P}} y_p^t = |\mathcal{P}| - k$; and the total y value for pages of any agent i is within its respective complementary reserve size, $\sum_{p\in\mathcal{P}(i)} y_p^t \le |\mathcal{P}(i)| - k_i$. When a request for page p_t arrives at time t, the algorithm fully fetches p_t into the cache by paying a fetch-cost of $y_{p_t}^t$ while simultaneously evicting a total of $y_{p_t}^t$ amount of other suitably chosen pages.

3.1 Fractional Algorithm

The complete algorithm (referred to as Algorithm $\mathcal{A}$ in the proofs) is presented in Algorithm 1. We present a high-level discussion here. At any time t, we say that an agent i is *tight* if $\sum_{p\in\mathcal{P}(i)} x_p^t = k_i$, i.e., the algorithm is not allowed to further evict any pages (even fractionally) of agent i. Conversely, an agent i is *non-tight* if $\sum_{p\in\mathcal{P}(i)} x_p^t > k_i$.

The algorithm is a fractional marking algorithm and runs in phases where each phase corresponds to a maximal sequence of page requests that can be served while maintaining feasibility and ensuring that no "marked" pages are evicted. Within each phase, the currently requested page p_t is fully fetched into cache by continuously evicting an infinitesimal amount of an "available" (described below) unmarked page q with the smallest y_q value; if there are multiple choices of q, then all of them are simultaneously evicted at the same rate. Page p_t gets marked after it has been served and this mark may only be erased at the end of a phase.

[1] If all of cache is reserved, the problem decomposes over agents into the standard caching task.

At the end of a phase, an agent i is designated as *isolated* if strictly fewer than k_i i-pages are marked in the cache at this time point. This designation changes to non-isolated as soon as k_i i-pages get marked at some point in the future. An isolated agent essentially runs a separate instance of caching on its own pages and in its own reserved space. At the end of a phase, the marks of pages owned by non-isolated agents (i.e., agents with at least k_i marked i-pages) are erased.

It remains to describe when a page q is considered available for eviction. Clearly, $y_q^t < 1$ must hold, since otherwise page q is already fully outside the cache. Moreover, $ag(q)$ must be *non-tight*, i.e., evicting page q must not violate the reserve constraint of the agent that owns it. The last condition for q to be considered available for eviction depends on whether the agent $i_t := ag(p_t)$ is *isolated* or not: (i) if agent i_t is isolated, then $ag(q) = i_t$ should hold, i.e., only unmarked i_t-pages are available for eviction; and (ii) if agent i_t is not isolated, then $ag(q)$ should also be non-isolated. We recall again that among all available pages for eviction, pages with the smallest y_q^t value are evicted first.

Notation. Let $\mathcal{I}(t) \subsetneq [m]$ denote the set of isolated agents at time t. For a global phase r_0, we use $\mathcal{I}(r_0)$ to denote the set of isolated agents at the end of phase r_0. Let $\mathcal{T}(t)$ denote the set of *tight* agents at time t. At any time t, let r_i^t denote the value of the local phase counter r_i for agent i, and let $R_i = r_i^T$ be the total number of local phases for agent i. By definition, for any agent $i \in [m]$, $P(i, r_i - 1)$ and $P(i, r_i)$ denote the set of i-pages in the cache (integrally) at the beginning of the r_ith local phase and the end of the r_ith local phase for agent i, respectively. For any agent $i \in [m]$, let $M(i,t) \subseteq \mathcal{P}(i)$ denote the set of marked i-pages in the cache and $U(i,t) = P(i, r_i^t - 1) \setminus M(i,t)$ denote the set of unmarked i-pages.

We emphasize that the notion of *unmarked* pages will only be relevant while referring to pages in $P(i, r_i^t - 1)$ for some i, t; in particular, every i-page $q \in \mathcal{P}(i) \setminus (P(i, r_i^t - 1) \cup M(i,t))$ is not *marked*, but we do not refer to it as *unmarked*. Analogous to the notion of clean and stale pages used by the randomized marking algorithm [9], we define *clean, pseudo-clean* and *stale* pages as follows. Fix an agent i and let r_i be its local phase counter at time t. Any i-page $q \in P(i, r_i - 1)$ is considered *stale*. The currently requested page p_t is said to be *clean* if $p_t \notin P(i, r_i - 1)$. Next, we say that the currently requested page p_t is *pseudo-clean* if $p_t \in P(i, r_i - 1)$ and $y_{p_t}^t = 1$ holds right before Algorithm $\mathcal{A}$ starts to fetch p_t into the cache. Lemmas 7 and 8 show that a pseudo-clean page necessarily belongs to an agent who was isolated at the start of (global) phase r_0 but is non-isolated at time t. To simplify notation, we drop the superscript t from all notation whenever the time index is clear from the context.

The following lemma compiles a list of key invariants that are maintained throughout the execution of the algorithm that follow directly from an examination of Algorithm 1.

▶ **Lemma 4.** *Algorithm 1 maintains the following invariants.*

- **(i)** *When a new phase begins, all marked pages belong to isolated agents.*
- **(ii)** *At any time t, all isolated agents are tight.*
- **(iii)** *At any time t and for any agent i, all unmarked pages of agent i have the same y value.*
- **(iv)** *Any page belonging to an isolated agent is (fractionally) evicted only in those timesteps when a different page of the same agent has been requested.*

The following lemmas show that the algorithm is well-defined and that the operations in Lines 12 and 18 of Algorithm 1 are always feasible.

[2] Agent i_t is considered non-tight here because fetching p_t while evicting other $q \in \mathcal{P}(i_t) \setminus p_t$ does not violate reserve feasibility.

Algorithm 1 Fractional Marking Algorithm for *Caching with Reserves*.

```
/* Initialization */
r_0 ← 1                                                  /* global phase counter */
r_i ← 1, ∀i ∈ [m]                                        /* local phase counters */
Let P(i,0) ⊂ P(i) be set of i-pages in the initial cache (assume |P(i,0)| ≥ k_i) ∀i ∈ [m]
All agents i ∈ [m] are non-isolated and all pages p in the cache are unmarked
for each page request p_t of agent i_t do
    if y_{p_t} = 0, i.e., x_{p_t} = 1, then
        Mark page p_t and serve the request.
    else if agent i_t is isolated, then
        /* Continuously fetch page p_t while uniformly evicting all unmarked
           i_t-pages. */
        Set y_{p_t} ← 0, mark page p_t and serve the request.
        Increase y_q at the same rate for all unmarked i_t-pages in the cache until the cache
          becomes feasible, i.e. Σ_{p∈P} y_p ≥ |P| − k holds.
        if agent i_t now has k_i marked pages then
            Designate i_t as non-isolated
    else if ∃ page q owned by some non-tight agent[2] and satisfying y_q < 1, then
        /* Continuously fetch page p_t while uniformly evicting all unmarked
           pages (belonging to any non-tight agent) with the least y-value. */
        Set y_{p_t} ← 0, mark page p_t and serve the request.
        Increase y_q at the same rate for all unmarked pages of non-tight agents with the
          smallest y values until the cache becomes feasible, i.e. Σ_{p∈P} y_p ≥ |P| − k holds.
    else
        /* End of phase */
        for each agent i ∈ [m] do
            if i has strictly fewer than k_i marked pages, then
                Designate i as isolated.
            else
                /* i is non-isolated and undergoes a phase reset */
                Set P(i, r_i) ← collection of all (integral and marked) i-pages in cache.
                Set r_i ← r_i + 1
                All marked i-pages are now unmarked
        Set r_0 ← r_0 + 1
        Re-process the current page request p_t in the new phase
```

▶ **Lemma 5.** *If the requested page p_t has $x^t_{p_t} \in [0,1)$ and agent i_t is isolated, then p_t can be fetched fully by evicting unmarked pages of agent i_t.*

Proof. As agent i_t is isolated when page p_t is requested, $\sum_{p\in\mathcal{P}(i)} x^t_p = k_i$ (by invariant (ii) in Lemma 4) and i_t has fewer than k_i marked pages in cache. Hence $\sum_{p\in U(i,t)} x^t_p \geq 1$ and $\sum_{p\in U(i,t)\setminus\{p_t\}} x^t_p \geq 1 - x^t_{p_t}$. ◀

▶ **Lemma 6.** *If the requested page p_t has $0 < x^t_{p_t} < 1$ and its owner i_t is non-isolated, then there is always enough fractional mass of pages belonging to non-tight agents that can be evicted to fully fetch page p_t. In particular, line 18 of Algorithm 1 is well-defined.*

Proof. Suppose page p_t is fetched in a continuous manner. To show that page p_t can be fetched fully, it suffices to show that at any instantaneous time t when $x^t_{p_t} < 1$, there always exists an unmarked page q belonging to a non-tight agent i such that $x^t_q > 0$, i.e. page

q can be evicted. Observe that $k = \sum_{q\in\mathcal{P}} x_q^t = \sum_{i\in\mathcal{T}(t)} k_i + \sum_{i\notin\mathcal{T}(t)} \sum_{q\in\mathcal{P}(i)} x_q^t$. Due to integrality of k and $\{k_i\}_{i\in[m]}$, we must have $\sum_{i\notin\mathcal{T}(t)} \sum_{q\in\mathcal{P}(i)} x_q^t$ is an integer. Since any marked page q always has $x_q^t = 1$, $\mu := \sum_{i\notin\mathcal{T}(t)} \sum_{q\in U(i,t)} x_q^t$ is also an integer. Further, since $i_t \notin \mathcal{T}(t)$ and $x_{p_t}^t > 0$, we must have $\mu \geq 1$ and hence there exists a page q belonging to some non-tight agent i with $x_q^t > 0$ as desired. ◀

Since we always evict an available page with the least y value, at any time step t, all available pages (i.e., unmarked pages q belonging to non-tight agents and satisfying $y_q < 1$) have the same y value at all times. We denote this common y-value by h^* and refer to the corresponding set of evictable pages (with y-value h^*) as the *frontier*. The following two key structural lemmas formalize this property.

▶ **Lemma 7.** *At any time t, let i be an agent that was isolated at the beginning of the current phase, and let q be one of its unmarked pages. Then $y_q^t < 1$ if and only if i is still isolated at time t.*

Proof. For the *if* direction, suppose that i is still isolated. By invariant (ii), it is also tight. By invariant (iii), all its unmarked pages have the same x-value and in total they occupy $k_i - |M(i,t)| > 0$ units of cache space. Thus, $x_q^t > 0$ and $y_q^t < 1$.

For the *only if* direction, suppose that i is no longer isolated. Just before the k_ith i-page to be marked was requested, $(k_i - 1)$ i-pages were marked. Since i was tight, the total x value of all its unmarked pages must have been 1. Then the algorithm replaced all of them with the k_ith marked page, and the x-value of all remaining unmarked pages became 0. Thus, the property holds for a newly non-isolated agent i. This property continues to hold for the rest of the phase since y_q never decreases for an unmarked page. ◀

▶ **Lemma 8.** *At any time t, there is a value $h_t^* \in [0,1]$ such that: for any agent i that was non-isolated at the beginning of the current phase and any unmarked i-page q, $y_q \leq h_t^*$ holds and $y_q = h_t^*$ holds whenever i is non-tight.*

Proof. We prove the lemma by induction. Clearly, the lemma holds at the start of the phase: all unmarked pages belonging to non-isolated agents have y-value 0. Now consider a time t during phase r such that the lemma holds for all timepoints before t in this phase. We may also assume that $y_{p_t} > 0$, since otherwise none of the variables are modified in this timestep.

By induction hypothesis, any unmarked i-page q satisfies $y_q \leq h_{t-1}^*$, and this inequality is tight whenever i is non-tight. If agent i_t is non-tight, then its unmarked pages are already part of the frontier. Otherwise, Algorithm $\mathcal{A}$ fetches p_t fully into the cache by increasing the y-value of other unmarked i_t-pages until one of the following happens: (a) p_t is fully fetched. In this case, i_t continues to remain tight; or (b) The y-value of unmarked i_t-pages becomes equal to the frontier's y-value, h_{t-1}^*. In the latter case, unmarked i_t-pages become part of the frontier and the y-value of the frontier is uniformly increased until p_t gets fully fetched into the cache. If some agent i' becomes tight before the fetch operation is completed, then its unmarked pages get excluded from the frontier and the corresponding y-values remain unchanged for the rest of this timestep. In all cases, the lemma continues to hold since the y-value of the frontier is never decreased and only tight agents get dropped from the frontier. ◀

▶ Remark 9. Within any phase, h_t^* is non-decreasing over time and takes values 0 and 1 at the endpoints. This follows from the fact that $\mathcal{A}$ never decreases y_q for an unmarked page $q \neq p_t$.

The following lemma shows that any page that is not completely in Algorithm $\mathcal{A}$'s cache must be evicted to at least a $1/k$ portion. This property will be useful to us in Sections 3.3 and 4.

▶ **Lemma 10.** *At the end of any time step t, for any page $p \in \mathcal{P}$, we have $y_p^t = 0$ or $y_p^t \geq 1/k$.*

Proof. First, note that for all marked pages, we have $y_p^t = 1 - x_p^t = 0$. Let $i \in \mathcal{T}(t)$ be any tight agent. Then we have $k_i = \sum_{p \in \mathcal{P}(i)} x_p^t = |M(i,t)| + \sum_{p \in U(i,t)} x_p^t$. By Lemma 4 (part iii), all unmarked pages of agent i have the same y value: $y_p^t = 1 - x_p^t = h_i$ (say). Since k_i is integral, we have either $h_i = 0$ or h_i. Rearranging, we have $|U(i,t)|h_i = |M(i,t)| + |U(i,t)| - k_i$. Since all terms on the RHS are integral, we have either $h_i = 0$ or $h_i \geq 1/|U(i,t)| \geq 1/k$.

By Lemma 8, all unmarked pages p belonging to non-tight agents satisfy $y_p^t = h_t^*$. Let $U(t)$ be the set of all unmarked pages belonging to all non-tight agents that were also non-isolated at the beginning of the phase. Recall that by definition, we have $|U(t)| \leq k$ since all pages in $U(t)$ must have been fully in the cache at the beginning of the current phase. Since we have $k = \sum_{i \in \mathcal{T}(t)} k_i + \sum_{i \notin \mathcal{T}(t)} \sum_{p \in \mathcal{P}(i)} x_p^t$, once again by integrality of k and $\{k_i\}$, we must have that $\sum_{p \in U(t)} y_p^t = \sum_{p \in U(t)} h_t^*$ is an integer. Hence, either $h_t^* = 0$ or $h_t^* \geq 1/|U(t)| \geq 1/k$. ◀

3.2 Analysis Overview

At any time t, we consider the set of y values of pages in $\bigcup_{i \in [m]} P(i, r_i - 1)$ as the *state* of the system. We define a non-negative potential function Ψ that is purely a function of this state. For any page request p_t, we attempt to bound the algorithm's cost by an increase in the potential function, thereby bounding the total cost incurred by the algorithm by the final value of the potential function. There are two difficulties with this approach: (i) when a phase ends, the potential function abruptly drops since all the unmarked pages that were fully evicted no longer contribute to the state, and (ii) when the agent i_t was isolated at the beginning of the phase but is now non-isolated, the change in potential is not sufficient to cover the fetch cost. In both these situations we charge the cost incurred by the online algorithm to a new quantity that is a function of the sets $\{P(i, r_i)\}$. To complete the analysis, we show that this quantity is upper-bounded by the cost of the optimal solution.

3.3 Potential Function Analysis

Consider the function $\phi : [0,1] \to \mathbb{R}_{\geq 0}$ defined as:

$$\phi(h) \triangleq 2h \cdot \ln(1 + kh) \tag{1}$$

As h goes from 0 to 1, $\phi(h)$ increases from 0 to $2\ln(1+k)$.

The potential at any time t is defined as follows:

$$\Psi(t) \triangleq \sum_{i=1}^{m} \sum_{p \in U(i,t)} \phi(y_p^t) \tag{2}$$

Note that only unmarked pages at any time t contribute to the potential. So when page p_t is fetched at time t and marked, it stops contributing to the potential. But since ϕ is monotone, the newly evicted pages increase their contribution to the potential. We remark that the potential is purely a function of the state of the system as defined by the y values of unmarked pages in the cache and is thus always bounded by a quantity independent of the length of the page request sequence.

▶ **Lemma 11.** *For any $h \geq 1/k$, we have $\phi(h) \geq h$ and $\phi'(h) \geq 1 + 2\ln(1+kh)$.*

Proof. The first conclusion follows from the logarithmic inequality $\ln(1+x) \geq x/(1+x)$ which holds for any nonnegative x: we have $\phi(h) = 2h\ln(1+kh) \geq h \cdot 2kh/(1+kh) \geq h$ whenever $kh \geq 1$. Next, $\phi'(h) = \frac{d\phi}{dh} = 2(1 - 1/(1+kh) + \ln(1+kh))$. So, for any $h \geq 1/k$ we have $1/2 \geq 1/(1+kh)$, which gives the other conclusion. ◀

The rest of this section is devoted to proving the following theorem where we bound the total cost incurred by the algorithm in terms of the sets $\{P(i, r_i)\}$ and the number of requests to pseudo-clean pages.

▶ **Theorem 12.** *The following bound holds on the cost incurred by $\mathcal{A}$ to process the first T page requests:*

$$\text{cost}_{\mathcal{A}}(\sigma) \leq 2\ln(1+k) \cdot \left(mk + \sum_{t=1}^{T} \mathbb{1}_{p_t \text{ is pseudo-clean}} + \sum_{i \in [m]} \sum_{r_i=1}^{R_i} |P(i, r_i - 1) \setminus P(i, r_i)| \right).$$

Recall that the algorithm incurs a cost of $y^t_{p_t}$ to fetch page p_t at time t. So the total cost incurred by the algorithm is simply $\text{cost}_{\mathcal{A}}(\sigma) = \sum_t y^t_{p_t}$. We first bound this cost for time steps when the requested page p_t is at least partially in the cache, i.e., $y^t_{p_t} < 1$. Recall by Lemmas 5 and 6, the algorithm does not undergo a phase transition in this time step.

▶ **Lemma 13.** *Consider any time step t such that $y^t_{p_t} < 1$ for the currently requested (unmarked) page p_t. Let $\Delta\Psi(t)$ denote the change in the potential function during time step t. Then $y^t_{p_t} \leq \Delta\Psi(t)$.*

Proof. We assume that $y^t_{p_t} \geq \frac{1}{k}$, since otherwise by Lemma 10, we must have $y^t_{p_t} = 0$ and the lemma follows trivially. Since $y^t_{p_t} < 1$, by Lemma 8, either agent i_t is tight or we have $y^t_q = y^t_{p_t}$ for every unmarked page q owned by any non-tight agent i that was non-isolated at the start of this phase. In either case, the pages that get evicted to make space for p_t have their initial y values at least $y^t_{p_t} \geq 1/k$. The potential function Ψ changes in this step due to two factors: (i) Ψ drops as page p_t stops contributing to the potential as soon as it gets marked; and (ii) Ψ increases as the y-value of (fractionally) evicted pages increases in this step.

Let $h \triangleq y^t_{p_t}$. At the beginning to time t, page p_t contributed exactly $\phi(h) = 2h\ln(1+kh)$ to the potential; This contribution is lost as soon as p_t gets marked. To prove the lemma, it suffices to show that the rate of increase in the potential function (without including p_t's contribution) is at least $1 + 2\ln(1+kh)$ throughout the eviction of an h amount of unmarked pages belonging to non-tight agents: the 1 term in total pays for the fetch-cost of h and the $2\ln(1+kh)$ term in total pays for the $2h\ln(1+kh)$ loss in potential. This directly follows from Lemma 11 from the fact that the y-values of pages that are fractionally evicted in this timestep were already at least $h \geq 1/k$. Here, we also use the monotonicity of the function $h' \mapsto \ln(1+kh')$. ◀

We still need to bound the cost incurred by the algorithm when the incoming request is to a page that is fully outside the cache. Note that the algorithm incurs exactly unit cost for all such time steps. The following lemma shows that the total cost incurred by the algorithm can be bounded by the drop in potential function at the end of a phase and by a term that depends only on the change in the potential function while processing a request to a page fully outside the cache.

▶ **Lemma 14.** *For any global phase r_0, let $\Delta\Psi(r_0)$ denote the change in the potential function at the end of phase r_0 (line 30 in Algorithm 1). Let R_0 denote the total number of global phases and T denote the time at the end of phase R_0. Then we have the following upper bound on the cost incurred by $\mathcal{A}$ for processing the first T page requests:*

$$\text{cost}_{\mathcal{A}}(\sigma) \leq 2mk\ln(1+k) + \sum_{t\in[T]:y^t_{p_t}=1} (1-\Delta\Psi(t)) - \sum_{r_0=1}^{R_0} \Delta\Psi(r_0)$$

Proof. We have:

$$\begin{aligned}
\text{cost}_{\mathcal{A}}(\sigma) &= \sum_{t\in[T]} y^t_{p_t} = \sum_{t\in[T]:y^t_{p_t}<1} y^t_{p_t} + |\{t\in[T] : y^t_{p_t}=1\}| \\
&\leq \sum_{t:y^t_{p_t}<1} \Delta\Psi(t) + |\{t : y^t_{p_t}=1\}| && \text{(Using Lemma 13)} \\
&= \Psi(T) - \Psi(0) - \sum_{t:y^t_{p_t}=1} \Delta\Psi(t) - \sum_{r_0=1}^{R_0} \Delta\Psi(r_0) + |\{t : y^t_{p_t}=1\}|.
\end{aligned}$$

The lemma follows since $\Psi(T) \leq 2mk\ln(1+k)$ and $\Psi(0)=0$. The bound on $\Psi(T)$ is because we have m agents each with $|P(i,r_i-1)| \leq k$, and $\phi(1) = 2\ln(1+k)$. ◀

So, it is enough to bound the total cost and drop in potential for time steps when the requested page is fully outside the cache and also to bound the drop in potential when the phase changes.

Proof of Theorem 12. Consider any time step t such that the currently requested page p_t is fully outside the cache, i.e. $y^t_{p_t}=1$. We differentiate such requests into two cases depending on whether the page p_t is in the set $P(i_t, r_{i_t}-1)$ at the time or not. In other words, we do a case analysis on p_t being clean or pseudo-clean. (Recall that only unmarked pages in $P(i_t, r_{i_t}-1)$ contribute to the potential).

Case 1: $p_t \notin P(i_t, r_{i_t}-1)$, i.e, p_t is clean. Since page $p_t \notin U(i,t)$, it does not contribute to the potential before (or after) the request has been served. Consider any page q that is evicted (fractionally) by the algorithm in this step. By Lemma 4, before the eviction, we have $y_q = 0$ or $y_q \geq 1/k$. In either case, by Lemma 11, we have $\Delta\phi(y_q) \geq \Delta y_q$ where Δy_q denotes the change in y-value of page q in this step. Since we have $\sum_q \Delta y_q = y^t_{p_t} = 1$, we have $\Delta\Psi(t) = \sum_q \Delta\phi(y_q) \geq 1$.

Case 2: $p_t \in P(i_t, r_{i_t}-1)$, i.e., p_t is pseudo-clean. By the same reasoning as above, we have $\sum_{q\neq p_t} \Delta\phi(y_q) \geq 1$. However, in this case, page p_t also contributed exactly $2\ln(1+k)$ to the potential at the beginning of the time step. So we have $\Delta\Psi(t) = \sum_{q\neq p_t} \Delta\phi(y_q) - 2\ln(1+k) \geq 1 - 2\ln(1+k)$.

Combining the two cases we get:

$$\sum_{t:y^t_{p_t}=1} (1-\Delta\Psi(t)) \leq 2\ln(1+k)\cdot|\{t : y_{p_t}=1 \text{ and } p_t \in P(i_t, r_{i_t}-1)\}| \tag{3}$$

Consider the end of some phase r_0 and let i be a non-isolated agent. Let r_i denote the current local phase of agent i that must also end along with the global phase r_0. Consider any unmarked page q in $U(i,t)$. As the phase r_0 is ending, page q must be fully evicted and thus contributes $\phi(1)$ to the potential. Once phase r_0 ends and phase r_0+1 begins,

page q no longer contributes to the potential. Note that the set of such unmarked pages is exactly $P(i, r_i - 1) \setminus P(i, r_i)$. Hence, the change in potential at the end of (global) phase r_0 is given by:

$$\Delta\Psi(r_0) = -2\ln(1+k) \cdot \sum_{i \notin \mathcal{I}(r_0)} |P(i, r_i - 1) \setminus P(i, r_i)|$$

Since an agent only changes its local phase when it is non-isolated at the end of a global phase, we have:

$$\sum_{r_0=1}^{R_0} \Delta\Psi(r_0) = -2\ln(1+k) \cdot \sum_{i \in [m]} \sum_{r_i=1}^{R_i} |P(i, r_i - 1) \setminus P(i, r_i)|. \tag{4}$$

The theorem now follows from Lemma 14. ◀

3.4 A Lower Bound on OPT through Dual Fitting

In this section, we give a novel LP-based lower bound on the cost of any offline algorithm for caching with reserves via dual-fitting. This lower bound analysis is new even for the classical unweighted paging setting. Crucially, the lower bound derived here perfectly matches the two terms used to bound the cost of the fractional algorithm $\mathcal{A}$ in Theorem 12, thereby completing the proof of our main result (Theorem 1).

We now describe the linear relaxation of the caching with reserves problem and its dual program. The following notation will be useful. For any page $q \in \mathcal{P}$, let $t_{q,1} < t_{q,2} < \ldots$ denote the time steps when q is requested in the online sequence. For an integer $a \geq 0$, define $I(q, a) = \{t_{q,a} + 1, \ldots, t_{q,a+1} - 1\}$ to be the time interval between the ath and $(a+1)$th requests for q. We define $t_{q,0} \triangleq 0$ for all pages. Let $a(q, t)$ denote the number of requests to page q that have been seen until time t (inclusive). Hence, by definition, for any time t and page $q \in \mathcal{P} \setminus \{p_t\}$, we have $t \in I(q, a(q, t))$. The primal LP has variables $y(q, a) \in [0, 1]$ which denote the portion of page q that is evicted between its ath and $(a+1)$th requests, i.e., $1 - y(q, a)$ portion of q is held in the cache during the time-interval $I(q, a)$. For convenience, we define $n \triangleq |\mathcal{P}|$ and $n_i \triangleq |\mathcal{P}(i)|$ for any $i \in [m]$. The first and second set of primal constraints encode the cache size constraint and the agent-level reserve constraints for all times. The dual LP has variables $\alpha(t)$ and $\beta(t, i)$ corresponding to these primal constraints. We also have dual variables $\gamma(q, a)$ corresponding to the primal constraint encoding $y(q, a) \leq 1$. Besides nonnegativity, the dual has a single constraint for each interval $I(q, a)$. The primal and dual LPs are stated below. We emphasize that we use these linear programs purely for analysis and the algorithm itself does not need to solve any linear program.

Primal LP

$$\min \quad \sum_{q \in \mathcal{P}} \sum_{a \geq 1} y(q, a)$$

subject to:

$$\sum_{q \in \mathcal{P}, q \neq p_t} y(q, a(q, t)) \geq n - k \qquad \forall t \tag{5}$$

$$\sum_{q \in \mathcal{P}(i), q \neq p_t} y(q, a(q, t)) \leq n_i - k_i \qquad \forall t, \forall i \tag{6}$$

$$y(q, a) \leq 1 \qquad \forall q, \forall a \tag{7}$$

$$y \geq 0 \tag{8}$$

Dual LP

$$\max \sum_t (n-k)\alpha(t) - \sum_{t,i} (n_i - k_i)\beta(t, i) - \sum_{q,a} \gamma(q, a)$$

subject to:

$$\sum_{t \in I(q,a)} \big(\alpha(t) - \beta(t, ag(q))\big) - \gamma(q, a) \leq 1 \ \forall q, \forall a \tag{9}$$

$$\alpha, \beta, \gamma \geq 0 \tag{10}$$

Consider time T that marks the end of a global phase R_0 for some integer R_0. Let $\mathrm{OPT} = \mathrm{cost}_{\mathrm{OPT}}(\sigma)$ denote the total cost incurred by an optimal offline algorithm. By weak LP duality, the objective function of the Dual LP yields a lower bound on OPT for any feasible dual solution. We now construct an explicit dual solution (α, β, γ) whose objective value is roughly equal to the total number of clean and pseudo-clean pages seen by the algorithm. See Section 3.1 to recall relevant notation and terminology. The dual solution is updated at the end of each (global) phase in two stages. Updates in the first stage, denoted $\mathsf{update}(r_0, 1)$, are simple and account for stale pages belonging to non-isolated agents that got evicted in the most recent local phase for that agent. Updates in the second stage, denoted $\mathsf{update}(r_0, 2)$, are more involved and account for the pseudo-clean pages of agents who lost their isolated status in the current phase. The dual solution that we maintain will always be approximately feasible up to $O(1)$ factors, so the objective value of this dual solution serves as a lower bound on $\mathrm{OPT}(T)$ within a constant factor. We remark that the assumption that T marks the end of a phase is without loss of generality since it can lead to at most an additive $O(k)$ loss in the lower bound. Formally, we show the following.

▶ **Theorem 15.** *Let T denote the timepoint when global phase R_0 ends, and let $(R_i)_{i\in[m]}$ denote the corresponding local phase counters. Let (α, β, γ) denote the dual solution that is constructed by the end of time T, i.e., the solution that arises from a sequential application of dual updates in the order $\mathsf{update}(1,1), \mathsf{update}(1,2), \mathsf{update}(2,1), \mathsf{update}(2,2), \ldots, \mathsf{update}(R_0,1)$, and $\mathsf{update}(R_0,2)$. We have:*

(a) *The dual solution is approximately feasible: for any i-page q and an integer $a \geq 0$, $\sum_{t\in I(q,a)}(\alpha(t) - \beta(t,i)) - \gamma(q,a) \leq 5$ holds.*

(b) *The dual objective value of (α, β, γ) is:*

$$\begin{aligned}\mathrm{dual}(R_0) &\triangleq \sum_{t=1}^{T}(n-k)\alpha(t) - \sum_{t=1}^{T}\sum_{i\in[m]}(n_i - k_i)\beta(t,i) - \sum_{q\in\mathcal{P}}\sum_{a=1}^{a(q,T)}\gamma(q,a)\\ &= \sum_{i\in[m]}\sum_{r_i=1}^{R_i}|P(i,r_i-1)\setminus P(i,r_i)| + \sum_{r_0=1}^{R_0}\sum_{i\in\mathcal{I}(r_0-1)\setminus\mathcal{I}(r_0)}(|P(i,r_i-1)\cup P(i,r_i)| - k_i).\end{aligned}$$

We first show how Theorem 15 implies that our fractional algorithm $\mathcal{A}$ is $O(\log k)$-competitive.

Proof of Theorem 1. In Theorem 12 we proved the following upper bound on the cost incurred by $\mathcal{A}$ for processing the first T page requests:

$$\mathrm{cost}_{\mathcal{A}}(\sigma) \leq 2\ln(1+k)\cdot\Big(mk + \sum_{t=1}^{T}\mathbb{1}_{p_t \text{ is pseudo-clean}} + \sum_{i\in[m]}\sum_{r_i=1}^{R_i}|P(i,r_i-1)\setminus P(i,r_i)|\Big).$$

Clearly, the second nontrivial term in the above cost-expression matches the first term in the expression for $\mathrm{dual}(R_0)$. Now consider an arbitrary global phase $r_0 \in \{1,\ldots,R_0\}$ and a timestep t in this phase. By definition, a pseudo-clean page p_t is necessarily stale, i.e., $p_t \in P(i_t, r_{i_t} - 1)$ holds, and it must be that agent i_t was isolated at the start of phase r_0 but is non-isolated by time t. Therefore, $i_t \in \mathcal{I}(r_0 - 1) \setminus \mathcal{I}(r_0)$ and the following holds:

$$|P(i,r_i-1)\cup P(i,r_i)| - k_i \geq |P(i,r_i)| - k_i \geq |\{t \in \text{phase } r_0 : p_t \text{ is pseudo-clean}\}|.$$

In the above, the final inequality is because among all pages in $P(i,r_i)$ (w.r.t. the order in which they were marked by $\mathcal{A}$), the first k_i pages are not pseudo-clean. Thus, the first nontrivial term in the cost-expression for $\mathcal{A}$ can be bounded by the second term in $\mathrm{dual}(R_0)$.

Overall, we have shown that $\mathrm{cost}_{\mathcal{A}}(\sigma) \leq 2\ln(1+k)\cdot\big(mk + \mathrm{dual}(R_0)\big)$ holds. Since the dual solution is $O(1)$-feasible, we get that $\mathcal{A}$ is $O(\log k)$-competitive. ◀

We now furnish the details of our dual updates. Initially, all our dual variables $\{\alpha(t)\}, \{\beta(i,t)\}, \{\gamma(q,a)\}$ with $t \in [T], i \in [m], q \in \mathcal{P}, a \in [a(q,T)]$ are set to zero. We assume that the dual updates are applied in the sequence given in Theorem 15. That is, the set of updates in $\{\mathsf{update}(r_0, s)\}_{r_0 \in [R_0], s \in \{1,2\}}$ are applied in increasing order of r_0 and within each phase first stage updates are applied first. With a slight abuse of notation, let $\text{dual}(r_0, s)$ denote the objective value of the dual solution right after updates until $\mathsf{update}(r_0, s)$ (inclusive) have been applied where $r_0 \in [R_0], s \in \{1, 2\}$. Note that $\text{dual}(R_0) = \text{dual}(R_0, 2)$. We also define $\text{dual}(0, 1) = \text{dual}(0, 2) := 0$. Throughout our updates, we ensure that the dual objective value never decreases, i.e., $0 \leq \text{dual}(1,1) \leq \text{dual}(1,2) \leq \cdots \leq \text{dual}(R_0, 1) \leq \text{dual}(R_0, 2)$ holds. We remark that β variables may decrease and this only happens in the second stage; However, the α and γ variables never decrease.

In Section 3.4, we describe the first stage of updates and show that the gain in the dual objective corresponds to the first term in Theorem 15(b). In Section 3.4, we describe the second stage of updates and show that the gain in the dual objective corresponds to the second term in Theorem 15(b). Lastly, in Section 3.4, we show that the dual solution that we maintain is always feasible up to constant factors and thus complete the proof of Theorem 15.

First Stage of Dual Updates

Fix a phase $r_0 \in [R_0]$ and consider the set $\mathcal{I}(r_0) \subsetneq [m]$ of agents that are designated as isolated at the end of phase r_0. Let $C(r_0)$ denote the set of timesteps t (in this phase) when the following two conditions hold: (a) $y^t_{p_t} = 1$ in the fractional algorithm $\mathcal{A}$ just before p_t is requested; and (b) i_t is not isolated at time t. Define $\ell^{(r_0)} \triangleq |C(r_0)|$. It is not hard to see that the following is an equivalent expression for $\ell^{(r_0)}$.

$$\ell^{(r_0)} := \sum_{i \notin \mathcal{I}(r_0-1) \cup \mathcal{I}(r_0)} |P(i, r_i) \setminus P(i, r_i - 1)| + \sum_{i \in \mathcal{I}(r_0-1) \setminus \mathcal{I}(r_0)} (|P(i, r_i)| - k_i). \tag{11}$$

Observe that for agents who are non-isolated both at the start and end of phase r_0, $\ell^{(r_0)}$ counts all their clean pages. However, for agents who were isolated at the start of this phase but are no longer isolated by the end, $\ell^{(r_0)}$ only counts clean and pseudo-clean pages that are requested *after* the agent has become non-isolated. Roughly speaking, the motivation for the definition of $\ell^{(r_0)}$ comes from the intuition that an offline algorithm should incur, on an average, a cost of $\Omega(\ell^{(r_0)})$ to serve page requests in phase r_0.

Description of $\mathsf{update}(r_0, 1)$. For each time $t \in C(r_0)$, we separately apply the following updates. First, we increase $\alpha(t)$ by $1/\ell^{(r_0)}$. Next, we increase $\beta(t, i)$ by $1/\ell^{(r_0)}$ for every agent $i \in \mathcal{I}(r_0)$. Last, for each agent $i \notin \mathcal{I}(r_0)$, we increase $\gamma(q, a(q,t))$ by $1/\ell^{(r_0)}$ for every i-page $q \in \mathcal{P}(i) \setminus (P(i, r_i - 1) \cup P(i, r_i))$.

It will be clear from the description of our updates that the α and β variables that were modified in $\mathsf{update}(r_0, 1)$ were previously at 0. However, no such guarantee holds for the affected γ variables. We also remark that the same $\gamma(q, a)$ variable can be increased more than once during $\mathsf{update}(r_0, 1)$; this happens when there are multiple times $t \in C(r_0)$ with the same $a(q, t)$ value. In fact, since the $\gamma(q, a)$ variables arise from intervals $I(q, a)$ that can possibly span across multiple phases, it is possible that the same γ variable is increased by different $1/\ell^{(r_0)}$ amounts across different $\mathsf{update}(r_0, 1)$ steps.

For convenience, let $t \in$ phase r_0 be a shorthand for all timepoints in phase r_0. The following result will be useful to us.

▶ **Lemma 16.** *Let (α, β, γ) denote the dual solution that is obtained right after* update$(r_0, 1)$ *has been applied. We have: (a) $\sum_{t \in phase\ r_0} \alpha(t) = 1$; and (b) $\sum_{t \in phase\ r_0} \beta(t,i) = 1$ for any agent $i \in \mathcal{I}(r_0)$.*

Proof. Follows directly from our choice of $\ell^{(r_0)} = |C(r_0)|$. ◀

Our key technical result in this section is that the gain in the dual objective value that comes from update$(r_0, 1)$ is equal to the number of stale pages owned by non-isolated agents that were not requested in their most recent local phases. For convenience, we use the prefix Δ to refer to changes that occured during update$(r_0, 1)$.

▶ **Lemma 17.** *We have* $\Delta\text{dual}(r_0, 1) = \sum_{i \notin \mathcal{I}(r_0)} |P(i, r_i - 1) \setminus P(i, r_i)|$, *where* $\Delta\text{dual}(r_0, 1) \triangleq \text{dual}(r_0, 1) - \text{dual}(r_0 - 1, 2)$ *is the change in the dual objective after* update$(r_0, 1)$.

Proof. Since the only affected $\alpha(t)$ and $\beta(t,i)$ variables have are those with $t \in C(r_0)$ and they are all increased by exactly $1/|C(r_0)|$, we get:

$$\begin{aligned}
\Delta\text{dual}(r_0, 1) &= \sum_t (n-k)\Delta\alpha(t) - \sum_{t,i} (n_i - k_i)\Delta\beta(t,i) - \sum_{q,a} \Delta\gamma(q,a) \\
&= (n-k) - \sum_{i \in \mathcal{I}(r_0)} (n_i - k_i) - \sum_{q,a} \Delta\gamma(q,a) \\
&= \Big(\sum_{i \notin \mathcal{I}(r_0)} n_i \Big) - k + \Big(\sum_{i \in \mathcal{I}(r_0)} k_i \Big) - \sum_{q,a} \Delta\gamma(q,a)
\end{aligned}$$

Now observe that for every $t \in C(r_0)$ and $i \notin \mathcal{I}(r_0)$, the update$(r_0, 1)$ step increases the $\gamma(q,a)$ variable corresponding to exactly $n_i - |P(i, r_i - 1) \cup P(i, r_i)|$ unique i-pages, each by an amount $1/\ell^{(r_0)}$. So we have $\sum_{q,a} \Delta\gamma(q,a) = (1/\ell^{(r_0)}) \cdot \sum_{t \in C(r_0)} \sum_{i \notin \mathcal{I}(r_0)} (n_i - |P(i, r_i - 1) \cup P(i, r_i)|) = \sum_{i \notin \mathcal{I}(r_0)} (n_i - |P(i, r_i - 1) \cup P(i, r_i)|)$. Substituting back into the equation above, we get:

$$\begin{aligned}
\Delta\text{dual}(r_0, 1) &= \Big(\sum_{i \notin \mathcal{I}(r_0)} n_i \Big) - k + \Big(\sum_{i \in \mathcal{I}(r_0)} k_i \Big) - \sum_{i \notin \mathcal{I}(r_0)} (n_i - |P(i, r_i - 1) \cup P(i, r_i)|) \\
&= \Big(-k + \sum_{i \in \mathcal{I}(r_0)} k_i + \sum_{i \notin \mathcal{I}(r_0)} |P(i, r_i)| \Big) + \sum_{i \notin \mathcal{I}(r_0)} |P(i, r_i - 1) \setminus P(i, r_i)|
\end{aligned}$$

The lemma follows from observing that the above group of terms within the parentheses is 0: this is because the cache (of size k) at the end of phase r_0 consists exactly k_i (fractional) pages for isolated agents $i \in \mathcal{I}(r_0)$ and exactly $|P(i, r_i)|$ (integral) pages for non-isolated agents $i \notin \mathcal{I}(r_0)$. ◀

Second Stage of Dual Updates

We now describe the second stage of dual updates that are carried out at the end of each phase $r_0 \in [R_0]$. Unlike the first stage, where we only increased the α and β variables of time steps in phase r_0, in the second stage we decrease the β variables of time steps in the previous phase $r_0 - 1$.

Description of update$(r_0, 2)$. These dual updates correspond to agents that were isolated at the end of phase $r_0 - 1$ but are no longer isolated at the end of phase r_0. Consider an agent $i \in \mathcal{I}(r_0 - 1) \setminus \mathcal{I}(r_0)$. For every time t in phase $r_0 - 1$ with $\beta(t,i) > 0$ (i.e., $t \in C(r_0 - 1)$), we do the following: we increase $\gamma(q, a(q,t))$ by $\beta(t,i)$ for all i-pages $q \in \mathcal{P}(i) \setminus \big(P(i, r_i - 1) \cup P(i, r_i)\big)$ followed by resetting $\beta(t,i)$ to 0.

For clarity, we note the following: (i) resetting $\beta(t,i)$ to zero is the only dual update when a variable is *decreased*; (ii) the $\beta(t,i)$ updates are applied to timepoints in phase r_0-1 (i.e., the previous phase); and (iii) the $\gamma(q,a(q,t))$ variables that we updated above were unchanged while applying $\mathsf{update}(r_0-1,1)$ at the end of phase r_0-1 because their owner i was designated as isolated at that time. The reason for decreasing the $\beta(t,i)$ variables is that it leads to an increase in the dual objective, which will be needed to pay for costs associated with pseudo-clean pages. We formalize this in the following lemma.

▶ **Lemma 18.** *We have* $\Delta\mathrm{dual}(r_0,2)=\sum_{i\in\mathcal{I}(r_0-1)\setminus\mathcal{I}(r_0)}\left(|P(i,r_i-1)\cup P(i,r_i)|-k_i\right)$ *where* $\Delta\mathrm{dual}(r_0,2)\triangleq\mathrm{dual}(r_0,2)-\mathrm{dual}(r_0,1)$ *is the change in the dual objective after* $\mathsf{update}(r_0,2)$.

Proof. Fix an agent $i\in\mathcal{I}(r_0-1)\setminus\mathcal{I}(r_0)$. In Lemma 16 we showed that after $\mathsf{update}(r_0-1,1)$, $\sum_{t\in\text{phase }r_0-1}\beta(t,i)=1$ and $\beta(t,i)\in\{0,1/\ell^{(r_0-1)}\}$. Consider any time t in phase r_0-1 with $\beta(t,i)>0$. By the definition of $\mathsf{update}(r_0,2)$, we decrease $\beta(t,i)$ by $1/\ell^{(r_0-1)}$ while increasing $\gamma(q,a(q,t))$ by the same amount for all pages $q\in\mathcal{P}(i)\setminus\big(P(i,r_i-1)\cup P(i,r_i)\big)$. Recalling the coefficients in the dual objective function, we see that the updates corresponding to agent i increases the dual objective by exactly:

$$(n_i-k_i)-|\mathcal{P}(i)\setminus(P(i,r_i)\cup P(i,r_i-1))|=|(P(i,r_i)\cup P(i,r_i-1))|-k_i.$$ ◀

Approximate Dual Feasibility

We finish this section by showing that the dual solution is always approximately feasible.

▶ **Lemma 19.** *Let* (α,β,γ) *denote the dual solution that is obtained right after* $\mathsf{update}(r_0,s)$ *has been applied for some* $r_0\in[R_0]$ *and* $s\in\{0,1\}$. *For any* i*-page* q *and an integer* $a\geq 1$ *satisfying* $a\leq a(q,T)$, *we have* $\sum_{t\in I(q,a)}(\alpha(t)-\beta(t,i))-\gamma(q,a)\leq 5$.

Proof. First of all, for the purposes of this proof, the specific values of r_0 and s are irrelevant, so we ignore them. Fix some i-page q and an integer a satisfying $a\leq a(q,T)$. Recall that $I(q,a)=\{t_{q,a}+1,\ldots,t_{q,a+1}-1\}$, where $t_{q,a'}$ denotes the time when q is requested for the a'th time; We redefine $t_{q,a+1}$ to be $T+1$ if $t_{q,a+1}>T$ holds.

The lemma holds trivially if $I(q,a)$ is empty, so we assume otherwise. Let $r_0^b,r_0^e\in[R_0]$ denote the global phases that contain timesteps $t_{q,a}+1$ and $t_{q,a+1}-1$, respectively. Clearly, $r_0^b\leq r_0^e$. Another easy case of the lemma is when $r_0^e\leq r_0^b+1$ holds. The desired conclusion follows easily because all the dual variables are nonnegative and the sum of all $\alpha(t)$ variables in any phase is at most 1 (by Lemma 16). Formally,

$$\sum_{t\in I(q,a)}\big(\alpha(t)-\beta(t,i)\big)-\gamma(q,a)\leq\sum_{t\in\text{phase }r_0^b}\alpha(t)+\sum_{t\in\text{phase }r_0^e}\alpha(t)\leq 2.$$

Now suppose that $r_0^b+2\leq r_0^e$ holds. Define $Z:=\{r_0^b+1,\ldots,r_0^e-1\}$. Repeating the above calculation, we get:

$$\sum_{t\in I(q,a)}(\alpha(t)-\beta(t,i))-\gamma(q,a)\leq 2+\left\{\sum_{r_0\in Z}\sum_{t\in\text{phase }r_0}\big(\alpha(t)-\beta(t,i)\big)\right\}-\gamma(q,a),$$

so the crux of the lemma is to show that the sum of $\delta(t)\triangleq\alpha(t)-\beta(t,i)$ over timepoints spanning phases in Z is not much larger than $\gamma(q,a)$. For a phase $r_0\in Z$, we overload the notation $\delta(r_0)$ to mean $\sum_{t\in\text{phase }r_0}\delta(t)$. Note that by nonnegativity of β variables and Lemma 17, $\delta(r_0)\leq 1$ for every $r_0\in Z$. We do a case analysis on phase $r_0\in Z$ to get a better handle on the changes that happens during our dual update procedures.

(a) Suppose that $i \in \mathcal{I}(r_0) \cap \mathcal{I}(r_0+1)$ holds. Since i is isolated by the end of phase r_0, we know that any increase in $\alpha(t)$ (as part of $\mathsf{update}(r_0, 1)$) for some $t \in C(r_0)$ is accompanied with the same increase in $\beta(t,i)$. Since $i \in \mathcal{I}(r_0+1)$ holds, $\mathsf{update}(r_0+1, 2)$ does not decrease/reset any of the $\{\beta(t,i)\}_{t \in \text{phase } r_0}$ variables to 0. Thus, $\delta(r_0) = 0$ holds. Note that there can be an arbitrary number of phases r_0 that fall under this case, but this is not a problem for us since $\delta(r_0) = 0$.

(b) Suppose that $i \in \mathcal{I}(r_0) \setminus \mathcal{I}(r_0+1)$ and $q \in P(i, r_i - 1) \cup P(i, r_i)$ hold. We rely on the trivial bound $\delta(r_0) \leq 1$ for this case. Since i is isolated by the end of phase r_0 but is non-isolated by the end of phase r_0+1, the local phase counter r_i goes up by 1 at the end of phase r_0+1, and subsequently the $P(i,\cdot)$ set gets updated with some new collection of marked i-pages. By definition of $I(q,a)$, there are no page-requests for q during any of the phases in Z. Thus, there can be at most 2 local phase increments for agent i before q gets dropped from the $P(i,\cdot)$ set; By the design of $\mathcal{A}$, page q cannot enter any of the future $P(i, r_i)$ until the next time it is requested, which does not happen during any of the phases in Z.

(c) Suppose that $i \in \mathcal{I}(r_0) \setminus \mathcal{I}(r_0+1)$ and $q \notin P(i, r_i - 1) \cup P(i, r_i)$ hold. Similar to case (a) above, we know that any increase in $\alpha(t)$ (as part of $\mathsf{update}(r_0, 1)$) for some $t \in C(r_0)$ is accompanied with the same increase in $\beta(t,i)$. Now, although $\mathsf{update}(r_0+1, 2)$ decreases/resets all the $\{\beta(t,i)\}_{t \in C(r_0)}$ variables to 0, it also increases $\gamma(q,a)$ by the same amount since $q \notin P(i, r_i - 1) \cup P(i, r_i)$. Thus, $\sum_{t \in \text{phase } r_0} \alpha(t)$ equals the increase in $\gamma(q,a)$ due to $\mathsf{update}(r_0+1, 2)$. So, the difference is essentially 0.

(d) Suppose $i \notin \mathcal{I}(r_0)$ and $q \in P(i, r_i - 1) \cup P(i, r_i)$ hold. We rely on the trivial bound $\delta(r_0) \leq 1$ for this case. Since i is non-isolated by the end of phase r_0, the local phase counter r_i goes up by 1 at the end of phase r_0. Repeating the argument from case (c), there can be at most 1 more local phase increment for agent i before q gets dropped from the $P(i,\cdot)$ set for the rest of the phases in Z.

(e) Suppose $i \notin \mathcal{I}(r_0)$ and $q \notin P(i, r_i - 1) \cup P(i, r_i)$ hold. Since i is non-isolated by the end of phase r_0 and q is not in $P(i, r_i - 1) \cup P(i, r_i)$, we know that any increase in $\alpha(t)$ (as part of $\mathsf{update}(r_0, 1)$) for some $t \in C(r_0)$ is accompanied with the same increase in $\gamma(q,a)$. Thus, $\sum_{t \in \text{phase } r_0} \alpha(t)$ equals the increase in $\gamma(q,a)$ due to $\mathsf{update}(r_0, 1)$. So, the difference is essentially 0.

From the above case analysis, it follows that $\{\sum_{r_0 \in Z} \sum_{t \in \text{phase } r_0} \big(\alpha(t) - \beta(t,i)\big)\} - \gamma(q,a)$ is bounded by the number of phases $r_0 \in Z$ for which case (b) or (d) hold. Since we argued that there can be at most 3 such occurences, $\sum_{t \in I(q,a)} (\alpha(t) - \beta(t,i)) - \gamma(q,a) \leq 2 + 3 = 5$ holds. ◀

We now prove the main theorem in this section by combining the above lemmas.

Proof of Theorem 15. The first part of the theorem follows from Lemma 19. The second part follows directly from Lemmas 17 and 18 since we have $\text{dual}(R_0) = \sum_{r_0=1}^{R_0} \big(\Delta\text{dual}(r_0, 1) + \Delta\text{dual}(r_0, 2)\big)$ ◀

4 Rounding

In this section we show how to convert the fractional Algorithm 1 into a randomized integral online algorithm for *Caching with Reserves*, each step of which runs in polynomial time.

The algorithm maintains a uniform distribution on $N = k^3$ valid cache states. In each step, these states are updated based on the actions of the fractional algorithm. The randomized algorithm selects one of these states uniformly at random in the beginning, and then follows it throughout the run.

Initially, all N cache states in the distribution are the same as the initial cache state in Algorithm 1. Given the fractional algorithm values x_p^t after each page request, the distribution is updated in two steps. First, we produce a discretized version of these fractions, $\tilde{x}_p^t$, which are also a feasible fractional solution. This is based on the technique in [2]. Second, we update the N cache states so that all of them remain valid, and for each page p, exactly $N \cdot \tilde{x}_p^t$ of the states contain p. This is based on the technique in [10]. We note that the rounding procedure can be done online, as it does not need the knowledge of any future page requests.

4.1 Discretization Procedure

In this subsection, we explain how to perform the first step: discretizing the fractional algorithm's values x_p^t into $\tilde{x}_p^t$ which are multiples of $\frac{1}{N}$. Our procedure is quite simple: we iterate over the pages in any order π that arranges all pages belonging to the same agent consecutively (i.e., order the agents arbitrarily and order each agent's pages arbitrarily, but do not interleave pages from different agents). Then for $i \in [|\mathcal{P}|]$, set:

$$\tilde{x}_{\pi(i)}^t \triangleq \left\lfloor \sum_{j=1}^{i} x_{\pi(j)}^t \right\rfloor_{1/N} - \left\lfloor \sum_{j=1}^{i-1} x_{\pi(j)}^t \right\rfloor_{1/N}$$

where $\lfloor a \rfloor_b$ denotes rounding a down to the nearest multiple of b; formally: $\lfloor a \rfloor_b \triangleq b \lfloor a/b \rfloor$.

▶ **Lemma 20.** *Discretization satisfies the following guarantees:*

1. $\tilde{x}_p^t$ *is a multiple of* $1/N$
2. $\left|\tilde{x}_p^t - x_p^t\right| < 1/N$
3. *for each agent* i, $\left|\sum_{p\in\mathcal{P}(i)} \tilde{x}_p^t - \sum_{p\in\mathcal{P}(i)} x_p^t\right| < 1/N$
4. *if* $x_p^t \in \{0,1\}$, *then* $\tilde{x}_p^t = x_p^t$

Due to space constraints, the proof of the above lemma is deferred to the full version [11].

▶ **Corollary 21.** *If* $\{x_p^t\}$ *satisfy total cache capacity* ($\sum_{p\in\mathcal{P}} x_p^t \le k$) *and reserve requirements* ($\sum_{p\in\mathcal{P}(i)} x_p^t \ge k_i$), *then so do* $\{\tilde{x}_p^t\}$.

Proof. The total cache capacity constraint continues to hold due to a telescoping argument:

$$\begin{aligned}\sum_{p\in\mathcal{P}} \tilde{x}_p^t &= \sum_{i\in[|\mathcal{P}|]} \tilde{x}_{\pi(i)}^t = \sum_{i\in[|\mathcal{P}|]} \left[\left\lfloor \sum_{j=1}^{i} x_{\pi(j)}^t \right\rfloor_{1/N} - \left\lfloor \sum_{j=1}^{i-1} x_{\pi(j)}^t \right\rfloor_{1/N} \right] \\ &= \left\lfloor \sum_{j=1}^{|\mathcal{P}|} x_{\pi(j)}^t \right\rfloor_{1/N} \le \sum_{j=1}^{|\mathcal{P}|} x_{\pi(j)}^t \le k\end{aligned}$$

Next, we will prove that reserve cache sizes are satisfied. For the sake of contradiction, suppose that for some agent i, $\sum_{p\in\mathcal{P}(i)} \tilde{x}_p^t < k_i$. Since the right-hand side of this inequality is an integer and therefore a multiple of $1/N$, the left-hand side, which is also a multiple of $1/N$ due to being a sum of multiples of $1/N$ (by Lemma 20's first guarantee), must be at least a full multiple of $1/N$ less than the right-hand side: $\sum_{p\in\mathcal{P}(i)} \tilde{x}_p^t \le k_i - 1/N$. But this contradicts Lemma 20's third guarantee, $\left|\sum_{p\in\mathcal{P}(i)} \tilde{x}_p^t - \sum_{p\in\mathcal{P}(i)} x_p^t\right| < 1/N$. Therefore for all agents i, $\sum_{p\in\mathcal{P}(i)} \tilde{x}_p^t \ge k_i$, completing the proof. ◀

▶ **Corollary 22.** $\sum_p \left|\tilde{x}_p^t - x_p^t\right| \le k^2/N$

Proof. Without loss of generality, there are at most k agents since we can combine all agents that do not have any reserve. Each agent i has at most k fractional pages by the algorithm (the i-pages that were in cache when i's local phase began). The x value of each fractional page is distorted by at most $1/N$ by Lemma 20's second guarantee, while not being distorted for non-fractional pages by the fourth guarantee. This completes the proof. ◀

▶ **Lemma 23.** *Let x_p^t and x_p^{t+1} be the amounts of each page p in cache in the fractional algorithm for two consecutive time steps, and $\tilde{x}_p^t$ and $\tilde{x}_p^{t+1}$ be the corresponding discretized values. Then the cost of cache update from $\tilde{x}^t$ to $\tilde{x}^{t+1}$ (call it $\tilde{c}$) is at most twice the cost of cache update from x^t to x^{t+1} (call it c).*

Proof. Lemma 10 implies that either $c = 0$ (i.e., the requested page was already in cache and there is no change to the cache state), or $c \ge 1/k$. In the first case, there is no change to the discretized cache state either, so $\tilde{c} = 0$. So we focus on the second case. By triangle inequality, for any page p,

$$|\tilde{x}_p^t - \tilde{x}_p^{t+1}| \le |\tilde{x}_p^t - x_p^t| + |x_p^t - x_p^{t+1}| + |x_p^{t+1} - \tilde{x}_p^{t+1}|.$$

We note that since cost is incurred for adding pages to cache, and both the original fractional solution and the discretized one add as much page mass to cache as they evict, $2c = \sum_p |x_p^t - x_p^{t+1}|$, and similarly for $\tilde{c}$. Summing the above inequality over p, we get

$$2\tilde{c} = \sum_p |\tilde{x}_p^t - \tilde{x}_p^{t+1}| \le \sum_p |x_p^t - x_p^{t+1}| + 2k^2/N = 2c + 2/k \le 4c,$$

where we used $\sum_p (|\tilde{x}_p^t - x_p^t| + |x_p^{t+1} - \tilde{x}_p^{t+1}|) \le 2k^2/N$ by Corollary 22, then $N = k^3$, then $c \ge 1/k$. ◀

4.2 Updating the Distribution of Cache States

In this subsection, we explain how to perform the second step: updating the N cache states. We would like (i) all cache states to be valid, (ii) exactly $N \cdot \tilde{x}_p^t$ (integral due to our discretization step) of the states to contain page p, and (iii) to not use too many evictions.

Formally, let $\mathcal{X}$ be a set of N cache states with k pages each, which corresponds to the discretized values $\tilde{x}_p^t$ for time step t. Given the discretized values $\tilde{x}_p^{t+1}$ for time step $t+1$, we show how to transform $\mathcal{X}$ into $\mathcal{X}'$, in which each page p appears in exactly $N \cdot \tilde{x}_p^{t+1}$ cache states and such that each cache state satisfies all the reserve requirements.

Let P be a multiset of pages whose fraction in the cache increased from time t to $t+1$, with each page p appearing $\max(0, (\tilde{x}_p^{t+1} - \tilde{x}_p^t)N)$ times. Let Q be an analagous multiset for decreases, with each page appearing $\max(0, (\tilde{x}_p^t - \tilde{x}_p^{t+1})N)$ times. Since the total amount of pages in the cache is unchanged, $|P| = |Q|$. We find a matching between pages in P and Q and use it to transform $\mathcal{X}$ into $\mathcal{X}'$ gradually, one pair at a time. The matching is constructed as follows. First, any pages from P and Q that belong to the same agent are matched up. Then, the remaining pages in P and Q are matched up arbitrarily.

▶ **Lemma 24.** *Let $(p_1, q_1), (p_2, q_2), \ldots$ be the matching between P and Q described above. Then for any j, the fractional solution that adds $1/N$ fraction of pages $p_1, \ldots, p_j$ to $\tilde{x}^t$ and removes $1/N$ fraction of pages $q_1, \ldots, q_j$ from it satisfies all the reserve requirements.*

We defer the proof of the above lemma to the full version [11].

We now show how to modify $\mathcal{X}$ with the next pair (p, q) from the matching. This follows the procedure in [10], with the difference that we work on a limited number of N sets, and the amount of increase in p and decrease in q is fixed at $1/N$.

Let $\mathcal{X}$ be the current set of cache states (possibly modified by the previous page pairs). If there is a cache state $S \in \mathcal{X}$ such that $p \notin S$ and $q \in S$, add p to S and remove q from S. Otherwise, find cache states $S \in \mathcal{X}$ and $T \in \mathcal{X}$ with $p \notin S$ and $q \in T$, add p to S and remove q from T. Next, move some page $r \in S \setminus T$ from S to T to adjust the set sizes back to k.

At this point, each page is in the correct number of cache states. However, reserve requirements could be violated by one page for $ag(q)$ or $ag(r)$ in the cache states from which the corresponding pages were removed. In such a case, suppose the requirement is violated for agent i in a cache state $V \in \mathcal{X}$. Since, by Lemma 24, each reserve requirement is satisfied on average, there must be another set $W \in \mathcal{X}$ which has strictly more than k_i pages belonging to agent i. We move one such page from W to V. Now V has $k + 1$ pages, so there must be an agent j which has more than k_j pages in V. We move one of j's pages from V to W to restore the sizes. This completes the update, resulting in new valid sets corresponding to the fractions $\tilde{x}^{t+1}$.

We conclude by bounding the cost of update to $\mathcal{X}$, and thus the expected cost of the randomized algorithm, relative to the cost of the fractional algorithm.

▶ **Lemma 25.** *The cost of update to $\mathcal{X}$ is at most 6 times the cost of fractional cache update from $\tilde{x}^t$ to $\tilde{x}^{t+1}$.*

Proof. Each pair (p, q) in the matching corresponds to a cost of $1/N$ incurred by the discretized fractional solution. In the updates to sets in $\mathcal{X}$, each time a page is removed from one of the cache states incurs a cost of $1/N$ to the randomized algorithm. At most, the following six removals are done: remove q from T; remove r from S; two pages each are swapped to fix the reserve requirements for $ag(q)$ and $ag(r)$. ◀

The proof of Theorem 2 follows by combining Lemmas 23 and 25.

References

1 Dimitris Achlioptas, Marek Chrobak, and John Noga. Competitive Analysis of Randomized Paging Algorithms. *Theoretical Computer Science*, 234(1):203–218, 2000. doi:10.1016/S0304-3975(98)00116-9.

2 Anna Adamaszek, Artur Czumaj, Matthias Englert, and Harald Räcke. An $O(\log k)$-Competitive Algorithm for Generalized Caching. *ACM Transactions on Algorithms*, 15(1):6:1–6:18, 2018. doi:10.1145/3280826.

3 Kunal Agrawal, Michael A. Bender, Rathish Das, William Kuszmaul, Enoch Peserico, and Michele Scquizzato. Tight Bounds for Parallel Paging and Green Paging. In *Proceedings of the 32nd Symposium on Discrete Algorithms*, pages 3022–3041, 2021. doi:10.1137/1.9781611976465.180.

4 Nikhil Bansal, Niv Buchbinder, and Joseph Seffi Naor. A Simple Analysis for Randomized Online Weighted Paging. *Unpublished Manuscript*, 2010.

5 Nikhil Bansal, Niv Buchbinder, and Joseph (Seffi) Naor. Towards the Randomized k-Server Conjecture: A Primal-Dual Approach: (Extended Abstract). In *Proceedings of the 21st Symposium on Discrete Algorithms*, pages 40–55, 2010. doi:10.1137/1.9781611973075.5.

6 Nikhil Bansal, Christian Coester, Ravi Kumar, Manish Purohit, and Erik Vee. Learning-Augmented Weighted Paging. In *Proceedings of the 33rd Symposium on Discrete Algorithms*, pages 67–89, 2022. doi:10.1137/1.9781611977073.4.

7 Jichuan Chang and Gurindar S Sohi. Cooperative Cache Partitioning for Chip Multiprocessors. In *Proceedings of the 21st ACM International Conference on Supercomputing*, pages 242–252, 2007. doi:10.1145/1274971.1275005.

8 Ashish Chiplunkar, Monika Henzinger, Sagar Sudhir Kale, and Maximilian Vötsch. Online Min-Max Paging. In *Proceedings of the 34th Symposium on Discrete Algorithms*, pages 1545–1565, 2023. doi:10.1137/1.9781611977554.ch57.

9 Amos Fiat, Richard M. Karp, Michael Luby, Lyle A. McGeoch, Daniel D. Sleator, and Neal E. Young. Competitive Paging Algorithms. *Journal of Algorithms*, 12(4):685–699, 1991. doi:10.1016/0196-6774(91)90041-V.

10 Sharat Ibrahimpur, Manish Purohit, Zoya Svitkina, Erik Vee, and Joshua R. Wang. Caching with Reserves. In *Proceedings of the 25th International Conference on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, volume 245, pages 52:1–52:16, 2022. doi:10.4230/LIPIcs.APPROX/RANDOM.2022.52.

11 Sharat Ibrahimpur, Manish Purohit, Zoya Svitkina, Erik Vee, and Joshua R. Wang. Efficient Caching with Reserves via Marking. *CoRR*, abs/2305.02508, 2023. doi:10.48550/arXiv.2305.02508.

12 Wu Kan, Tu Kaiwei, Patel Yuvraj, Sen Rathijit, Park Kwanghyun, Arpaci-Dusseau Andrea, and Remzi Arpaci-Dusseau. NyxCache: Flexible and Efficient Multi-tenant Persistent Memory Caching. In *Proceedings of the 20th USENIX Conference on File and Storage Technologies (FAST)*, pages 1–16, 2022. URL: https://www.usenix.org/conference/fast22/presentation/wu.

13 Mayuresh Kunjir, Brandon Fain, Kamesh Munagala, and Shivnath Babu. ROBUS: Fair Cache Allocation for Data-parallel Workloads. In *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD)*, pages 219–234, 2017. doi:10.1145/3035918.3064018.

14 Lyle A. McGeoch and Daniel D. Sleator. A Strongly Competitive Randomized Paging Algorithm. *Algorithmica*, 6:816–825, 1991. doi:10.1007/BF01759073.

15 Qifan Pu, Haoyuan Li, Matei Zaharia, Ali Ghodsi, and Ion Stoica. FairRide: Near-Optimal, Fair Cache Sharing. In *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 393–406, 2016. URL: https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/pu.

16 Harold S. Stone, John Turek, and Joel L. Wolf. Optimal Partitioning of Cache Memory. *IEEE Transactions on Computers*, 41(9):1054–1068, 1992. doi:10.1109/12.165388.

17 G Edward Suh, Larry Rudolph, and Srinivas Devadas. Dynamic Partitioning of Shared Cache Memory. *The Journal of Supercomputing*, 28(1):7–26, 2004. doi:10.1023/B:SUPE.0000014800.27383.8f.

18 Yinghao Yu, Wei Wang, Jun Zhang, and Khaled Ben Letaief. LACS: Load-Aware Cache Sharing with Isolation Guarantee. In *Proceedings of the 39th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 207–217. IEEE, 2019. doi:10.1109/ICDCS.2019.00029.

Rerouting Planar Curves and Disjoint Paths

Takehiro Ito
Graduate School of Information Sciences, Tohoku University, Sendai, Japan

Yuni Iwamasa
Graduate School of Informatics, Kyoto University, Japan

Naonori Kakimura
Faculty of Science and Technology, Keio University, Yokohama, Japan

Yusuke Kobayashi
Research Institute for Mathematical Sciences, Kyoto University, Japan

Shun-ichi Maezawa
Department of Mathematics, Tokyo University of Science, Japan

Yuta Nozaki
Faculty of Environment and Information Sciences, Yokohama National University, Japan
SKCM^2, Hiroshima University, Japan

Yoshio Okamoto
Graduate School of Informatics and Engineering, The University of Electro-Communications, Tokyo, Japan

Kenta Ozeki
Faculty of Environment and Information Sciences, Yokohama National University, Japan

Abstract

In this paper, we consider a transformation of k disjoint paths in a graph. For a graph and a pair of k disjoint paths $\mathcal{P}$ and $\mathcal{Q}$ connecting the same set of terminal pairs, we aim to determine whether $\mathcal{P}$ can be transformed to $\mathcal{Q}$ by repeatedly replacing one path with another path so that the intermediates are also k disjoint paths. The problem is called Disjoint Paths Reconfiguration. We first show that Disjoint Paths Reconfiguration is PSPACE-complete even when $k = 2$. On the other hand, we prove that, when the graph is embedded on a plane and all paths in $\mathcal{P}$ and $\mathcal{Q}$ connect the boundaries of two faces, Disjoint Paths Reconfiguration can be solved in polynomial time. The algorithm is based on a topological characterization for rerouting curves on a plane using the algebraic intersection number. We also consider a transformation of disjoint s-t paths as a variant. We show that the disjoint s-t paths reconfiguration problem in planar graphs can be determined in polynomial time, while the problem is PSPACE-complete in general.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Disjoint paths, combinatorial reconfiguration, planar graphs

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.81

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2210.11778

Funding *Takehiro Ito*: JSPS KAKENHI Grant Numbers JP18H04091, JP19K11814, JP20H05793.
Yuni Iwamasa: JSPS KAKENHI Grant Numbers JP20K23323, JP20H05795, JP22K17854.
Naonori Kakimura: JSPS KAKENHI Grant Numbers JP20H05795, JP21H03397, JP22H05001.
Yusuke Kobayashi: JSPS KAKENHI Grant Numbers JP20K11692, JP20H05795, JP22H05001.
Shun-ichi Maezawa: JSPS KAKENHI Grant Numbers JP20H05795, JP22K13956.
Yuta Nozaki: JSPS KAKENHI Grant Numbers JP20H05795, JP20K14317, JP23K12974.
Yoshio Okamoto: JSPS KAKENHI Grant Numbers JP20H05795, JP20K11670, JP23K10982.
Kenta Ozeki: JSPS KAKENHI Grant Numbers JP18K03391, JP19H01803, JP20H05795, 23K03195.

Acknowledgements We thank Naoyuki Kamiyama for the discussion and the anonymous referees for their helpful comments.

50th International Colloquium on Automata, Languages, and Programming (ICALP 2023).
Editors: Kousha Etessami, Uriel Feige, and Gabriele Puppis; Article No. 81; pp. 81:1–81:19

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

1.1 Disjoint Paths and Reconfiguration

The *disjoint paths problem* is a classical and important problem in algorithmic graph theory and combinatorial optimization. In the problem, the input consists of a graph $G = (V, E)$ and $2k$ distinct vertices $s_1, \ldots, s_k, t_1, \ldots, t_k$, called *terminals*, and the task is to find k vertex-disjoint paths $P_1, \ldots, P_k$ such that P_i connects s_i and t_i for $i = 1, \ldots, k$ if they exist. A tuple $\mathcal{P} = (P_1, \ldots, P_k)$ of paths satisfying this condition is called a *linkage*. The disjoint paths problem has attracted attention since the 1970s because of its practical applications to transportation networks, network routing [46], and VLSI-layout [16, 29]. When the number k of terminal pairs is part of the input, the disjoint paths problem was shown to be NP-hard by Karp [25], and it remains NP-hard even for planar graphs [30]. For the case when the graph is undirected and k is a fixed constant, Robertson and Seymour [42] gave a polynomial-time algorithm based on the graph minor theory, which is one of the biggest achievements in this area. Although the setting of the disjoint paths problem is quite simple and easy to understand, a deep theory in discrete mathematics is required to solve the problem, which is a reason why this problem has attracted attention in the theoretical study of algorithms.

In this paper, we consider a transformation of linkages in a graph. Roughly, in a transformation, we pick up one path among the k paths in a linkage and replace it with another path to obtain a new linkage. To give a formal definition, suppose that G is a graph and $s_1, \ldots, s_k, t_1, \ldots, t_k$ are distinct terminals. For two linkages $\mathcal{P} = (P_1, \ldots, P_k)$ and $\mathcal{Q} = (Q_1, \ldots, Q_k)$, we say that $\mathcal{P}$ is *adjacent* to $\mathcal{Q}$ if there exists $i \in \{1, \ldots, k\}$ such that $P_j = Q_j$ for $j \in \{1, \ldots, k\} \setminus \{i\}$ and $P_i \neq Q_i$. We say that a sequence $\langle \mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_\ell \rangle$ of linkages is a *reconfiguration sequence* from $\mathcal{P}_1$ to $\mathcal{P}_\ell$ if $\mathcal{P}_i$ and $\mathcal{P}_{i+1}$ are adjacent for $i = 1, \ldots, \ell - 1$. If such a sequence exists, we say that $\mathcal{P}_1$ is *reconfigurable* to $\mathcal{P}_\ell$. In this paper, we focus on the following reconfiguration problem, which we call Disjoint Paths Reconfiguration.

> Disjoint Paths Reconfiguration
> **Input.** A graph $G = (V, E)$, distinct terminals $s_1, \ldots, s_k, t_1, \ldots, t_k$, and two linkages $\mathcal{P}$ and $\mathcal{Q}$.
> **Question.** Is $\mathcal{P}$ reconfigurable to $\mathcal{Q}$?

The problem can be regarded as the problem of deciding the reachability between linkages via rerouting paths. Such a problem falls in the area of *combinatorial reconfiguration*; see Section 1.3 for prior work on combinatorial reconfiguration. Note that Disjoint Paths Reconfiguration is a decision problem that just returns "YES" or "NO" and does not necessarily find a reconfiguration sequence when the answer is YES[1].

Although our study is motivated by a theoretical interest in the literature on combinatorial reconfiguration, the problem can model a rerouting problem in a telecommunication network as follows. Suppose that a linkage represents routing in a telecommunication network, and we want to modify linkage $\mathcal{P}$ to another linkage $\mathcal{Q}$ which is better than $\mathcal{P}$ in some sense. If we can change only one path in a step in the network for some technical reasons, and we have to keep a linkage in the modification process, then this situation is modeled as Disjoint Paths Reconfiguration.

[1] Our positive results in this paper hold also for the problem of finding a reconfiguration sequence.

We also study internally vertex-disjoint s-t paths instead of disjoint paths. In the *disjoint s-t paths problem*, for a graph and two terminals s and t, we seek for k internally vertex-disjoint paths connecting s and t. It is well-known that the disjoint s-t paths problem can be solved in polynomial time. The study of disjoint s-t paths originated from Menger's min-max theorem [33] and the max-flow algorithm by Ford and Fulkerson [14]. Faster algorithms for finding maximum disjoint s-t paths or a maximum s-t flow have been actively studied in particular for planar graphs; see e.g. [12, 24, 26, 51].

In the same way as Disjoint Paths Reconfiguration, we consider a reconfiguration of internally vertex-disjoint s-t paths. Let $G = (V, E)$ be a graph with two distinct terminals s and t. We say that a set $\mathcal{P} = \{P_1, \ldots, P_k\}$ of k paths in G is an *s-t linkage* if $P_1, \ldots, P_k$ are internally vertex-disjoint s-t paths. Note that $\mathcal{P}$ is not a tuple but a set, that is, we ignore the ordering of the paths in $\mathcal{P}$. We say that s-t linkages $\mathcal{P}$ and $\mathcal{Q}$ are *adjacent* if $\mathcal{Q} = (\mathcal{P} \setminus P) \cup \{Q\}$ for some s-t paths P and Q with $P \neq Q$. We define the reconfigurability of s-t linkages in the same way as linkages. We consider the following problem.

Disjoint s-t Paths Reconfiguration
Input. A graph $G = (V, E)$, distinct terminals s and t, and two s-t linkages $\mathcal{P}$ and $\mathcal{Q}$.
Question. Is $\mathcal{P}$ reconfigurable to $\mathcal{Q}$?

1.2 Our Contributions

Since finding disjoint s-t paths is an easy combinatorial optimization problem, we may wonder whether Disjoint s-t Paths Reconfiguration is also tractable. In this paper, we show that Disjoint s-t Paths Reconfiguration is PSPACE-hard even when $k = 2$.

▶ **Theorem 1.** *The Disjoint s-t Paths Reconfiguration is* PSPACE*-complete even when $k = 2$ and the maximum degree of G is four.*

Note that Disjoint s-t Paths Reconfiguration can be easily reduced to Disjoint Paths Reconfiguration by splitting each of s and t into k terminals. Thus, this theorem implies the PSPACE-hardness of Disjoint Paths Reconfiguration with $k = 2$.

In this paper, we mainly focus on the problems in planar graphs. To better understand Disjoint Paths Reconfiguration in planar graphs, we show a topological necessary condition.

Topological conditions play important roles in the disjoint paths problem. If there exist disjoint paths connecting terminal pairs in a graph embedded on a surface Σ, then there must exist disjoint curves on Σ connecting them. For example, when terminals s_1, s_2, t_1 and t_2 lie on the outer face F in a plane graph G in this order, there exist no disjoint curves connecting the terminal pairs in the disk $\Sigma = \mathbb{R}^2 \setminus F$, and hence we can conclude that G contains no disjoint paths. Such a topological condition is used to design polynomial-time algorithms for the disjoint paths problem with $k = 2$ [44, 45, 49], and to deal with the problem on a disk or a cylinder [40]. When Σ is a plane (or a sphere), we can always connect terminal pairs by disjoint curves on Σ, and hence nothing is derived from the above argument. Indeed, Robertson and Seymour [41] showed that if the input graph is embedded on a surface and the terminals are mutually "far apart," then desired disjoint paths always exist.

In contrast, as we will show below in Theorem 2, there exists a topological necessary condition for the reconfigurability of disjoint paths. Thus, even when the terminals are mutually far apart, the reconfiguration of disjoint paths is not always possible. This shows a difference between the disjoint paths problem and Disjoint Paths Reconfiguration.

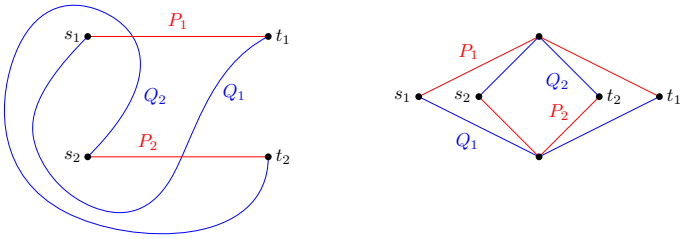


Figure 1 (Left) An example on the plane where (P_1, P_2) is not reconfigurable to (Q_1, Q_2). (Right) An example in a graph where the condition in Theorem 2 holds but (P_1, P_2) is not reconfigurable to (Q_1, Q_2).

To formally discuss the topological necessary condition, we consider the reconfiguration of curves on a surface. Suppose that Σ is a surface and let $s_1, \ldots, s_k, t_1, \ldots, t_k$ be distinct points on Σ. By abuse of notation, we say that $\mathcal{P} = (P_1, \ldots, P_k)$ is a *linkage* if it is a collection of disjoint simple curves on Σ such that P_i connects s_i and t_i. We also define the adjacency and reconfiguration sequences for linkages on Σ in the same way as linkages in a graph. Then, the reconfigurability between two linkages on a plane can be characterized with a word w_j associated to Q_j which is an element of the free group[2] F_k generated by $x_1, \ldots, x_k$ as follows; see Section 3 for the definition of w_j.

▶ **Theorem 2.** *Let $\mathcal{P} = (P_1, \ldots, P_k)$ and $\mathcal{Q} = (Q_1, \ldots, Q_k)$ be linkages on a plane (or a sphere). Then, $\mathcal{P}$ is reconfigurable to $\mathcal{Q}$ if and only if $w_j \in \langle x_j \rangle$ for any $j \in \{1, \ldots, k\}$, where $\langle x_j \rangle$ denotes the subgroup generated by x_j.*

See Figure 1 (left) for an example. It is worth noting that, if $k = 2$ and Σ is a connected orientable closed surface of genus $g \geq 1$, then such a topological necessary condition does not exist, i.e., the reconfiguration is always possible; see the full version [20].

For a graph embedded on a plane, we can identify paths and curves. Then, Theorem 2 gives a topological necessary condition for Disjoint Paths Reconfiguration in planar graphs. However, the converse does not necessarily hold: even when the condition in Theorem 2 holds, an instance of Disjoint Paths Reconfiguration may have no reconfiguration sequence. See Figure 1 (right) for a simple example. The polynomial solvability of Disjoint Paths Reconfiguration in planar graphs is open even for the case of $k = 2$.

With the aid of the topological necessary condition, we design polynomial-time algorithms for special cases, in which all the terminals are on a single face (called *one-face instances*), or $s_1, \ldots, s_k$ are on some face and $t_1, \ldots, t_k$ are on another face (called *two-face instances*). Note that one/two-face instances have attracted attention in the disjoint paths problem [40, 47, 48], in the multicommodity flow problem [18, 35, 36], and in the shortest disjoint paths problem [8, 9, 11, 28]. We show that any one-face instance of Disjoint Paths Reconfiguration has a reconfiguration sequence (Proposition 13). Moreover, we prove a topological characterization for two-face instances of Disjoint Paths Reconfiguration with a certain condition (Theorem 14), which leads to a polynomial-time algorithm in this case.

▶ **Theorem 3.** *When the instances are restricted to two-face instances, Disjoint Paths Reconfiguration can be solved in polynomial time.*

[2] Each element of the free group can be expressed as a word consisting of $x_1, x_1^{-1}, \ldots, x_k, x_k^{-1}$ in which x_i and x_i^{-1} are not adjacent.

Based on this theorem, we give a polynomial-time algorithm for Disjoint s-t Paths Reconfiguration in planar graphs.

▶ **Theorem 4.** *There is a polynomial-time algorithm for Disjoint s-t Paths Reconfiguration in planar graphs.*

Note that the number k of paths in Theorems 3 and 4 can be part of the input.

It is well known that G has an s-t linkage of size k if and only if G has no s-t separator of size $k-1$ (Menger's theorem). The characterization for two-face instances (Theorem 14) implies the following theorem, which is interesting in the sense that one extra s-t connectivity is sufficient to guarantee the existence of a reconfiguration sequence.

▶ **Theorem 5.** *Let $G=(V,E)$ be a planar graph with distinct vertices s and t, and let $\mathcal{P}$ and $\mathcal{Q}$ be s-t linkages of size k. If there is no s-t separator of size k, then $\mathcal{P}$ is reconfigurable to $\mathcal{Q}$.*

As mentioned above, the polynomial solvability of Disjoint Paths Reconfiguration in planar graphs is open even for the case of $k=2$. On the other hand, when k is not bounded, Disjoint Paths Reconfiguration is PSPACE-complete as the next theorem shows.

▶ **Theorem 6.** *The Disjoint Paths Reconfiguration is PSPACE-complete when the graph G is planar and of bounded bandwidth.*

Here, we recall the definition of the bandwidth of a graph. Let $G=(V,E)$ be an undirected graph. Consider an injective map $\pi\colon V\to\mathbb{Z}$. Then, the *bandwidth* of π is defined as $\max\{|\pi(u)-\pi(v)| \mid \{u,v\}\in E\}$. The *bandwidth* of G is defined as the minimum bandwidth of all injective maps $\pi\colon V\to\mathbb{Z}$.

1.3 Related Work

There are a lot of studies on the disjoint paths problem and its variant. For the case of $k=2$, polynomial-time algorithms were presented in [44, 45, 49], while the directed variant was shown to be NP-hard [15]. In the early stages of the study of the disjoint paths problem, for the case when G is embedded on a plane and all the terminals are on one face or two faces, polynomial-time algorithms were given in [40, 47, 48]. For fixed k, Robertson and Seymour [41] gave a polynomial time algorithm for the disjoint paths problem on a plane or a fixed surface. By extending this result, for the case when the graph is undirected and k is a fixed constant, Robertson and Seymour [42] gave a polynomial-time algorithm based on the graph minor theory, which is one of the biggest achievements in this area. For the planar case, faster algorithms were presented in [1, 38, 39]. The directed variant of the problem can be solved in polynomial time if the input digraph is planar and k is a fixed constant; an XP algorithm was given by Schrijver [43] and an FPT algorithm was given by Cygan et al. [10] for the parameter k.

Combinatorial reconfiguration is an emerging field in discrete mathematics and theoretical computer science. In typical problems of combinatorial reconfiguration, we consider two discrete structures and ask whether one can be transformed to the other by a sequence of local changes. See surveys of Nishimura [34] and van den Heuvel [50]. Refer to [22] for a general solver.

Path reconfiguration problems have been studied in this framework. The first problem is the shortest path reconfiguration, introduced by Kaminski et al. [23]. In this problem, we are given an undirected graph with two designated vertices s, t and two s-t shortest paths P and

Q. Then, we want to decide whether P can be transformed to Q by a sequence of one-vertex changes in such a way that all the intermediate s-t paths remain the shortest. Bonsma [6] proved that the shortest path reconfiguration is PSPACE-complete, but polynomial-time solvable when the input graph is chordal or claw-free. Bonsma [7] further proved that the problem is polynomial-time solvable for planar graphs. Wrochna [52] proved that the problem is PSPACE-complete even for graphs of bounded bandwidth. Gajjar et al. [17] proved that the problem is polynomial-time solvable for circle graphs, circular-arc graphs, permutation graphs, and hypercubes. They also considered a variant where a change can involve k successive vertices; in this variant, they proved that the problem is PSPACE-complete even for line graphs. Properties of the adjacency relation in the shortest path reconfiguration have also been studied [4, 5].

Another path reconfiguration problem has been introduced by Amiri et al. [3] who were motivated by a problem in software-defined networks. In their setup, we are given a directed graph with edge capacity and two designated vertices s, t. We are also given k pairs of s-t paths (P_i, Q_i), $i = 1, 2, \ldots, k$, where the number of paths among $P_1, P_2, \ldots, P_k$ (and among $Q_1, Q_2, \ldots, Q_k$ respectively) traversing an edge is at most the capacity of the edge. The problem is to determine whether one set of paths can be transformed into the other set of paths by a sequence of the following type of changes: specify one vertex v and then switch the usable outgoing edges at v from those in the P_i to those in the Q_i. In each of the intermediate situations, there must be a unique path through usable edges in $P_i \cup Q_i$ for each i. See [3] for the precise problem specification. Amiri et al. [3] proved that the problem is NP-hard even when $k = 2$. For directed acyclic graphs, they also proved that the problem is NP-hard (for unbounded k) but fixed-parameter tractable with respect to k. A subsequent work [2] studied an optimization variant in which the number of steps is to be minimized when a set of "disjoint" changes can be performed simultaneously.

Matching reconfiguration in bipartite graphs can be seen as a certain type of disjoint paths reconfiguration problems. In matching reconfiguration, we are given two matchings (with extra properties) and want to determine whether one matching can be transformed to the other matching by a sequence of local changes. There are several choices for local changes. One of the most studied local change rules is the token jumping rule, where we remove one edge and add one edge at the same time. Ito et al. [19] proved that the matching reconfiguration (under the token jumping rule) can be solved in polynomial time.[3]

To see a connection of matching reconfiguration with disjoint paths reconfiguration, consider the matching reconfiguration problem in bipartite graphs G under the token jumping rule, where we are given two matchings M, M' of G. Then, we add two extra vertices s, t to G, and for each edge $e \in M$ (and M') we construct a unique s-t path of length three that passes through e. This way, we obtain two s-t linkages $\mathcal{P}$ and $\mathcal{P}'$ from M and M', respectively. It is easy to observe that $\mathcal{P}$ can be reconfigured to $\mathcal{P}'$ in Disjoint s-t Paths Reconfiguration if and only if M can be reconfigured to M' in the matching reconfiguration problem in G.

1.4 Organization

In Section 2, we introduce some notation and basic concepts in topology. Section 3 deals with rerouting disjoint curves, giving the proof of Theorem 2. In Sections 4 and 5, we prove Theorems 3, 4, and 5. Hardness results (Theorems 1 and 6) are proven in the full version [20].

[3] The theorem by Ito et al. [19] only gave a polynomial-time algorithm for a different local change, the so-called token addition and removal rule. However, their result can easily be adapted to the token jumping rule, too. See [21].

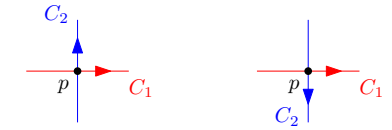


Figure 2 Local intersection numbers of curves C_1 and C_2 at p.

2 Preliminaries

For a positive integer k, let $[k] = \{1, 2, \ldots, k\}$.

Let $G = (V, E)$ be a graph. For a subgraph H of G, the vertex set of H is denoted by $V(H)$. Similarly, for a path P, let $V(P)$ denote the set of vertices in P. For $X \subseteq V$, let $N(X) = \{v \in V \setminus X \mid \{u, v\} \in E \text{ for some } u \in X\}$. For a vertex set $U \subseteq V$, let $G \setminus U$ denote the graph obtained from G by removing all the vertices in U and the incident edges. For a path P in G, we denote $G \setminus V(P)$ by $G \setminus P$ to simplify the notation. For disjoint vertex sets $X, Y \subseteq V$, we say that a vertex subset $U \subseteq V \setminus (X \cup Y)$ *separates* X and Y if $G \setminus U$ contains no path between X and Y. For distinct vertices $s, t \in V$, $U \subseteq V \setminus \{s, t\}$ is called an *s-t separator* if U separates $\{s\}$ and $\{t\}$.

For Disjoint Paths Reconfiguration (resp. Disjoint s-t Paths Reconfiguration), an instance is denoted by a triplet $(G, \mathcal{P}, \mathcal{Q})$, where G is a graph and $\mathcal{P}$ and $\mathcal{Q}$ are linkages (resp. s-t linkages). Note that we omit the terminals because they are determined by $\mathcal{P}$ and $\mathcal{Q}$. Since any instance has a trivial reconfiguration sequence when $k = 1$, we may assume that $k \geq 2$. For linkages (resp. s-t linkages) $\mathcal{P}$ and $\mathcal{Q}$, we denote $\mathcal{P} \leftrightarrow \mathcal{Q}$ if $\mathcal{P}$ and $\mathcal{Q}$ are adjacent. Recall that $\mathcal{P} = (P_1, \ldots, P_k)$ is *adjacent* to $\mathcal{Q} = (Q_1, \ldots, Q_k)$ if there exists $i \in [k]$ such that $P_j = Q_j$ for $j \in [k] \setminus \{i\}$ and $P_i \neq Q_i$.

For a graph G embedded on a surface Σ, each connected region of $\Sigma \setminus G$ is called a *face* of G. For a face F, its boundary is denoted by ∂F. When a graph G is embedded on a surface Σ, a path in G is sometimes identified with the corresponding curve in Σ. A graph embedded on a plane is called a *plane graph*. A graph is said to be *planar* if it has a planar embedding.

The following notion is well-known in topology. See [13, Section 1.2.3] for instance.

▶ **Definition 7.** *Let C_1 and C_2 be piecewise smooth oriented curves on an oriented surface and let $p \in C_1 \cap C_2$ be a transverse double point*[4]*. The* local intersection number *$\varepsilon_p(C_1, C_2)$ of C_1 and C_2 at p is defined by $\varepsilon_p(C_1, C_2) = 1$ if C_1 crosses C_2 from left to right and $\varepsilon_p(C_1, C_2) = -1$ if C_1 crosses C_2 from right to left (see Figure 2). When $\partial C_1 \cap C_2 = C_1 \cap \partial C_2 = \emptyset$, the* algebraic intersection number *$\mu(C_1, C_2) \in \mathbb{Z}$ is defined to be the sum of $\varepsilon_p(C_1, C_2)$ over all $p \in C_1 \cap C_2$ (after a small perturbation if necessary). Note that ∂C_i denotes the set of endpoints of C_i.*

When a graph is embedded on an oriented surface, paths in the graph are piecewise smooth curves, and hence we can define the algebraic intersection number for a pair of paths (see Figure 3).

[4] Intuitively, a "transverse double point" means that at the intersection two curves are not tangent with each other and no three segments of curves do not intersect simultaneously.

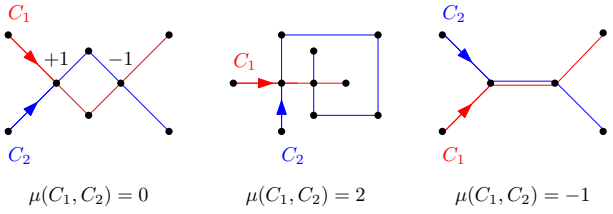


Figure 3 Algebraic intersection numbers of paths C_1 and C_2 on a graph.

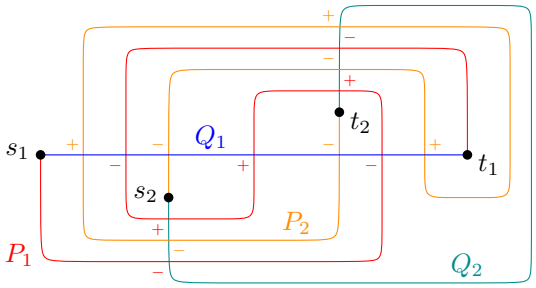


Figure 4 An example of linkages with $w_1 = x_2x_1^{-1}x_2^{-1}x_1x_2^{-1}x_1^{-1}x_2$ and $w_2 = x_1x_2^{-1}x_1^{-1}x_2x_1^{-1}x_2^{-1}x_1$.

3 Curves on a Plane

In this section, we consider the reconfiguration of curves on a plane and prove Theorem 2. Suppose that we are given distinct points $s_1, \dots, s_k, t_1, \dots, t_k$ on a plane and linkages $\mathcal{P}$ and $\mathcal{Q}$ that consist of curves on the plane connecting s_i and t_i.

Throughout this section, all intersections of curves are assumed to be transverse double points. Fix $j \in [k]$ and let $\bigcup_{i\in[k]} P_i \cap Q_j = \{s_j, p_1, \dots, p_n, t_j\}$, where the $n+2$ points are aligned on Q_j in this order. We now define $w_j \in F_k$ by

$$w_j = \prod_{\ell\in[n]} x_{i_\ell}^{\varepsilon_{p_\ell}(P_{i_\ell}, Q_j)},$$

where $i_\ell \in [k]$ satisfies $p_\ell \in P_{i_\ell} \cap Q_j$. Recall that F_k denotes the free group generated by $x_1, \dots, x_k$. We give an example in Figure 4.

▶ Remark 8. Let $\mathrm{ab}\colon F_k \to \mathbb{Z}^k$ denote the abelianization, that is, the ℓth entry of $\mathrm{ab}(w)$ is the sum of the exponents of x_ℓ's in w. For distinct $i, j \in [k]$, the ith entry of $\mathrm{ab}(w_j)$ is equal to the algebraic intersection number $\mu(P_i, Q_j) \in \mathbb{Z}$ of P_i and Q_j. Thus, $w_j \in \langle x_j \rangle$ implies that $\mu(P_i, Q_j) = 0$ for any $i \in [k] \setminus \{j\}$.

In the following two lemmas, we observe the behavior of w_j under certain moves of curves. For $j \in [k]$, let w'_j denote the word defined by a linkage $\mathcal{P}'$ and the curve Q_j.

▶ **Lemma 9.** *Let $i \in [k]$ and let $\mathcal{P}' = (P'_1, \dots, P'_k)$ be a linkage such that $P'_\ell = P_\ell$ if $\ell \neq i$, and P'_i is isotopic[5] to P_i relative to $\{s_i, t_i\}$ in $\mathbb{R}^2 \setminus \bigcup_{\ell\neq i} P_\ell$. Then, $w'_j = w_j$ for $j \in [k] \setminus \{i\}$, and $w'_i = x_i^{e_1} w_i x_i^{e_2}$ for some $e_1, e_2 \in \mathbb{Z}$.*

[5] Intuitively, this means P'_i can be obtained from P_i by a continuous deformation in the plane that fixes the endpoints s_i and t_i and avoids passing any point in $\bigcup_{\ell\neq i} P_\ell$.

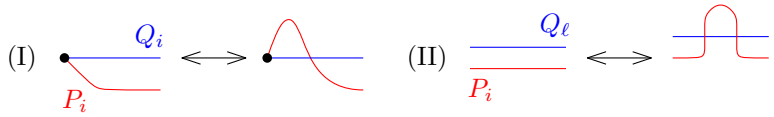


Figure 5 Local pictures of isotopies of P_i.

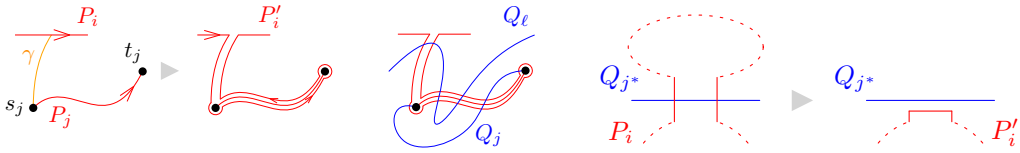


Figure 6 (Left) A move of P_i along γ. (Right) Intersections of P_i' and $\bigcup_\ell Q_\ell$.

Figure 7 A reconfiguration of P_i to P_i'.

Proof. By the definition of an isotopy (see [13, Section 1.2.5]), P_i' is obtained from P_i by a finite sequence of the moves illustrated in Figure 5. By (I), one intersection of P_i and Q_i is created or eliminated, and thus (I) changes w_i to $w_i x_i^{\pm 1}$ or $x_i^{\pm 1} w_i$. In (II), two intersections of P_i and Q_ℓ are created or eliminated for some $\ell \in [k]$. Since $x_i^{\pm 1} x_i^{\mp 1} = 1$, w_j is unchanged under (II) for any $j \in [k]$. ◀

Recall here that $\langle x_\ell \rangle$ denotes the subgroup of F_k generated by x_ℓ.

▶ **Lemma 10.** *Let γ be a simple curve connecting P_i and s_j $(i \neq j)$ whose interior is disjoint from $\bigcup_{\ell \in [k]} P_\ell$, and define P_i' as illustrated in Figure 6. Let $\mathcal{P}'$ be the linkage obtained from $\mathcal{P}$ by replacing P_i with P_i'. For $\ell \in [k]$, if $w_\ell \in \langle x_\ell \rangle$, then $w_\ell' = w_\ell$.*

Proof. Define a group homomorphism $f_{ij} \colon F_k \to F_k$ by $f_{ij}(x_\ell) = x_\ell$ if $\ell \neq j$, and $f_{ij}(x_j) = x_i x_j x_i^{-1}$. Then, one can check that $w_\ell' = f_{ij}(w_\ell)$ if $\ell \neq j$, and $w_j' = x_i^{-1} f_{ij}(w_j) x_i$ (see Figure 6). Since $w_\ell = x_\ell^{e_\ell}$ for some $e_\ell \in \mathbb{Z}$ by the assumption, we have $w_\ell' = w_\ell$ if $\ell \neq j$. Also, one has

$$w_j' = x_i^{-1} f_{ij}(w_j) x_i = x_i^{-1} (x_i x_j x_i^{-1})^{e_j} x_i = w_j.$$

This completes the proof. ◀

As a consequence of Lemmas 9 and 10, we obtain the following key lemma.

▶ **Lemma 11.** *Suppose that $\mathcal{P}$ is reconfigurable to $\mathcal{P}'$. For $j \in [k]$, if $w_j \in \langle x_j \rangle$, then $w_j' \in \langle x_j \rangle$.*

Proof. It suffices to consider the case when there is $i \in [k]$ such that $P_\ell' = P_\ell$ if $\ell \neq i$, and $P_i' \neq P_i$. Since P_i is isotopic to P_i' (relative to $\{s_i, t_i\}$) in $\mathbb{R}^2$, the curve P_i' is obtained from P_i by the moves in Lemmas 9 and 10. Therefore, these lemmas imply that if $w_j \in \langle x_j \rangle$ then $w_j' \in \langle x_j \rangle$. ◀

With this key lemma, we can prove Theorem 2 stating that $\mathcal{P}$ is reconfigurable to $\mathcal{Q}$ if and only if $w_j \in \langle x_j \rangle$ for any $j \in [k]$.

Proof of Theorem 2. First suppose that $\mathcal{P}$ is reconfigurable to $\mathcal{Q}$, namely $\mathcal{P}$ is reconfigurable to $\mathcal{P}'$ such that $P_i' \cap Q_i = \{s_i, t_i\}$ and $P_i' \cap Q_j = \emptyset$ for $j \in [k] \setminus \{i\}$. Then, $w_j' = 1$ for any $j \in [k]$. Since $\mathcal{P}'$ is reconfigurable to $\mathcal{P}$, Lemma 11 implies that $w_j \in \langle x_j \rangle$ for any $j \in [k]$.

The converse is shown by induction on the number, say n, of intersections of $\mathcal{P}$ and $\mathcal{Q}$ except their endpoints. The case $n = 0$ is obvious. Let us consider the case $n \geq 1$. If $P_i \cap Q_j = \emptyset$ for any pair of distinct $i, j \in [k]$, then the reconfiguration is obviously

possible. Otherwise, there exists $x_i x_i^{-1}$ or $x_i^{-1} x_i$ in the product of the definition of w_{j^*} for some $i, j^* \in [k]$ (possibly $i = j^*$). This means that P_i can be reconfigured to a curve P_i' as illustrated in Figure 7. This process eliminates at least two intersections and we have $w_j' \in \langle x_j \rangle$ for any $j \in [k]$ by Lemma 11. Thus, the induction hypothesis concludes that $\mathcal{P}'$ is reconfigurable to $\mathcal{Q}$. ◀

By Theorem 2 and Remark 8, we obtain the following corollary.

▶ **Corollary 12.** *Let $\mathcal{P} = (P_1, \ldots, P_k)$ and $\mathcal{Q} = (Q_1, \ldots, Q_k)$ be linkages on a plane (or a sphere). If $\mathcal{P}$ is reconfigurable to $\mathcal{Q}$, then $\mu(P_i, Q_j) = 0$ for any distinct $i, j \in [k]$.*

It is worth mentioning that the converse is not necessarily true as illustrated in Figure 4. This means that a "non-commutative" tool such as the free group F_k is essential to describe the complexity of the reconfiguration of curves on a plane.

4 Algorithms for Planar Graphs

In this section, we consider the reconfiguration in planar graphs and prove Theorems 3, 4, and 5. We deal with one-face instances and two-face instances of Disjoint Paths Reconfiguration in Section 4.1. Then, we discuss Disjoint s-t Paths Reconfiguration in Section 4.2. A proof of a key theorem (Theorem 14) is postponed to Section 5.

4.1 One-Face Instance and Two-Face Instance

We say that an instance $(G, \mathcal{P}, \mathcal{Q})$ of Disjoint Paths Reconfiguration is a *one-face instance* if G is a plane graph and all the terminals are on the boundary of some face. We show that $\mathcal{P}$ is always reconfigurable to $\mathcal{Q}$ in a one-face instance, whose proof is given in the full version [20].

▶ **Proposition 13.** *For any one-face instance $(G, \mathcal{P}, \mathcal{Q})$ of Disjoint Paths Reconfiguration, $\mathcal{P}$ is reconfigurable to $\mathcal{Q}$.*

Let $k \geq 2$. We say that an instance $(G, \mathcal{P}, \mathcal{Q})$ of Disjoint Paths Reconfiguration is a *two-face instance* if $G = (V, E)$ is a plane graph, $s_1, \ldots, s_k$ are on the boundary of some face S, and $t_1, \ldots, t_k$ are on the boundary of another face T. The objective of this subsection is to present a polynomial-time algorithm for two-face instances.

It suffices to consider the case when the graph is 2-connected since otherwise we can easily reduce to the 2-connected case. Hence, we may assume that the boundary of each face forms a cycle. For ease of explanation, without loss of generality, we assume that G is embedded on $\mathbb{R}^2$ so that S is an inner face and T is the outer face. Furthermore, we may assume that $s_1, \ldots, s_k$ lie on the boundary of S clockwise in this order and $t_1, \ldots, t_k$ lie on the boundary of T clockwise in this order, because there is a linkage.

A vertex set $U \subseteq V$ is called a *terminal separator* if U separates $\{s_1, \ldots, s_k\}$ and $\{t_1, \ldots, t_k\}$. For two curves (or paths) P and Q between ∂S and ∂T that share no endpoints, define $\mu(P, Q)$ as in Definition 7. That is, $\mu(P, Q)$ is the number of times P crosses Q from left to right minus the number of times P crosses Q from right to left, where we suppose that P and Q are oriented from ∂S to ∂T. Since $\mu(P_i, Q_j)$ takes the same value for distinct $i, j \in [k]$ (see the full version [20] for details), this value is denoted by $\mu(\mathcal{P}, \mathcal{Q})$. Roughly, $\mu(\mathcal{P}, \mathcal{Q})$ indicates the difference in the numbers of rotations around S of the linkages.

The existence of a linkage shows that the graph has no terminal separator of size less than k. If the graph has no terminal separator of size k, then we can characterize the reconfigurability by using $\mu(\mathcal{P}, \mathcal{Q})$. The following is a key theorem in our algorithm, whose proof is given in Section 5.

▶ **Theorem 14.** *Let $k \geq 2$. Suppose that a two-face instance $(G, \mathcal{P}, \mathcal{Q})$ of* Disjoint Paths Reconfiguration *has no terminal separator of size k. Then, $\mathcal{P}$ is reconfigurable to $\mathcal{Q}$ if and only if $\mu(\mathcal{P}, \mathcal{Q}) = 0$.*

By using this theorem, we can design a polynomial-time algorithm for two-face instances of Disjoint Paths Reconfiguration and prove Theorem 3.

Proof of Theorem 3. Suppose that we are given a two-face instance $I = (G, \mathcal{P}, \mathcal{Q})$ of Disjoint Paths Reconfiguration.

We first test whether I has a terminal separator of size k, which can be done in polynomial time by a standard minimum cut algorithm. If there is no terminal separator of size k, then Theorem 14 shows that we can easily solve Disjoint Paths Reconfiguration by checking whether $\mu(\mathcal{P}, \mathcal{Q}) = 0$ or not.

Suppose that we obtain a terminal separator U of size k. Then, we obtain subgraphs G_1 and G_2 of G such that $G = G_1 \cup G_2$, $V(G_1) \cap V(G_2) = U$, $\{s_1, \ldots, s_k\} \subseteq V(G_1)$, and $\{t_1, \ldots, t_k\} \subseteq V(G_2)$. We test whether $V(P_i) \cap U = V(Q_i) \cap U$ holds for any $i \in [k]$ or not, where we note that each of $V(P_i) \cap U$ and $V(Q_i) \cap U$ consists of a single vertex. If this does not hold, then we can immediately conclude that $\mathcal{P}$ is not reconfigurable to $\mathcal{Q}$, because $V(P_i) \cap U$ does not change in the reconfiguration. If $V(P_i) \cap U = V(Q_i) \cap U$ for $i \in [k]$, then we consider the instance $I_i = (G_i, \mathcal{P}_i, \mathcal{Q}_i)$ for $i = 1, 2$, where $\mathcal{P}_i$ and $\mathcal{Q}_i$ are the restrictions of $\mathcal{P}$ and $\mathcal{Q}$ to G_i. That is, I_i is the restriction of I to G_i. Then, we see that $\mathcal{P}$ is reconfigurable to $\mathcal{Q}$ if and only if $\mathcal{P}_i$ is reconfigurable to $\mathcal{Q}_i$ for $i = 1, 2$. Since I_1 and I_2 are one-face or two-face instances, by solving them recursively, we can solve the original instance I in polynomial time. ◀

4.2 Reconfiguration of s-t Paths

In this subsection, for Disjoint s-t Paths Reconfiguration in planar graphs, we show results that are analogous to Theorems 14 and 3, which have been already stated in Section 1.2.

▶ **Theorem 5.** *Let $G = (V, E)$ be a planar graph with distinct vertices s and t, and let $\mathcal{P}$ and $\mathcal{Q}$ be s-t linkages of size k. If there is no s-t separator of size k, then $\mathcal{P}$ is reconfigurable to $\mathcal{Q}$.*

Proof. Suppose that G, s, t, $\mathcal{P}$, and $\mathcal{Q}$ are as in the statement, and assume that there is no s-t separator of size k. We fix an embedding of G on the plane. If there is an edge connecting s and t, then s and t are on the boundary of some face, and hence $\mathcal{P}$ is reconfigurable to $\mathcal{Q}$ in the same way as Proposition 13. Thus, it suffices to consider the case when there is no edge connecting s and t.

We now construct an instance of Disjoint Paths Reconfiguration by replacing s and t with large "grids" as follows. Let $e_1, e_2, \ldots, e_\ell$ be the edges incident to s clockwise in this order. Note that $\ell \geq k + 1$ holds, because G has no s-t separator of size k. For $i \in [\ell]$, we subdivide e_i by introducing p new vertices $v_i^1, v_i^2, \ldots, v_i^p$ such that they are aligned in this order and v_i^1 is closest to s, where p is a sufficiently large integer (e.g., $p \geq |V|^2$). For $i \in [\ell]$ and for $j \in [p]$, we introduce a new edge connecting v_i^j and v_{i+1}^j, where $v_{\ell+1}^j = v_1^j$. Define $s_i = v_i^1$ for $i \in [k]$ and remove s. Then, the graph is embedded on the plane and $s_1, \ldots, s_k$ are on the boundary of some face clockwise in this order; see Figure 8. By applying a similar procedure to t, we modify the graph around t and define $t_1, \ldots, t_k$ that are on the boundary of some face counter-clockwise in this order. Let G' be the obtained graph. Observe that G' contains no terminal separator of size k, because G has no s-t separator of size k.

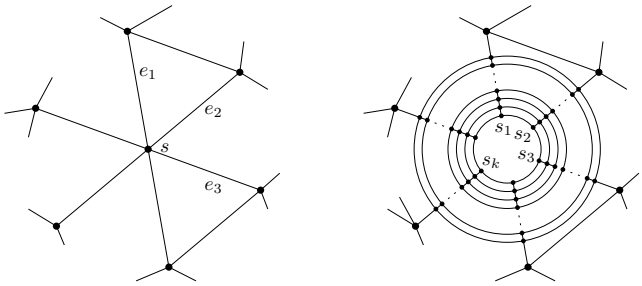


Figure 8 (Left) Original graph G. (Right) Modification around s.

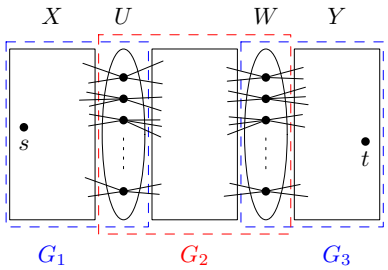


Figure 9 Construction of G_1, G_2, and G_3.

By rerouting the given s-t linkages $\mathcal{P}$ and $\mathcal{Q}$ around s and t, we obtain linkages $\mathcal{P}'$ and $\mathcal{Q}'$ from $\{s_1, \dots, s_k\}$ to $\{t_1, \dots, t_k\}$ in G'. Note that the restrictions of $\mathcal{P}$ and $\mathcal{Q}$ to $G \setminus \{s, t\}$ coincide with those of $\mathcal{P}'$ and $\mathcal{Q}'$, respectively. Then, we can take $\mathcal{P}'$ and $\mathcal{Q}'$ so that $|\mu(\mathcal{P}', \mathcal{Q}')| \leq |V|$. Furthermore, using at most $|V|$ concentric cycles around s and t, we can reroute the linkages so that the value $\mu(\mathcal{P}', \mathcal{Q}')$ decreases or increases by one. Therefore, using $p \geq |V|^2$ concentric cycles, we can reroute $\mathcal{P}'$ and $\mathcal{Q}'$ so that $\mu(\mathcal{P}', \mathcal{Q}')$ becomes zero.

By Theorem 14, $\mathcal{P}'$ is reconfigurable to $\mathcal{Q}'$ in G' (in terms of Disjoint Paths Reconfiguration). Then, the reconfiguration sequence from $\mathcal{P}'$ to $\mathcal{Q}'$ corresponds to that from $\mathcal{P}$ to $\mathcal{Q}$ in G (in terms of Disjoint s-t Paths Reconfiguration). Therefore, $\mathcal{P}$ is reconfigurable to $\mathcal{Q}$ in G. ◀

▶ **Theorem 4.** *There is a polynomial-time algorithm for Disjoint s-t Paths Reconfiguration in planar graphs.*

Proof. Suppose that we are given a planar graph $G = (V, E)$ with $s, t \in V$ and s-t linkages $\mathcal{P} = \{P_1, \dots, P_k\}$ and $\mathcal{Q} = \{Q_1, \dots, Q_k\}$ in G. We first test whether G has an s-t separator of size k. If there is no such a separator, then we can immediately conclude that $\mathcal{P}$ is reconfigurable to $\mathcal{Q}$ by Theorem 5.

Suppose that G has an s-t separator of size k. Let X be the inclusionwise minimal vertex set subject to $s \in X$ and $N(X)$ is an s-t separator of size k. Note that such X is uniquely determined by the submodularity of $|N(X)|$ and it can be computed in polynomial time by a standard minimum cut algorithm. Similarly, let Y be the unique inclusionwise minimal vertex set subject to $t \in Y$ and $N(Y)$ is an s-t separator of size k. Let $U = N(X)$, $W = N(Y)$, $G_1 = G[X \cup U]$, $G_2 = G \setminus (X \cup Y)$, and $G_3 = G[Y \cup W]$; see Figure 9. Since $V(P_i) \cap U$ and $V(P_i) \cap W$ do not change in the reconfiguration, we can consider the reconfiguration in G_1, G_2, and G_3, separately.

We first consider the reconfiguration in G_1. Observe that each path in $\mathcal{P}$ contains exactly one vertex in U, and the restriction of $\mathcal{P}$ to G_1 consists of k paths from s to U that are vertex-disjoint except at s. The same for $\mathcal{Q}$. By the minimality of X, G_1 contains no vertex set of size k that separates $\{s\}$ and U. Therefore, by the same argument as Theorem 5, the restriction of $\mathcal{P}$ to G_1 is reconfigurable to that of $\mathcal{Q}$.

If $U \cap W \neq \emptyset$, then $G \setminus X$ contains no vertex set of size k that separates U and $\{t\}$ by the minimality of Y. In such a case, by applying the same argument as above, the restriction of $\mathcal{P}$ to $G \setminus X$ is reconfigurable to that of $\mathcal{Q}$. By combining the reconfiguration in G_1 and that in $G \setminus X$, we obtain a reconfiguration sequence from $\mathcal{P}$ to $\mathcal{Q}$.

Therefore, it suffices to consider the case when $U \cap W = \emptyset$. In the same way as G_1, we see that the restriction of $\mathcal{P}$ to G_3 is reconfigurable to that of $\mathcal{Q}$. This shows that the reconfigurability from $\mathcal{P}$ to $\mathcal{Q}$ in G is equivalent to that in G_2. By changing the indices if necessary, we may assume that $P_i \cap U = Q_i \cap U$ for $i \in [k]$. If $P_i \cap W \neq Q_i \cap W$ for some $i \in [k]$, then we can conclude that $\mathcal{P}$ is not reconfigurable to $\mathcal{Q}$. Otherwise, let $\mathcal{P}'$ and $\mathcal{Q}'$ be the restrictions of $\mathcal{P}$ and $\mathcal{Q}$ to G_2, respectively. Since $(G_2, \mathcal{P}', \mathcal{Q}')$ is a one-face or two-face instance of Disjoint Paths Reconfiguration, we can solve it in polynomial time by Proposition 13 and Theorem 3. Therefore, we can test the reconfigurability from $\mathcal{P}$ to $\mathcal{Q}$ in polynomial time; ◀

5 Proof of Theorem 14

The necessity ("only if" part) in Theorem 14 is immediately derived from Corollary 12.

In what follows in this section, we show the sufficiency ("if" part) in Theorem 14, which is one of the main technical contributions of this paper. Assume that $\mu(\mathcal{P}, \mathcal{Q}) = 0$ and there is no terminal separator of size k. The objective is to show that $\mathcal{P}$ is reconfigurable to $\mathcal{Q}$. Our proof is constructive, and based on topological arguments. A similar technique is used in [27, 32, 31, 37].

5.1 Preliminaries for the Proof

Let C be a simple curve connecting the boundaries of S and T such that C contains no vertex in G, C intersects the boundaries of S and T only at its endpoints, and $\mu(P_i, C) = 0$ for $i \in [k]$. Note that such C always exists, because the last condition is satisfied if C is disjoint from $\mathcal{P}$. Note also that $\mu(Q_i, C) = 0$ holds for $i \in [k]$, because $\mu(\mathcal{P}, \mathcal{Q}) = 0$.

Since T is the outer face, $\mathbb{R}^2 \setminus (S \cup T)$ forms an annulus (or a cylinder).[6] Thus, by cutting it along C, we obtain a rectangle whose boundary consists of ∂S, ∂T, and two copies of C. We take infinite copies of this rectangle and glue them together to obtain an infinitely long strip R. That is, for $j \in \mathbb{Z}$, let C^j be a copy of C, let R^j be a copy of the rectangle whose boundary contains C^j and C^{j+1}, and define $R = \bigcup_{j \in \mathbb{Z}} R^j$; see Figure 10. By taking C appropriately, we may assume that the copies of $s_1, \ldots, s_k$ lie on the boundary of R^j in this order so that s_1 is closest to C^j and s_k is closest to C^{j+1}. The same for $t_1, \ldots, t_k$. Note that R is called the *universal cover* of $\mathbb{R}^2 \setminus (S \cup T)$ in the terminology of topology.

Since G is embedded on $\mathbb{R}^2 \setminus (S \cup T)$, this operation naturally defines an infinite periodic graph $\hat{G} = (\hat{V}, \hat{E})$ on R that consists of copies of G. A path in $\hat{G}$ is identified with the corresponding curve in R. For $v \in V$ and $j \in \mathbb{Z}$, let $v^j \in \hat{V}$ denote the unique vertex in R^j

[6] More precisely, the annulus is degenerated when $\partial S \cap \partial T \neq \emptyset$, but the same argument works even for this case.

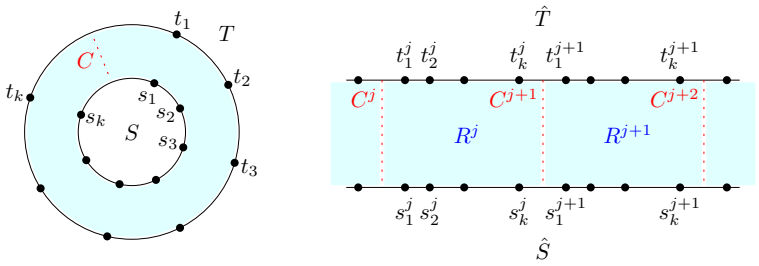


Figure 10 (Left) Curve C in $\mathbb{R}^2 \setminus (S \cup T)$. (Right) Construction of R.

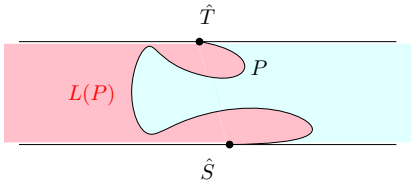


Figure 11 Definition of $L(P)$.

that corresponds to v. Since $\mu(P_i, C) = 0$ for $i \in [k]$, each path in $\hat{G}$ corresponding to P_i is from s_i^j to t_i^j for some $j \in \mathbb{Z}$, and we denote such a path by P_i^j. We define Q_i^j in the same way. Since $\mathcal{P}$ and $\mathcal{Q}$ are linkages in G, $\{P_i^j \mid i \in [k],\ j \in \mathbb{Z}\}$ and $\{Q_i^j \mid i \in [k],\ j \in \mathbb{Z}\}$ are sets of vertex-disjoint paths in $\hat{G}$.

A path in $\hat{G}$ connecting the boundary of R corresponding to ∂S and that corresponding to ∂T is called an $\hat{S}$-$\hat{T}$ *path*. For an $\hat{S}$-$\hat{T}$ path P, let $L(P)$ be the region of $R \setminus P$ that is on the "left-hand side" of P. Formally, let r be a point in R^j for sufficiently small j, and define $L(P)$ as the set of points $x \in R \setminus P$ such that any curve in R between r and x crosses P an even number of times; see Figure 11. For two $\hat{S}$-$\hat{T}$ paths P and Q, we denote $P \preceq Q$ if $L(P) \subseteq L(Q)$, and denote $P \prec Q$ if $L(P) \subsetneq L(Q)$. For two linkages $\mathcal{P} = (P_1, \ldots, P_k)$ and $\mathcal{Q} = (Q_1, \ldots, Q_k)$ in G with $\mu(P_i, C) = \mu(Q_i, C) = 0$ for $i \in [k]$, we denote $\mathcal{P} \preceq \mathcal{Q}$ if $P_i^j \preceq Q_i^j$ for any $i \in [k]$ and $j \in \mathbb{Z}$, and denote $\mathcal{P} \prec \mathcal{Q}$ if $\mathcal{P} \preceq \mathcal{Q}$ and $\mathcal{P} \neq \mathcal{Q}$.

5.2 Case When $\mathcal{P} \preceq \mathcal{Q}$

In this subsection, we consider the case when $\mathcal{P} \preceq \mathcal{Q}$, and the general case will be dealt with in Section 5.3. To show that $\mathcal{P}$ is reconfigurable to $\mathcal{Q}$, we show the following lemma.

▶ **Lemma 15.** *If $\mathcal{P} \prec \mathcal{Q}$, then there exists a linkage $\mathcal{P}'$ such that $\mathcal{P} \leftrightarrow \mathcal{P}'$ and $\mathcal{P} \prec \mathcal{P}' \preceq \mathcal{Q}$.*

Proof. Let $\hat{W} := \{\hat{v} \in \hat{V} \mid \hat{v} \in P_i^j \setminus Q_i^j \text{ for some } i \in [k] \text{ and } j \in \mathbb{Z}\}$ and let W be the subset of V corresponding to $\hat{W}$. If $W = \emptyset$, then take an index $i \in [k]$ such that $P_i \neq Q_i$ and let $\mathcal{P}' = (P_1', \ldots, P_k')$ be the set of paths obtained from $\mathcal{P}$ by replacing P_i with Q_i. Since $\mathcal{Q}$ is a linkage and all the vertices in P_h' are contained in Q_h for $h \in [k]$, $\mathcal{P}'$ is a desired linkage. Thus, it suffices to consider the case when $W \neq \emptyset$.

Let $u \in W$. Let $\hat{u} \in \hat{W}$ be a vertex corresponding to u and let $i \in [k]$ and $j \in \mathbb{Z}$ be the indices such that $\hat{u} \in P_i^j \setminus Q_i^j$. Since $\hat{u} \in P_i^j \setminus Q_i^j$ implies $\hat{u} \in L(Q_i^j) \setminus L(P_i^j)$, there exists a face $\hat{F}$ of $\hat{G}$ such that $\partial \hat{F}$ contains an edge of P_i^j incident to $\hat{u}$ and $\hat{F} \subseteq L(Q_i^j) \setminus L(P_i^j)$. Define $(P_i^j)'$ as the s_i^j-t_i^j path in $\hat{G}$ with maximal $L((P_i^j)')$ subject to $(P_i^j)' \subseteq P_i^j \cup \partial \hat{F}$; see Figure 12. Note that such a path is uniquely determined, it satisfies $P_i^j \prec (P_i^j)' \preceq Q_i^j$, and it can be found in polynomial time.

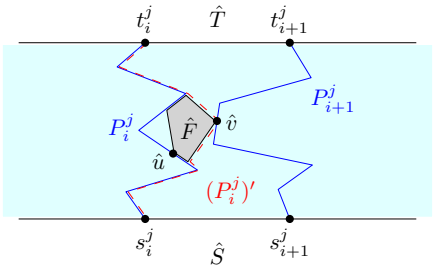


Figure 12 The blue thick paths are P_i^j and P_{i+1}^j, and the red dashed path is $(P_i^j)'$. There exists a vertex $\hat{v} \in \partial\hat{F} \cap P_{i+1}^j$.

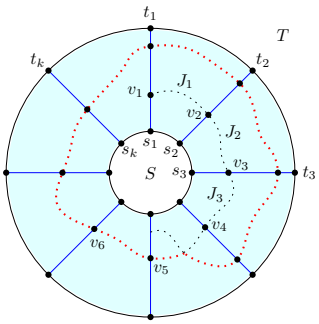


Figure 13 Each blue path represents P_i. The dotted curve is part of J and the red dotted thick curve is C^*.

Let P_i' be the s_i-t_i path in G that corresponds to $(P_i^j)'$. If P_i' is disjoint from P_h for any $h \in [k] \setminus \{i\}$, then we can obtain a desired linkage $\mathcal{P}'$ from $\mathcal{P}$ by replacing P_i with P_i'. Otherwise, P_i' intersects P_h for some $h \in [k] \setminus \{i\}$. This together with $P_i^j \prec (P_i^j)'$ shows that $(P_i^j)'$ intersects P_{i+1}^j, where P_{k+1}^j means P_1^{j+1}. Since P_i^j and P_{i+1}^j are vertex-disjoint, the intersection of $(P_i^j)'$ and P_{i+1}^j is contained in $\partial\hat{F}$, which implies that $\partial\hat{F} \cap P_{i+1}^j$ contains a vertex $\hat{v} \in \hat{V}$; see Figure 12 again. Since $\hat{F} \subseteq L(Q_i^j)$, we obtain $\hat{v} \in L(Q_i^j) \cup Q_i^j \subseteq L(Q_{i+1}^j)$, and hence $\hat{v} \notin Q_{i+1}^j$. Let v and F be the vertex and the face of G that correspond to $\hat{v}$ and $\hat{F}$, respectively. Then, $\hat{v} \in P_{i+1}^j \setminus Q_{i+1}^j$ implies that $\hat{v} \in \hat{W}$ and $v \in W$. Note that there exists a curve in F from u to v.

By the above argument, for any $u \in W$, we can obtain

- **(i)** a desired linkage $\mathcal{P}'$, or
- **(ii)** a vertex $v \in W$ on P_{i+1} and a curve from u to v contained in some face of G, where i is the index with $u \in V(P_i)$.

Therefore, it suffices to show that we obtain the outcome (i) for some $u \in W$. To derive a contradiction, assume to the contrary that we obtain the outcome (ii) for any $u \in W$.

By using the outcome (ii) repeatedly and by shifting the indices of P_i if necessary, we obtain v_i and J_i for $i = 1, 2, \ldots$ such that $v_i \in W$ is on P_i (where the index is modulo k) and J_i is a curve from v_i to v_{i+1} contained in some face. We consider the curve J obtained by concatenating $J_1, J_2, \ldots$ in this order. Since $|W|$ is finite, this curve visits the same point more than once, and hence it contains a simple closed curve C^*. Since C^* is simple and visits vertices on $P_i, P_{i+1}, \ldots$ in this order, C^* surrounds S exactly once in the clockwise direction; see Figure 13. In particular, C^* contains exactly one vertex on P_i for each $i \in [k]$. Let U be the set of vertices in V contained in C^*. Then, $|U| = k$ and $G \setminus U$ has no path between $\{s_1, \ldots, s_k\}$ and $\{t_1, \ldots, t_k\}$ by the choice of C^*. Furthermore, U contains no terminals, because $U \subseteq W$ and W contains no terminals. Therefore, U is a terminal separator of size k, which contradicts the assumption. ◀

As long as $\mathcal{P} \neq \mathcal{Q}$, we apply this lemma and replace $\mathcal{P}$ with $\mathcal{P}'$, repeatedly. Then, this procedure terminates when $\mathcal{P} = \mathcal{Q}$, and gives a reconfiguration sequence from $\mathcal{P}$ to $\mathcal{Q}$. This completes the proof for the case when $\mathcal{P} \preceq \mathcal{Q}$.

We now give a remark on the length of the reconfiguration sequence. For $\hat{S}$-$\hat{T}$ paths P and Q with the same endpoints, we see that $L(Q) \setminus L(P)$ contains $O(|V|^2)$ faces of $\hat{G}$. Therefore, in the reconfiguration sequence from $\mathcal{P}$ to $\mathcal{Q}$ obtained above, each path in $\mathcal{P}$ is replaced with another path $O(|V|^2)$ times, which shows that the number of applications of Lemma 15 is $O(k|V|^2)$ in total.

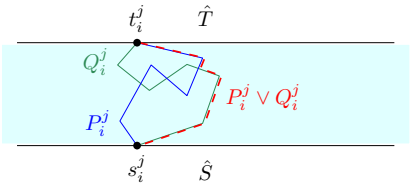


Figure 14 Construction of $P_i^j \vee Q_i^j$.

5.3 General Case

In this subsection, we consider the case when $\mathcal{P} \preceq \mathcal{Q}$ does not necessarily hold. For $i \in [k]$ and $j \in \mathbb{Z}$, define $P_i^j \vee Q_i^j$ as the s_i^j-t_i^j path in $\hat{G}$ with maximal $L(P_i^j \vee Q_i^j)$ subject to $P_i^j \vee Q_i^j \subseteq P_i^j \cup Q_i^j$; see Figure 14. Note that such a path is uniquely determined, $P_i^j \preceq P_i^j \vee Q_i^j$, and $Q_i^j \preceq P_i^j \vee Q_i^j$. Since $\hat{G}$ is periodic, for any $j \in \mathbb{Z}$, $P_i^j \vee Q_i^j$ corresponds to a common s_i-t_i walk $P_i \vee Q_i$ in G. Actually, $\mathcal{P} \vee \mathcal{Q} := (P_1 \vee Q_1, \dots, P_k \vee Q_k)$ is a linkage in G.

▶ **Lemma 16.** *$\mathcal{P} \vee \mathcal{Q}$ is a linkage in G.*

Proof. We first show that $P_i \vee Q_i$ is a path for each $i \in [k]$. Assume to the contrary that $P_i \vee Q_i$ visits a vertex $v \in V$ more than once. Then, for $j \in \mathbb{Z}$, there exist $j_1, j_2 \in \mathbb{Z}$ with $j_1 < j_2$ such that $P_i^j \vee Q_i^j$ contains both v^{j_1} and v^{j_2}. Since the path $P_i^j \vee Q_i^j$ is contained in the subgraph $P_i^j \cup Q_i^j$, without loss of generality, we may assume that P_i^j contains v^{j_2}. This shows that $v^{j_1} \in L(P_i^j) \subseteq L(P_i^j \vee Q_i^j)$, which contradicts that v^{j_1} is contained in $P_i^j \vee Q_i^j$.

We next show that $P_1 \vee Q_1, \dots, P_k \vee Q_k$ are pairwise vertex-disjoint. Assume to the contrary that $P_i \vee Q_i$ and $P_{i'} \vee Q_{i'}$ contain a common vertex $v \in V$ for distinct $i, i' \in [k]$. Since $\hat{G}$ is periodic, there exist $j, j' \in \mathbb{Z}$ such that $P_i^j \vee Q_i^j$ and $P_{i'}^{j'} \vee Q_{i'}^{j'}$ contain v^0 (i.e., the copy of v in R^0). We may assume that (j, i) is smaller than (j', i') in the lexicographical ordering, that is, either $j < j'$ holds or $j = j'$ and $i < i'$ hold. Since $P_i^j \vee Q_i^j \subseteq P_i^j \cup Q_i^j$, we may also assume that $v^0 \in P_i^j$ by changing the roles of P_i^j and Q_i^j if necessary. Then, we obtain $v^0 \in P_i^j \subseteq L(P_{i'}^{j'}) \subseteq L(P_{i'}^{j'} \vee Q_{i'}^{j'})$, which contradicts that v^0 is contained in $P_{i'}^{j'} \vee Q_{i'}^{j'}$. ◀

We also see that $\mu(P_i \vee Q_i, C) = 0$ for $i \in [k]$ by definition, and hence $\mu(\mathcal{P}, \mathcal{P} \vee \mathcal{Q}) = 0$. Since $\mathcal{P} \preceq \mathcal{P} \vee \mathcal{Q}$ and $\mu(\mathcal{P}, \mathcal{P} \vee \mathcal{Q}) = 0$, $\mathcal{P}$ is reconfigurable to $\mathcal{P} \vee \mathcal{Q}$ as described in Section 5.2. Similarly, $\mathcal{Q}$ is reconfigurable to $\mathcal{P} \vee \mathcal{Q}$, which implies that $\mathcal{P} \vee \mathcal{Q}$ is reconfigurable to $\mathcal{Q}$. By combining them, we see that $\mathcal{P}$ is reconfigurable to $\mathcal{Q}$, which completes the proof of the sufficiency in Theorem 14.

Note that the reconfiguration sequence from $\mathcal{P}$ to $\mathcal{Q}$ can be constructed in polynomial time by the discussion in Section 5.2.

6 Concluding Remarks

Although Disjoint Paths Reconfiguration and Disjoint s-t Paths Reconfiguration are decision problems, the proofs for our positive results (Theorems 3, 4, 5, and 14) show that we can find a reconfiguration sequence in polynomial time if it exists.

We leave several open problems for future research. We proved that Disjoint Paths Reconfiguration can be solved in polynomial time when the problem is restricted to the two-face instances. On the other hand, we do not know whether Disjoint Paths Reconfiguration in planar graphs can be solved in polynomial time for fixed k, and even when $k = 2$, if we drop the requirement that inputs are two-face instances.

We did not try to minimize the number of reconfiguration steps when a reconfiguration sequence exists. It is an open problem whether a shortest reconfiguration sequence can be found in polynomial time for Disjoint Paths Reconfiguration restricted to planar two-face instances.

A natural extension of our studies is to consider a higher-genus surface. As a preliminary result, in the full version [20], we give a proof (sketch) to show that when the number k of curves is two, the reconfiguration is *always* possible for any connected orientable closed surface Σ_g of genus $g \geq 1$. Note that this result does not refer to graphs embedded on Σ_g, but only refers to the case when curves can pass through any points on the surface. It is not clear what we can say for Disjoint Paths Reconfiguration for graphs embedded on Σ_g, $g \geq 1$.

References

1 Isolde Adler, Stavros G Kolliopoulos, Philipp Klaus Krause, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M Thilikos. Irrelevant vertices for the planar disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 122:815–843, 2017. doi:10.1016/j.jctb.2016.10.001.

2 Saeed Akhoondian Amiri, Szymon Dudycz, Mahmoud Parham, Stefan Schmid, and Sebastian Wiederrecht. On polynomial-time congestion-free software-defined network updates. In *2019 IFIP Networking Conference, Networking 2019, Warsaw, Poland, May 20-22, 2019*, pages 1–9. IEEE, 2019. doi:10.23919/IFIPNetworking.2019.8816833.

3 Saeed Akhoondian Amiri, Szymon Dudycz, Stefan Schmid, and Sebastian Wiederrecht. Congestion-free rerouting of flows on DAGs. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 143:1–143:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.143.

4 John Asplund, Kossi D. Edoh, Ruth Haas, Yulia Hristova, Beth Novick, and Brett Werner. Reconfiguration graphs of shortest paths. *Discret. Math.*, 341(10):2938–2948, 2018. doi:10.1016/j.disc.2018.07.007.

5 John Asplund and Brett Werner. Classification of reconfiguration graphs of shortest path graphs with no induced 4-cycles. *Discret. Math.*, 343(1):111640, 2020. doi:10.1016/j.disc.2019.111640.

6 Paul S. Bonsma. The complexity of rerouting shortest paths. *Theor. Comput. Sci.*, 510:1–12, 2013. doi:10.1016/j.tcs.2013.09.012.

7 Paul S. Bonsma. Rerouting shortest paths in planar graphs. *Discret. Appl. Math.*, 231:95–112, 2017. doi:10.1016/j.dam.2016.05.024.

8 Glencora Borradaile, Amir Nayyeri, and Farzad Zafarani. Towards single face shortest vertex-disjoint paths in undirected planar graphs. In *Proceedings of 23rd Annual European Symposium on Algorithms (ESA)*, volume 9294 of *Lecture Notes in Computer Science*, pages 227–238, 2015. doi:10.1007/978-3-662-48350-3_20.

9 Éric Colin de Verdière and Alexander Schrijver. Shortest vertex-disjoint two-face paths in planar graphs. *ACM Transactions on Algorithms*, 7(2):1–12, 2011. doi:10.1145/1921659.1921665.

10 Marek Cygan, Dániel Marx, Marcin Pilipczuk, and Michal Pilipczuk. The planar directed k-vertex-disjoint paths problem is fixed-parameter tractable. In *54th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2013)*, pages 197–206, 2013. doi:10.1109/FOCS.2013.29.

11 Samir Datta, Siddharth Iyer, Raghav Kulkarni, and Anish Mukherjee. Shortest k-disjoint paths via determinants. In *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018)*, volume 122, pages 19:1–19:21, 2018. doi:10.4230/LIPIcs.FSTTCS.2018.19.

12 Julian Enoch, Kyle Fox, Dor Mesica, and Shay Mozes. A faster algorithm for maximum flow in directed planar graphs with vertex capacities. In Hee-Kap Ahn and Kunihiko Sadakane, editors, *32nd International Symposium on Algorithms and Computation, ISAAC 2021, December 6-8, 2021, Fukuoka, Japan*, volume 212 of *LIPIcs*, pages 72:1–72:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ISAAC.2021.72.

13 Benson Farb and Dan Margalit. *A primer on mapping class groups*, volume 49 of *Princeton Mathematical Series*. Princeton University Press, Princeton, NJ, 2012.

14 Lester R. Ford and Delbert R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. doi:10.4153/CJM-1956-045-5.

15 Steven Fortune, John Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111–121, 1980. doi:10.1016/0304-3975(80)90009-2.

16 András Frank. Packing paths, circuits, and cuts – A survey. In *Paths, Flows, and VLSI-Layout*, pages 47–100. Springer, 1990.

17 Kshitij Gajjar, Agastya Vibhuti Jha, Manish Kumar, and Abhiruk Lahiri. Reconfiguring shortest paths in graphs. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022*, pages 9758–9766. AAAI Press, 2022.

18 Refael Hassin. On multicommodity flows in planar graphs. *Networks*, 14(2):225–235, 1984. doi:10.1002/net.3230140204.

19 Takehiro Ito, Erik D. Demaine, Nicholas J. A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theor. Comput. Sci.*, 412(12-14):1054–1065, 2011. doi:10.1016/j.tcs.2010.12.005.

20 Takehiro Ito, Yuni Iwamasa, Naonori Kakimura, Yusuke Kobayashi, Shun ichi Maezawa, Yuta Nozaki, Yoshio Okamoto, and Kenta Ozeki. Rerouting planar curves and disjoint paths. arXiv:2210.11778, 2022. doi:10.48550/arXiv.2210.11778.

21 Takehiro Ito, Naonori Kakimura, Naoyuki Kamiyama, Yusuke Kobayashi, and Yoshio Okamoto. Reconfiguration of maximum-weight b-matchings in a graph. *J. Comb. Optim.*, 37(2):454–464, 2019. doi:10.1007/s10878-018-0289-3.

22 Takehiro Ito, Jun Kawahara, Yu Nakahata, Takehide Soh, Akira Suzuki, Junichi Teruyama, and Takahisa Toda. ZDD-based algorithmic framework for solving shortest reconfiguration problems. arXiv:2207.13959, 2022. doi:10.48550/arXiv.2207.13959.

23 Marcin Kaminski, Paul Medvedev, and Martin Milanic. Shortest paths between shortest paths. *Theor. Comput. Sci.*, 412(39):5205–5210, 2011. doi:10.1016/j.tcs.2011.05.021.

24 Haim Kaplan and Yahav Nussbaum. Maximum flow in directed planar graphs with vertex capacities. *Algorithmica*, 61(1):174–189, 2011. doi:10.1007/s00453-010-9436-7.

25 Richard M Karp. On the computational complexity of combinatorial problems. *Networks*, 5(1):45–68, 1975. doi:10.1002/net.1975.5.1.45.

26 Samir Khuller and Joseph Naor. Flow in planar graphs with vertex capacities. *Algorithmica*, 11(3):200–225, 1994. doi:10.1007/BF01240733.

27 Yusuke Kobayashi and Kensuke Otsuki. Max-flow min-cut theorem and faster algorithms in a circular disk failure model. In *2014 IEEE Conference on Computer Communications (INFOCOM)*, pages 1635–1643, 2014. doi:10.1109/INFOCOM.2014.6848100.

28 Yusuke Kobayashi and Christian Sommer. On shortest disjoint paths in planar graphs. *Discrete Optimization*, 7(4):234–245, 2010. doi:10.1016/j.disopt.2010.05.002.

29 Mark R Kramer. The complexity of wire-routing and finding minimum area layouts for arbitrary VLSI circuits. *Advances in Computing Research*, 2:129–146, 1984.

30 James F Lynch. The equivalence of theorem proving and the interconnection problem. *ACM SIGDA Newsletter*, 5(3):31–36, 1975. doi:10.1145/1061425.1061430.

31 Colin J. H. McDiarmid, Bruce A. Reed, Alexander Schrijver, and F. Bruce Shepherd. Non-interfering network flows. In *Proceedings of the Third Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 245–257, 1992. doi:10.1007/3-540-55706-7_21.

32 Colin J. H. McDiarmid, Bruce A. Reed, Alexander Schrijver, and F. Bruce Shepherd. Induced circuits in planar graphs. *Journal of Combinatorial Theory, Series B*, 60(2):169–176, 1994. doi:10.1006/jctb.1994.1011.

33 Karl Menger. Zur allgemeinen Kurventheorie. *Fundamenta Mathematicae*, 10:96–115, 1927.

34 Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018. doi:10.3390/a11040052.

35 Haruko Okamura. Multicommodity flows in graphs. *Discrete Applied Mathematics*, 6(1):55–62, 1983. doi:10.1016/0166-218X(83)90100-2.

36 Haruko Okamura and Paul D. Seymour. Multicommodity flows in planar graphs. *Journal of Combinatorial Theory, Series B*, 31(1):75–81, 1981. doi:10.1016/S0095-8956(81)80012-3.

37 Kensuke Otsuki, Yusuke Kobayashi, and Kazuo Murota. Improved max-flow min-cut algorithms in a circular disk failure model with application to a road network. *Eur. J. Oper. Res.*, 248(2):396–403, 2016. doi:10.1016/j.ejor.2015.07.035.

38 Bruce Reed. Rooted routing in the plane. *Discrete Applied Mathematics*, 57:213–227, 1995. doi:10.1016/0166-218X(94)00104-L.

39 Bruce Reed, Neil Robertson, Alexander Schrijver, and Paul D. Seymour. Finding disjoint trees in planar graphs in linear time. In *Contemporary Mathematics 147*, pages 295–301. American Mathematical Society, 1993. doi:10.1090/conm/147/01180.

40 Neil Robertson and Paul D. Seymour. Graph minors. VI. Disjoint paths across a disc. *Journal of Combinatorial Theory, Series B*, 41(1):115–138, 1986. doi:10.1016/0095-8956(86)90031-6.

41 Neil Robertson and Paul D. Seymour. Graph minors. VII. Disjoint paths on a surface. *Journal of Combinatorial Theory, Series B*, 45(2):212–254, 1988. doi:10.1016/0095-8956(88)90070-6.

42 Neil Robertson and Paul D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of combinatorial theory, Series B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

43 Alexander Schrijver. Finding k disjoint paths in a directed planar graph. *SIAM Journal on Computing*, 23(4):780–788, 1994. doi:10.1137/S0097539792224061.

44 Paul D. Seymour. Disjoint paths in graphs. *Discrete Mathematics*, 29:293–309, 1980. doi:10.1016/0012-365X(80)90158-2.

45 Yossi Shiloach. A polynomial solution to the undirected two paths problem. *Journal of the ACM*, 27(3):445–456, 1980. doi:10.1145/322203.322207.

46 Anand Srinivas and Eytan Modiano. Finding minimum energy disjoint paths in wireless ad-hoc networks. *Wireless Networks*, 11(4):401–417, 2005. doi:10.1007/s11276-005-1765-0.

47 Hitoshi Suzuki, Takehiro Akama, and Takao Nishizeki. Algorithms for finding internally disjoint paths in a planar graph. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 72(10):55–67, 1989. doi:10.1002/ecjc.4430721006.

48 Hitoshi Suzuki, Takehiro Akama, and Takao Nishizeki. Finding Steiner forests in planar graphs. In *Proceedings of the first annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 444–453, 1990.

49 Carsten Thomassen. 2-linked graphs. *European Journal of Combinatorics*, 1:371–378, 1980. doi:10.1016/S0195-6698(80)80039-4.

50 Jan van den Heuvel. The complexity of change. In Simon R. Blackburn, Stefanie Gerke, and Mark Wildon, editors, *Surveys in Combinatorics 2013*, volume 409 of *London Mathematical Society Lecture Note Series*, pages 127–160. Cambridge University Press, 2013. doi:10.1017/CBO9781139506748.005.

51 Yipu Wang. Max flows in planar graphs with vertex capacities. *ACM Trans. Algorithms*, 18(1), 2022. doi:10.1145/3504032.

52 Marcin Wrochna. Reconfiguration in bounded bandwidth and tree-depth. *J. Comput. Syst. Sci.*, 93:1–10, 2018. doi:10.1016/j.jcss.2017.11.003.

Hardness of Finding Combinatorial Shortest Paths on Graph Associahedra

Takehiro Ito
Graduate School of Information Sciences, Tohoku University, Japan

Naonori Kakimura
Faculty of Science and Technology, Keio University, Yokohama, Japan

Naoyuki Kamiyama
Institute of Mathematics for Industry, Kyushu University, Fukouka, Japan

Yusuke Kobayashi
Research Institute for Mathematical Sciences, Kyoto University, Japan

Shun-ichi Maezawa
Department of Mathematics, Tokyo University of Science, Japan

Yuta Nozaki
Faculty of Environment and Information Sciences, Yokohama National University, Japan
SKCM^2, Hiroshima University, Japan

Yoshio Okamoto
Graduate School of Informatics and Engineering, The University of Electro-Communications, Tokyo, Japan

Abstract

We prove that the computation of a combinatorial shortest path between two vertices of a graph associahedron, introduced by Carr and Devadoss, is NP-hard. This resolves an open problem raised by Cardinal. A graph associahedron is a generalization of the well-known associahedron. The associahedron is obtained as the graph associahedron of a path. It is a tantalizing and important open problem in theoretical computer science whether the computation of a combinatorial shortest path between two vertices of the associahedron can be done in polynomial time, which is identical to the computation of the flip distance between two triangulations of a convex polygon, and the rotation distance between two rooted binary trees. Our result shows that a certain generalized approach to tackling this open problem is not promising. As a corollary of our theorem, we prove that the computation of a combinatorial shortest path between two vertices of a polymatroid base polytope cannot be done in polynomial time unless P = NP. Since a combinatorial shortest path on the matroid base polytope can be computed in polynomial time, our result reveals an unexpected contrast between matroids and polymatroids.

2012 ACM Subject Classification Mathematics of computing → Combinatorics; Theory of computation → Problems, reductions and completeness

Keywords and phrases Graph associahedra, combinatorial shortest path, NP-hardness, polymatroids

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.82

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2304.14782`

Funding *Takehiro Ito*: JSPS KAKENHI Grant Numbers JP18H04091, JP19K11814, JP20H05793.
Naonori Kakimura: JSPS KAKENHI Grant Numbers JP20H05795, JP21H03397, JP22H05001.
Naoyuki Kamiyama: JSPS KAKENHI Grant Number JP20H05795.
Yusuke Kobayashi: JSPS KAKENHI Grant Numbers JP20K11692, JP20H05795, JP22H05001.
Shun-ichi Maezawa: JSPS KAKENHI Grant Numbers JP20H05795, JP22K13956.
Yuta Nozaki: JSPS KAKENHI Grant Numbers JP20H05795, JP20K14317, JP23K12974.
Yoshio Okamoto: JSPS KAKENHI Grant Numbers JP20H05795, JP20K11670, JP23K10982.

Acknowledgements This work was motivated by the talks of Vincent Pilaud and Jean Cardinal at Workshop "Polytope Diameter and Related Topics," held online on September 2, 2022. We thank them for inspiration. We are also grateful to Yuni Iwamasa and Kenta Ozeki for the discussion and to the anonymous reviewers for their helpful comments.

1 Introduction

Graph associahedra were introduced by Carr and Devadoss [8]. These polytopes generalize associahedra. In an associahedron, each vertex corresponds to a binary tree over a set of n elements, and each edge corresponds to a rotation operation between two binary trees. For the historical account of associahedra, see the introduction of the paper by Ceballos, Santos, and Ziegler [9].

A binary tree can be obtained from a labeled path. Let $V = \{1, 2, \ldots, n\}$ be the set of vertices of the path, and $E = \{\{i, i+1\} \mid 1 \leq i \leq n-1\}$ be the set of edges of the path. To construct a labeled binary tree, we choose an arbitrary vertex from the path. Let it be $i \in V$. Then, the removal of i from the path results in at most two connected components: the left subpath and the right subpath, which may be empty. Then, in the corresponding binary tree, we set i as a root, and append recursively a binary tree of the left subpath as a left subtree and a binary tree of the right subpath as a right subtree. Note that in this construction, each node of the binary tree is labeled by a vertex of the path.

In the construction of graph associahedra, we follow the same idea. Since we are only interested in the graph structure of graph associahedra in this work, we only describe their vertices and edges. To define a graph associahedron, we first fix a connected undirected graph $G = (V, E)$.[1] Then, in the G-associahedron, the vertices correspond to the so-called elimination trees of G, and the edges correspond to swap operations between two elimination trees. The following description follows that of Cardinal, Merino, and Mütze [5].

An *elimination tree* of a connected undirected graph $G = (V, E)$ is a rooted tree defined as follows. It has V as the vertex set and is composed of a root $v \in V$ that has as children elimination trees for each connected component of $G - v$ (Figure 1). A swap from an elimination tree T to another elimination tree T' of G is defined as follows. Let v be a non-root vertex of T, and let u be the parent of v in T. Denote by H the subgraph of G induced by the subtree rooted at v in T. Then, the swap of u with v transforms T to T' as follows. (1) The tree T' has v as the parent of u, and T' has v as a child of the parent of u in T. (2) The subtrees rooted at u in T remain subtrees rooted at u in T'. (3) A subtree S rooted at v in T remains a subtree rooted at v in T', unless the vertices of S belong to the same connected component of $H - v$ as u, in which case S becomes a subtree rooted at u in T'. The G-associahedron for a claw G is given in Figure 2. Note that a swap operation is reversible.

In this paper, among graph properties of graph associahedra, we concentrate on the computation of a combinatorial shortest path (i.e., the graph-theoretic distance) between two vertices of the polytope, which we call the combinatorial shortest path problem on graph associahedra. In this problem, we are given a graph G and two elimination trees T, T' of G, and want to compute the shortest length of a graph-theoretic path from T to T' on the G-associahedron. In the literature, we only find the studies in the case where G is a complete graph or (a generalization of) a star. When G is a complete graph, the G-associahedron is called a permutahedron, and each of its vertices corresponds to a permutation. Since an edge corresponds to an adjacent transposition, the graph-theoretic distance between two vertices is equal to the number of inversions between the corresponding permutations. This can be computed in polynomial time. When G is a star, the G-associahedron is called stellohedron [26]. Recently, Cardinal, Pournin, and Valencia-Pabon [7] gave a polynomial-

[1] A graph associahedron can also be defined for disconnected graphs, but in this paper, we concentrate on connected graphs.

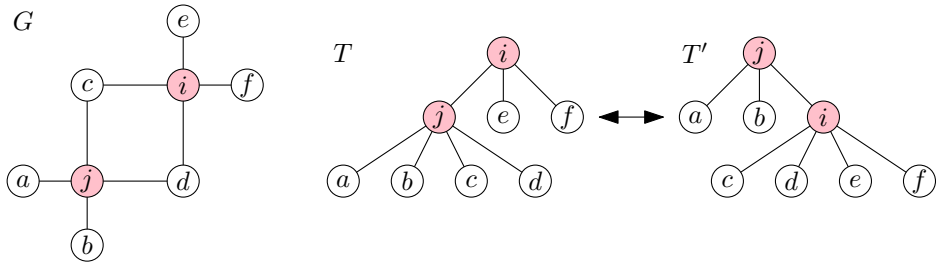


Figure 1 An example of elimination trees. Two trees T and T' are elimination trees of the graph G. The tree T' is obtained from T by the swap of i with j. The example is borrowed from Cardinal, Merino, and Mütze [5].

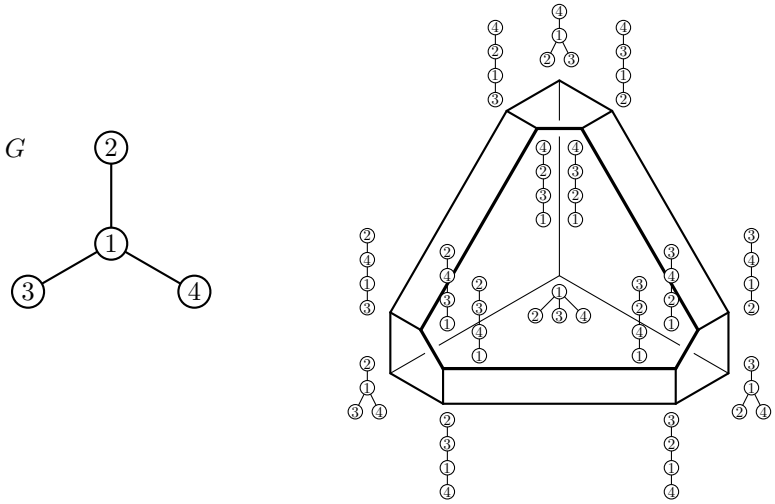


Figure 2 An example of a graph associahedron. Each vertex of the polytope corresponds to an elimination tree of the graph G.

time algorithm to compute a combinatorial shortest path on stellohedra, and they generalize the algorithm when G is a complete split graph (i.e., a graph obtained from a star by replacing the center vertex with a clique).

On the other hand, it is a tantalizing open problem whether a combinatorial shortest path can be computed in polynomial time when G is a path. In this case, the graph-theoretic distance corresponds to the rotation distance between two binary trees. By Catalan correspondences, this is equivalent to the flip distance between two triangulations of a convex polygon. A possible strategy to resolve this open problem is to generalize the problem and solve the generalized problem. In our case, a generalization is achieved by considering graph associahedra for general graphs.

Our main result states that the combinatorial shortest path problem on G-associahedra is NP-hard when G is also given as part of the input. This implies that the strategy mentioned above is bound to fail.

First, we formally state the problem Combinatorial Shortest Path on Graph Associahedra as follows.

Combinatorial Shortest Path on Graph Associahedra
Input: A graph G and two elimination trees $T_{\mathrm{ini}}, T_{\mathrm{tar}}$ of G
Output: The distance between T_{ini} and T_{tar} on the graph of the G-associahedron

Our first theorem states the NP-hardness of Combinatorial Shortest Path on Graph Associahedra. This solves an open problem raised by Cardinal (see [3, Section 4.2]).

▶ **Theorem 1.** *Combinatorial Shortest Path on Graph Associahedra is* NP*-hard.*

Theorem 1 yields the following corollary, which is related to polymatroids introduced by Edmonds [13]. A pair (U, ρ) of a finite set U and a function $\rho\colon 2^U \to \mathbb{R}$ is called a *polymatroid* if ρ satisfies the following conditions: (P1) $\rho(\emptyset) = 0$; (P2) if $X \subseteq Y \subseteq U$, then $\rho(X) \leq \rho(Y)$; (P3) if $X, Y \subseteq U$, then $\rho(X \cup Y) + \rho(X \cap Y) \leq \rho(X) + \rho(Y)$. The function ρ is called the *rank function* of the polymatroid (U, ρ).

For a polymatroid (U, ρ), we define the *base polytope* of (U, ρ) as

$$\mathbf{B}(\rho) := \{x \in \mathbb{R}^U \mid x(X) \leq \rho(X)\,(\forall\, X \subseteq U),\, x(U) = \rho(U)\},$$

where we define $x(X) := \sum_{u \in X} x(u)$ for each subset $X \subseteq U$. Then, $\mathbf{B}(\rho)$ is a polytope because $0 \leq \rho(U) - \rho(U \setminus \{u\}) = x(U) - \rho(U \setminus \{u\}) \leq x(u) \leq \rho(\{u\})$ for every element $e \in E$.

A polymatroid is seen as a polyhedral generalization of a matroid. For example, the greedy algorithm for matroids can be treated as an algorithm to maximize a linear function over the base polytope of a matroid,[2] and the algorithm is readily generalized to the base polytope of a polymatroid. A lot of combinatorial properties of the base polytopes of matroids also hold for the base polytopes of polymatroids. Since it is known that a combinatorial shortest path on the base polytope of a matroid can be computed in polynomial time [18], we are interested in generalizing this result to polymatroids, which leads to the following problem definition.

Combinatorial Shortest Path on Polymatroids
Input: An oracle access to a polymatroid (U, ρ) and two extreme points $x_{\mathrm{ini}}, x_{\mathrm{tar}}$ of the base polytope $\mathbf{B}(\rho)$
Output: The distance between x_{ini} and x_{tar} on $\mathbf{B}(\rho)$

We note that a polymatroid (U, ρ) is given as an oracle access that returns the value $\rho(X)$ for any set $X \subseteq U$. The running time of an algorithm is also measured in terms of the number of oracle calls. This is a standard assumption when we deal with polymatroids [16] since if we would input the function ρ as a table of the values $\rho(X)$ for all $X \subseteq U$, then it would already take at least $2^{|U|}$ space. We also note that the adjacency of two extreme points of the base polytope of a polymatroid can be tested in polynomial time [31].

The next theorem states that this problem is hard, which is proved as a corollary of Theorem 1, and reveals an unexpected contrast between matroids and polymatroids.

▶ **Theorem 2.** *There exists no polynomial-time algorithm for Combinatorial Shortest Path on Polymatroid unless* P = NP*.*

Our proof relies on the fact that graph associahedra can be realized as the base polytopes of some polymatroids [26]. However, we need careful treatment since in the reduction we require the rank function of our polymatroid to be evaluated in polynomial time. To this end, we give an explicit inequality description of the realization of a graph associahedron due to Devadoss [12],[3] which is indeed the base polytope of a polymatroid.

[2] This can further be seen as a generalization of Kruskal's algorithm for the minimum spanning tree problem.

[3] We note here that the original definition of a graph associahedron by Carr and Devadoss [8] does not give explicit vertex coordinates of the polytope. Therefore, we rely on the realization by Devadoss [12] who gave the explicit vertex coordinates.

Related Work

Paths on polytopes have been studied thoroughly. One of the initial motivations for this research direction is to design and understand path-following algorithms for linear optimization such as simplex methods. In his chapter of *Handbook of Discrete and Computational Geometry* [17], Kalai stated as an open problem "Find an efficient routing algorithm for convex polytopes." Here, a routing algorithm means one that finds a short path from a given initial vertex to a given target vertex.

Paths on graph associahedra have been receiving much attention. The diameter is perhaps the most frequently studied quantity, which is defined as the maximum distance between two vertices of the polytope. A famous result by Sleator, Tarjan, and Thurston [30] states that the diameter of the $(n-1)$-dimensional associahedron (i.e., a graph associahedron of an n-vertex path) is at most $2n-6$ when $n \geq 11$ and this bound is tight for all sufficiently large n. This bound is refined by Pournin [27], who proved that the diameter of the $(n-1)$-dimensional associahedron is exactly $2n-6$ when $n \geq 11$.

For a general n-vertex graph G, Manneville and Pilaud [24] proved that the diameter of G-associahedron is at most $\binom{n}{2}$ and at least $\max\{m, 2n-20\}$, when m is the number of edges of G. The upper bound is attained by the case where G is a complete graph (i.e., the G-associahedron is a permutahedron). When G is an n-vertex star (i.e., $K_{1,n-1}$), $n \geq 6$, Manneville and Pilaud [24] showed that the diameter is $2n-2$. When G is a cycle (i.e., the polytope is a cyclohedron), Pournin [28] gave the asymptotically exact diameter. When G is a tree, Manneville and Pilaud [24] gave the upper bound $O(n \log n)$ while Cardinal, Langerman, and Pérez-Lantero [4] gave an example of trees for which the diameter is $\Omega(n \log n)$ (such an example is chosen as a complete binary tree). Cardinal, Pournin, and Valencia-Pabon [6] proved that the diameter is $\Theta(m)$ for m-edge trivially perfect graphs, and gave upper and lower bounds for the diameter in terms of treewidths, pathwidths, and treedepths of graphs. Berendsohn [2] proved that the diameter is $\Theta(n+mH)$ for a caterpillar with n spine vertices, m leg vertices, and the Shannon entropy H of the so-called leg distribution.

To the authors' knowledge, the complexity of computing the diameter of graph associahedra has not been investigated. When polytopes are not restricted to graph associahedra, a few hardness results have been known. Frieze and Teng [15] proved that computing the diameter of a polytope, given by inequalities, is weakly NP-hard. Sanità [29] proved that computing the diameter of the fractional matching polytope of a given graph is strongly NP-hard. Kaibel and Pfetsch [19] raised an open problem about the complexity of computing the diameter of simple polytopes.

The computation of a combinatorial shortest path on a G-associahedron has also been studied. It is a long-standing open problem whether a combinatorial shortest path in an associahedron (i.e., a G-associahedron when G is a path) can be computed in polynomial time. Polynomial-time algorithms are only known when G is a complete graph (folklore), a star, or a complete split graph (Cardinal, Pournin, and Valencia-Pabon [7]). When G is a path, a polynomial-time approximation algorithm of factor two [11] and fixed-parameter algorithms when the distance is a parameter [10, 20, 21, 23] are known.

Since a combinatorial shortest path on an associahedron is equivalent to a shortest flip sequence of triangulations of a convex polygon, the computation of a shortest flip sequence of triangulations has been studied in more general setups. For triangulations of point sets, the problem is NP-hard [22] and even APX-hard [25]. For triangulations of simple polygons, the problem is also NP-hard [1].

Elimination trees have appeared in a lot of branches of mathematics and computer science. For a good summary, see Cardinal, Merino, and Mütze [5].

Technique Overview

To prove the hardness of the combinatorial shortest path problem on graph associahedra, we first consider a "weighted" version of the combinatorial shortest path problem on graph associahedra, which is newly introduced in this paper for our reduction. In this problem, each vertex of a given graph has a positive weight, and the swap of two vertices incurs the weight that is defined as the product of the weights of these two vertices. The weight of a swap sequence is defined as the sum of weights of swaps in the sequence. As our intermediate theorem, we prove that this weighted version is strongly NP-hard.

To this end, we reduce the NP-hard problem of finding a balanced minimum s-t cut in a graph [14] to the weighted version of the combinatorial shortest path problem on graph associahedra. In the balanced minimum s-t cut problem, we want to determine whether there exists a minimum s-t cut of a given graph G that is a bisection of the vertex set. In the reduction, we construct a vertex-weighted graph H from G and two elimination trees T_{ini}, T_{tar} of H. The weighted graph H is constructed by replacing s and t by large cliques, subdividing each edge, and duplicating each vertex; the weights are assigned so that the subdivision vertices receive small weights, and duplicated vertices and vertices in large cliques receive large weights. Elimination trees T_{ini} and T_{tar} are constructed so that swapping two vertices of large weights is forbidden and identifying a few vertices of small weights (that corresponds to a balanced minimum s-t cut of G) gives a shortest path.

In the second step, we reduce the weighted version to the original unweighted version of the problem. To this end, a vertex of weight w is replicated by a clique of size w. We want to make sure that a swap of the vertices u, v of weights $w(u), w(v)$, respectively in the weighted instance is mapped to consecutive $w(u) \cdot w(v)$ swaps of the vertices of cliques that correspond to u and v in the constructed unweighted instance. This is proved via the useful operation of projections combined with the averaging argument.

2 Preliminaries

For a positive integer k, let $[k]$ denote $\{1, 2, \ldots, k\}$.

For a graph $G = (V, E)$ and two elimination trees T_{ini} and T_{tar} of G, we say that a sequence $\mathbf{T} = \langle T_0, T_1, \ldots, T_\ell \rangle$ of elimination trees of G is a *reconfiguration sequence from* T_{ini} *to* T_{tar} if $T_0 = T_{\mathrm{ini}}$, $T_\ell = T_{\mathrm{tar}}$, and T_i is obtained from T_{i-1} by applying a single swap operation for $i \in [\ell]$. We sometimes regard $\mathbf{T}$ as a sequence of swap operations if no confusion may arise. The length of $\mathbf{T}$ is defined as the number ℓ of swaps in $\mathbf{T}$, which we denote length($\mathbf{T}$). Then, Combinatorial Shortest Path on Graph Associahedra is the problem of finding a reconfiguration sequence $\mathbf{T}$ from T_{ini} to T_{tar} that minimizes length($\mathbf{T}$).

When $u \in V$ is a child of $v \in V$ in an elimination tree T, an operation swapping u and v is represented by $\mathbf{swap}(u, v)$. Note that we distinguish $\mathbf{swap}(u, v)$ and $\mathbf{swap}(v, u)$. For an elimination tree T and for a vertex $v \in V(T)$, let $\mathbf{anc}_T(v)$ (resp. $\mathbf{des}_T(v)$) denote the set of all ancestors (descendants) of v in T. Note that $u \in \mathbf{anc}_T(v)$ if and only if $v \in \mathbf{des}_T(u)$. Note also that neither $\mathbf{anc}_T(v)$ nor $\mathbf{des}_T(v)$ contains v. We say that distinct vertices u and v are *comparable* in T if $u \in \mathbf{anc}_T(v)$ or $v \in \mathbf{anc}_T(u)$. Otherwise, they are called *incomparable* in T. A linear ordering $\prec$ on V defines an elimination tree T uniquely so that $u \in \mathbf{anc}_T(v)$ implies $u \prec v$.

Let $G = (V, E)$ be an undirected graph. For $X \subseteq V$, let $\delta_G(X)$ denote the set of edges between X and $V \setminus X$. For $s, t \in V$, we say that $X \subseteq V$ is an *s-t cut* if $s \in X$ and $t \notin X$. An edge set $F \subseteq E$ is called an *s-t cut set* if $F = \delta_G(X)$ for some s-t cut $X \subseteq V$. A *minimum s-t cut* is an s-t cut X minimizing $|\delta_G(X)|$. For $X \subseteq V$, let $G[X]$ denote the subgraph induced by X and let $E[X]$ denote its edge set.

3 Hardness of the Weighted Problem

We consider a weighted variant of Combinatorial Shortest Path on Graph Associahedra, which we call Weighted Combinatorial Shortest Path on Graph Associahedra. In the problem, we are given a graph $G = (V, E)$, two elimination trees T_{ini} and T_{tar}, and a weight function $w\colon V \to \mathbb{Z}_{>0}$. For $u, v \in V$, the *weight* of $\mathbf{swap}(u, v)$ is defined as $w(u) \cdot w(v)$. This value is sometimes denoted by $w(\mathbf{swap}(u, v))$. The *weighted length* (or simply the *weight*) of a reconfiguration sequence $\mathbf{T}$ is defined as the total weight of swaps in $\mathbf{T}$, which we denote by $\mathrm{length}_w(\mathbf{T})$. The objective of Weighted Combinatorial Shortest Path on Graph Associahedra is to find a reconfiguration sequence $\mathbf{T}$ from T_{ini} to T_{tar} that minimizes $\mathrm{length}_w(\mathbf{T})$.

In this section, we show that the weighted variant is strongly NP-hard.

▶ **Theorem 3.** *Weighted Combinatorial Shortest Path on Graph Associahedra is strongly* NP*-hard, that is, it is* NP*-hard even when the input size is* $\sum_{v \in V} w(v)$.

3.1 Reduction

To show Theorem 3, we reduce Balanced Minimum s-t Cut to Weighted Combinatorial Shortest Path on Graph Associahedra. In Balanced Minimum s-t Cut, the input consists of a connected graph $G = (V, E)$ with $s, t \in V$, and the objective is to determine whether G contains a minimum s-t cut X with $|X| = |V \setminus X|$. Without loss of generality, we may assume that $|V|$ is even. Let $V = \{s, t, v_1, v_2, \ldots, v_{2n}\}$ and $E = \{e_1, \ldots, e_m\}$, where $|V| = 2n + 2$ and $|E| = m$. It is known that Balanced Minimum s-t Cut is NP-hard [14]. For an instance of Balanced Minimum s-t Cut, we construct an instance of Weighted Combinatorial Shortest Path on Graph Associahedra as follows.

Let N be a sufficiently large integer (e.g., $N = 10n^3m$). We first subdivide each edge $e \in E$ by introducing a new vertex u_e. Then, for each $v \in V$, we introduce a copy v' of v. We replace s with a clique $\{s_1, \ldots, s_{N^3}\}$ of size N^3 and replace t with another clique $\{t_1, \ldots, t_{N^3}\}$ of size N^3. Let H be the obtained graph. Formally, the graph $H = (V(H), E(H))$ is defined as follows:

$$
\begin{aligned}
V(H) &= (V \setminus \{s, t\}) \cup \{v' \mid v \in V\} \cup \{u_e \mid e \in E\} \cup \{s_1, \ldots, s_{N^3}\} \cup \{t_1, \ldots, t_{N^3}\},\\
E(H) &= \{\{v, u_e\} \mid v \in V \setminus \{s, t\},\ e \in \delta_G(v)\} \cup \{\{v', u_e\} \mid v \in V,\ e \in \delta_G(v)\}\\
&\quad \cup \{\{s_i, s_j\} \mid i, j \in [N^3],\ i \neq j\} \cup \{\{t_i, t_j\} \mid i, j \in [N^3],\ i \neq j\}\\
&\quad \cup \{\{s_i, u_e\} \mid i \in [N^3],\ e \in \delta_G(s)\} \cup \{\{t_i, u_e\} \mid i \in [N^3],\ e \in \delta_G(t)\}.
\end{aligned}
$$

Define $w\colon V(H) \to \mathbb{Z}_{>0}$ as follows:

$$
\begin{aligned}
w(v) &= N && (v \in V \setminus \{s, t\}),\\
w(v') &= N^8 && (v \in V),\\
w(u_e) &= 1 && (e \in E),\\
w(s_i) &= w(t_i) = N^4 && (i \in [N^3]).
\end{aligned}
$$

The initial elimination tree T_{ini} is defined by the following linear ordering:

$$
\begin{aligned}
&v_1 \prec \cdots \prec v_{2n} \prec s_1 \prec t_1 \prec s_2 \prec t_2 \prec \cdots \prec s_{N^3} \prec t_{N^3}\\
&\quad \prec u_{e_1} \prec \cdots \prec u_{e_m} \prec v'_1 \prec \cdots \prec v'_{2n} \prec s' \prec t'.
\end{aligned}
$$

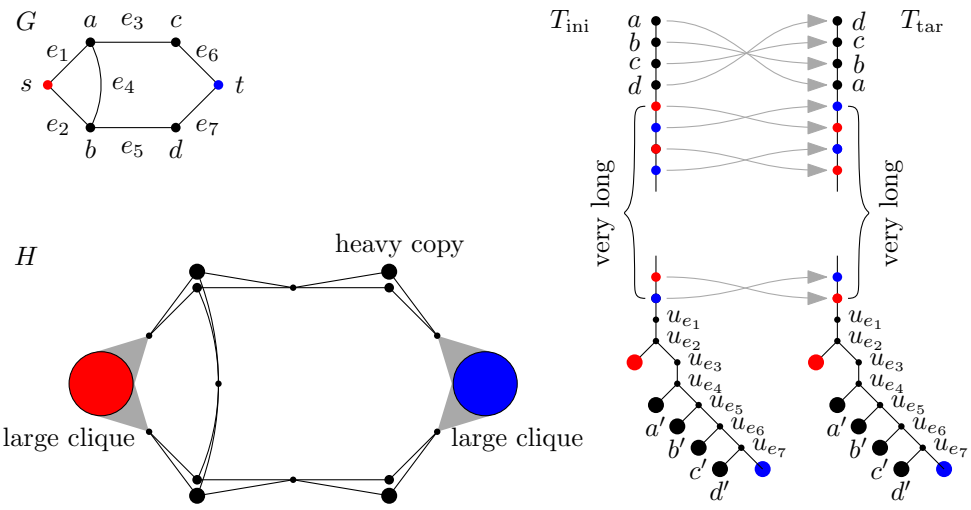


Figure 3 Reduction for Theorem 3.

Note that, in T_{ini}, the vertices $v_1, \ldots, v_{2n}, s_1, t_1, s_2, t_2, \ldots, s_{N^3}, t_{N^3}$ are aligned on a path, while the other elements are not necessarily aligned sequentially. The target elimination tree T_{tar} is the elimination tree defined by the following linear ordering:

$$v_{2n} \prec \cdots \prec v_1 \prec t_1 \prec s_1 \prec t_2 \prec s_2 \prec \cdots \prec t_{N^3} \prec s_{N^3} \\ \prec u_{e_1} \prec \cdots \prec u_{e_m} \prec v'_1 \prec \cdots \prec v'_{2n} \prec s' \prec t'.$$

We consider an instance $(H, w, T_{\mathrm{ini}}, T_{\mathrm{tar}})$ of Weighted Combinatorial Shortest Path on Graph Associahedra. In this instance, we reverse the ordering of the first $2n$ elements and reverse the ordering of s_i and t_i for each i. See Figure 3 for an illustration.

To prove Theorem 1, it suffices to show the following proposition.

▶ **Proposition 4.** *Let λ be the minimum size of an s-t cut set in G. There is a reconfiguration sequence from T_{ini} to T_{tar} of weight less than $4\lambda N^7 + (n^2 - n + 1)N^2$ if and only if G has a minimum s-t cut X with $|X| = |V \setminus X|$.*

3.2 Proof of Proposition 4

Sufficiency ("if" part)

Suppose that G has a minimum s-t cut X with $|X| = |V \setminus X| = n + 1$. Let $U = \{u_e \mid e \in \delta_G(X)\}$. Note that $|U| = |\delta_G(X)| = \lambda$. Starting from T_{ini}, we swap an element in U and its parent repeatedly so that we obtain an elimination tree T_1 in which each element in U is an ancestor of $V(H) \setminus U$. See Figure 4. The total weight of swaps from T_{ini} to T_1 is at most $|U|(2nN + 2N^7 + m)$. Since $G - \delta_G(X)$ consists of two connected components, so does $H - U$. Thus, $T_1 - U$ consists of two elimination trees T_s and T_t such that T_s contains $(X \setminus \{s\}) \cup \{s_1, \ldots, s_{N^3}\} \cup \{u_e \mid e \in E[X]\} \cup \{v' \mid v \in X\}$ and T_t contains $((V \setminus X) \setminus \{t\}) \cup \{t_1, \ldots, t_{N^3}\} \cup \{u_e \mid e \in E[V \setminus X]\} \cup \{v' \mid v \in V \setminus X\}$.

In T_s, by swapping u and v for every pair of $u, v \in X \setminus \{s\}$, we obtain an elimination tree in which v_i is an ancestor of v_j for $v_i, v_j \in X \setminus \{s\}$ with $i > j$. The total weight of these swaps is $\binom{|X|-1}{2} \cdot N^2$. Similarly, by applying swaps with weight $\binom{|V \setminus X|-1}{2} \cdot N^2$ to T_t, we obtain an elimination tree in which v_i is an ancestor of v_j for $v_i, v_j \in (V \setminus X) \setminus \{t\}$ with $i > j$. Let T_2 be the elimination tree obtained from T_1 by applying these operations.

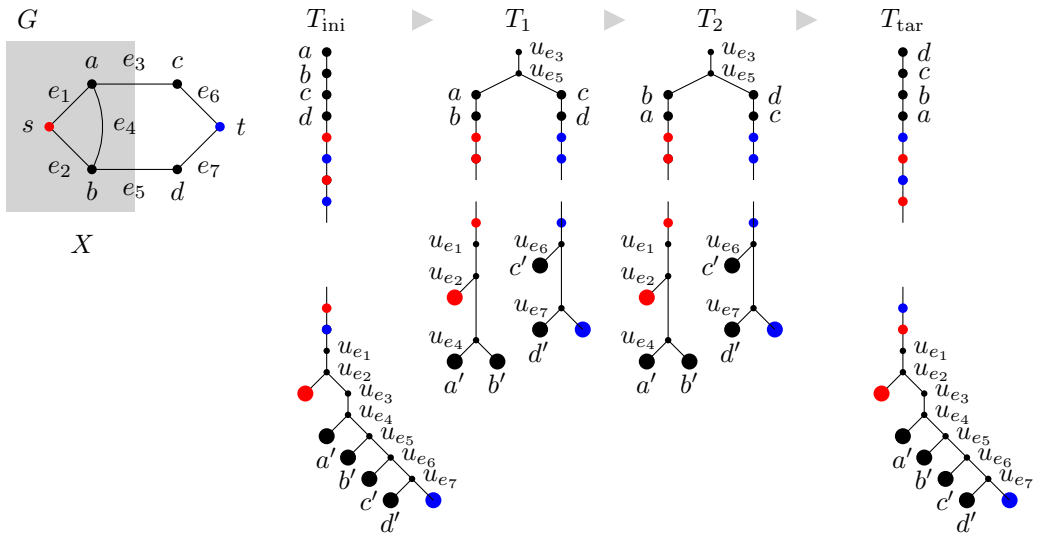


Figure 4 A reconfiguration sequence from T_{ini} to T_{tar}.

Starting from T_2, we swap an element in U and its child repeatedly so that we obtain an elimination tree T_{tar}. This can be done by applying swaps whose total weight is at most $|U|(2nN + 2N^7 + m)$.

Therefore, the total weight of the above swaps from T_{ini} to T_{tar} is at most

$$\begin{aligned} &2|U|(2nN + 2N^7 + m) + \binom{|X| - 1}{2} \cdot N^2 + \binom{|V \setminus X| - 1}{2} \cdot N^2 \\ &\quad = 4\lambda N^7 + n(n-1)N^2 + 4\lambda nN + 2\lambda m \\ &\quad < 4\lambda N^7 + (n^2 - n + 1)N^2, \end{aligned}$$

where we note that $|U| = \lambda$ and $|X| = |V \setminus X| = n + 1$. This shows the sufficiency.

Necessity ("only if" part)

Let $\mathbf{T}$ be a reconfiguration sequence from T_{ini} to T_{tar} whose weight is less than $4\lambda N^7 + (n^2 - n + 1)N^2$. Since this weight is less than N^8, we observe the following.

▶ **Observation 5.** *For $v \in V$, vertex v' is not swapped with other vertices in $\mathbf{T}$. For $i, j \in [N^3]$, none of $\mathbf{swap}(s_i, s_j)$, $\mathbf{swap}(t_i, t_j)$, $\mathbf{swap}(s_i, t_j)$, and $\mathbf{swap}(t_i, s_j)$ is applied in $\mathbf{T}$.*

By Observation 5, we cannot swap s_1 and t_1 directly, and hence $\mathbf{T}$ contains an elimination tree T^* in which s_1 and t_1 are incomparable. Then, there exists a vertex set $V^* \subseteq V(H)$ such that s_1 and t_1 are contained in different connected components of $H - V^*$, and each vertex in V^* is an ancestor of s_1 and t_1 in T^*. By Observation 5 again, V^* does not contain v' for $v \in V$, that is, $V^* \subseteq V \cup \{u_e \mid e \in E\}$. Note that removing $V^* \cap V$ does not affect the connectedness of H since each vertex $v \in V^* \cap V$ has its copy v' in H. Let

$$F := \{e \in E \mid u_e \in V^*\}.$$

Then, s and t are contained in different connected components of $G - F$, i.e., F contains an s-t cut set in G.

Since removing V does not affect the connectedness of H, we also observe the following.

▶ **Observation 6.** *Let T and T' be elimination trees in $\mathbf{T}$ and let $e_1, e_2 \in E$ be distinct edges. If $u_{e_1} \in \mathbf{anc}_T(u_{e_2})$ and $u_{e_2} \in \mathbf{anc}_{T'}(u_{e_1})$, then $\mathbf{swap}(u_{e_3}, u_{e_4})$ is applied for some $e_3, e_4 \in E$ (possibly $\{e_1, e_2\} \cap \{e_3, e_4\} \neq \emptyset$) between T and T'.*

We divide $\mathbf{T}$ into two reconfiguration sequences $\mathbf{T}_1$ and $\mathbf{T}_2$, where $\mathbf{T}_1$ is from T_{ini} to T^* and $\mathbf{T}_2$ is from T^* to T_{tar}. By symmetry, we may assume that

$$\mathrm{length}_w(\mathbf{T}_1) \leq \frac{\mathrm{length}_w(\mathbf{T})}{2} < 2\lambda N^7 + N^3.$$

For $i \in [N^3]$, define

$$L_i = \{e \in E \mid \mathbf{swap}(u_e, s_i) \text{ is applied in } \mathbf{T}_1\},$$
$$R_i = \{e \in E \mid \mathbf{swap}(u_e, t_i) \text{ is applied in } \mathbf{T}_1\}.$$

For $i \in [N^3]$, let $\mathbf{swap}(L_i)$ denote the set of all swaps $\mathbf{swap}(u_e, s_i)$ in $\mathbf{T}_1$ with $e \in L_i$. Similarly, let $\mathbf{swap}(R_i)$ denote the set of all swaps $\mathbf{swap}(u_e, t_i)$ in $\mathbf{T}_1$ with $e \in R_i$.

▷ Claim 7. For $i \in [N^3]$, we have the following:
- if an edge $e \in E$ is contained in the connected component of $G - L_i$ containing s, then $u_e \in \mathbf{des}_T(s_i)$ for any elimination tree T in $\mathbf{T}_1$, and
- if an edge $e \in E$ is contained in the connected component of $G - R_i$ containing t, then $u_e \in \mathbf{des}_T(t_i)$ for any elimination tree T in $\mathbf{T}_1$.

Proof. For each edge $e \in E$ in the connected component of $G - L_i$ containing s, vertices s_i and u_e are contained in the same connected component in $H - \{u_f \mid f \in L_i\}$. Since $u_e \in \mathbf{des}_{T_{\mathrm{ini}}}(s_i)$ holds and $\mathbf{swap}(u_e, s_i)$ is not applied in $\mathbf{T}_1$ as $e \notin L_i$, we have that $u_e \in \mathbf{des}_T(s_i)$ for any elimination tree T in $\mathbf{T}_1$. The same argument works for the second statement. ◁

To simplify the notation, let $L_0 = R_0 = F$. For $i \in [N^3] \cup \{0\}$, let $X_i \subseteq V$ be the vertex set of the connected component of $G - L_i$ containing s. Similarly, let $Y_i \subseteq V$ be the vertex set of the connected component of $G - R_i$ containing t.

▷ Claim 8. For $i, j \in [N^3] \cup \{0\}$ with $j > i$, we have the following:
(i) $(E[X_j] \setminus L_j) \cap L_i = \emptyset$, and
(ii) $(E[Y_j] \setminus R_j) \cap R_i = \emptyset$.

Proof. To show (i), assume to the contrary that there exists $e \in (E[X_j] \setminus L_j) \cap L_i$ for some $j > i$. Note that $j \in [N^3]$. Since $e \in E[X_j] \setminus L_j$, Claim 7 shows that $u_e \in \mathbf{des}_T(s_j)$ for any elimination tree T in $\mathbf{T}_1$. If $i \geq 1$, then since $u_e \in \mathbf{des}_T(s_j)$ and $s_i \in \mathbf{anc}_T(s_j)$, we see that u_e and s_i are not adjacent in T. This implies that $\mathbf{swap}(u_e, s_i)$ is not applied in $\mathbf{T}_1$, which contradicts $e \in L_i$. If $i = 0$, then $e \in L_0 = F$ implies that $u_e \in \mathbf{anc}_{T^*}(s_1) \subseteq \mathbf{anc}_{T^*}(s_j)$, which contradicts $u_e \in \mathbf{des}_T(s_j)$ for any T. The same argument works for (ii). ◁

▷ Claim 9. $X_0 \supseteq X_1 \supseteq X_2 \supseteq \cdots \supseteq X_{N^3}$ and $Y_0 \supseteq Y_1 \supseteq Y_2 \supseteq \cdots \supseteq Y_{N^3}$.

Proof. Let $i, j \in [N^3] \cup \{0\}$ be indices with $j > i$. Since $(E[X_j] \setminus L_j) \cap L_i = \emptyset$ by Claim 8 (i), all vertices in X_j are contained in the same connected component of $G - L_i$. Since both X_i and X_j contain s, we obtain $X_j \subseteq X_i$. This shows that $X_0 \supseteq X_1 \supseteq X_2 \supseteq \cdots \supseteq X_{N^3}$. Similarly, we obtain $Y_0 \supseteq Y_1 \supseteq Y_2 \supseteq \cdots \supseteq Y_{N^3}$. ◁

▷ Claim 10. For $i \in [N^3]$, we have $|L_i| = |R_i| = \lambda$, $L_i = \delta_G(X_i)$, and $R_i = \delta_G(Y_i)$.

Proof. Since F contains an s-t cut set, it holds that $X_0 \subseteq V \setminus \{t\}$. For $i \geq 1$, since $s \in X_i \subseteq X_0 \subseteq V \setminus \{t\}$ by Claim 9, we see that $\delta_G(X_i)$ is an s-t cut set contained in L_i. Similarly, R_i contains an s-t cut set in G. Therefore, we obtain $|L_i|, |R_i| \geq \lambda$ for any $i \in [N^3]$. By considering the weight of $\mathbf{T}_1$, we obtain

$$\begin{aligned} 2\lambda N^7 + N^3 &> \mathrm{length}_w(\mathbf{T}_1) \\ &\geq \sum_{i=1}^{N^3} (w(s_i)|L_i| + w(t_i)|R_i|) \\ &= 2\lambda N^7 + N^4 \sum_{i=1}^{N^3} ((|L_i| - \lambda) + (|R_i| - \lambda)), \end{aligned}$$

which shows that $|L_i| = |R_i| = \lambda$ for any $i \in [N^3]$. Therefore, each of L_i and R_i is a minimum s-t cut set in G, and hence $L_i = \delta_G(X_i)$ and $R_i = \delta_G(Y_i)$ hold. ◁

Since the total weight of $\mathbf{swap}(L_i)$ and $\mathbf{swap}(R_i)$ is at least $2\lambda N^7$ by this claim, we see that u_e and s_i (resp. t_i) are swapped exactly once in $\mathbf{T}_1$ for $e \in L_i$ (resp. $e \in R_i$) and for $i \in [N^3]$. For $i \in [N^3]$, let T_i (resp. T'_i) be the elimination tree that appears in $\mathbf{T}_1$ after all the swaps in $\mathbf{swap}(L_i)$ (resp. $\mathbf{swap}(R_i)$) are just applied.

▷ Claim 11. Elimination trees $T'_{N^3}, T_{N^3}, T'_{N^3-1}, T_{N^3-1}, \ldots, T'_1, T_1$ appear in this order in $\mathbf{T}_1$.

Proof. We first show that T_i appears after T'_i for $i \in [N^3]$. Let $e \in R_i$ be the edge such that $\mathbf{swap}(u_e, t_i)$ is applied just before obtaining T'_i. Ler $f \in L_i \setminus (R_i \setminus \{e\})$, where the existence of such f is guaranteed by $|L_i| = |R_i|$. Since R_i is a minimum s-t cut set by Claim 10, we see that $G - (R_i \setminus \{e\})$ is connected. Then, for any elimination tree T before T'_i, we have $u_{e'} \in \mathbf{des}_T(t_i)$ for any $e' \in E \setminus (R_i \setminus \{e\})$. In particular, $u_f \in \mathbf{des}_T(t_i)$. Since $s_i \in \mathbf{anc}_T(t_i)$, we see that u_f and s_i are not adjacent in T. This shows that we cannot apply $\mathbf{swap}(u_f, s_i)$ before T'_i. Therefore, T_i appears after T'_i in $\mathbf{T}_1$.

By the same argument, we can show that T'_i appears after T_{i+1} for $i \in [N^3 - 1]$. Therefore, T'_{N^3}, T_{N^3}, T'_{N^3-1}, $T_{N^3-1}, \ldots, T'_1$, T_1 appear in this order. ◁

▷ Claim 12. For $i \in [N^3]$, we have the following:

- $u_e \in \mathbf{anc}_{T_i}(u_{e'})$ for any $e \in L_i$ and $e' \in E \setminus L_i$, and
- $u_e \in \mathbf{anc}_{T'_i}(u_{e'})$ for any $e \in R_i$ and $e' \in E \setminus R_i$.

Proof. Let T be an elimination tree in $\mathbf{T}_1$ just before T_i. Then, there exists an edge $f \in L_i$ such that T_i is obtained from T by applying $\mathbf{swap}(u_f, s_i)$. Since $G - (L_i \setminus \{f\})$ is connected by Claim 10, we have $u_e \in \mathbf{anc}_T(s_i)$ for $e \in L_i \setminus \{f\}$ and $u_{e'} \in \mathbf{des}_T(s_i)$ for $e' \in E \setminus (L_i \setminus \{f\})$. Therefore, after applying $\mathbf{swap}(u_f, s_i)$, we obtain $u_e \in \mathbf{anc}_{T_i}(f)$ for $e \in L_i \setminus \{f\}$ and $u_{e'} \in \mathbf{des}_{T_i}(f)$ for $e' \in E \setminus L_i$. This shows that $u_e \in \mathbf{anc}_{T_i}(u_{e'})$ for any $e \in L_i$ and $e' \in E \setminus L_i$. By the same argument, we obtain $u_e \in \mathbf{anc}_{T'_i}(u_{e'})$ for any $e \in R_i$ and $e' \in E \setminus R_i$. ◁

▷ Claim 13. $X_1 = V \setminus Y_1$.

Proof. Observe that X_0 and Y_0 are disjoint since $F = L_0 = R_0$ contains an s-t cut set in G. Since $X_1 \subseteq X_0$ and $Y_1 \subseteq Y_0$ by Claim 9, we see that X_1 and Y_1 are disjoint. To derive a contradiction, assume that $X_1 \neq V \setminus Y_1$, that is, X_1 and Y_1 are disjoint sets with $X_1 \cup Y_1 \subsetneq V$. Then, by Claim 9, we obtain $X_i \neq V \setminus Y_i$ for any $i \in [N^3]$. This shows that $L_i \neq R_i$ for any $i \in [N^3]$. Since $|L_i| = |R_i| = \lambda$, there exist $f_i \in L_i \setminus R_i$ and $f'_i \in R_i \setminus L_i$. By Claim 12, we

obtain $u_{f_i} \in \mathbf{anc}_{T_i}(u_{f'_i})$ and $u_{f'_i} \in \mathbf{anc}_{T'_i}(u_{f_i})$. By Observation 6, $\mathbf{swap}(u_e, u_{e'})$ is applied for some $e, e' \in E$ between T'_i and T_i. Since such a swap is required for each $i \in [N^3]$, by Claim 11, we have to swap pairs in $\{u_e \mid e \in E\}$ at least N^3 times in $\mathbf{T}_1$. Therefore, we obtain

$$\mathrm{length}_w(\mathbf{T}_1) \geq \sum_{i=1}^{N^3} (w(s_i)|L_i| + w(t_i)|R_i|) + N^3 = 2\lambda N^7 + N^3,$$

which contradicts $\mathrm{length}_w(\mathbf{T}_1) < 2\lambda N^7 + N^3$. ◁

▷ Claim 14. $F = \delta_G(X_1) = \delta_G(Y_1)$.

Proof. Claims 10 and 13 show that $L_1 = R_1 = \delta_G(X_1) = \delta_G(Y_1)$. This together with Claim 8 shows that $F \cap E[X_1] = F \cap E[Y_1] = \emptyset$. Since F contains an s-t cut set in G, we obtain $F = \delta_G(X_1) = \delta_G(Y_1)$. ◁

▷ Claim 15. Let T be an elimination tree in $\mathbf{T}_1$. If two vertices $u, v \in V \setminus \{s, t\}$ are contained in the same connected component in $G - F$, then u and v are comparable in T.

Proof. By Claim 14, $G - F$ consists of two connected components $G[X_1]$ and $G[Y_1]$. We first consider the case when $u, v \in X_1 \setminus \{s\}$. By Claims 7 and 14, we obtain $u_e \in \mathbf{des}_T(s_1)$ for any $e \in E[X_1]$. Furthermore, since $\mathrm{length}_w(\mathbf{T}_1) < 2\lambda N^7 + N^3$ holds and the total weight of $\mathbf{swap}(L_i)$ and $\mathbf{swap}(R_i)$ is $2\lambda N^7$, neither $\mathbf{swap}(s_1, u)$ nor $\mathbf{swap}(s_1, v)$ is applied in $\mathbf{T}_1$, because $w(s_1)w(u) = w(s_1)w(v) = N^5$. Therefore, we obtain $u \in \mathbf{anc}_T(s_1)$ and $v \in \mathbf{anc}_T(s_1)$, which shows that u and v are comparable in T. The same argument works when $u, v \in Y_1 \setminus \{t\}$. ◁

Since the weight of $\mathbf{T}_1$ is at least $\sum_{i=1}^{N^3}(w(s_i)|L_i| + w(t_i)|R_i|) = 2\lambda N^7$, we obtain

$$\mathrm{length}_w(\mathbf{T}_2) = \mathrm{length}_w(\mathbf{T}) - \mathrm{length}_w(\mathbf{T}_1) < 2\lambda N^7 + N^3.$$

Hence, the above argument (Claims 7–15) can be applied also to the reverse sequence of $\mathbf{T}_2$. In particular, Claim 15 holds even if $\mathbf{T}_1$ is replaced with $\mathbf{T}_2$. Therefore, if two vertices $u, v \in V \setminus \{s, t\}$ are contained in the same connected component in $G - F$, then u and v are comparable in any elimination tree in $\mathbf{T}$. For such a pair of vertices u and v, the only way to reverse the ordering of u and v is to apply $\mathbf{swap}(u, v)$ or $\mathbf{swap}(v, u)$.

Recall that $G - F$ consists of two connected components $G[X_1]$ and $G[Y_1]$ by Claim 14. Since the ordering of $v_1, \ldots, v_{2n}$ are reversed from T_{ini} to T_{tar}, we see that $\mathbf{swap}(u, v)$ or $\mathbf{swap}(v, u)$ has to be applied in $\mathbf{T}$ if $u, v \in X_1 \setminus \{s\}$ or $u, v \in Y_1 \setminus \{t\} = (V \setminus X_1) \setminus \{t\}$. Furthermore, we have to swap some elements in $\{u_e \mid e \in E\}$ and $\{s_1, t_1, \ldots, s_{N^3}, t_{N^3}\}$ in $\mathbf{T}_2$, whose total weight is at least $2\lambda N^7$ in the same way as $\mathbf{T}_1$. With these observations, we evaluate the weight of $\mathbf{T}$ as follows, where we denote $k = |X_1|$ to simplify the notation:

$$\begin{aligned}
\mathrm{length}_w(\mathbf{T}) &\geq 2\lambda N^7 + 2\lambda N^7 + \binom{|X_1| - 1}{2} \cdot N^2 + \binom{|V \setminus X_1| - 1}{2} \cdot N^2 \\
&= 4\lambda N^7 + \frac{(k-1)(k-2)}{2} N^2 + \frac{(2n-k+1)(2n-k)}{2} N^2 \\
&= 4\lambda N^7 + (k^2 - 2(n+1)k + 2n^2 + n + 1)N^2 \\
&= 4\lambda N^7 + (n^2 - n)N^2 + (k - n - 1)^2 N^2.
\end{aligned}$$

This together with $\mathrm{length}_w(\mathbf{T}) < 4\lambda N^7 + (n^2 - n + 1)N^2$ shows that $(k - n - 1)^2 < 1$, and hence $k = n + 1$ by the integrality of k and n.

Therefore, we obtain $|X_1| = k = n + 1$ and $|Y_1| = |V \setminus X_1| = n + 1$. Since $|\delta_G(X_1)| = |L_1| = \lambda$, this shows that X_1 is a desired s-t cut in G.

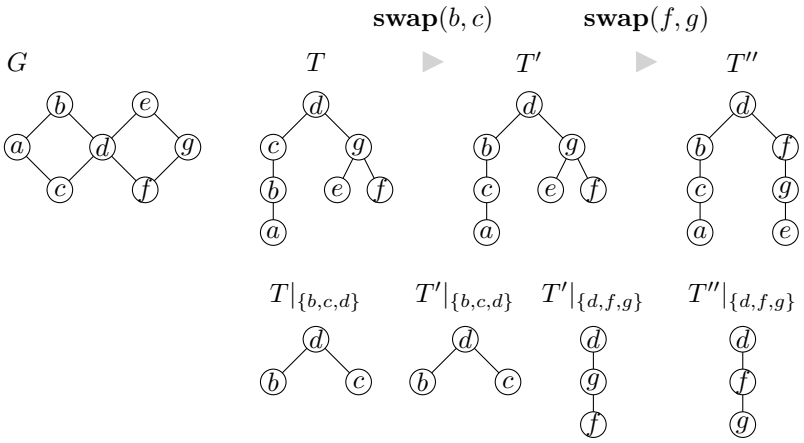


Figure 5 An example of projections. Note that $T|_{\{b,c,d\}} = T'|_{\{b,c,d\}}$ since b and c are incomparable in $T|_{\{b,c,d\}}$, and $T''|_{\{d,f,g\}}$ is obtained from $T'|_{\{d,f,g\}}$ by swapping f and g since f and g are adjacent in $T'|_{\{d,f,g\}}$.

4 Hardness of the Unweighted Problem (Proof of Theorem 1)

To show Theorem 1, we reduce Weighted Combinatorial Shortest Path on Graph Associahedra to Combinatorial Shortest Path on Graph Associahedra. An operation called *projection* (e.g. [6]) plays an important role in our validity proof.

4.1 Useful Operation: Projection

Let $G = (V, E)$ be a graph and let T be an elimination tree associated with G. For $U \subseteq V$ such that $G[U]$ is connected, let $T|_U$ be the elimination tree associated with $G[U]$ that preserves the ordering in T. That is, $u \in \mathbf{anc}_{T|_U}(v)$ if and only if $u \in \mathbf{anc}_T(v)$ and u are v are connected in $G[U] - \mathbf{anc}_T(u)$ for $u, v \in U$. Note that such $T|_U$ is uniquely determined. We call $T|_U$ the *projection* of T to U. See Figure 5 for illustration.

▶ **Lemma 16.** *Let $U \subseteq V$ be a vertex set such that $G[U]$ is connected. Let T and T' be elimination trees associated with G such that T' is obtained from T by applying* $\mathbf{swap}(u, v)$*, where $u, v \in V$.*

1. *If $\{u, v\} \subseteq U$, then either $T'|_U = T|_U$ or $T'|_U$ is obtained from $T|_U$ by applying* $\mathbf{swap}(u, v)$*.*
2. *Otherwise, $T'|_U = T|_U$.*

Proof. Since all the vertices in $V \setminus U$ are removed when we construct $T|_U$, $\mathbf{swap}(u, v)$ affects $T|_U$ only if $\{u, v\} \subseteq U$, which proves the second item. For the first item, suppose that $\{u, v\} \subseteq U$. Then, u and v are adjacent or incomparable in $T|_U$. If they are adjacent, then $T'|_U$ is obtained from $T|_U$ by applying $\mathbf{swap}(u, v)$. If they are incomparable, then $T'|_U = T|_U$. ◀

4.2 Reduction

Suppose we are given a graph $G = (V, E)$, two elimination trees T_{ini} and T_{tar}, and a weight function $w\colon V \to \mathbb{Z}_{>0}$, which form an instance of Weighted combinatorial Shortest Path on Graph Associahedra. Then, we replace each vertex $v \in V$ with a clique of size $w(v)$. Formally, consider a graph $G' = (V', E')$ such that $V' = \{v_i \mid v \in V,\ i \in \{1, \ldots, w(v)\}\}$,

and $\{u_i, v_j\} \in E'$ if $\{u, v\} \in E$ or $u = v$. Let T'_{ini} (resp. T'_{tar}) be the elimination tree obtained from T_{ini} (resp. T_{tar}) by replacing a vertex $v \in V$ with a path $v_1, v_2, \ldots, v_{w(v)}$. That is, for distinct $u, v \in V$, there is an arc (u, v) in T_{ini} (resp. T_{tar}) if and only if $(u_{w(u)}, v_1)$ is an arc of T'_{ini} (resp. T'_{tar}). Note that the obtained elimination tree is associated with G'. This defines an instance of Combinatorial Shortest on Graph Associahedra.

4.3 Validity

In what follows, we show that the obtained instance of Combinatorial Shortest Path on Graph Associahedra has a reconfiguration sequence of length at most ℓ if and only if the original instance of Weighted Combinatorial Shortest Path on Graph Associahedra has a reconfiguration sequence $\mathbf{T}$ with $\mathrm{length}_w(\mathbf{T}) \leq \ell$.

Sufficiency ("if" part)

Suppose that the original instance of Weighted Combinatorial Shortest Path on Graph Associahedra has a reconfiguration sequence $\mathbf{T}$ from T_{ini} to T_{tar}. Then, we construct a reconfiguration sequence $\mathbf{T}'$ from T'_{ini} to T'_{tar} by replacing each swap $\mathbf{swap}(u, v)$ in $\mathbf{T}$ with $w(u) \cdot w(v)$ swaps $\{\mathbf{swap}(u_i, v_j) \mid i \in [w(u)],\ j \in [w(v)]\}$. This gives a reconfiguration sequence from T'_{ini} to T'_{tar} whose length is $\mathrm{length}_w(\mathbf{T})$, which shows the sufficiency.

Necessity ("only if" part)

Suppose that the obtained instance of Combinatorial Shortest Path on Graph Associahedra has a reconfiguration sequence $\mathbf{T}'$ from T'_{ini} to T'_{tar} of length at most ℓ. For any $v \in V$, since $v_1, \ldots, v_{w(v)}$ form a clique, they are comparable in any elimination tree in $\mathbf{T}'$. Furthermore, since $v_1, \ldots, v_{w(v)}$ are aligned in this order in both T'_{ini} and T'_{tar}, we may assume that $\mathbf{swap}(v_i, v_j)$ is not applied in $\mathbf{T}'$ for any $i, j \in [w(v)]$.

Let Φ be the set of all maps $\phi\colon V \to \mathbb{Z}$ such that $\phi(v) \in \{1, \ldots, w(v)\}$ for any $v \in V$. Note that $|\Phi| = \prod_{v \in V} w(v)$. For $\phi \in \Phi$, define $U_\phi = \{v_{\phi(v)} \mid v \in V\}$. Note that $G'[U_\phi]$ is isomorphic to G, and hence it is connected. By projecting each elimination tree in $\mathbf{T}'$ to U_ϕ, we obtain a sequence of elimination trees. Lemma 16 shows that this forms a reconfiguration sequence, say $\mathbf{T}_\phi$, if we remove duplications when the same elimination tree appears consecutively. Since $G'[U_\phi]$ is isormorphic to G, by idenfitying $v_{\phi(v)}$ with v for each $v \in V$, we can regard $\mathbf{T}_\phi$ as a recofiguration sequence from T_{ini} to T_{tar}. That is, $\mathbf{T}_\phi$ is regarded as a feasible solution of the original instance of Weighted Combinatorial Shortest Path on Graph Associahedra.

In what follows, we consider reconfiguration sequences $\{\mathbf{T}_\phi \mid \phi \in \Phi\}$ and show that a desired sequence exists among them. Suppose that $\mathbf{swap}(u_i, v_j)$ is applied in $\mathbf{T}'$, where $u, v \in V$, $i \in [w(u)]$, and $j \in [w(v)]$. Then, Lemma 16 shows that the corresponding swap operation $\mathbf{swap}(u_i, v_j)$, which is identified with $\mathbf{swap}(u, v)$, is applied in $\mathbf{T}_\phi$ only if $\phi(u) = i$ and $\phi(v) = j$. Thus, such a swap is applied in at most $|\{\phi \in \Phi \mid \phi(u) = i,\ \phi(v) = j\}| = |\Phi|/(w(u) \cdot w(v))$ sequences in $\{\mathbf{T}_\phi \mid \phi \in \Phi\}$. Therefore, we obtain

$$\begin{aligned}
\sum_{\phi \in \Phi} \mathrm{length}_w(\mathbf{T}_\phi) &= \sum_{\phi \in \Phi} \sum_{\mathbf{swap}(u,v) \in \mathbf{T}_\phi} w(\mathbf{swap}(u, v)) \\
&\leq \sum_{\mathbf{swap}(u_i, v_j) \in \mathbf{T}'} w(\mathbf{swap}(u, v)) \cdot \frac{|\Phi|}{w(u) \cdot w(v)} \\
&= \mathrm{length}(\mathbf{T}') \cdot |\Phi| \leq \ell \cdot |\Phi|,
\end{aligned}$$

where each reconfiguration sequence is regarded as a multiset of swaps. Therefore,

$$\min_{\phi\in\Phi}(\mathrm{length}_w(\mathbf{T}_\phi)) \leq \frac{1}{|\Phi|}\sum_{\phi\in\Phi}\mathrm{length}_w(\mathbf{T}_\phi) \leq \ell.$$

Hence, there exists $\phi \in \Phi$ such that $\mathbf{T}_\phi$ is a desired sequence. This shows the necessity.

Therefore, the weighted problem can be reduced to the unweighted problem, and hence Theorem 3 implies Theorem 1.

5 Hardness for Polymatroids (Proof of Theorem 2)

In this section, we give a proof sketch of Theorem 2.

We reduce Combinatorial Shortest Path on Graph Associahedra to Combinatorial Shortest Path on Polymatroids. Assume that we are given an instance $G = (V, E)$, T_{ini}, and T_{tar} of Combinatorial Shortest Path on Graph Associahedra. To this end, we construct a polymatroid (V, f) satisfying the following conditions.

1. $\mathbf{B}(f)$ is a realization of the G-associahedron.
2. For each subset $X \subseteq V$, we can evaluate the value $f(X)$ in time bounded by a polynomial in the size of G.
3. We can find the extreme points $x_{\mathrm{ini}}, x_{\mathrm{tar}}$ of $\mathbf{B}(f)$ corresponding to $T_{\mathrm{ini}}, T_{\mathrm{tar}}$, respectively, in time bounded by a polynomial in the size of G.

We first argue that the conditions above suffice for our proof. Suppose the existence of a polymatroid (V, f) with the properties above. Then, we may construct a polynomial-time algorithm for Combinatorial Shortest Path on Graph Associahedra with a fictitious polynomial-time algorithm for Combinatorial Shortest Path on Polymatroids as follows. Let $(G, T_{\mathrm{ini}}, T_{\mathrm{tar}})$ be an instance of Combinatorial Shortest Path on Graph Associahedra. From Properties 1 and 3, we can construct an instance $((V, f), x_{\mathrm{ini}}, x_{\mathrm{tar}})$ of Combinatorial Shortest Path on Polymatroids in polynomial time. By the fictitious polynomial-time algorithm, we can solve the instance in time bounded by a polynomial in $|V|$ and the number of oracle calls to f. By Property 2, this running time is bounded by a polynomial in $|V|$. Thus, we find a solution to $(G, T_{\mathrm{ini}}, T_{\mathrm{tar}})$ in polynomial time, and the proof is completed.

In our construction of such a polymatroid (V, f), we use the realization of the G-associahedron by Devadoss [12], which can be described as follows. Let T be an elimination tree of G. For each vertex $v \in V$, we define $T(v)$ as the vertex set of the subtree of T rooted at v. Then, we define the vector $x^T \in \mathbb{R}^V$ by choosing the coordinate $x^T(v)$ at every vertex of v from the leaves to the root according to the following rule.

- If v is a leaf of T, then we define $x^T(v) := 0$.
- If v is not a leaf of T, then we define $x^T(v)$ so that

$$\sum_{u\in T(v)} x^T(u) = 3^{|T(v)|-2}.$$

Define $\mathcal{E} := \{x^T \mid T \text{ is an elimination tree of } G\}$. Then, Devadoss [12] proved that the convex hull of $\mathcal{E}$ is a realization of the G-associahedron, and for each elimination tree T of G, the point x^T is an extreme point of the G-associahedron.

In our proof, we define the function $f\colon 2^V \to \mathbb{R}$ by

$$f(X) := 3^{|V|-2} - \sum_{C \in \mathcal{C}^*(X)} 3^{|C|-2}$$

for each subset $X \subseteq V$, where $\mathcal{C}^*(X)$ is the family of connected components of $G - X$ with at least two vertices.

Properties 2 and 3 above are immediate: it is not difficult to see that we can evaluate the values of the function f in time bounded by a polynomial in the size of G; we can construct x_{ini} and x_{tar} from T_{ini} and T_{tar}, respectively, as $x_{\mathrm{ini}} = x^{T_{\mathrm{ini}}}$ and $x_{\mathrm{tar}} = x^{T_{\mathrm{tar}}}$. In the full version, we prove that (V, f) is a polymatroid and $\mathbf{B}(f)$ coincides with the convex hull of $\mathcal{E}$. This completes the reduction. Therefore, Theorem 2 follows from Theorem 1.

6 Conclusion

We prove that the combinatorial shortest path computation is hard on graph associahedra and base polytopes of polymatroids. This evaporates our hope for resolving an open problem to obtain a polynomial-time algorithm for finding a shortest flip sequence between two triangulations of convex polygons and the rotation distance between two binary trees by generalizing the setting to graph associahedra. However, that open problem is still open, and we should pursue another way of attacking it.

References

1. Oswin Aichholzer, Wolfgang Mulzer, and Alexander Pilz. Flip distance between triangulations of a simple polygon is NP-complete. *Discret. Comput. Geom.*, 54(2):368–389, 2015. doi:10.1007/s00454-015-9709-7.
2. Benjamin Aram Berendsohn. The diameter of caterpillar associahedra. In Artur Czumaj and Qin Xin, editors, *18th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2022, June 27-29, 2022, Tórshavn, Faroe Islands*, volume 227 of *LIPIcs*, pages 14:1–14:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SWAT.2022.14.
3. Maike Buchin, Anna Lubiw, Arnaud de Mesmay, and Saul Schleimer. Computation and reconfiguration in low-dimensional topological spaces (Dagstuhl Seminar 22062). *Dagstuhl Reports*, 12(2):17–66, 2022. doi:10.4230/DagRep.12.2.17.
4. Jean Cardinal, Stefan Langerman, and Pablo Pérez-Lantero. On the diameter of tree associahedra. *Electron. J. Comb.*, 25(4):4, 2018. doi:10.37236/7762.
5. Jean Cardinal, Arturo I. Merino, and Torsten Mütze. Efficient generation of elimination trees and graph associahedra. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 2128–2140. SIAM, 2022. doi:10.1137/1.9781611977073.84.
6. Jean Cardinal, Lionel Pournin, and Mario Valencia-Pabon. Diameter estimates for graph associahedra. *Ann. Comb.*, 26:873–902, 2022. doi:10.1007/s00026-022-00598-z.
7. Jean Cardinal, Lionel Pournin, and Mario Valencia-Pabon. The rotation distance of brooms, 2022. doi:10.48550/arXiv.2211.07984.
8. Michael Carr and Satyan L. Devadoss. Coxeter complexes and graph-associahedra. *Topology and its Applications*, 153(12):2155–2168, 2006. doi:10.1016/j.topol.2005.08.010.
9. Cesar Ceballos, Francisco Santos, and Günter M. Ziegler. Many non-equivalent realizations of the associahedron. *Combinatorica*, 35(5):513–551, 2015. doi:10.1007/s00493-014-2959-9.
10. Sean Cleary and Katherine St. John. Rotation distance is fixed-parameter tractable. *Inf. Process. Lett.*, 109(16):918–922, 2009. doi:10.1016/j.ipl.2009.04.023.

11 Sean Cleary and Katherine St. John. A linear-time approximation algorithm for rotation distance. *J. Graph Algorithms Appl.*, 14(2):385–390, 2010. doi:10.7155/jgaa.00212.

12 Satyan L. Devadoss. A realization of graph associahedra. *Discrete Mathematics*, 309(1):271–276, 2009. doi:10.1016/j.disc.2007.12.092.

13 Jack Edmonds. Submodular functions, matroids, and certain polyhedra. In Richard Guy, Haim Hanani, Norbert Sauer, and Johanan Schönheim, editors, *Combinatorial Structures and their Applications*, pages 69–87. Gordon and Breach, New York, NY, 1970.

14 Uriel Feige and Mohammad Mahdian. Finding small balanced separators. In Jon M. Kleinberg, editor, *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 375–384. ACM, 2006. doi:10.1145/1132516.1132573.

15 Alan M. Frieze and Shang-Hua Teng. On the complexity of computing the diameter of a polytope. *Comput. Complex.*, 4:207–219, 1994. doi:10.1007/BF01206636.

16 Satoru Fujishige. *Submodular Functions and Optimization*, volume 58 of *Annals of Discrete Mathematics*. Elsevier, Amsterdam, The Netherlands, 2nd edition, 2005.

17 Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Tóth, editors. *Handbook of Discrete and Computational Geometry, Third Edition*. CRC Press LLC, 2017.

18 Takehiro Ito, Erik D. Demaine, Nicholas J. A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theor. Comput. Sci.*, 412(12-14):1054–1065, 2011. doi:10.1016/j.tcs.2010.12.005.

19 Volker Kaibel and Marc E. Pfetsch. Some algorithmic problems in polytope theory. In Michael Joswig and Nobuki Takayama, editors, *Algebra, Geometry, and Software Systems [outcome of a Dagstuhl seminar]*, pages 23–47. Springer, 2003. doi:10.1007/978-3-662-05148-1_2.

20 Iyad A. Kanj, Eric Sedgwick, and Ge Xia. Computing the flip distance between triangulations. *Discret. Comput. Geom.*, 58(2):313–344, 2017. doi:10.1007/s00454-017-9867-x.

21 Haohong Li and Ge Xia. An $\mathcal{O}(3.82^k)$ time FPT algorithm for convex flip distance. In Petra Berenbrink, Patricia Bouyer, Anuj Dawar, and Mamadou Moustapha Kanté, editors, *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023, March 7-9, 2023, Hamburg, Germany*, volume 254 of *LIPIcs*, pages 44:1–44:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.STACS.2023.44.

22 Anna Lubiw and Vinayak Pathak. Flip distance between two triangulations of a point set is NP-complete. *Comput. Geom.*, 49:17–23, 2015. doi:10.1016/j.comgeo.2014.11.001.

23 Joan M. Lucas. An improved kernel size for rotation distance in binary trees. *Inf. Process. Lett.*, 110(12-13):481–484, 2010. doi:10.1016/j.ipl.2010.04.022.

24 Thibault Manneville and Vincent Pilaud. Graph properties of graph associahedra. *Seminaire Lotharingien de Combinatoire*, 73:B73d, 2015.

25 Alexander Pilz. Flip distance between triangulations of a planar point set is APX-hard. *Comput. Geom.*, 47(5):589–604, 2014. doi:10.1016/j.comgeo.2014.01.001.

26 Alex Postnikov, Victor Reiner, and Lauren Williams. Faces of generalized permutohedra. *Doc. Math.*, 13:207–273, 2008.

27 Lionel Pournin. The diameter of associahedra. *Advances in Mathematics*, 259:13–42, 2014. doi:10.1016/j.aim.2014.02.035.

28 Lionel Pournin. The asymptotic diameter of cyclohedra. *Israel Journal of Mathematics*, 219(2):609–635, 2017. doi:10.1007/s11856-017-1492-0.

29 Laura Sanità. The diameter of the fractional matching polytope and its hardness implications. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 910–921. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00090.

30 Daniel D. Sleator, Robert E. Tarjan, and William P. Thurston. Rotation distance, triangulations, and hyperbolic geometry. *J. Amer. Math. Soc.*, 1:647–681, 1988. doi:10.1090/S0894-0347-1988-0928904-4.

31 Donald M. Topkis. Adjacency on polymatroids. *Math. Program.*, 30(2):229–237, 1984. doi:10.1007/BF02591887.

Searching for Regularity in Bounded Functions

Siddharth Iyer
University of Washington CSE, Seattle, WA, USA

Michael Whitmeyer
University of Washington CSE, Seattle, WA, USA

Abstract

Given a function f on $\mathbb{F}_2^n$, we study the following problem. What is the largest affine subspace $\mathcal{U}$ such that when restricted to $\mathcal{U}$, all the non-trivial Fourier coefficients of f are very small?

For the natural class of bounded Fourier degree d functions $f : \mathbb{F}_2^n \to [-1, 1]$, we show that there exists an affine subspace of dimension at least $\widetilde{\Omega}(n^{1/d!}k^{-2})$, wherein all of f's nontrivial Fourier coefficients become smaller than 2^{-k}. To complement this result, we show the existence of degree d functions with coefficients larger than $2^{-d \log n}$ when restricted to any affine subspace of dimension larger than $\Omega(dn^{1/(d-1)})$. In addition, we give explicit examples of functions with analogous but weaker properties.

Along the way, we provide multiple characterizations of the Fourier coefficients of functions restricted to subspaces of $\mathbb{F}_2^n$ that may be useful in other contexts. Finally, we highlight applications and connections of our results to parity kill number and affine dispersers.

2012 ACM Subject Classification Mathematics of computing

Keywords and phrases regularity, bounded function, Boolean function, Fourier analysis

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.83

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2207.13312`

Funding *Siddharth Iyer*: Research supported by NSF grant CCF-2131899.
Michael Whitmeyer: Research supported by NSF grant CCF-2006359.

Acknowledgements We thank Anup Rao for posing the question that launched this project and for his invaluable advice and feedback. We are also grateful Paul Beame for his extremely helpful advice, discussions, and feedback. Finally, we thank Sandy Kaplan for detailed feedback on this writeup.

1 Introduction

The search for structure within large objects is an old one that lies at the heart of Ramsey theory. For example, a famous corollary of Ramsey's theorem is that any graph on n vertices must contain a clique or an independent set of size $\Omega(\log n)$. Another example is Roth's[1] theorem [19] on 3-term arithmetic progressions, which essentially says that *every* subset of $\{1, \ldots, n\}$ of density $\delta > \Omega(1/\log\log n)$ must contain a 3-term arithmetic progression.[2]

Szemerédi's Regularity Lemma is also a well known example of this phenomenon. Roughly speaking, it states that any graph G can be partitioned into $k := M(\delta)$ parts $V_1, \ldots, V_k$, wherein most pairs of parts (V_i, V_j) are δ-regular. In this setting, the δ-regularity of (V_i, V_j) roughly corresponds to saying that the bipartite graph induced across V_i and V_j appears as though its edges were sampled randomly. This powerful statement has found applications

[1] The related Hales-Jewett theorem [10] is also a classic result in Ramsey theory.
[2] See also the recent quantitative improvement due to Kelley and Meka [13] which gives the same result for all subsets of density at least $\Omega(2^{-\log^{1/11}(n)})$.

in both pure mathematics (e.g., Szemerédi's [23] generalization of Roth's result to k-term arithmetic progressions) and theoretical computer science (to test triangle-freeness in dense graphs [20, 1, 22]).

Similar to the definition of regular partitions in Szemerédi's Regularity Lemma, one can also define a notion of regularity for functions. In particular, for functions $f : \mathbb{F}_2^n \to \mathbb{R}$, we follow Green [9] and O'Donnell [17] and define a function to be δ-*regular* if all its nontrivial Fourier coefficients are at most δ in magnitude.[3] This definition can be viewed as a pseudorandomness condition; in particular, a randomly chosen Boolean function $f : \mathbb{F}_2^n \to \{\pm 1\}$ is δ-regular with very high probability, even for $\delta = 2^{-\Omega(n)}$.[4]

The prior works surrounding graph regularity [23, 7, 22] and function regularity [9, 12] have been concerned with obtaining δ-regular *partitions*, which, roughly speaking, are partitions of the object at hand into (mostly) pseudorandom parts. Often, these results have quite poor dependencies on the parameter δ so as not to be practical for any reasonably small value of δ (see Proposition 3 and Proposition 4 for detailed statements). Motivated by this, and by applications in theoretical computer science, we relax our requirement and look to find just *one* δ-regular part. Namely, we seek to understand the following quantity:

$$\mathsf{r}(f, \delta) := \min\{\operatorname{codim}(\mathcal{U}) : \mathcal{U} \text{ is an affine subspace such that } f_{\mathcal{U}} \text{ is } \delta\text{-regular}\},$$

where here and throughout this work $f_{\mathcal{U}} : \mathcal{U} \to \mathbb{R}$ denotes the restriction of f to inputs coming from $\mathcal{U}$.

Before stating our main results as well as prior work, we make a few remarks about the quantity $\mathsf{r}(f, \delta)$. In the special case when $\delta = 0$, the quantity $\mathsf{r}(f, 0)$ has been previously studied in the literature, under the name of *parity kill number* [18]. This is the smallest number of parities that need to be fixed in order to make f constant. The value $\mathsf{r}(f, 0)$ is also a measure associated with affine dispersers, objects that have received significant attention in the study of pseudorandomness, see e.g. [21, 14, 5, 6, 3]. An affine disperser of dimension k is a coloring of $\mathbb{F}_2^n$ such that no affine subspace of dimension k is monochromatic. If we view an affine disperser as a function $f : \mathbb{F}_2^n \to \{0, 1, \ldots, C\}$, then its dimension is just $n - \mathsf{r}(f, 0) + 1$.

Now, we briefly discuss the bounds on $\mathsf{r}(f, \delta)$ most relevant to our work. For a general function $f : \mathbb{F}_2^n \to [-1, 1]$, it is known that $\mathsf{r}(f, \delta) \leq 1/\delta$; this follows from a well-known density-increment argument, see [15] (for a short proof of this, see Proposition 5). One might ask if $\mathsf{r}(f, \delta)$ is small when we assume f is structured, and a natural example of such functions is the class of functions with low Fourier degree. For general degree d functions $f : \mathbb{F}_2^n \to [-1, 1]$, the best bound on $\mathsf{r}(f, \delta)$ until this work was just the above mentioned bound of $1/\delta$. However, for the class of degree d Boolean functions, we know that $\mathsf{r}(f, \delta) \leq \mathsf{r}(f, 0) = O(d^3)$; this follows from the polynomial relationship between Fourier degree and decision tree depth, see [16], and [2, 4] for surveys. We emphasize that this result relies crucially on Booleanity (and is independent of δ), and one can ask if the more general class of degree d functions bounded in the interval $[-1, 1]$ also have small $\mathsf{r}(f, \delta)$ values. Our main result answers exactly this question, and provides an upper bound for $\mathsf{r}(f, \delta)$ in this setting.

▶ **Theorem 1.** *For any $\delta \in (0, 1)$ and any degree d function $f : \mathbb{F}_2^n \to [-1, 1]$, we have* $\mathsf{r}(f, \delta) \leq n - \Omega\left(n^{1/d!}(\log(n/\delta))^{-2}\right)$.

[3] For a formal definition, see Definition 11, and for more background on Fourier analysis, see Section 2.
[4] See for example [17], Exercise 1.7 and Proposition 6.1.

Note that the general bound $\mathsf{r}(f,\delta) \leq 1/\delta$ that we mentioned earlier, is only meaningful when $\delta > 1/n$, however, our theorem allows for δ to be much smaller. The regime of small δ is particularly interesting from the perspective of pseudorandomness. Indeed, in a qualitative sense, we see that by decreasing δ, we are asking for affine subspaces where the restricted function looks increasingly like a random function. Using Theorem 1 together with our connection between $\mathsf{r}(f,0)$ and the dimension of affine disperse, we obtain the following corollary which says that low degree polynomials cannot serve as good affine dispersers.

▶ **Corollary 2.** *If $f : \mathbb{F}_2^n \to \{0,\dots,C\}$ has Fourier degree d, then f cannot be an affine disperser of dimension k for any $k \geq \Omega\left(n^{1/d!}(d+\log(nC))^{-2}\right)$.*

Lower Bounds on $\mathsf{r}(f,\delta)$. To complement Theorem 1, we present in Table 1 several examples of functions (bounded as well as Boolean) for which $\mathsf{r}(f,\delta)$ is large. For each row in the table, we exhibit a class of functions (whose degree and range is as specified), such that for any $\delta' \leq \delta$, *no* affine subspace of dimension larger than $n - \mathsf{r}(f,\delta)$ is δ'-regular.

Table 1 Table of functions with large $\mathsf{r}(f,\delta)$ values.

$\delta \leq$	$\mathsf{r}(f,\delta) \geq$	$\deg(f)$	range(f)	Ref.
$1/n$	$n/2-1$	1	$[-1,1]$	Lemma 26
$\binom{n}{d}^{-1}$	$n - 2dn^{1/(d-1)}$	d	$[-1,1]$	Lemma 27
$\Theta(n^{-1/2})$	$\Theta(\sqrt{n})$	n	$\{\pm 1\}$	Lemma 32
$\frac{1}{2}\cdot n^{-d}$ (for $d \leq \frac{\log n}{\log\log n+1}$)	$n - 2dn^{1/(d-1)}$	$\Omega(n)$	$\{\pm 1\}$	Corollary 30
$1/2^{2^k+1}$ (for integer k)	$\Omega\left((\log\frac{1}{\delta})^{\log_2(3)}\right)$	2^k	$\{\pm 1\}$	Lemma 31

Observe that Lemma 27 provides a somewhat of a converse to Theorem 1. However there is a noticeable gap between the two results, and we conjecture that Lemma 27 is closer to being tight, and that Theorem 1 could be improved. We also note that Lemma 27 and Corollary 30 are not explicit – it would be interesting to find more explicit examples.

1.1 Related Work

To the best of our knowledge, $\mathsf{r}(f,\delta)$ has not been explicitly studied before. However, it is closely related to well-studied notions of function regularity as well as the concepts of parity kill number and affine dispersers. In this section, we give a detailed description of both these connections.

Parity Kill Number and Affine Dispersers. As we have already mentioned, $\mathsf{r}(f,0)$ has been studied under the name of *parity kill number*, denoted $C^{\oplus}_{\min}[f]$ (see [18]). Parity kill number can be considered as a further generalization of the *minimum certificate complexity* of f, denoted $C_{\min}[f]$, which is the minimum number of bits one must fix in order to make f constant. In particular, for any $\delta \geq 0$, we have $\mathsf{r}(f,\delta) \leq \mathsf{r}(f,0) \leq C_{\min}[f]$. The minimum certificate complexity is one of several natural complexity measures that have been well studied for Boolean functions $f : \mathbb{F}_2^n \to \{\pm 1\}$ (see [4, 2] for surveys).

As we have already alluded to, the quantity $\mathsf{r}(f,0)$ is also closely related to efficacy of $f : \mathbb{F}_2^n \to \{0,\dots,C\}$ as an affine disperser. In the case of $C = 1$, Cohen and Tal [6] rule out $\mathbb{F}_2$-polynomials of degree d as affine dispersers by showing that any such function satisfies $\mathsf{r}(f,0) \leq n - \Omega(d \cdot n^{1/(d-1)})$. This result resembles our Corollary 2; however, the two results are incomparable for two reasons. First, degree d functions over $\mathbb{F}_2$ can have very large

Fourier degree; moreover, the corresponding result of [6] applies to functions whose range is $\mathbb{F}_2$, while ours applies to functions that take values in the set $\{0, \ldots, C\}$, which can have a much larger size. Furthermore, for $f : \mathbb{F}_2^n \to \{0, \ldots, C\}$, a standard argument (analogous to the one in [16]) shows that $\mathsf{r}(f, 0) \leq O(Cd^3)$, where d here is the Fourier degree. However, this does not address the case where $C = \Omega(n)$, which is when Corollary 2 becomes useful.

Pseudorandom Partitions. As we have mentioned, much prior work on function regularity has been focused on finding pseudorandom partitions of $\mathbb{F}_2^n$. To the best of our knowledge, the earliest result in this direction is due to Green [9]; below, the notation $\mathrm{twr}(x)$ refers to an exponential tower of 2's $2^{2^{\cdot^{\cdot^{\cdot}}}}$ of height x.

▶ **Proposition 3** (Theorem 2.1 in [9]). *For any $f : \mathbb{F}_2^n \to \{0, 1\}$ and $\delta > 0$, there exists a subspace $\mathcal{V}$ of co-dimension $M(\delta) \leq \mathrm{twr}(\lceil 1/\delta^3 \rceil)$ such that for all but a δ-fraction of the affine subspaces $\mathcal{U} = \alpha + \mathcal{V}$, $f_{\mathcal{U}}$ is δ-regular.*

In the same paper, Green showed that $M(\delta) \geq \mathrm{twr}(\Omega(\log(1/\delta)))$ was necessary. Subsequently, Hosseini et al. [12] exhibited a better counterexample showing co-dimension $M(\delta) \geq \mathrm{twr}(\lceil 1/16\delta \rceil)$ is required.

In the above upper and lower bound of [9, 12], the partition of $\mathbb{F}_2^n$ is of a specific form – namely, it is every affine shift of a given subspace. Given this observation, one can ask if there is a partition of $\mathbb{F}_2^n$ into affine subspaces of smaller co-dimension so that in most parts f is δ-regular. As the next proposition, due to Girish et al. [8] shows, this is indeed the case.

▶ **Proposition 4** (Proposition A.1 in [8]). *For any $f : \mathbb{F}_2^n \to [0, 1]$ and $\delta > 0$, there exists a partition Π of $\mathbb{F}_2^n$, where every $\pi \in \Pi$ is an affine subspace of co-dimension at most $\frac{1}{\delta^3}$ such that for all but a δ-fraction of the parts, f_π is δ-regular.*

The proof of Proposition 4 is based on a simple algorithm that greedily fixes the parities corresponding to the largest Fourier coefficients; it is included in Appendix A.1 for completeness.

Although, both these results partition $\mathbb{F}_2^n$ into several affine subspaces where f is δ-regular, they are only meaningful when δ is relatively large. Indeed, Proposition 3 is trivial when $\delta < (\log^*(n))^{-1/3}$, and Proposition 4 when $\delta < n^{-1/3}$. As we mentioned earlier, if we relax our requirement to finding just one affine subspace, there is a simple upper bound on $\mathsf{r}(f, \delta)$ based on a density-increment argument, which goes back to the works of Roth [19] and Meshulam [15].

▶ **Proposition 5** (Folklore). *For any $f : \mathbb{F}_2^n \to [-1, 1]$, we have $\mathsf{r}(f, \delta) \leq \frac{1}{\delta}$.*

We provide a proof of Proposition 5 in Appendix A.1 for completeness.

1.2 Techniques

Upper bound on $\mathsf{r}(f, \delta)$. We give a brief proof sketch of Theorem 1. The proof proceeds by induction over the Fourier degree. The base case corresponds to degree one functions. Our intuition is derived from the following fact. If we have any k real numbers $a_1, \ldots, a_k$ such that the sum of any subset of them has magnitude at most one, then by the pigeonhole principle, there is a non-empty subset $S \subseteq [k]$, and a signing of the numbers in S so that the signed sum has magnitude at most $2^{-\Omega(k)}$. In the degree one case, we partition $\{1, \ldots, n\}$ into consecutive disjoint intervals of size $k = O(\log 1/\delta)$. We apply the above intuition to the k Fourier coefficients in each interval, to obtain signed sums that have small magnitude.

Then, by appropriately choosing an affine subspace, $\mathcal{U}$ of dimension $\Omega(n/\log(1/\delta))$, we show that these signed sums are exactly the Fourier coefficients of the function restricted to $\mathcal{U}$ (see Proposition 13 for a more general statement). We give a more detailed description of how this works in Section 3.

At a high level, we reduce the problem for degree d functions to degree $d-1$ by restricting to an affine subspace of dimension $\widetilde{\Omega}\left(n^{1/d}\right)$, where the function is degree d and all Fourier coefficients at the d-th level are extremely small $\ll \delta/n^d$. For a detailed statement, see Lemma 19. When we use the inductive hypothesis for $d-1$, the last constraint ensures that the degree d coefficients cannot increase the new coefficients by more than $O(\delta)$, even if they combine in the most constructive way possible.

Lemma 19 is also obtained by repeatedly applying the pigeonhole principle. However, the key issue now is that several Fourier coefficients could be affected when we apply a restriction, unlike the degree one case. To avoid this, we apply restrictions iteratively so that each one preserves the small Fourier coefficients from past iterations while still ensuring that *several* new Fourier coefficients are also small. The cost of this procedure is that, in each step, we must apply the pigeonhole principle over larger and larger subsets of coordinates.

Lower Bounds. Here, we give a very high level overview of our lower bounds on $\mathsf{r}(f,\delta)$. The basic idea is to consider functions f with the property that their Fourier spectrum is concentrated on a small number of Fourier coefficients. It turns out (see Proposition 13) that when we restrict to an affine subspace, say $\mathcal{U}$, the Fourier coefficients of $f_{\mathcal{U}}$ are simply signed sums of the Fourier coefficients of f. By our choice of f, if the restricted function was δ-regular, then the large coefficients of f involved in the signed sums somehow cancelled each other out. We show that by choosing the vectors corresponding to the large Fourier coefficients in f appropriately, such a cancellation would imply that the co-dimension of $\mathcal{U}$ must be large. For more detailed sketches of the entries in Table 1, see Appendix A.2.

2 Preliminaries

Notation. $\mathbb{1}\{\cdot\}$ denotes an indicator function that takes the value 1 if the clause is satisfied and 0 otherwise. For a set $J \subseteq [n]$, we use $\mathsf{span}(J)$ to denote the subspace spanned by the standard basis vectors corresponding to the elements in J. We refer to the L_1 norm of $\gamma \in \mathbb{F}_2^n$ by $\|\gamma\|_1$. Given a subset $S \subseteq \mathbb{F}_2^n$, we denote $S^{=t} := S \cap \{u : \|u\|_1 = t\}$. Further, we define the degree of a function $f : \mathbb{F}_2^n \to \mathbb{R}$ to be $\max\{\|\gamma\|_1 : \widehat{f}(\gamma) \neq 0\}$. We frequently interpret a linear transformation $M : \mathbb{F}_2^n \to \mathbb{F}_2^n$ as a matrix and refer to the linear map obtained by taking the transpose of the matrix as M^{T}. At several points, we consider the compositions of functions with linear maps. For a function f and a map $M : \mathbb{F}_2^n \to \mathbb{F}_2^n$, we denote by $f \circ M$ the composition of the functions f with M. In particular, $f \circ M(x) = f(M(x))$.

Probability. The following basic facts from probability theory are useful for us.

▶ **Fact 6** (Hoeffding, [11]). *Suppose $X_1, \ldots, X_n$ are such that $a \leq X_i \leq b$ for all i. Let $M = \frac{X_1 + \ldots X_n}{n}$. Then,*

$$\Pr\Big[|M_n - \mathbf{E}M_n| \geq t\Big] \leq 2\exp\left(\frac{-2t^2 n}{|b-a|}\right).$$

▶ **Definition 7** (Statistical Distance). *Let X and Y be two random variables taking values in a set $\mathcal{S}$. Then we define the statistical distance between X and Y as*

$$|X - Y| := \max_{\mathcal{T} \subseteq \mathcal{S}} \Big| \Pr[X \in \mathcal{T}] - \Pr[Y \in \mathcal{T}] \Big| = \frac{1}{2} \sum_{s \in \mathcal{S}} \Big| \Pr[X = s] - \Pr[Y = s] \Big|.$$

Linear Algebra. We recap two concepts from linear algebra, namely, orthogonal subspaces and direct sum, since they become useful for studying the Fourier spectrum of functions defined over subspaces of $\mathbb{F}_2^n$. For a subspace $\mathcal{A}$ of $\mathbb{F}_2^n$, we denote the **orthogonal subspace** of $\mathcal{A}$ as $\mathcal{A}^\perp = \{\gamma \in \mathbb{F}_2^n : \langle \gamma, \gamma' \rangle = 0, \, \forall \, \gamma' \in \mathcal{A}\}$. We denote by $\dim(\mathcal{A})$, the dimension of $\mathcal{A}$ and $\operatorname{codim}(\mathcal{A}) = n - \dim(\mathcal{A})$.

We now define the notion of the direct sum of two subspaces.

▶ **Definition 8** (Independence, Direct Sum). *Two subspaces $\mathcal{A}, \mathcal{B}$ are* independent *if $a + b \neq 0$ for any non-trivial choice of $a \in \mathcal{A}$ and $b \in \mathcal{B}$. In addition, if $\{a + b : a \in \mathcal{A} \text{ and } b \in \mathcal{B}\} = \mathbb{F}_2^n$, we say that $\mathbb{F}_2^n$ is a* direct sum *of $\mathcal{A}$ and $\mathcal{B}$, written as $\mathcal{A} \oplus \mathcal{B} = \mathbb{F}_2^n$.*[5]

If $\mathcal{A} \oplus \mathcal{B} = \mathbb{F}_2^n$, then $\dim(\mathcal{A}) + \dim(\mathcal{B}) = n$. It is also well known that $\dim(\mathcal{A}^\perp) + \dim(\mathcal{A}) = n$. Note, however, that $\mathcal{A}^\perp$ and $\mathcal{A}$ need not be independent,[6] and often in fact must not be.

▶ **Fact 9.** *Let $\mathcal{A}, \mathcal{B}$ be independent subspaces of $\mathbb{F}_2^n$. Then for all distinct $b, b' \in \mathcal{B}$, the affine subspaces $b + \mathcal{A}$ and $b' + \mathcal{A}$ are mutually disjoint.*

Proof. If $b + a = b' + a'$, then a non-trivial sum of a vector from each $\mathcal{A}$ and $\mathcal{B}$ equals zero, contradicting the fact that $\mathcal{A} \oplus \mathcal{B} = \mathbb{F}_2^n$. ◀

Fourier Analysis. For $f : \mathbb{F}_2^n \to \mathbb{R}$, we can write f in the Fourier representation as

$$f(x) = \sum_{\gamma \in \mathbb{F}_2^n} \widehat{f}(\gamma) \chi_\gamma(x),$$

where $\chi_\gamma(x) := (-1)^{\langle \gamma, x \rangle}$ and $\widehat{f}(\gamma) = \mathbf{E}_x[f(x)\chi_\gamma(x)]$. We say f has degree d if $\max_{\gamma : \widehat{f}(\gamma) \neq 0} \|\gamma\|_1 = d$, and we refer to the degree d part of f by $f^{=d}(x) := \sum_{\|\gamma\|_1 = d} \widehat{f}(\gamma)\chi_\gamma(x)$. For more on this topic, see [17], which uses notation consistent with ours.

Restrictions. We are ultimately concerned with understanding the Fourier coefficients of a function when it is restricted to some affine subspace of $\mathbb{F}_2^n$. In the special case where the coordinates in a set $J \subseteq [n]$ are fixed using the vector $b \in \mathbb{F}_2^J$, we denote the restriction of f thus obtained as the function $f_{J \leftarrow b} : \mathsf{span}(\overline{J}) \to \mathbb{R}$, which can be written as $f_{J \leftarrow b}(x) = f(x + b)$. Next, we recall the formula of the Fourier coefficients of the restricted function. Note that $\{\chi_\gamma(x) := (-1)^{\langle \gamma, x \rangle} : \gamma \in \mathsf{span}(\overline{J})\}$ is a Fourier basis of the restricted function.

▶ **Fact 10** (Fourier Coefficients of Restricted Functions (see [17], Proposition 3.21)). *For every $\gamma \in \mathsf{span}(\overline{J})$ and $b \in \mathsf{span}(J)$,*

$$\widehat{f_{J \leftarrow b}}(\gamma) = \sum_{\beta \in \mathsf{span}(J)} \widehat{f}(\beta + \gamma)\chi_\beta(b).$$

---

[5] Such a subspace $\mathcal{B}$ is sometimes called a **complement** of $\mathcal{A}$. However, this term can be confused with the orthogonal subspace/complement, so we avoid using this terminology.

[6] this might be unexpected at first for those used to working over the reals, but it is essentially because the inner product over $\mathbb{F}_2$ allows self-orthogonal vectors in $\mathbb{F}_2^n$.

2.1 Fourier Analysis on Subspaces

We move to the general setting of restricting functions to arbitrary affine subspaces.[7] Let $\mathcal{U} = \mathcal{V} + \alpha$ be an affine subspace of $\mathbb{F}_2^n$. By the restriction of f to $\mathcal{U}$, we mean the function $f_{\mathcal{U}} : \mathcal{V} \to \mathbb{R}$ defined as $f_{\mathcal{U}}(x) = f(x + \alpha)$.

For the remainder of this section (and paper), let $\mathcal{W}$ be such that $\mathcal{W} \oplus \mathcal{V}^\perp = \mathbb{F}_2^n$. For each element $\gamma \in \mathcal{W}$, consider the function $\chi_\gamma : \mathcal{V} \to \{\pm 1\}$ as $\chi_\gamma(x) = (-1)^{\langle \gamma, x \rangle}$. It is easy to verify that $\{\chi_\gamma : \gamma \in \mathcal{W}\}$ form an orthonormal basis of real-valued functions defined over $\mathcal{V}$ under the inner product given by $\langle p, q \rangle = \mathbf{E}_{x \in \mathcal{V}}[p(x)q(x)]$. We can therefore uniquely associate each vector $\gamma \in \mathcal{W}$ with the function χ_γ, and for $\mathcal{U} = \alpha + \mathcal{V}$, we can write

$$f_{\mathcal{U}}(x) = \sum_{\gamma \in \mathcal{W}} \widehat{f_{\mathcal{U}}}(\gamma)(-1)^{\langle \gamma, x \rangle}. \tag{1}$$

We now state the formal definition of δ-regularity.

▶ **Definition 11** (δ-regularity). *Let $\mathcal{V}$ be a subspace of $\mathbb{F}_2^n$ and $g : \mathcal{V} \to \mathbb{R}$. For $\delta \geq 0$, we say g is δ-regular if $\max_{\gamma \neq 0} |\widehat{g}(\gamma)| \leq \delta$.*

In this section, we present three separate formulas (Fact 12, Proposition 13 and Proposition 16) for the Fourier coefficients of $f_{\mathcal{U}}$, each of which is useful in different contexts. With the exception of Fact 12, which is direct, the proofs of the statements in this section can be found in Appendix B.

First, using the above observations, we have the following simple formula for the Fourier coefficients of $f_{\alpha+\mathcal{V}}$, which follows from the orthogonality of the χ_γ we have defined.

▶ **Fact 12.** *Let $\mathcal{V}, \mathcal{W}$ be subspaces such that $\mathcal{W} \oplus \mathcal{V}^\perp = \mathbb{F}_2^n$ and $\mathcal{U} = \alpha + \mathcal{V}$. For any $\gamma \in \mathcal{W}$, we have that*

$$\widehat{f_{\mathcal{U}}}(\gamma) = \mathbf{E}_{x \in \mathcal{V}}[f(x + \alpha) \cdot (-1)^{\langle \gamma, x \rangle}] = (-1)^{\langle \gamma, \alpha \rangle} \mathbf{E}_{x \in \mathcal{U}}[f(x) \cdot (-1)^{\langle \gamma, x \rangle}].$$

Fact 12 represents a simple and analogous formula for Fourier coefficients of functions restricted to affine subspaces. It also highlights that the magnitude of the Fourier coefficients of a restricted function are unaffected by the choice for shift α as long it corresponds to the same affine subspace.

Our next formula, which shows how the Fourier coefficients of $f_{\mathcal{U}}$ can be written in terms of the Fourier coefficients of f, is an easy consequence of Fact 12.

▶ **Proposition 13.** *Let $\mathcal{V}, \mathcal{W}$ be subspaces such that $\mathcal{W} \oplus \mathcal{V}^\perp = \mathbb{F}_2^n$ and $\mathcal{U} = \alpha + \mathcal{V}$. For any $\gamma \in \mathcal{W}$, we have*

$$\widehat{f_{\mathcal{U}}}(\gamma) = \sum_{\beta \in \gamma + \mathcal{V}^\perp} \widehat{f}(\beta) \cdot (-1)^{\langle \beta, \alpha \rangle}.$$

For a proof of Proposition 13 as well as proofs for the rest of the statements in this section, see Appendix B. We note that Proposition 13 gives a formula analogous to Fact 10 for restrictions to general affine subspaces. This fact will be useful to construct functions and argue that they never become δ-regular when restricted to any sufficiently large subspace. Before we give our final formula, we highlight one particular choice of $\mathcal{W}$ such that $\mathcal{W} \oplus \mathcal{V}^\perp = \mathbb{F}_2^n$.

[7] For an arbitrary subspace $\mathcal{V}$, there is no canonical mapping between vectors and characters when $\mathcal{V} \neq \mathbb{F}_2^n$, and we cannot simply define the vectors χ_γ, for each $\gamma \in \mathcal{V}$, as we did in the case of $\mathbb{F}_2^n$ to be the characters of $\mathcal{V}$.

▶ **Definition 14** (M mapping $\mathcal{V}$ to $\mathsf{span}(J)$)**.** *Given a k-dimensional subspace $\mathcal{V}$, let $B = \{\beta_1, \ldots, \beta_n\}$ be a basis for $\mathbb{F}_2^n$ such that $\mathcal{V} = \mathsf{span}(\{\beta_1, \ldots, \beta_k\}$. For any subset $J \subseteq [n]$ of size k, let $M : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be an invertible linear map such that $\{M\beta_i : i \in [k]\} = \{e_j : j \in J\}$.*

▶ **Proposition 15** (Choice of $\mathcal{W}$)**.** *Let $\mathcal{V}$, M and J be defined as in Definition 14. The subspaces $\mathcal{W} = \{M^{\mathsf{T}}\gamma : \gamma \in \mathsf{span}(J)\}$ and $\mathcal{V}^{\perp}$ are independent, and $\mathcal{W} \oplus \mathcal{V}^{\perp} = \mathbb{F}_2^n$.*

Finally, we show that the Fourier coefficients of a function restricted to an affine subspace are the same as the Fourier coefficients of the function $f \circ M$ under a suitable (normal) restriction and for a particular choice of M.

▶ **Proposition 16.** *Let $\mathcal{V}$, M and J be defined as in Definition 14 and $\mathcal{U} = \alpha + \mathcal{V}$. For any $\gamma \in \mathsf{span}(J)$, we have $\left|\widehat{f_{\mathcal{U}}}(M^{\mathsf{T}}\gamma)\right| = \left|\widehat{h_{\mathcal{U}'}}(\gamma)\right|$, where $h = f \circ M^{-1}$ and $\mathcal{U}' = \{Mu : u \in \mathcal{U}\} = M\alpha + \mathsf{span}(J)$ is a standard restriction.*

Proposition 16 implies the following important corollary.

▶ **Corollary 17.** *There exists an affine subspace $\mathcal{U}$ of dimension k such that $f_{\mathcal{U}}$ is δ-regular if and only if there exists an invertible linear map $M : \mathbb{F}_2^n \to \mathbb{F}_2^n$, a set $J \subseteq [n]$ of size k, and a fixing of coordinates outside J given by $b \in \mathbb{F}_2^{\overline{J}}$ such that the function $h_{\overline{J} \leftarrow b}$ is δ-regular, where $h = f \circ M$.*

We use Corollary 17 crucially in the proof of Theorem 1, wherein we construct M and b such that $f \circ M_{\overline{[k]} \leftarrow b}$ has small Fourier coefficients. In the proof of this theorem we must understand the Fourier coefficients of $f \circ M$ in terms of the Fourier coefficients of f. The following fact gives an identity relating the Fourier coefficients of the two functions. For completeness, we include the proof in Appendix B.

▶ **Fact 18** ([17], Exercise 3.1)**.** *Let M be an invertible linear transformation, and consider the function $g = f \circ M^{-1} : \mathbb{F}_2^n \to \mathbb{R}$. Then we have that $\widehat{g}(\gamma) = \widehat{f}(M^{\mathsf{T}}\gamma)$.*

3 Upper Bound on $\mathsf{r}(f, \delta)$

Now we prove our main theorem, restated here for convenience.

▶ **Theorem 1.** *For any $\delta \in (0, 1)$ and any degree d function $f : \mathbb{F}_2^n \to [-1, 1]$, we have $\mathsf{r}(f, \delta) \leq n - \Omega\left(n^{1/d!}(\log(n/\delta))^{-2}\right)$.*

First, we gain some intuition from degree one functions.

Base Case/Toy Example. Suppose f is a Fourier degree one function. In this case our function has the form

$$f(x) = \widehat{f}(0) + \sum_i \widehat{f}(e_i)(-1)^{x_i}.$$

For a parameter $t \geq 1$ and a subset $S \subseteq [t]$, consider the sum $g_S = \widehat{f}(0) + \sum_{i \in S} \widehat{f}(e_i)$. Note that $g_S = \mathbf{E}[f(x) | x_i = 0 \ \forall i \in S] \in [-1, 1]$. The pigeonhole principle implies that for $t = \Omega(\log 1/\delta)$ there must exist two distinct sets S, S' such that the difference $|g_S - g_{S'}| \leq \delta$. We can further write $g_S - g_{S'} = \sum_{i \in S \triangle S'} \widehat{f}(e_i)(-1)^{|\{i\} \cap S'|}$.

We now use the set $S \triangle S'$ and the signs to construct an affine subspace where at least one Fourier coefficient will have small magnitude. Assume without loss of generality that $1 \in S \setminus S'$ and $S \triangle S' = [t']$ for some $t' \leq t$. Consider restricting f to the affine subspace $\mathcal{U}$

defined by the linear equations $x_1 + x_i = b_i$ for each $i \in \{2, \dots, t'\}$, where $b_i = |\{i\} \cap S'|$. We can reason about the Fourier spectrum of $f_{\mathcal{U}}$ by plugging in $x_i = b_i + x_1$. Under this restriction, we see that the Fourier coefficients of $e_{t'+1}, \dots, e_n$ stay the same, and the new Fourier coefficient of e_1 is exactly equal to

$$\widehat{f}(e_1) + \sum_{i=2}^{t'} \widehat{f}(e_i)(-1)^{b_i} = g_S - g_{S'},$$

which we observed has magnitude at most δ. Repeatedly applying this argument roughly $n(\log(1/\delta))^{-1}$ times for the remaining standard basis vectors and fixing remaining coordinates arbitrarily, we obtain an affine subspace of dimension at least $\Omega\left(\frac{n}{\log(1/\delta)}\right)$.

Theorem 1 is proved via induction using the following lemma.

▶ **Lemma 19.** *For $\tau \in (0,1)$ and any degree d function $f : \mathbb{F}_2^n \to [-1,1]$, there exists an invertible linear map $M : \mathbb{F}_2^n \to \mathbb{F}_2^n$, a set $J \subseteq [n]$ with size at least $\frac{d}{4e}\left(\frac{n}{\log 5/\tau}\right)^{1/d}$, and $b \in \mathsf{span}(\overline{J})$ such that $h = f \circ M$ satisfies*

$$\left|\widehat{h_{\overline{J}\leftarrow b}}(\gamma)\right| \leq \begin{cases} \tau & \text{if } \|\gamma\|_1 = d, \\ 0 & \text{for all } \|\gamma\|_1 > d. \end{cases}$$

We now prove Theorem 1 using Lemma 19.

Proof. The proof proceeds by induction over the degree. Our inductive hypothesis is that for any $\delta > 0$ and any degree d function f, there exists an invertible linear map M, a set $I \subseteq [n]$, and $b \in \mathsf{span}(\overline{I})$ such that the following two items hold:

1. $h_{\overline{I}\leftarrow b}$ is δ-regular, where $h = f \circ M$, and
2. for $C_d = \sum_{i=1}^{d} (i!)^{-1}$, we have

$$|I| \geq \frac{n^{1/d!}}{(8e)^{C_{d-1}} \left(\log(n/\delta)\right)^{C_d}}.$$

Note that $C_d \leq e - 1 < 2$ for all $d \geq 1$. The existence of the desired affine subspace is then given by Corollary 17, and its dimension is equal to $|I| \geq \Omega\left(n^{1/d!} \left(\log(n/\delta)\right)^{-2}\right)$.

The base case corresponds to the degree being one. Let us apply Lemma 19 for degree one with $\tau = \delta$ and denote $g = f \circ M$, where M is the linear map M promised by the lemma. Additionally, we have a set J of size at least $\frac{n}{4e \log 5/\delta} \geq \Omega\left(\frac{n}{\log n/\delta}\right)$, and $b \in \mathsf{span}(\overline{J})$ such that

$$\left|\widehat{g_{\overline{J}\leftarrow b}}(\gamma)\right| \leq \begin{cases} \tau & \text{if } \|\gamma\|_1 = 1, \\ 0 & \text{for all } \|\gamma\|_1 > 1. \end{cases} \implies \left|\widehat{g_{\overline{J}\leftarrow b}}(\gamma)\right| \leq \delta, \text{ for all } \gamma \neq 0.$$

Assuming both items hold for some degree $d-1$, we show them for degree d. Applying Lemma 19 with degree d and $\tau = n^{-d}\delta/3$, we denote $p := (f \circ M)_{\overline{J}\leftarrow b}$, where M, J and b are as promised by the lemma. Note that, by Lemma 19, p has degree at most d, and for any γ with $\|\gamma\|_1 = d$, we have, $|\widehat{p}(\gamma)| \leq \delta/(3n^d)$. Consider the functions $p^{<d}$ and $p^{=d}$, which are the degree at most $d-1$ part of p and the degree d part of p, respectively. We note that $\frac{p^{<d}}{(1+\delta/3)}$ is bounded in the interval $[-1,1]$ because for any x,

$$\left|p^{<d}(x)\right| \leq |p(x)| + \left|p^{=d}(x)\right| \leq 1 + \sum_{\gamma : \|\gamma\|_1 = d} |\widehat{p}(\gamma)| \leq 1 + \frac{\delta}{3}.$$

Applying the inductive hypothesis[8] to $\frac{p^{<d}}{1+\delta/3}$ for the choice of $\delta/3$, we get a linear map M', a set $I \subseteq J$, and $b' \in \mathsf{span}(J \setminus I)$ such that $\left(\frac{q}{1+\delta/3}\right)_{J\setminus I \leftarrow b'}$ is $\delta/3$-regular, where $q := p^{<d} \circ M'$. Therefore, for any $\gamma \neq 0$, we have $\left|\widehat{q_{J\setminus I\leftarrow b'}}(\gamma)\right| \leq \left(1+\frac{\delta}{3}\right)\frac{\delta}{3} < \frac{2\delta}{3}$. Denoting $p' := p \circ M'$ and $r := p^{=d} \circ M'$, we have for any $\gamma \neq 0$ that

$$\left|\widehat{p'_{J\setminus I\leftarrow b'}}(\gamma)\right| \leq \left|\widehat{q_{J\setminus I\leftarrow b'}}(\gamma)\right| + \left|\widehat{r_{J\setminus I\leftarrow b'}}(\gamma)\right| < 2\delta/3 + \sum_{\beta:\|\beta\|_1=d} |\widehat{g}(\beta)| \leq \delta.$$

This shows that ${p'}_{J\setminus I\leftarrow b'}$ is δ-regular. Moreover, if we extend M' to act as the identity map on the coordinates in $\overline{J}$, we can write

$$\begin{aligned}{p'}_{J\setminus I\leftarrow b'}(x) = (p\circ M')_{J\setminus I\leftarrow b'}(x) &= p(M'(x+b')) \\ &= (f\circ M)_{J\leftarrow b}(M'(x+b')) = f(MM'(x+b'+b)),\end{aligned}$$

which implies that item 1 of the inductive hypothesis is satisfied by applying the linear map MM' and restricting to the set I by fixing the coordinates outside according to $b+b'$.

We now show that the size of I satisfies item 2 above. Note that Lemma 19 promises that $|J| \geq \frac{d}{4e}\left(\frac{n}{\log(15n^d/\delta)}\right)^{1/d}$. Moreover, we have

$$\log(15n^d/\delta) \leq d\log n/\delta + \log 15 \leq 4d\log n/\delta,$$

where the last inequality follows for sufficiently large n. Therefore, $|J| \geq \frac{1}{8e}\left(\frac{n}{\log(n/\delta)}\right)^{1/d}$. Moreover, we assume without loss of generality that $3|J| \leq n$ because, if not, we can arbitrarily fix coordinates in J until it is, which does not affect the crucial property that all remaining degree d Fourier coefficients have small magnitude. Using the bounds on $|J|$ and applying item 2 of the inductive hypothesis for degree $d-1$, we get

$$\begin{aligned}|I| \geq \frac{|J|^{1/(d-1)!}}{(8e)^{C_{d-2}}\left(\log(3|J|/\delta)\right)^{C_{d-1}}} &\geq \frac{n^{1/d!}}{(8e)^{C_{d-1}}\log(n/\delta)^{1/d!}\left(\log(3|J|/\delta)\right)^{C_{d-1}}} \\ &\geq \frac{n^{1/d!}}{(8e)^{C_{d-1}}\left(\log(n/\delta)\right)^{C_d}}.\end{aligned}$$

This shows item 2 of the inductive hypothesis as desired. ◀

To prove Lemma 19, we need the following claim, which ultimately lets us bound Fourier coefficients in certain affine subspaces.

▷ Claim 20 (Pigeonhole Principle). Let $f : \mathbb{F}_2^n \to [-1,1]$ be degree d. For every $K \subseteq [n]$ of size k such that $n-k \geq \binom{k}{d-1}\log(5/\tau)$, there exists $S \subseteq [n]\setminus K$ and $z \in \{\pm 1\}^S$ such that

1. $\forall \gamma \in \mathsf{span}(K)$ with $\|\gamma\|_1 = d-1$, we have $\left|\sum_{j\in S}\widehat{f}(\gamma+e_j)\cdot z_j\right| \leq \tau$, and
2. $1 < |S| \leq \binom{k}{d-1}\log(5/\tau)$.

Proof. Consider any subset of $T \subseteq \overline{K}$ of size $\binom{k}{d-1}\log(5/\tau)$. For any $U \subseteq T$, consider the sum

$$a_U(\gamma) := \widehat{f}(\gamma) + \sum_{j\in U}\widehat{f}(\gamma+e_j).$$

[8] Technically, $\frac{p^{<d}}{1+\delta/3} : \mathsf{span}(J) \to [-1,1]$. However, we can abuse notation slightly and consider it as a function from $\mathbb{F}_2^J$ to $[-1,1]$ in order to apply the inductive hypothesis.

We must have that $a_U(\gamma) \in [-1,1]$ since it is exactly equal to the Fourier coefficient corresponding to γ if we restricted everything in U to be one. This follows because f is degree d.

Now, divide the interval $[-1,1]$ into $2/\tau$ intervals of length τ. For a fixed $U \subseteq T$ of even size, consider putting the values of $a_U(\gamma)$ for all $\gamma \in \mathsf{span}(K)^{=d-1}$ into a vector v_U of length $\binom{k}{d-1}$. First, note that the number of even subsets of T is at least $2^{\binom{k}{d-1}\log(5/\tau)-1} > (2/\tau)^{\binom{k}{d-1}}$. Moreover, the number of possible interval vectors is at most $(2/\tau)^{\binom{k}{d-1}}$. Therefore, by the pigeonhole principle, there must be two distinct sets $U, U' \subseteq T$ such that $\|v_U - v_{U'}\|_\infty \leq \tau$.

Thus, we have that

$$\|v_U - v_{U'}\|_\infty \leq \tau \iff \sum_{i \in U \triangle U'} (-1)^{|\{i\}\cap U'|} \widehat{f}(\gamma + e_i) \leq \tau \quad \forall\, \gamma \in \mathsf{span}(K)^{=d-1}.$$

Since U, U' have even size and are not equal, $U \triangle U'$ has even size as well, so we can set our $S = U \triangle U' \subseteq T$ and $z_i = (-1)^{|\{i\}\cap U'|}$, and the claim follows. ◁

We can now prove Lemma 19.

Proof of Lemma 19. We build the map M, the set J, and the vector b iteratively. Throughout the iterations, we seek to maintain a set K of coordinates for which (under a suitable linear transformation M) every Fourier coefficient corresponding to a vector of weight d in $\mathsf{span}(K)$ has magnitude at most τ. We build K one coordinate at a time by repeatedly invoking Claim 20 and arguing that the quantities guaranteed to be small by Claim 20 are exactly the (new) Fourier coefficients. When we can no longer add more coordinates to K, we fix any remaining coordinates (outside of K that are still alive), and we are left with a function, over only the coordinates in K, that has the desired property.

Note that we can start with K being an arbitrary subset of size $d-1$ (w.l.o.g. let it be $[d-1]$) since any such subset has no Fourier coefficients of degree d. Therefore, we can assume without loss of generality that $\tau \geq 5 \cdot 2^{-n/(4e)^d}$, since otherwise $\frac{d}{4e}\left(\frac{n}{\log(5/\tau)}\right)^{1/d} < d$ and the lemma becomes trivial. In each iteration, we maintain the following invariant for M, J and b. In iteration i, there exists some $K \subseteq J$ of size $d+i-1$ such that the function $g = (f \circ M)_{\overline{J} \leftarrow b}$ satisfies

$$|\widehat{g}(\gamma)| \leq \begin{cases} \tau & \text{if } \gamma \in \mathsf{span}(K) \text{ and } \|\gamma\|_1 = d, \\ 0 & \text{for all } \|\gamma\|_1 > d. \end{cases}$$

Assume without loss of generality that $J = [j]$ for some $j \leq n$ and $K = [d+i-1] \subseteq J$. Since g has degree d, we can apply Claim 20 to g and obtain a subset $S \subseteq J \setminus K$ of size at most $\binom{d+i-1}{d-1}(\log(5/\tau)$ and a sign vector $z \in \{\pm 1\}^S$ so that

$$\left|\sum_{j \in S} \widehat{g}(\gamma + e_j) \cdot z_j\right| \leq \tau, \qquad \text{for all } \gamma \in \mathsf{span}([d+i-1]) \text{ such that } \|\gamma\|_1 = d-1. \tag{2}$$

We can also assume that $d+i \in S$ and $z_{d+i} = 1$. Now consider the invertible linear transformation $M_i : \mathbb{F}_2^n \to \mathbb{F}_2^n$ that maps e_{d+i} to $\sum_{j \in S} e_j$ and behaves as the identity map on the remaining standard basis vectors. Further, denote $J_i := S \setminus \{d+i\}$ and let $b_i \in \mathsf{span}(J_i)$, where $(b_i)_j := (1 - z_j)/2$ for each $j \in J_i$. Intuitively, applying the linear transformation M_i and then fixing the coordinates in J_i to b_i corresponds to restricting the affine subspace described by the equations $x_j + x_{d+i} = (1 - z_j)/2$ for all $j \in J_i$.

After this iteration, we show that if we set $M' \leftarrow MM_i$, $J' \leftarrow J \setminus J_i$ and $b' \leftarrow b + b_i$, the invariant holds with $K' \leftarrow K \cup \{d+i\}$. For these choices, we have

$$\begin{aligned}(f \circ M')_{\overline{J'} \leftarrow b'}(x) = f \circ M(M_i(x + b')) &= f \circ M(M_i(x + b_i + b)) \\ &= f \circ M(M_i(x + b_i) + b) \\ &= g \circ M_i(x + b_i) = (g \circ M_i)_{J_i \leftarrow b_i}(x),\end{aligned}$$

and it therefore suffices to show that $(g \circ M_i)_{J_i \leftarrow b_i}$ – denoted by h henceforth, for shorthand – is degree d and $\left|\widehat{h}(\gamma)\right| \leq \tau$ for all $\gamma \in \mathsf{span}([d+i])$ with $\|\gamma\|_1 = d$. We start by analyzing the Fourier coefficients of h, for which by Fact 10 we have

$$\widehat{h}(\gamma) = \sum_{\beta \in \mathsf{span}(J_i)} \widehat{g \circ M_i}(\gamma + \beta)(-1)^{\langle \beta, b_i \rangle}. \tag{3}$$

Next, we observe the following relation between the Fourier coefficients of $g \circ M_i$ and those of g, which we use to simplify Equation (3). Denoting $v := \sum_{j \in J_i} e_j$, we claim that, for any γ,

$$\widehat{g \circ M_i}(\gamma) = \widehat{g}(\gamma + e_{d+i}\langle \gamma, v \rangle). \tag{4}$$

Before proving Equation (4), we use it to prove that h has the desired properties. Note that since g is degree d, Equation (4) implies that if $\widehat{g \circ M_i}(\gamma) \neq 0$, then $\|\gamma + e_{d+i}\langle \gamma, v \rangle\|_1 \leq d$, which in turn implies that $\|\gamma\|_1 \leq d+1$. This immediately tells us that $g \circ M_i$ has degree at most $d+1$; therefore, h also has degree at most $d+1$ since the degree cannot increase under restrictions. Now, for any γ, Equation (3) reduces to

$$\begin{aligned}\widehat{h}(\gamma) &= \sum_{\substack{\beta \in \mathsf{span}(J_i), \\ \|\beta\|_1 \leq d+1-\|\gamma\|_1}} \widehat{g \circ M_i}(\gamma + \beta)(-1)^{\langle \beta, b_i \rangle} \\ &= \widehat{g \circ M_i}(\gamma) + \sum_{\substack{\beta \in \mathsf{span}(J_i), \\ 0 < \|\beta\|_1 \leq d+1-\|\gamma\|_1}} \widehat{g \circ M_i}(\gamma + \beta)(-1)^{\langle \beta, b_i \rangle} \\ &= \widehat{g}(\gamma + e_{d+i}\langle \gamma, v \rangle) + \sum_{\substack{\beta \in \mathsf{span}(J_i), \\ 0 < \|\beta\|_1 \leq d+1-\|\gamma\|_1}} \widehat{g}(\gamma + \beta + e_{d+i}\langle \gamma + \beta, v \rangle)(-1)^{\langle \beta, b_i \rangle}, \end{aligned} \tag{5}$$

where, in the first equality, we used the fact that if $\|\beta\|_1 > d+1-\|\gamma\|_1$, then $\|\beta+\gamma\|_1 > d+1$ and the corresponding Fourier coefficient in $g \circ M_i$ is just zero, and in the last equality, we used Equation (4). Moreover, for any $\gamma \in \mathsf{span}(J \setminus J_i)$, we have $\langle \gamma, v \rangle = 0$, which means that $\widehat{g}(\gamma + e_{d+i}\langle \gamma, v \rangle) = \widehat{g}(\gamma)$. We can now conclude that h has degree at most d. Indeed, if $\|\gamma\|_1 \geq d+1$, then Equation (5) implies that $\widehat{h}(\gamma) = \widehat{g}(\gamma) = 0$ since g has degree at most d.

Next, we show that for any $\gamma \in \mathsf{span}([d+i])$ with $\|\gamma\|_1 = d$, it must be that $|\widehat{h}(\gamma)| \leq \tau$. Applying Equation (5) for such γ, we note that

$$\begin{aligned}\widehat{h}(\gamma) &= \widehat{g}(\gamma) + \sum_{j \in J_i} \widehat{g}(\gamma + e_j + e_{d+i}\langle \gamma + e_j, v \rangle)(-1)^{\langle e_j, b_i \rangle} \\ &= \widehat{g}(\gamma) + \sum_{j \in J_i} \widehat{g}(\gamma + e_j + e_{d+i}) z_j.\end{aligned}$$

We now consider two cases. First, when $\gamma_{d+i} = 0$, the above equation implies that $\widehat{h}(\gamma) = \widehat{g}(\gamma)$ since $\|\gamma + e_{d+i} + e_j\|_1 = d+2$ for every $j \in J_i$, and g has degree at most d. Therefore, in

this case, $\left|\widehat{h}(\gamma)\right| = |\widehat{g}(\gamma)| \leq \tau$ by the inductive hypothesis. Otherwise, $\gamma_{d+i} = 1$, and now using both Equation (2) and the fact that $\gamma + e_{d+i} \in \mathsf{span}(\{e_1, \ldots, e_{d+i-1}\})$, we conclude that $\left|\widehat{h}(\gamma)\right| = \left|\sum_{j \in S} \widehat{g}((\gamma + e_{d+i}) + e_j) z_j\right| \leq \tau$.

It remains to show Equation (4). We start by observing that $M_i = M_i^{-1}$, which can be verified by noting that $M_i^{-1} e_{d+i} = M_i^{-1}(e_{d+i} + v + v) = e_{d+i} + v$ and M_i^{-1} acts as the identity map on the remaining standard basis vectors. From Fact 18, we know that $\widehat{g \circ M_i}(\gamma) = \widehat{g \circ M_i^{-1}}(\gamma) = \widehat{g}(M_i^{\mathsf{T}}\gamma)$. Since the rows of M_i^{T} are the same as the columns of M_i, we have

$$(M_i^{\mathsf{T}}\gamma)_j = \begin{cases} \langle v + e_{d+i}, \gamma \rangle & \text{if } j = d+i, \\ \gamma_j & \text{otherwise.} \end{cases}$$

Therefore, we can write $M_i^{\mathsf{T}}\gamma = \sum_{j \neq d+i} \gamma_j e_j + e_{d+i}\langle v + e_{d+i}, \gamma \rangle = \gamma + e_{d+i}\langle v, \gamma \rangle$, as claimed.

We conclude the argument by calculating how many times we can repeat the above procedure. Note that, in the i-th iteration, we fixed at most $\binom{d+i-1}{d-1} \log 5/\tau - 1$ coordinates and we added exactly one coordinate to K. We can thus continue this process until iteration t for the largest value of t such that

$$\log(5/\tau) \cdot \left(\sum_{i=1}^{t} \binom{d+i-1}{d-1} \right) \leq n - d + 1.$$

Simplifying the binomial sum, we get

$$\begin{aligned} \sum_{i=1}^{t} \binom{d+i-1}{d-1} = \sum_{i=1}^{t} \binom{d+i-1}{i} &= \sum_{i=1}^{t} \binom{d+i-1}{i} + \binom{d}{0} - 1 \\ &= \binom{d+t}{t} - 1 < \left(\frac{e(d+t)}{d} \right)^d, \end{aligned}$$

where the last equality follows by repeatedly using the identity $\binom{a}{i} + \binom{a}{i-1} = \binom{a+1}{i}$. Thus, we can set $t = \frac{d}{e}\left(\frac{n-d+1}{\log 5/\tau}\right)^{1/d} - d$. Adding in the initial $d-1$ coordinates, at the end of the t iterations, we can bound $|K|$ as,

$$\begin{aligned} |K| &= \frac{d}{e}\left(\frac{n-d+1}{\log 5/\tau}\right)^{1/d} - d + d - 1 \\ &\geq \frac{d}{e}\left(\frac{n}{\log 5/\tau}\left(1 - \frac{d-1}{n}\right)\right)^{1/d} - 1 \\ &\geq \frac{d}{e}\left(\frac{n}{\log 5/\tau} \cdot \frac{1}{d}\right)^{1/d} - 1 \\ &\geq \frac{d}{2e}\left(\frac{n}{\log 5/\tau}\right)^{1/d} - 1 && (d^{1/d} \leq 2 \quad \forall d \geq 1) \\ &= \frac{d}{4e}\left(\frac{n}{\log 5/\tau}\right)^{1/d} + \frac{d}{4e}\left(\frac{n}{\log 5/\tau}\right)^{1/d} - 1 \\ &\geq \frac{d}{4e}\left(\frac{n}{\log 5/\tau}\right)^{1/d} + d - 1 && (\text{since } \tau \geq 5 \cdot 2^{-n/(4e)^d}) \\ &\geq \frac{d}{4e}\left(\frac{n}{\log 5/\tau}\right)^{1/d}. && (d \geq 1) \end{aligned}$$

At the end of t iterations, we can fix any coordinates outside the set K arbitrarily to ensure that the only non-zero Fourier coefficients with L_1 norm d in the resulting function must correspond to vectors in $\mathsf{span}(K)$, which do not change under the restriction. ◀

4 Applications

We now present an application of Theorem 1 that shows a tradeoff between the dimension of a disperser and its Fourier degree, and a connection to extractors, as well. First, we introduce a definition that generalizes Boolean functions and helps us reason about the Fourier spectrum of dispersers.

▶ **Definition 21.** *We say a function $f : \mathbb{F}_2^n \to \mathbb{R}$ is G-granular if for every $x \in \mathbb{F}_2^n$, we have that $f(x)$ is an integer multiple of G.*

▷ Claim 22. If a degree d function $f : \mathbb{F}_2^n \to \mathbb{R}$ is G-granular, then for every $\gamma \in \mathbb{F}_2^n$, we have that $\widehat{f}(\gamma)$ is an integer multiple of $2^{-d} \cdot G$.

We defer the proof of Claim 22 to the full version. Assuming the claim, we now show that low degree granular functions cannot have a large parity kill number. As a consequence, we get that low-degree affine dispersers cannot have small dimension (Corollary 2).

▶ **Lemma 23.** *Every degree d function $f : \mathbb{F}_2^n \to [-1, 1]$ that is G-granular satisfies*

$$C^{\oplus}_{\min}[f] \leq n - \Omega\left(n^{1/d!}(d + \log n/G)^{-2}\right).$$

Proof. If f is G-granular and degree d, then from Claim 22 we know that all its Fourier coefficients must be integer multiples of $2^{-d} \cdot G$. Moreover, a Fourier coefficient of f in any affine subspace is simply a signed sum of the Fourier coefficients of f and therefore it must also be an integer multiple of $2^{-d} \cdot G$. This shows that if f is δ-regular in some affine subspace $\mathcal{U}$ with $\delta < 2^{-d} \cdot G$, then $f_{\mathcal{U}}$ must be constant. The lemma follows by using Theorem 1 for $\delta = 2^{-d-1} \cdot G$. ◀

Proof of Corollary 2. Using f, we can construct a degree d function $h : \mathbb{F}_2^n \to [-1, 1]$ as $h(x) = 1 - \frac{2f(x)}{C}$. Noting that h is $2/C$-granular and using the above lemma, it follows that

$$C^{\oplus}_{\min}[f] = C^{\oplus}_{\min}[h] \leq n - \Omega\left(n^{1/d!}(d + \log(nC))^{-2}\right),$$

which shows that there is some affine subspace of dimension at least $\Omega\left(n^{1/d!}(2d + \log(nC))^{-2}\right)$ where f is constant. ◀

▶ Remark 24. Affine dispersers can be viewed as a relaxation of affine extractors, objects that have been well studied in the pseudorandomness literature. In a similar vein, we observe a connection between the notion of δ-regularity and affine extractors. In particular, we note that affine extractors can be viewed as functions that are δ-regular in *all* affine subspaces of sufficiently large dimension. We detail this connection in the full version of this work.

5 Lower Bounds on $\mathsf{r}(f, \delta)$

In this section, we prove lower bounds on $\mathsf{r}(f, \delta)$. With the exception of Lemma 26, we defer the proofs of all the statements to the appendices. We start with lower bounds for functions f that are bounded in the interval $[-1, 1]$; in the subsequent section, we give lower bounds for Boolean functions. For detailed sketches of all the results in this section, see Appendix A.2.

5.1 Bounded Functions

We begin with a simple bound on the number of standard basis vectors in low-dimensional affine subspaces, which is crucial in the analysis of the lower bounds.

▶ **Observation 25.** *For a subspace $\mathcal{V} \subseteq \mathbb{F}_2^n$ of co-dimension C, and $\mathcal{W}$ such that $\mathcal{W} \oplus \mathcal{V}^\perp = \mathbb{F}_2^n$, there exists a set $S \subseteq \mathcal{W}$ of size at least $n - C$ such that for every $u \in S$,*

$$|\left(u + \mathcal{V}^\perp\right)^{=1}| \geq 1.$$

Moreover, there exists a subset $S_1 \subseteq S$ of size at least $n - 2C$ whose corresponding shifts contain exactly *one standard basis vector.*

The proof of Observation 25 can be found in Appendix C. Using this observation, we provide an example of a degree one function f that witnesses large values for $\mathsf{r}(f, \delta)$.

▶ **Lemma 26.** *There is a degree one function $f : \mathbb{F}_2^n \to [-1, 1]$ for which $\mathsf{r}(f, \delta) \geq n/2$, for all $\delta < 1/n$.*

Proof. The counterexample is given by the function $f(x) = \frac{1}{n} \cdot \sum_i (-1)^{e_i \cdot x}$. Let $\mathcal{V}$ be a subspace of $\mathbb{F}_2^n$ of co-dimension C, and suppose we restrict the function to the affine subspace $\mathcal{U} = \alpha + \mathcal{V}$. By Observation 25, if $C \leq n/2 - 1$, there exists at least two vectors $\gamma, \gamma' \in \mathcal{W}$ (where $\mathcal{W}$ is such that $\mathcal{W} \oplus \mathcal{V}^\perp = \mathbb{F}_2^n$) such that $|(\gamma + \mathcal{V}^\perp)^{=1}| = |(\gamma' + \mathcal{V}^\perp)^{=1}| = 1$. Assume without loss of generality that $\gamma \neq 0$. Then, by Proposition 13, we have that

$$|\widehat{f_{\mathcal{U}}}(\gamma)| = \Big| \sum_{\eta \in u + \mathcal{V}^\perp} \widehat{f}(\eta)(-1)^{\langle \eta, \alpha \rangle} \Big| = \frac{1}{n} > \delta,$$

which follows by observing that exactly one of the summands in the last sum corresponds to a weight one vector and is non-zero. Therefore, $\mathsf{r}(f, \delta) \geq n/2$. ◀

We next show that Lemma 26 can be generalized to degree d bounded functions.

▶ **Lemma 27.** *For $d > 2$ and $\delta < \binom{n}{d}^{-1}$, there exists a degree d function $f : \mathbb{F}_2^n \to [-1, 1]$ for which $\mathsf{r}(f, \delta) \geq n - 2dn^{1/(d-1)}$.*

The proof of Lemma 27 can also be found in Appendix C. We note that unlike the degree one case, the functions achieving the lower bound in the above lemma are not explicit. Additionally, when $d = 2$, we see that Lemma 27 is trivial; it would be interesting to obtain a tighter result in this case.

5.2 Boolean Functions

This section has two parts. The first gives non-explicit lower bounds on $\mathsf{r}(f, \delta)$ for Boolean functions, and the second gives explicit lower bounds.

5.2.1 Non-explicit Lower Bounds on $\mathsf{r}(f, \delta)$

We can turn our lower bounds on $\mathsf{r}(f, \delta)$ for bounded functions into (non-explicit) lower bounds for Boolean functions. To do so, we use the following simple but powerful lemma of [12], which states that given a bounded function with a large $\mathsf{r}(f, \delta)$, there must exist some Boolean function g with similarly a large $\mathsf{r}(g, 2\delta)$.

▶ **Proposition 28** ([12], Claim 1.2). *Let $\tau > 0$ and $f : \mathbb{F}_2^n \to [-1, 1]$. There exists a Boolean function $g : \mathbb{F}_2^n \to \{\pm 1\}$ satisfying, for every affine subspace $\mathcal{U}$ such that $|\mathcal{U}| \geq \frac{4n^2}{\tau^2}$ and any $\gamma \in \mathbb{F}_2^n$, that $\left|\widehat{f_{\mathcal{U}}}(\gamma) - \widehat{g_{\mathcal{U}}}(\gamma)\right| \leq \tau$.*

Using Proposition 28, we have the following lemma.

▶ **Lemma 29.** *For all $d \geq 3$ and $\delta < \frac{1}{2} \cdot \binom{n}{d}^{-1}$, there exists a Boolean function f with*

$$\mathsf{r}(f, \delta) \geq n - \max\left\{2d \cdot n^{1/(d-1)}, \log\left(16n^2/\delta^2\right)\right\}.$$

Proof. By Lemma 27, there exists a bounded f that is not δ-regular in any affine subspace of dimension at least $2dn^{1/(d-1)}$ for all $\delta < \binom{n}{d}^{-1}$. Proposition 28 tells us that there exists a Boolean function g whose Fourier coefficients agree up to an additive error $\delta/2$ with the Fourier coefficients of f on all affine subspaces of dimension at least $\log\left(16n^2/\delta^2\right)$. Therefore, if f is not δ-regular on all of these affine subspaces, then g is also not $\delta/2$-regular on any of these subspaces. ◀

We can plug some parameters into Lemma 29 and achieve the following more parsable corollary.

▶ **Corollary 30.** *For every $3 \leq d \leq \frac{\log n}{\log\log n + 1}$ and $\delta = \frac{1}{2} \cdot n^{-d}$, there exists a Boolean function f with $\mathsf{r}(f, \delta) \geq n - 2d \cdot n^{1/(d-1)}$.*

We include the proofs of Proposition 28 and Corollary 30 in Appendix C.

5.2.2 Explicit Lower Bounds on $\mathsf{r}(f, \delta)$

We now turn to lower bounds on $\mathsf{r}(f, \delta)$ given by explicit Boolean functions. Our first example comes from analyzing certain Boolean functions studied by [18] that provide lower bounds for $\mathsf{r}(f, 0)$.

▶ **Lemma 31** (Related to Corollary 1.1 in [18]). *For each $\delta > 0$, there exists an explicit Boolean function $f : \mathbb{F}_2^n \to \{0, 1\}$ with $\mathsf{r}(f, \delta) = \Omega\left((\log \frac{1}{\delta})^{\log_2(3)}\right)$.*

Our second example is the majority function; we show that the majority function, denoted by MAJ_n, has a large $\mathsf{r}(f, \delta)$ value when $\delta = O(1/\sqrt{n})$. This in particular rules out the possibility of proving a bound of the form $\mathsf{r}(f, \delta) \leq \mathsf{poly}(\log(1/\delta))$ for Boolean f.

▶ **Lemma 32.** *There is an absolute constant $C > 0$, such that for all sufficiently large n, $\mathsf{r}(\mathsf{MAJ}_n, \delta) \geq \Omega(n^{1/2})$ for any $\delta \leq C/\sqrt{n}$.*

The proofs of Lemma 31 and Lemma 32 can be found in the full version of this work.

References

1. Noga Alon, Eldar Fischer, Michael Krivelevich, and Mario Szegedy. Efficient testing of large graphs. *Combinatorica*, 20, May 2001. doi:10.1007/s004930070001.
2. Shalev Ben David. *Quantum speedups in query complexity.* PhD thesis, Massachusetts Institute of Technology, 2017.
3. Eli Ben-Sasson and Swastik Kopparty. Affine dispersers from subspace polynomials. *SIAM J. Comput.*, 41(4):880–914, 2012.
4. Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002. doi:10.1016/S0304-3975(01)00144-X.
5. Eshan Chattopadhyay, Jesse Goodman, and Jyun-Jie Liao. Affine extractors for almost logarithmic entropy. In *FOCS*, pages 622–633. IEEE, 2021.

6 Gil Cohen and Avishay Tal. Two structural results for low degree polynomials and applications. In *APPROX-RANDOM*, volume 40 of *LIPIcs*, pages 680–709. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015.

7 Alan M. Frieze and Ravi Kannan. The regularity lemma and approximation schemes for dense problems. In *FOCS*, pages 12–20. IEEE Computer Society, 1996.

8 Uma Girish, Justin Holmgren, Kunal Mittal, Ran Raz, and Wei Zhan. Parallel repetition for the GHZ game: A simpler proof. In *APPROX-RANDOM*, volume 207 of *LIPIcs*, pages 62:1–62:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

9 Ben Green. A szemerédi-type regularity lemma in abelian groups. *Geometric and Functional Analysis*, 15:340–376, January 2005. doi:10.1007/s00039-005-0509-8.

10 A. W. Hales and R. I. Jewett. Regularity and positional games. *Transactions of the American Mathematical Society*, 106(2):222–229, 1963.

11 Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. URL: http://www.jstor.org/stable/2282952.

12 Kaave Hosseini, Shachar Lovett, Guy Moshkovitz, and Asaf Shapira. An improved lower bound for arithmetic regularity. *Mathematical Proceedings of the Cambridge Philosophical Society*, 161(2):193–197, 2016. doi:10.1017/S030500411600013X.

13 Zander Kelley and Raghu Meka. Strong bounds for 3-progressions, 2023. arXiv:2302.05537.

14 Xin Li. Improved two-source extractors, and affine extractors for polylogarithmic entropy. In *FOCS*, pages 168–177. IEEE Computer Society, 2016.

15 Roy Meshulam. On subsets of finite abelian groups with no 3-term arithmetic progressions. *J. Comb. Theory Ser. A*, 71(1):168–172, July 1995. doi:10.1016/0097-3165(95)90024-1.

16 Gatis Midrijanis. Exact quantum query complexity for total boolean functions, 2004. doi:10.48550/arXiv.quant-ph/0403168.

17 Ryan O'Donnell. Analysis of boolean functions. *CoRR*, abs/2105.10386, 2021. arXiv:2105.10386.

18 Ryan O'Donnell, John Wright, Yu Zhao, Xiaorui Sun, and Li-Yang Tan. A composition theorem for parity kill number. In *Computational Complexity Conference*, pages 144–154. IEEE Computer Society, 2014.

19 K. F. Roth. On certain sets of integers. *Journal of the London Mathematical Society*, s1-28(1):104–109, 1953. doi:10.1112/jlms/s1-28.1.104.

20 I. Ruzsa and E. Szemerédi. Triple systems with no six points carrying three triangles. *Combinatorica*, 18, January 1976.

21 Ronen Shaltiel. Dispersers for affine sources with sub-polynomial entropy. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 247–256, 2011. doi:10.1109/FOCS.2011.37.

22 Asaf Shapira. *Graph Property Testing and Related Problems*. University of Tel-Aviv, 2006.

23 Endre Szemerédi. Regular partitions of graphs. Technical report, Stanford Univ Calif Dept of Computer Science, 1975.

24 Avishay Tal. Properties and applications of boolean function composition. In *ITCS*, pages 441–454. ACM, 2013.

A Section 1 Omissions

A.1 Omitted Proofs

In this section we provide the proofs of Proposition 4 and Proposition 5. We first begin with a corollary of Proposition 13 which will be useful in the analysis of the claims.

▶ **Corollary 33.** *If $\mathcal{V}$ has dimension $n-1$ and $\mathcal{V}^{\perp} = \mathsf{span}(\{\gamma\})$, we have that $\widehat{f_{\alpha+\mathcal{V}}}(0) = \widehat{f}(0) \pm \widehat{f}(\gamma)$. Moreover, there exists a choice of α such that $|\widehat{f_{\alpha+\mathcal{V}}}(0)| = |\widehat{f}(0)| + |\widehat{f}(\gamma)|$.*

Proof. By Proposition 13, we have

$$\widehat{f_{\alpha+\mathcal{V}}}(0) = (-1)^{\langle 0,\alpha\rangle}\widehat{f}(0) + (-1)^{\langle \gamma,\alpha\rangle}\widehat{f}(\gamma) = \widehat{f}(0) + (-1)^{\langle \gamma,\alpha\rangle}\widehat{f}(\gamma).$$

This immediately implies both parts of the corollary. ◀

Proof of Proposition 4. Given some $f : \mathbb{F}_2^n \to [-1,1]$, consider the following simple procedure:

- While at least δ fraction of $\pi \in \Pi$ have some γ_π such that $|\widehat{f_\pi}(\gamma_\pi)| > \delta$, further partition each π into $\pi \cap \{x : \langle \gamma_\pi, x\rangle = 0\}$ and $\pi \cap \{x : \langle \gamma_\pi, x\rangle = 1\}$.

We would like to show that we cannot perform the above partitioning action more that $\frac{1}{\delta^3}$ times. Towards this end, define the potential function $\Phi(\Pi) := \mathbf{E}_{\pi\in\Pi}\widehat{f_\pi}(0)^2 = \mathbf{E}_{\pi\in\Pi}[(\mathbf{E}f_\pi)^2] \in [0,1]$. Whenever we partition further, by Corollary 33 each $|\widehat{f_\pi}(0)|$ is updated to either $|\widehat{f_\pi}(0) + \widehat{f_\pi}(\gamma_\pi)|$ or $|\widehat{f_\pi}(0) - \widehat{f_\pi}(\gamma_\pi)|$. Therefore, the contribution of π to Φ in one step of the partitioning process is

$$\frac{1}{2}\left((\widehat{f_\pi}(0) + \widehat{f_\pi}(\gamma_\pi))^2 + (\widehat{f_\pi}(0) - \widehat{f_\pi}(\gamma_\pi))^2\right) - \widehat{f_\pi}(0)^2 = \widehat{f_\pi}(\gamma_\pi)^2.$$

Since we assume at least δ fraction of $\pi \in \Pi$ had some γ_π such that $|\widehat{f_\pi}(\gamma_\pi)| > \delta$, at each step of the refinement Φ must increase by at least δ^3, completing the proof. ◀

Proof of Proposition 5. Suppose without loss of generality, $\mathbf{E}f \geq 0$. Start with the trivial subspace, $\pi_0 = \mathbb{F}_2^n$. While there exists γ such that $|\widehat{f_{\pi_t}}(\gamma)| > \delta$, by Corollary 33 we can fix the parity corresponding to γ in such a way that ensures that $|\widehat{f_{\pi_{t+1}}}(0)| = \left|\widehat{f_{\pi_t}}(0) + |\widehat{f_{\pi_t}}(\gamma)|\right| > \widehat{f_{\pi_t}}(\gamma) + \delta$. Since $\widehat{f_\pi}(0) \leq 1$ for all π, this process can happen at most $\frac{1}{\delta}$ times. ◀

A.2 Omitted Sketches

We give the main ideas behind the lower bounds in Table 1.

Sketch of Lemma 26. The proof of this claim is based on the homogeneous degree-one function $f(x) = \frac{1}{n}\sum_i (-1)^{x_i}$. Its key idea comes from Observation 25, which we use to show that if the dimension of $\operatorname{codim}(\mathcal{V}) < n/2$, then at least one shift of $\mathcal{V}^\perp$ must contain *exactly* one standard basis vector. By the preceding discussion, this implies that $f_{\alpha+\mathcal{V}}$ has a non-trivial Fourier coefficient with magnitude exactly $1/n > \delta$.

We remark that Lemma 26 is tight. The function f is symmetric, and for any such function, we can fix $n/2$ parities to obtain an affine subspace where every vector has weight $n/2$, which in turn fixes the function.

Sketch of Lemma 27 and Corollary 30. To achieve Lemma 27, one might expect to extend the above argument to the homogeneous degree d function $f(x) = \binom{n}{d}^{-1}\sum_{\gamma:\|\gamma\|_1=d}(-1)^{\langle\gamma,x\rangle}$. Unfortunately, this function is symmetric, and we have $r(f,0) \leq n/2$. We therefore consider a random homogeneous degree d function $f_{\mathbf{z}}(x) = \binom{n}{d}^{-1}\sum_{\gamma:\|\gamma\|_1=d}\mathbf{z}_\gamma\cdot(-1)^{\langle\gamma,x\rangle}$, where each $\mathbf{z}_\gamma$ is a random sign. A simple argument, again utilizing Observation 25, shows that there must be at least $\binom{k}{d}$ affine subspaces of $\mathcal{V}^\perp$ with at least one vector of weight d. By our earlier reasoning, each of those subspaces must in fact contain at least two vectors of weight d so that the restricted function would have a non-trivial Fourier coefficient with magnitude $\binom{n}{d}^{-1} > \delta$. Moreover, the probability (over the signs $\mathbf{z}_\gamma$'s) that each of the $\binom{k}{d}$ signed sums cancels is at most $2^{-\binom{k}{d}}$, and a union bound over all the possible affine subspaces of dimension $k = \Theta(dn^{1/(d-1)})$ completes the argument.

If we restrict our attention to Boolean functions, we might hope to obtain strong upper bounds for $\mathsf{r}(f,\delta)$; however, Corollary 30 rules this out. The proof of this claim is based on a simple lemma of [12] (Proposition 28), which uses the probabilistic method to convert a bounded function that is not δ-regular in large affine subspaces to a Boolean function with the same property. Applying this lemma to the lower bound from Lemma 27 achieves the result.

Sketch of Lemma 32. This lower bound is based on the majority function. Its key idea is that there exists a non-trivial affine subspace of $\mathcal{V}^{\perp}$ containing *exactly* one weight-1 vector and relatively few vectors of higher weight (see the full version of the work for details on this). Then, we use properties of the Fourier spectrum of the majority function to show that the signed sum of the Fourier coefficients of majority corresponding to vectors in this affine subspace, is on the order of $\left|\widehat{f}(e_1)\right| = \Omega(n^{-1/2})$. Specifically, we argue that even if the coefficients coming from higher weight vectors in the aforementioned sum combined in the most constructive way possible, they cannot combine to more than $\left|\widehat{f}(e_1)\right|/2$. We also note that Lemma 32 is tight up to constant factors via Proposition 5. Conversely, Lemma 32 implies that for $\delta \geq n^{-1/2}$, the majority function on $O(1/\delta^2)$ variables is an explicit Boolean function for which $\mathsf{r}(f,\delta) \geq \Omega(1/\delta)$.

Rationale for Lemma 31. The last entry in the table corresponds to Lemma 31 and is based on a simple function f on 4 inputs that is composed with itself k times. We use key properties of the composition of Boolean functions (from [24, 18]) to achieve the bound. The function itself is the same one considered in [18], and we use their main theorem crucially to obtain our lower bound. We present a slightly generalized version of the main theorem of [18], so we include a proof this in the full version of this work.

We make some final comments about the lower bounds from Corollary 30. The Boolean functions that achieve the lower bounds share the property that the magnitudes of their Fourier coefficients are extremely close to their bounded counterparts in Lemma 27. However, even though the bounded functions themselves have low degree, the Boolean functions are very far from being low-degree functions; in fact, *almost all* their Fourier mass comes from the high-degree terms. Notably, these functions are also non-explicit affine dispersers with small dimension, and it would be interesting to find explicit Boolean functions with similar strong lower bounds on the $\mathsf{r}(f,\delta)$.

B Section 2 Omitted Proofs

Proof of Proposition 13. Using Fact 12, we can write

$$\begin{aligned}\widehat{f_{\mathcal{U}}}(\gamma) = \mathbf{E}_{x\in\mathcal{V}}[f(x+\alpha)\cdot(-1)^{\langle\gamma,x\rangle}] &= \mathbf{E}_{x\in\mathcal{V}}\sum_{\beta}\widehat{f}(\beta)(-1)^{\langle\beta,x+\alpha\rangle}(-1)^{\langle\gamma,x\rangle}\\ &= \sum_{\beta}\widehat{f}(\beta)(-1)^{\langle\beta,\alpha\rangle}\mathbf{E}_{x\in\mathcal{V}}[(-1)^{\langle\beta+\gamma,x\rangle}]\\ &= \sum_{\beta\in\gamma+\mathcal{V}^{\perp}}\widehat{f}(\beta)(-1)^{\langle\beta,\alpha\rangle},\end{aligned}$$

where the last equality follows by observing that $\mathbf{E}_{x\in\mathcal{V}}\left[(-1)^{\langle\gamma+\beta,x\rangle}\right] = 1$ if $\beta\in\gamma+\mathcal{V}^{\perp}$, and zero otherwise. ◀

Proof of Proposition 15. We first show that $\mathcal{W}$ and $\mathcal{V}^\perp$ are independent. Suppose that $M^\mathsf{T}\gamma + u = 0$, where $\gamma \in \mathsf{span}(J)$ and $u \in \mathcal{V}^\perp$. For any $v \in \mathcal{V}$ such that $v \neq 0$, we have

$$0 = \langle v, M^\mathsf{T}\gamma + u\rangle = \langle v, M^\mathsf{T}\gamma\rangle = \langle Mv, \gamma\rangle,$$

which is impossible unless $\gamma = 0$ since this implies $Mv \in \mathsf{span}(J)^\perp = \mathsf{span}(\overline{J})$ and $Mv \neq 0$. This in turn implies that $u = 0$ and therefore that $\mathcal{W}$ and $\mathcal{V}^\perp$ are independent. The claim follows by noting that $\dim(\mathcal{W} \oplus \mathcal{V}^\perp) = \dim(\mathcal{W}) + \dim(\mathcal{V}^\perp) = k + n - k = n$. ◀

Proof of Proposition 16. Repeatedly using Fact 12, we have that

$$\begin{aligned}\left|\widehat{f_{\mathcal{U}}}(M^\mathsf{T}\gamma)\right| = \left|\mathbf{E}_{x\in\mathcal{U}}\left[f(x)(-1)^{\langle M^\mathsf{T}\gamma, x\rangle}\right]\right| &= \left|\mathbf{E}_{x\in\mathcal{U}}\left[f(x)(-1)^{\langle \gamma, Mx\rangle}\right]\right| \\ &= \left|\mathbf{E}_{z\in\mathcal{U}'}\left[f(M^{-1}z)(-1)^{\langle \gamma, z\rangle}\right]\right| = \left|\widehat{g_{\mathcal{U}'}}(\gamma)\right|.\end{aligned}$$ ◀

Proof of Fact 18. We have that

$$\begin{aligned}\widehat{g}(\gamma) = \mathbf{E}_x[g(x)\chi_\gamma(x)] &= \mathbf{E}[f(Mx)\chi_\gamma(x)] \\ &= \mathbf{E}_y[f(y)\chi_\gamma(M^{-1}y)] \\ &= \mathbf{E}_y[f(y)\chi_{M^{-\mathsf{T}}\gamma}(y)] = \widehat{f}(M^{-\mathsf{T}}\gamma),\end{aligned}$$

where we have used the fact that $\chi_\gamma(M^{-1}y) = (-1)^{\langle \gamma, M^{-1}y\rangle} = (-1)^{\langle M^{-\mathsf{T}}\gamma, y\rangle}$. ◀

C Proofs from Section 5

C.1 Proofs from Section 5.1

Proof of Observation 25. Let $S = \{u : u \in \mathcal{W} \text{ and } |(u+\mathcal{V}^\perp)^{=1}| \geq 1\}$. Since every standard basis vector can be expressed as $u+v$ for some $u \in S$ and $v \in \mathcal{V}^\perp$, we have that $\dim(\mathsf{span}(S \cup \mathcal{V}^\perp)) = n$. However, we also know that $\dim(\mathsf{span}(S \cup \mathcal{V}^\perp)) \leq |S| + C$, and rearranging we get $|S| \geq n - C$. Next, let $S_1 = \{u \in S : |(u + \mathcal{V}^\perp)^{=1}| = 1\}$. By Fact 9, for any $u, u' \in S$, we have $u + \mathcal{V}^\perp \neq u' + \mathcal{V}^\perp$. Therefore,

$$n = \sum_{u\in S} |(u+\mathcal{V}^\perp)^{=1}| = \sum_{u\in S_1} |(u+\mathcal{V}^\perp)^{=1}| + \sum_{u\in S\setminus S_1} |(u+\mathcal{V}^\perp)^{=1}| \geq |S_1| + 2(|S| - |S_1|),$$

and rearranging, we get $|S_1| \geq 2|S| - n \geq n - 2C$. ◀

Fully Dynamic Shortest Paths and Reachability in Sparse Digraphs

Adam Karczmarz
University of Warsaw, Poland
IDEAS NCBR, Warsaw, Poland

Piotr Sankowski
University of Warsaw, Poland
IDEAS NCBR, Warsaw, Poland

Abstract

We study the exact fully dynamic shortest paths problem. For real-weighted directed graphs, we show a deterministic fully dynamic data structure with $\widetilde{O}(mn^{4/5})$ worst-case update time processing arbitrary s,t-distance queries in $\widetilde{O}(n^{4/5})$ time. This constitutes the first non-trivial update/query tradeoff for this problem in the regime of *sparse weighted* directed graphs.

Moreover, we give a Monte Carlo randomized fully dynamic *reachability* data structure processing single-edge updates in $\widetilde{O}(n\sqrt{m})$ worst-case time and queries in $O(\sqrt{m})$ time. For sparse digraphs, such a tradeoff has only been previously described with amortized update time [Roditty and Zwick, SIAM J. Comp. 2008].

2012 ACM Subject Classification Theory of computation → Dynamic graph algorithms; Theory of computation → Shortest paths

Keywords and phrases dynamic shortest paths, dynamic reachability, dynamic transitive closure

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.84

Category Track A: Algorithms, Complexity and Games

Funding *Adam Karczmarz*: Partially supported by the ERC CoG grant TUgbOAT no 772346.
Piotr Sankowski: Partially supported by the ERC CoG grant TUgbOAT no 772346 and National Science Center (NCN) grant no. 2020/37/B/ST6/04179.

1 Introduction

Computing all-pairs shortest paths (APSP) is among the most fundamental algorithmic problems on directed graphs. This classical problem is often generalized into a data structure "oracle" variant: given a graph G, preprocess G so that efficient point-to-point distance or shortest paths queries are supported. Computing APSP can be viewed as an extreme solution to the oracle variant; if one precomputes the answers to all the n^2 possible queries in $\widetilde{O}(nm)$ time, the queries can be answered in constant time. The other extreme solution is to not preprocess G at all and run near-linear-time Dijkstra's algorithm upon each query. Interestingly, for general directed weighted graphs, no other tradeoffs for the exact oracle variant of static APSP beyond these trivial ones are known.

In this paper, we consider the exact APSP problem, and its easier relative *all-pairs reachability* (or, in other words, *transitive closure*), in the *fully dynamic* setting, where the input graph G evolves by both edge insertions and deletions.

1.1 Prior work

There has been extensive previous work on APSP and transitive closure in the fully dynamic setting. Notably, Demetrescu and Italiano [16] showed that APSP in a real-weighted digraph can be maintained deterministically in $\widetilde{O}(n^2)$ amortized time per vertex update

(changing all edges incident to a single vertex). Thorup [36] later slightly improved and simplified their result. These data structures maintain an explicit distance matrix and the corresponding collection of shortest paths, and thus allow querying distances and shortest paths in optimal time. Similar amortized bounds have been earlier obtained for transitive closure [17, 30, 32] albeit using different combinatorial techniques. Polynomially worse (but nevertheless subcubic) *worst-case* update bounds for real-weighted fully dynamic APSP are also known: randomized $\widetilde{O}(n^{2+2/3})$ [2, 24] and slightly worse deterministic $\widetilde{O}(n^{2+41/61})$ [13].

For dense *unweighted* digraphs, non-trivial fully dynamic data structures for all-pairs reachability and APSP can be obtained using algebraic techniques. Via a reduction to dynamic matrix inverse, Sankowski [35] obtained $O(n^2)$ *worst-case* update bound for explicitly maintaining the transitive closure, and also gave update/query tradeoffs. In particular, he showed a reachability data structure with subquadratic $O(n^{1.529})$ update time and sublinear $O(n^{0.529})$ query time. Using the same general algebraic framework, van den Brand, Nanongkai, and Saranurak [40] showed $O(n^{1.407})$ worst-case update bound for *st*-reachability (that is, fixed single-pair reachability), whereas van den Brand, Forster, and Nazari [38] gave an $O(n^{1.704})$ worst-case update bound for maintaining exact *st*-distance in unweighted digraphs.[1] That framework, however, inherently leads to Monte Carlo randomized solutions and does not generally allow reporting (shortest) paths within the stated query bounds.[2]

Interestingly, neither the known fully dynamic APSP data structures for *real-weighted* digraphs (or even for integer weights between 1 and n) nor the algebraic data structures tailored to dense graphs yield any improvement over the extreme recompute-from-scratch approaches for *sparse* graphs with $m = \widetilde{O}(n)$. This is especially unfortunate as such graphs are ubiquitous in real-world applications. Indeed, for $m = \widetilde{O}(n)$, recomputing APSP from scratch takes $\widetilde{O}(n^2)$ worst-case update time and $O(1)$ query time (which matches the amortized bound in [17, 36]), whereas naively running Dijkstra's algorithm upon query costs $\widetilde{O}(n)$ time (which already improves upon the update bound of the algebraic *st*-distance data structure of [38]). The only non-trivial fully dynamic APSP data structure in the sparse regime has been described by Roditty and Zwick [34]. Their randomized data structure has $\widetilde{O}(m\sqrt{n})$ amortized update time and $O(n^{3/4})$ query time. Unfortunately, it works only for *unweighted digraphs*. To the best of our knowledge, no non-trivial update/query tradeoffs for fully dynamic APSP in sparse weighted digraphs have been described to date. A step towards this direction has been made by Karczmarz [27] who showed that some *fixed* – in a crucial way – m distance pairs can be maintained in $\widetilde{O}(mn^{2/3})$ worst-case time per update.

For the simpler fully dynamic reachability problem, the $O(n^{1.529})$ update time and $O(n^{0.529})$ query time algebraic tradeoff of [35] is already non-trivial for all graph densities. However, specifically for sparse graphs, a deterministic and combinatorial tradeoff of Roditty and Zwick [33] is more efficient; they showed a data structure with $O(m\sqrt{n})$ amortized update time and $O(\sqrt{n})$ query time. Moreover, the data structure of [35] requires fast matrix multiplication algorithms [3, 21] and these are considered impractical. That being said, the downside of [33] is that the update bound holds only in the amortized sense.

[1] The single-pair data structures [40, 38] can be easily extended to support arbitrary-pair queries. Then, the query time matches the update time.

[2] As shown quite recently, reporting (shortest) paths in subquadratic time can be possible via a combination of algebraic and combinatorial techniques [8, 28]. However, this comes with a polynomial time overhead.

1.2 Our results

Dynamic shortest paths. Most importantly, we show the first fully dynamic APSP data structure with non-trivial update and query bounds for *sparse weighted* digraphs.

▶ **Theorem 1.** *Let G be a real-weighted directed graph. There exists a deterministic data structure maintaining G under fully dynamic vertex updates and answering arbitrary s,t-distance queries with $\widetilde{O}(mn^{4/5})$ worst-case update time and $\widetilde{O}(n^{4/5})$ query time and using $\widetilde{O}(n^2)$ space. The queries are supported only when G has no negative cycles. After answering a distance query, some corresponding shortest path $P = s \to t$ can be reported in $O(|P|)$ time.*

Compared to the data structure of Roditty and Zwick [34] for the unweighted case, our obtained update/query bounds are polynomially higher. However, our data structure has some very significant advantages. It is deterministic, handles real-edge-weighted graphs (possibly with negative edge weights and negative cycles), and the update time bounds holds in the worst case, as opposed to only in the amortized sense in [34]. Moreover, if path reporting is required, then the bounds in [34] hold only against an oblivious adversary. We also remark that a slightly more efficient variant of Theorem 1, with $\widetilde{O}(mn^{3/4})$ worst-case update time and $\widetilde{O}(n^{3/4})$ query time, can be obtained for the unweighted case.

The near-quadratic space requirement in Theorem 1 is clearly undesirable in the sparse setting, but also applies to all the other known fully dynamic reachability and shortest paths data structures. Moreover, this phenomenon is not specific to the dynamic setting. To the best of our knowledge, even for the *static* transitive closure problem, it is not known whether one can preprocess a general sparse directed graph into a data structure of size $O(n^{2-\epsilon})$ supporting arbitrary reachability queries in $O(n^{1-\epsilon})$ time.[3]

Dynamic reachability. For fully dynamic all-pairs reachability in sparse digraphs, we show that the amortized update bound of Roditty and Zwick [33] can also hold in the worst case.

▶ **Theorem 2.** *Let G be a directed graph. Let $t \in [1, \sqrt{m}]$. There exist a Monte Carlo randomized data structure maintaining G subject to fully dynamic single-edge updates with $\widetilde{O}(mn/t)$ worst-case update time and supporting arbitrary-pair reachability queries in $O(t)$ time. The answers produced are correct with high probability[4].*

Note that for $t = \sqrt{m}$, Theorem 2 yields $O(n^{2-\epsilon})$ update time and $O(n^{1-\epsilon})$ for some $\epsilon > 0$ for all but dense graphs. The data structure of Roditty and Zwick [33], on the other hand, has amortized update time at least $\Theta(m\sqrt{n})$, which is $o(n^2)$ only if $m = o(n^{3/2})$. However, the downsides of Theorem 2 compared to [33] are: supporting more restricted single edge (as opposed to vertex-) updates, using randomization, and not being able to report the underlying path efficiently.

Our data structure should also be compared with the $O(n^{1.529})/O(n^{0.529})$ worst-case update/query bounds obtained in [35]. Theorem 2 gives polynomially better bounds for very sparse graphs, with $m = O(n^{1.057})$. Moreover, although it is also algebraic in nature, it does not rely on fast matrix multiplication [3, 21], thus avoiding this potential practical efficiency bottleneck.

[3] Such a tradeoff is possible, for example, if the graph has a sublinear *minimum path cover*, see, e.g., [31].
[4] That is, with probability at least $1 - 1/n^c$, where the constant $c \geq 1$ can be set arbitrarily. We will also use the standard abbreviation w.h.p.

1.3 Technical overview

Shortest paths. In order to obtain a basic randomized variant of Theorem 1, we combine ideas from the known data structures for fully dynamic APSP with subcubic worst-case update bound [2, 24, 27]. These data structures all build upon hitting set arguments (dating back to the work of Ullman and Yannakakis [37]) yielding a sublinear $\widetilde{O}(n/h)$-sized set of vertices of the graph that lie on the shortest paths whose number of edges (*hops*) is at least $h = \operatorname{poly} n$. With this in hand, the main challenge is to recompute pairwise *small-hop* shortest paths, i.e., those with at most h hops, under edge deletions. As usual, edge insertions are rather easy to handle since the potential new paths created by insertions necessarily pass through the inserted edges' endpoints.

For efficient recomputation of small-hop paths, our data structure once in a while chooses a collection Π of n^2 pairwise $\leq h$-hop paths in G, and a set $C \subseteq V$ of *congested vertices* of truly sublinear (in n) size, so that the chosen paths are at least as short as shortest $\leq h$-hop paths in $G - C$ (i.e., the graph G with edges incident to the vertices C removed). The congested vertices are picked in such a way that no individual vertex $v \in V$ appears on the chosen paths too often. As a result, the number of precomputed paths destroyed by a vertex deletion that have to be restored is bounded. This idea is due to Probst Gutenberg and Wulff-Nilsen [24]. However, as opposed to [24], we cannot afford to recompute shortest $\leq h$-hop paths upon update in a hierarchical way which is inherently quadratic in n (albeit advantageous in the case of dense graphs). Instead, recomputation upon deletions is performed using a Dijkstra-like procedure (as in [2]), crucially with the sparsity-aware enhancements of [27] (such as the degree-weighted congestion scheme). These techniques, combined with the standard random hitting set argument [37] are enough to get the stated bounds, albeit Monte Carlo randomized.

Derandomization. Randomization above is only required for the sake of the hitting set argument. Curiously, we do not (and do not know how to) exploit the often-used property that a random hitting set, once sampled, is valid through multiple versions of the evolving graph as long as the adversary is oblivious to the hitting set. Therefore, we may as well sample the hitting set from scratch after each update. This is as opposed to [2, 27], where avoiding that leads to polynomially better bounds. If a fresh hitting set can be used upon each update, the standard derandomization method is to use a folklore greedy algorithm (see Lemma 8) for constructing a minimum hitting set that is $O(\log n)$-approximate, first used in the context of static and dynamic APSP algorithms in [30, 42]. The greedy algorithm runs in linear time in the input size. For constructing a hitting set of explicitly given pairwise $\leq h$-hop paths, this gives an $O(n^2 h)$ time bound per update. This is enough for deterministic variants of [2] and [42]. However, the incurred cost is prohibitive in the sparse case.

Derandomization of our data structure without a polynomial slowdown turns out to be non-trivial and requires some new tools. First, when precomputing $\leq h$-hop paths Π, we construct a hitting set H_0 of those paths in Π that have $\Theta(h)$ hops. When G is subject to deletions, H_0 hits the precomputed paths in Π that are not destroyed as a result of deletions. Hence, in order to lift H_0 into a hitting set after an update, it is enough to extend it so that it hits all the restored paths. If we wanted to run the greedy algorithm on the restored paths, the data structure would suffer from a factor-h polynomial slowdown. This is because the representation of the restored paths (constructed using Dijkstra's algorithm) can be computed more efficiently that their total hop-length and encoded using a collection of shortest paths trees $\mathcal{Z}$. The goal can be thus achieved by finding a hitting set of all $\Theta(h)$-hop root-leaf paths in $\mathcal{Z}$. King [30] gave a variant of the aforementioned deterministic greedy

algorithm precisely for this task. The algorithm of [30] runs in $O(\min(Nh, |\mathcal{Z}|n))$ time, where N denotes the total size of trees in $\mathcal{Z}$. While this is optimal when $\mathcal{Z}$ contains $\Theta(n)$ trees of size $\Theta(n)$ (as required in [30]), for small enough N and large enough $|\mathcal{Z}|$, this is not better than the standard greedy algorithm which could also solve the task in $O(Nh)$ time.

We deal with this problem by designing a novel near-optimal deterministic algorithm computing an $\widetilde{O}(n/h)$-sized hitting set of h-hop root-leaf path in a collection of trees that runs in $O(N \log^2 N)$ time independent of h (see Theorem 9). We believe that this algorithm might be of independent interest. The main idea here is to simulate the greedy algorithm only approximately, which enables taking advantage of dynamic tree data structures [4].

Reachability and sparse matrix inverse. Our improved worst-case bounds for fully dynamic reachability in sparse digraphs are obtained via a small change in the subquadratic update-sublinear query tradeoff of [35] based on dynamic matrix inverse. That algorithm once in a while explicitly recomputes the inverse of a certain matrix associated with the graph using fast rectangular matrix multiplication. That inverse encodes the transitive closure of the graph G. We observe that for sparse graphs, it is beneficial to recompute the inverse in a more naive way, entirely from scratch. This is because for large enough finite fields (with more than n^2 elements), it is in fact possible to compute the inverse of a sparse matrix with $m = \widetilde{\Theta}(n)$ non-zero elements in near-optimal $\widetilde{O}(mn)$ time without fast Strassen-style matrix multiplication algorithms (see Theorem 13). This is a relatively easy consequence of the classical work of Kaltofen and Pan [26], and has been, to the best of our knowledge, overlooked and not explicitly stated before. Sparse matrix inversion has been recently viewed (see, e.g., [12, 18]) mainly through the lens of *black-box matrix computations*, i.e., parameterized by the cost $\phi(n)$ of multiplying the input matrix (or its transpose) by a vector. For sparse matrices, we clearly have $\phi(n) = \widetilde{O}(n)$, but the best described bound for sparse matrix inversion in finite fields in that literature seems to be $O(n^{2.214})$ [12]. However, $\phi(n) = \widetilde{O}(n)$ holds for sparse matrices even in a less general so-called *straight-line program* computation model (also called the *algebraic circuit* model) which allows employing powerful tools such as the Baur-Strassen theorem [7].

1.4 Further related work

Exact all-pairs shortest paths in unweighted graphs have been studied also in partially dynamic settings: incremental [5] and decremental [6, 19]. Fully dynamic data structures are also known for $(1+\epsilon)$-approximate distances in weighted directed graphs [9, 39]. A significant research effort has been devoted to finding fully- and partially dynamic (approximate) all-pairs shortest paths data structures for *undirected* graphs, e.g., [10, 14, 15, 20, 38].

Dynamic reachability and shortest paths problems have also been studied from the perspective of conditional lower bounds [1, 23, 25, 34, 40].

2 Preliminaries

We work with directed graphs $G = (V, E)$. We denote by $w_G(e) \in \mathbb{R}$ the weight of an edge $uv = e \in E$. The graph G is called *unweighted* if $w_G(e) = 1$ for all $e \in E$. If the graph whose edge we refer to is clear from the context, we may sometimes skip the subscript and write $w(e)$. For simplicity, we do not allow parallel directed edges between the same endpoints of G, as those with non-minimum weights can be effectively ignored in reachability and shortest paths problems we study. As a result, we sometimes write $w_G(uv)$ or $w(uv)$.

For $u, v \in V$, an $u \to v$ path P in G is formally a sequence of vertices $v_1 \dots v_k \in V$, where $k \geq 1$, $u = v_1$, $v = v_k$, such that $v_i v_{i+1} \in E$ for all $i = 1, \dots, k-1$. The hop-length $|P|$ of P equals $k-1$. The length $\ell(P)$ of P is defined as $\sum_{i=1}^{k-1} w_G(v_i v_{i+1})$. P is a *simple path* if $|V(P)| = |E(P)| + 1$. We sometimes view P as a subgraph of G with vertices $\{v_1, \dots, v_k\}$ and edges (hops) $\{v_1 v_2, \dots, v_{k-1} v_k\}$.

For any $k \geq 0$, $\delta_G^k(s,t)$ is the minimum length of an $s \to t$ path in G with at most k hops. A *shortest k-hop-bounded $s \to t$ path* in G is an $s \to t$ path with length $\delta_G^k(s,t)$ and at most k hops. We define the s,t-distance $\delta_G(s,t)$ as $\inf_{k \geq 0} \delta_G^k(s,t)$. For $s, t \in V$, we say that t is *reachable* from s in G if there exists an $s \to t$ path in G, that is, $\delta_G(s,t) < \infty$. If $\delta_G(s,t)$ is finite, there exists a simple $s \to t$ path of length $\delta_G(s,t)$. Then, we call any $s \to t$ path of length $\delta_G(s,t)$ a *shortest s,t-path*.

If G contains no negative cycles, then $\delta_G(s,t) = \delta_G^{n-1}(s,t)$ for all $s, t \in V$. Moreover, in such a case there exists a *feasible price function* $p : V \to \mathbb{R}$ such that reduced weight $w_p(e) := w(e) + p(u) - p(v) \geq 0$ for all $uv = e \in E$. For any path $s \to t = P \subseteq G$, the reduced length $\ell_p(P)$ (i.e., length wrt. weights w_p) is non-negative and differs from the original length $\ell(P)$ by the value $p(s) - p(t)$ which does not depend on the shape of P.

For any $S \subseteq V$, we denote by $G - S$ the subgraph of G on V obtained from G by removing all edges incident to vertices S.

We sometimes talk about rooted out-trees T with all edges directed from a parent to a child. In such a tree T with root s, a *root path* $T[s \to t]$ is the unique path from the root to the vertex t of T. A subtree of T rooted in some of its vertices v is denoted by $T[v]$.

3 Fully dynamic shortest paths data structure

This section is devoted to proving the main theorem of this paper.

▶ **Theorem 1.** *Let G be a real-weighted directed graph. There exists a deterministic data structure maintaining G under fully dynamic vertex updates and answering arbitrary s,t-distance queries with $\widetilde{O}(mn^{4/5})$ worst-case update time and $\widetilde{O}(n^{4/5})$ query time and using $\widetilde{O}(n^2)$ space. The queries are supported only when G has no negative cycles. After answering a distance query, some corresponding shortest path $P = s \to t$ can be reported in $O(|P|)$ time.*

First, let us assume that all the edge weights are non-negative. Let us also make a simplifying assumption that any shortest k-hop-bounded $s \to t$ path in G always has a minimum possible number of hops and is simple. If there are no negative cycles, this is easy to guarantee by replacing each edge weight $w(e)$ in G with a pair $(w(e), 1)$, adding weights coordinate-wise, and comparing them lexicographically. We discuss how to extend the data structure to also handle negative edge weights and negative cycles later in Section 3.7.

We will first present a simple Monte Carlo randomized data structure, and show how to make it deterministic with no asymptotic time penalty (wrt. $\widetilde{O}(\cdot)$ notation) in Section 3.5.

Some further variants of the data structure are sketched in the Appendix. A variant for unweighted digraphs is given in Section A.1. In the weighted case, one can also achieve polynomially faster update at the cost of polynomially slower query and randomization. For details, see Section A.2.

The data structure operates in phases of Δ vertex updates. At the beginning of each phase, we apply a rather costly preprocessing described in the next subsection.

3.1 Preprocessing at the beginning of a phase

The preprocessing follows the general approach of [24] adjusted with some ideas from [27].

Let $h \in [2, n]$, and let τ be a *congestion threshold*, to be set later. We compute a certain collection of paths Π in G containing, for every pair $s, t \in V$, at most one $s \to t$ path $\pi_{s,t}$, satisfying $|\pi_{s,t}| = O(h)$, and a subset $C \subseteq V$ of *congested vertices*.

First of all, the collection Π and the set C satisfy:

$$\delta^h_G(s,t) \leq \ell(\pi_{s,t}) \leq \delta^h_{G-C}(s,t), \text{ for all } s,t \in V. \tag{1}$$

Above, we abuse the notation a bit and set $\ell(\pi_{s,t}) := \infty$ if there is no path $\pi_{s,t}$ in Π. Moreover, for any $v \in V$, let us define:

$$\Pi(v) := \{\pi_{s,t} \in \Pi : v \in V(\pi_{s,t})\},$$
$$\alpha(v) := \sum_{\pi_{s,t} \in \Pi(v)} \deg(t).$$

Crucially, Π additionally satisfies:

$$\alpha(v) \leq \tau, \text{ for all } v \in V. \tag{2}$$

▶ **Lemma 3.** *Let $h \in [1, n]$. For any $\tau \geq 2m$, in $O(nmh)$ time one can compute the congested set $C \subseteq V$ and a set of paths Π satisfying conditions (1) and (2) so that $|C| = O(nmh/\tau)$.*

Proof. We start with empty sets C and Π. Note that (2) is satisfied initially since all values $\alpha(\cdot)$ are zero. We will gradually add new paths to Π while maintaining (2) and ensuring that (1) holds for more and more pairs s, t. While introducing new paths to Π, we will also maintain the values $\alpha(v)$ (as defined above) for all $v \in V$.

We process source vertices $s \in V$ one by one, in arbitrary order. For each such s, we first move to C all the vertices $v \in V \setminus C$ with $\alpha(v) > \tau/2$. Next, we compute, for all $t \in V$, a shortest h-hop-bounded path $\pi_{s,t} = s \to t$ in $G - C$ (if such a path exists). For a fixed s, all the paths $\pi_{s,t}$ can be computed in $O(mh)$ time using a variant of Bellman-Ford algorithm. We add the newly computed paths to Π. Afterwards, (1) clearly holds for s and all $t \in V$. Moreover, (1) also holds for all $\pi_{s',t'} \in \Pi$ that have been added for a source s' processed earlier than s. Indeed, extending the set C only weakens the upper bound in (1). The values $\alpha(v)$ can be updated easily in $O(nh)$ time. Observe that for any $v \in V \setminus C$, $\alpha(v)$ grows by at most $\sum_{t \in V} \deg(t) = m$ when processing s. As a result, after processing s, we have $\alpha(v) \leq \tau/2 + m \leq \tau/2 + \tau/2 = \tau$ and hence (2) is satisfied. At the same time, since we use paths from $G - C$, for any $y \in C$, $\alpha(y)$ does not increase and thus we still have $\alpha(y) \leq \tau$.

Finally, note that for any $\pi_{s,t}$ added to Π, since $|\pi_{s,t}| \leq h$, $\alpha(v)$ grows by $\deg(t)$ for at most h distinct vertices v. As a result, we have $\sum_{v \in V} \alpha(v) \leq \sum_{t \in V} \deg(t) \cdot \left(\sum_{s \in V} h\right) \leq m \cdot nh$. But for each $y \in C$, we have $\alpha(y) > \tau/2$, so there is at most $2nmh/\tau$ such vertices y. ◀

Applying Lemma 3 constitutes the only preprocessing that we apply at the beginning of a phase in the Monte Carlo randomized variant. The computed paths Π are stored explicitly and thus the used space might be $\Theta(n^2 h)$. Note that with the help of additional $O\left(\sum_{\pi_{s,t} \in \Pi} |\pi_{s,t}|\right) = O(n^2 h)$ vertex-path pointers, we can report the elements of any $\Pi(v)$, $v \in V$, in constant time per element. We will discuss how to improve the space to $\widetilde{O}(n^2)$ using a trick due to Probst Gutenberg and Wulff-Nilsen [24] in Section 3.6.

3.2 Update

When a phase proceeds, let D be the set of at most Δ *affected* vertices in the current phase, that is, D contains every v such that a vertex update around v has been issued in this phase.

In the query procedure, we will separately consider paths going through $C \cup D$, and those lying entirely in $G - (C \cup D)$. To handle the former, upon each update we simply compute single-source shortest-path trees from and to each $s \in C \cup D$ in the current graph G. This takes $\widetilde{O}(|C \cup D|m)$ worst-case time using Dijkstra's algorithm.

As a matter of fact, we will not quite compute shortest paths in $G - (C \cup D)$, but instead, we will find paths in $G - D$ that are not longer than the distances between their corresponding endpoints in $G - (C \cup D)$. This is acceptable since $G - D \subseteq G$.

To prepare for queries about the paths in $G - (C \cup D)$, we do the following. We will separately handle *short* $\leq h$-hop shortest paths, and *long* $> h$-hop shortest paths.

Short paths. Denote by G_0 the graph at the beginning of the phase. Recall that we use G to refer to the current graph. Clearly, we have $G - D \subseteq G_0$. Fix some $s \in V$. First of all, note that if for some $t \in V$, $V(\pi_{s,t}) \cap D = \emptyset$, then $\pi_{s,t} \subseteq G - D$, so by (1):

$$\delta_{G-D}(s,t) \leq \ell(\pi_{s,t}) \leq \delta^h_{G_0-C}(s,t) \leq \delta^h_{G-(C\cup D)}(s,t).$$

The paths $\pi_{s,t}$ going through D are not preserved in $G - (C \cup D)$ and thus we cannot use them. We replace them with other paths $\pi'_{s,t}$ constructed using the following lemma.

▶ **Lemma 4.** *For $s \in V$, let Q_s contain all t such that $V(\pi_{s,t}) \cap D \neq \emptyset$. In $\widetilde{O}\left(\sum_{t\in Q_s} \deg(t)\right)$ time we can compute a representation of paths $\pi'_{s,t} \subseteq G - D$ (where $t \in Q_s$), each with possibly $\Theta(n)$ hops, satisfying:*

$$\delta_{G-D}(s,t) \leq \ell(\pi'_{s,t}) \leq \delta^h_{G-(C\cup D)}(s,t).$$

The representation is a tree T_s rooted at s such that:

(1) *some edges $sv \in E(T_s)$ represent paths $\pi_{s,v} \subseteq G - D$ from Π and have corresponding weights $\ell(\pi_{s,v})$,*

(2) *all other edges of T_s come from $E(G - D)$,*

(3) *for all $t \in Q_s$, $\pi'_{s,t}$ equals $T_s[s \to t]$ with possibly the first edge sw of that path uncompressed into the corresponding path $\pi_{s,w} \in \Pi$.*

Proof. Let Y be an edge-induced directed graph obtained as follows. For all $t \in Q_s$, and every of at most $\deg(t)$ edges $vt \in E(G - D)$, we add to Y the following:

- the edge vt itself (with the same weight),
- if $V(\pi_{s,v}) \cap D = \emptyset$, an edge sv of weight $\ell(\pi_{s,v})$ corresponding to the path $\pi_{s,v} \in \Pi$.

The algorithm is to simply compute a shortest paths tree T_s from s in Y in $\widetilde{O}(|E(Y)|) = \widetilde{O}\left(\sum_{t\in Q_s} \deg(t)\right)$ time using Dijkstra's algorithm. Clearly, any path $T_s[s \to t]$ corresponds to an $s \to t$ path in $G - D$. It is thus sufficient to prove that for all $t \in Q_s$, we have $\ell(T_s[s \to t]) \leq \delta^h_{G-(C\cup D)}(s,t)$.

If t is unreachable in $G - (C \cup D)$ from s using a path with at most h hops, there is nothing to prove. Otherwise, let a simple path P be a shortest h-hop-bounded $s \to t$ path in $G - (C \cup D)$. Let p be the last vertex on P such that $V(\pi_{s,p}) \cap D = \emptyset$, that is, $p \notin Q_s$. Note that p exists since $\delta^h_{G-C}(s,v) \neq \infty$ for all $v \in V(P)$ (which implies $\pi_{s,v} \in \Pi$) and $p \neq t$. Let P' be the $s \to p$ subpath of P. Let $e_1, \ldots, e_k \in E(G - (C \cup D))$ be the edges following p on P. Here, p is the tail of e_1. By the definition of Y and p, we have $e_i \in E(Y)$ for all $i = 1, \ldots, k$ since the head of each e_i is in Q_s. Moreover, there is an edge sp of weight $\ell(\pi_{s,p})$ in Y. Now, since $\pi_{s,p}$ is a path in $G - D$ of length at most $\delta^h_{G_0-C}(s,p)$, whereas the path $P' \subseteq G - (C \cup D) = G_0 - (C \cup D)$ has less than h hops, we obtain $\ell(\pi_{s,p}) \leq \ell(P')$ and hence:

$$\ell(T_s[s \to t]) = \delta_Y(s,t) \leq \ell(\pi_{s,p}) + \sum_{i=1}^{k} w(e_i) \leq \ell(P') + \sum_{i=1}^{k} w(e_i) = \ell(P) = \delta^h_{G-(C\cup D)}(s,t). \qquad \blacktriangleleft$$

We compute the paths $\pi'_{s,t}$ from Lemma 4 for all $s \in V$, $t \in Q_s$. Recall that $t \in Q_s$ implies that $V(\pi_{s,t}) \cap D \neq \emptyset$ and thus $\pi_{s,t} \in \Pi(d)$ for some $d \in D$. Therefore, the time needed for computing the paths $\pi'_{s,t}$ can be bounded as follows:

$$\widetilde{O}\left(\sum_{s\in V}\sum_{t\in Q_s}\deg(t)\right) = \widetilde{O}\left(\sum_{d\in D}\sum_{\pi_{s,t}\in\Pi(d)}\deg(t)\right) = \widetilde{O}\left(\sum_{d\in D}\alpha(d)\right) = \widetilde{O}(|D|\tau) = \widetilde{O}(\Delta\tau).$$

Note that the sets Q_s can also be constructed within this bound, since they can be read from $\bigcup_{d\in D}\Pi(d)$ which also has size $\widetilde{O}(\Delta\tau)$ and the paths from any $\Pi(v)$ can be reported in $O(1)$ time per path.

For all $s \in V$ and $t \notin Q_s$, let us simply set $\pi'_{s,t} := \pi_{s,t}$ and put $\Pi' = \{\pi'_{s,t} : s,t \in V\}$. To summarize, in $\widetilde{O}(\Delta\tau)$ time we can find, for all $s,t \in V$, a representation of paths $\pi'_{s,t}$ in $G - D$ that are at least as short as the corresponding shortest h-hop-bounded $s \to t$ paths in $G - (C \cup D)$. Storing a representation of the paths $\Pi' \setminus \Pi$ requires $\widetilde{O}(\min(\Delta\tau, n^2))$ additional space since, by the construction of Lemma 4, each of these paths can be encoded using its last edge and a pointer to another path in Π' with less hops.

Long paths. In order to handle long paths, we use the following standard hitting set trick from [37].

▶ **Lemma 5.** *Let $G = (V, E)$ be a directed graph with no negative cycles. For any $s,t \in V$, fix some simple shortest $s \to t$ path $p_{s,t}$ in G. Let $H \subseteq V$ be obtained by sampling, uniformly and independently (also from the choice of paths $p_{s,t}$), $c \cdot (n/h)\log n$ elements of V, where $c \geq 1$ is a constant. Then, with high probability (controlled by the constant c), for all $s,t \in V$, if $|p_{s,t}| \geq h$, then $V(p_{s,t}) \cap H \neq \emptyset$.*

On update, we simply apply Lemma 5 to the graph $G - (C \cup D)$ and an arbitrary choice of pairwise shortest paths therein. This way, with high probability, we obtain an $\widetilde{O}(n/h)$-sized hitting set H of shortest paths in $G - (C \cup D)$ that have at least h hops. Finally, we simply compute shortest paths trees from and to the vertices H in $G - (C \cup D)$ in $\widetilde{O}(|H|m) = \widetilde{O}(mn/h)$ worst-case time using Dijkstra's algorithm.

3.3 Query

To answer a query about s,t distance in the current graph, we simply return:

$$\min\left(\min_{v\in C\cup D}\left\{\delta_G(s,v)+\delta_G(v,t)\right\}, \min_{v\in H}\left\{\delta_{G-(C\cup D)}(s,v)+\delta_{G-(C\cup D)}(v,t)\right\}, \ell(\pi'_{s,t})\right). \quad (3)$$

The first term above is responsible for considering all s,t paths in G going through $C \cup D$. If all shortest s,t paths in G do not pass through $C \cup D$, then the second term captures (with high probability) one of such paths provided that it has at least h hops. Finally, if every shortest s,t path in G does not go through $C \cup D$ and has less than h hops, then $\delta_G(s,t) = \delta_{G-D}(s,t) = \delta^h_{G-(C\cup D)}(s,t)$. Moreover, by Lemma 4, $\pi'_{s,t}$ belongs to $G - D \subseteq G$ and $\delta_G(s,t) = \delta_{G-D}(s,t) \leq \ell(\pi'_{s,t}) \leq \delta^h_{G-(C\cup D)}(s,t) = \delta_G(s,t)$, so indeed $\delta_G(s,t) = \ell(\pi'_{s,t})$.

Finally, note that finding the minimizer in (3) allows for reconstruction of some shortest s,t path P in G in $O(|P|)$ time using the stored data structures.

3.4 Time analysis

The total time spent handling a single update is:

$$\widetilde{O}\left((|D|+|C|+|H|)m+\Delta\tau\right)=\widetilde{O}(m\Delta+nm^2h/\tau+mn/h+\Delta\tau).$$

There is also an $O(mnh)$ preprocessing cost spent every Δ updates which yields an amortized cost of $\widetilde{O}(mnh/\Delta)$ per update. Since $\tau \geq 2m$, the term $m\Delta$ is negligible above.

Balancing the terms mnh/Δ and mn/h yields $\Delta = h^2$. Next, balancing with $\Delta\tau$ yields $\tau = mn/h^3$ under the assumption $h = O(n^{1/3})$. Finally, balancing mn/h and $nm^2h/\tau = mh^4$ yields $h = n^{1/5}$, $\Delta = n^{2/5}$, and $\tau = mn^{2/5}$. For such a choice of parameters, the amortized update time is $\widetilde{O}(mn^{4/5})$. Since the only source of amortization here is a costly preprocessing step happening in a coordinated way every Δ updates, the bounds can be made worst-case using a standard technique, see, e.g., [2, 40].

The query time is $O(|C|+|D|+|H|+1)$. For the obtained parameters, the bound becomes $\widetilde{O}(\Delta+nmh/\tau+n/h)=\widetilde{O}(h^4+n/h)=\widetilde{O}(n^{4/5})$.

▶ Remark 6. In the above analysis, we have silently assumed that the "current" number of edges m does not decrease significantly (say, by more than a constant factor) during a phase due to vertex deletions, so that $m = \Omega(m_0)$ holds at all times, where $m_0 = |E(G_0)|$. Since the preprocessing of Lemma 3 is applied to G_0, for the chosen parameters h, Δ, and $\tau = m_0 n^{2/5}$, the update bound should more precisely be bounded by $\widetilde{O}(\max(m, m_0) \cdot n^{4/5})$. In general, it might happen that m becomes polynomially smaller that m_0 while the phase proceeds, e.g., if $m_0 = O(n\Delta)$. This could make the update bound higher than $\widetilde{O}(mn^{4/5})$.

There is a simple fix to this shortcoming, described in [27]: when a phase starts, it is enough to put aside a set $B \subseteq V$ of Δ vertices with largest degrees in G_0 and preprocess the graph $G_0 - B$ instead. The edges incident to vertices B can be viewed as added during the first Δ "auxiliary" updates in the phase, and effectively included in the affected set D from the beginning of the phase. One can easily prove that this guarantees that $m = \Omega(m_0)$ throughout the phase, where m_0 is now defined as $|E(G_0 - B)|$.

3.5 Derandomization

The only source of randomization so far was sampling a subset of vertices that hits shortest paths in $G - (C \cup D)$ with at least h hops. To derandomize the data structure, we will construct a hitting set H of size $\widetilde{O}(n/h)$ such that H hits all the paths in $\Pi' = \{\pi'_{s,t} : s, t \in V\}$ (constructed during update) with at least h distinct vertices. Recall that the paths Π' have been used to handle short paths so far. We first show that a hitting set H defined this way can serve the same purpose as the randomly sampled hitting set.

▶ **Lemma 7.** *Let $H \subseteq V$ be such that for all $s, t \in V$ satisfying $|V(\pi'_{s,t})| \geq h$, $V(\pi'_{s,t}) \cap H \neq \emptyset$ holds. Let $a, b \in V$ be such that every shortest $a \to b$ path in G has more than h hops and* does not *go through $C \cup D$. Then there exists a shortest $a \to b$ path in G that goes through a vertex of H.*

Proof. Let Q be the shortest $a \to b$ path in G that has the minimum number of hops. By the assumption, $|Q| > h$ and $V(Q) \cap (C \cup D) = \emptyset$. Let $Q = RS$, where $R = a \to c$ is the h-hop prefix of Q. We have $R \subseteq G - (C \cup D)$ and, since Q is a shortest path in G, R is also shortest in G and

$$\ell(R) = \delta_G(a, c) = \delta^h_G(a, c) = \delta^h_{G-(C\cup D)}(a, c).$$

Since $\delta^h_{G-(C\cup D)}(a,c)$ is finite, the path $\pi'_{a,c} \subseteq G - D \subseteq G$ satisfies

$$\delta_G(a,c) \leq \delta_{G-D}(a,c) \leq \ell(\pi'_{a,c}) \leq \delta^h_{G-(C\cup D)}(a,c) = \delta_G(a,c).$$

We conclude that the path $Q' = \pi'_{a,c} \cdot S$ satisfies $\ell(Q') = \ell(Q)$ and thus Q' is also a shortest $a \to b$ path in G. Since G has no negative cycles, one can obtain a *simple* $a \to c$ path $\pi''_{a,c}$ from $\pi'_{a,c}$ by eliminating zero-weight cycles, so that $\ell(\pi''_{a,c}) = \ell(\pi'_{a,c}) = \delta_G(a,c)$ and $V(\pi''_{a,c}) \subseteq V(\pi'_{a,c})$. By the definition of Q, $|V(\pi'_{a,c})| \geq |\pi''_{a,c}| \geq |R| \geq h$, since otherwise Q would not have a minimum number of hops. By the assumption we have $V(\pi'_{a,c}) \cap H \neq \emptyset$, so Q' is a shortest $a \to b$ path in G going through a vertex of H. ◀

Additional preprocessing. When a phase starts, we additionally do the following. Let Π_0 be a set of paths obtained as follows. For all $\pi_{s,t} \in \Pi$, if $|\pi_{s,t}| \geq h/2$, we add $\pi_{s,t}$ to Π_0.

Let us now recall a folklore greedy algorithm (used, e.g., in [42]) for computing a hitting set of a collection of sufficiently large sets over a common ground set, summarized by the following lemma.

▶ **Lemma 8.** *Let X be a ground set of size n and let $\mathcal{Y}$ be a family of subsets of X, each with at least k elements. Then, in $O\left(\sum_{Y\in\mathcal{Y}} |Y|\right)$ time one can deterministically compute a* hitting set *$H \subseteq X$ of size $O(n/k \cdot \log n)$ such that $H \cap Y \neq \emptyset$ for all $Y \in \mathcal{Y}$.*

We skip the proof of Lemma 8 since we later prove a more general result in Theorem 9. Using Lemma 8 we can compute a hitting set $H_0 \subseteq V$ of Π_0 in $O(n^2 h)$ time. H_0 has size $\widetilde{O}(n/h)$.

Computing a hitting set upon update. To compute a hitting set $H \subseteq V \setminus D$ as required by Lemma 7, we perform the following additional steps upon update. Recall that the precomputed set $H_0 \subseteq V$ hits all (simple) paths in $\Pi \cap \Pi'$ with at least $h/2$ hops, and thus also those that have at least h distinct vertices. We will augment H_0 into H so that it also hits all the paths in $\Pi' \setminus \Pi$ with at least h distinct vertices.

Recall from Lemma 4 that for a fixed $s \in V$, all the paths $\pi'_{s,t}$, where $t \in Q_s$, are encoded using a tree T_s. By construction, for each edge e of T_s, we have that the tail of e is s, or the head of e is in Q_s. Consider a subtree $T_s[u]$ rooted at some child u of s in T_s. If the edge su in T_s corresponds to the path $\pi_{s,u}$ with $|\pi_{s,u}| \geq h/2$ then H_0 hits $\pi_{s,u}$. As a result, for all $t \in Q_s \cap V(T_s[u])$, $V(\pi_{s,u}) \subseteq V(\pi'_{s,t})$ and hence if $|V(\pi'_{s,t})| \geq h$ then H_0 hits $V(\pi'_{s,t})$. Otherwise either su is a single edge from $G - D$, or it corresponds to a path $\pi_{s,u}$ with $|\pi_{s,u}| < h/2$. In either of these cases, if some t is at depth less than $h/2 - 1$ in $T_s[u]$, then $|V(\pi'_{s,t})| < h/2 + 1 + (h/2 - 1) = h$, so the path $\pi'_{s,t}$ does not need to be hit by H. Consequently, observe that it is enough for H to hit all the $(h/2-1)$-hop root paths in $T_s[u]$ in order to have $V(\pi'_{s,t}) \cap H \neq \emptyset$ for each $t \in T_s[u]$ with $|V(\pi'_{s,t})| \geq h$.

Let $\mathcal{Z}$ be the collection of all the subtrees $T_s[u]$, where $s \in V$ and $su \in E(T_s)$. It is now enough to compute an $\widetilde{O}(n/h)$-sized hitting set H_1 of each of the $(h/2-1)$-hop root paths in all trees in $\mathcal{Z}$. Then, $H_0 \cup H_1$ will form a desired hitting set H of all the paths in Π' with at least h distinct vertices. To this end, we could use a well-known variant of Lemma 8 due to King [30, Lemma 5.2]. However, the running time of that algorithm cannot be easily bounded with the total size N of $\mathcal{Z}$ (i.e., $N = \sum_{T\in\mathcal{Z}} |T|$) exclusively; its running time is $O\left(N + \sum_{T\in\mathcal{Z}} \min(n \log n, |T|k)\right) = O(\min(Nk, |\mathcal{Z}|n \log n))$ if one desires to hit k-hop root paths. Though, for some important cases, e.g., when $\mathcal{Z}$ contains n trees with $\Theta(n)$ vertices each, the running time is near-linear in N for any k. Unfortunately, this might not be the case in our scenario. Instead, we present a more sophisticated near-linear (independent of k) time algorithm for this task.

▶ **Theorem 9.** *Let V be a vertex set of size n and let $\mathcal{Z}$ be a family of trees on V. Let $N = \sum_{T\in\mathcal{Z}} |T|$. For any $k \in [1, n]$, in $O(N \log^2 n)$ time one can deterministically compute an $O(n/k \cdot \log n)$-sized* hitting set *$H \subseteq V$ of all the k-hop root paths in all the trees in $\mathcal{Z}$.*

Proof. We first iteratively prune the trees in $\mathcal{Z}$ of all the leaves at depths not equal to k: this does not alter the set of subpaths required to be hit. Afterwards, the task is to hit all the root-leaf paths in the collection $\mathcal{Z}$, each of exactly k hops.

Similarly as in [30], we would like to simulate the greedy algorithm behind Lemma 8, that is, repeatedly pick a vertex $v \in V$ hitting the largest number of paths not yet hit, and add it to the constructed set H. However, we cannot afford to follow this approach directly. Instead, when $L \geq 1$ paths are remaining to be hit, and there is $n' \geq k+1$ vertices $V \setminus H$ that have not yet been chosen, we pick a vertex hitting at least $\frac{L(k+1)}{2n'}$ remaining paths. Note that there always exists a vertex hitting at least $\frac{L(k+1)}{n'}$ remaining paths, since otherwise some of the remaining paths would contain a vertex from outside $V \setminus H$, a contradiction. A single step in our approach reduces L to at most $\left(1 - \frac{k+1}{2n'}\right) L$, so $\lceil 2n'/(k+1) \rceil = O(n/k)$ steps reduce L to at most L/e. Hence, since L is an integer, after $O\left(\frac{n}{k} \ln L\right) = O\left(\frac{n}{k} \ln N\right)$ steps L will drop to 0, i.e., all required paths will be hit.

Our strategy can be also rephrased as follows: maintain 2-approximate counters $\{c_v : v \in V\}$ such that the vertex v hits between c_v and $2c_v$ of the remaining paths, and repeatedly pick a vertex z with the maximum value of c_z. By the above discussion, the picked z will always satisfy $c_z \geq L(k+1)/2n'$, as desired. To implement this strategy, we proceed as follows.

For each $T \in \mathcal{Z}$, and $v \in V(T)$, let $d_{v,T}$ be the *exact* number of *previously not hit* root-leaf paths in T that v hits. Note that through the entire collection, v hits $D_v := \sum_{T\in\mathcal{Z}} d_{v,T}$ paths not yet hit. Observe that when a root-leaf path to the leaf l in T is hit for the first time, the value $d_{v,T}$ of all the ancestors v of l gets decreased by one. In fact, the algorithm of [30] can be seen to maintain such values $d_{v,T}$ and D_v explicitly. However, this is too costly for us; we will instead maintain the exact values $d_{v,T}$ only implicitly, in a data structure.

For each $T \in \mathcal{Z}$, we keep $V(T)$ (explicitly) partitioned into subsets $V_{T,0}, \ldots, V_{T,\ell}$, where $\ell = O(\log |V(T)|)$, so that $v \in V_{T,i}$ iff $d_{v,T} \in [2^i, 2^{i+1})$. Throughout the process, the values $d_{v,T}$ will only decrease, so a vertex $v \in V(T)$ can only move $O(\log n)$ times to a subset $V_{T,j}$ with a lower value j. Let us first argue that maintaining such partitions yields the desired 2-approximate counters rather straightforwardly.

For $v \in V$, let us define $c_v = \sum_{T,j:v\in V_{T,j}} 2^j$. Then, we have:

$$D_v = \sum_{T\in\mathcal{Z}} d_{v,T} \geq \sum_{T,j:v\in V_{T,j}} 2^j = c_v = \sum_{T,j:v\in V_{T,j}} 2^j = \frac{1}{2} \sum_{T,j:v\in V_{T,j}} 2^{j+1} > \frac{1}{2} \sum_{T\in\mathcal{Z}} d_{v,T} = \frac{1}{2} D_v.$$

As a result, the counters c_v indeed 2-approximate the values D_v and can be maintained subject to changes in the partitions $V_{T,i}$, for all T, i, in $O\left(\sum_{T\in\mathcal{Z}} |T| \log n\right) = O(N \log n)$ time.

Fix some $T \in \mathcal{Z}$. To maintain the partition $V_{T,0}, \ldots, V_{T,\ell}$, we maintain the values $d_{v,T}$ using ℓ data structures $S_{T,0}, \ldots, S_{T,\ell}$. The data structure $S_{T,i}$ associates (implicitly) the following vertex weights to the individual vertices v of T. If $d_{v,T} \geq 2^i$, then v has weight $d_{v,T}$ in $S_{T,i}$. Otherwise, if $d_{v,T} < 2^i$, then v has weight ∞ in $S_{T,i}$. In particular, $S_{T,0}$ associates the exact values $d_{v,T}$ to the vertices of T.

Fix some $i = 0, \ldots, \ell$. $S_{T,i}$ is implemented using, e.g., a top-tree [4, Theorem 2.4] that allows performing the following operations, both in $O(\log n)$ time[5]:

[5] As a matter of fact, in [4] this is shown for edge weights. However, vertex weights can be simulated easily using edge weights by assigning each vertex its parent edge, and explicitly maintaining the weight of the root.

(1) adding the same $\delta \in \mathbb{R}$ to the weights of vertices on some specified path in the tree, and
(2) querying for a vertex of the tree with minimum weight.

Clearly, $S_{T,i}$ can be initialized at the beginning of the process in $O(|T| \log n)$ time. When a new vertex z is added to H, and $z \in V(T)$, we iterate through all the (previously unvisited) descendants of z to identify the (original) leaves y at depth k such that the root-to-y path in T has not been previously hit. For each such y, we decrease the weights of all the ancestors of y in T (all lying on a single path in T) by 1. This requires a single top-tree operation on $S_{T,i}$. Afterwards, for all $w \in V(T)$ whose value $d_{w,T}$ was at least 2^i before adding z to H, $S_{T,i}$ contains (in an implicit way) the correctly updated exact value $d_{w,T}$. Some of these values in $S_{T,i}$ might drop below 2^i, though. To deal with this, we repeatedly attempt to extract the minimum-valued vertex $x \in V(T)$ from $S_{T,i}$. If the value of x is less than 2^i, we reset the value of x in $S_{T,i}$ to ∞. Otherwise, we stop; at this point all the values in $S_{T,i}$ are at least 2^i; the invariant posed on $S_{T,i}$ is fixed.

The above update procedure is performed for each i. Observe that $v \in V_{T,i}$ iff i is the maximum index such that v has assigned a finite value in $S_{T,i}$. Since for all i we can explicitly track which vertices in $S_{T,i}$ are assigned ∞ while performing updates, the time needed to maintain the partition $V_{T,0}, \ldots, V_{T,\ell}$ can be charged to the cost of maintaining the data structures $S_{T,0}, \ldots, S_{T,\ell}$.

Let us now analyze the time cost of this algorithm. For each $T \in \mathcal{Z}$, we iterate through every vertex of T at most $O(1)$ times. For $i = 0, \ldots, \ell$, at most $O(|V(T)| + |H \cap V(T)|) = O(|V(T)|)$ top-tree operations are performed on $S_{T,i}$. Hence, the cost of maintaining all $S_{T,i}$ for all $i = 0, \ldots, O(\log n)$ is $O(|T| \log^2 n)$. Through all $T \in \mathcal{Z}$, this is $O(N \log^2 n)$.

To implement finding a next vertex $z \in H$ with the largest c_z, one may simply store the counters c_z in a priority queue. Since the counters are updated $O(N \log n)$ times in total, the priority queue operations cost is $O(N \log^2 n)$ as well. ◀

Observe that through all s, the total number of edges in trees added to $\mathcal{Z}$ can be bounded by the number of edges in the (compressed) trees T_s of Lemma 4, and thus also by $\widetilde{O}(\min(\tau\Delta, n^2))$. As a result, by Theorem 9, the desired set H hitting all paths $\pi'_{s,t}$ with at least h distinct vertices can be computed in $\widetilde{O}(\tau\Delta)$ time, using at most quadratic space. This does not increase the running time of the update procedure in the asymptotic sense.

3.6 Reducing the space usage

So far, the space used by the preprocessing phase could only be bounded by $O(n^2 h)$ as we have explicitly stored the $O(n^2)$ preprocessed paths $\pi_{s,t} \in \Pi$, each with $O(h)$ hops.

We do not, however, need to store the paths $\pi_{s,t} \in \Pi$ explicitly. For performing updates and answering distance queries, we only require storing the values $\ell(\pi_{s,t})$, $|\pi_{s,t}|$, and being able to efficiently access the sets $\Pi(v)$, for any $v \in V$. If we want to also support path queries, then constant-time reporting of the subsequent edges of $\pi_{s,t}$ is also needed. Probst Gutenberg and Wulff-Nilsen [24, Section 4.2] showed an elegant way of achieving that in a slightly relaxed way using only $\widetilde{O}(n^2 \log h)$ space.

▶ **Lemma 10** ([24]). *Let $G = (V, E)$ be a real-weighted digraph with no negative cycles. Let $s \in V$ and let $h \in [1, n]$. Using $O(mh)$ time and $O(nh)$ space, one can build an $\widetilde{O}(n)$-space data structure representing a collection $\{\pi_t : t \in V\}$ of (not necessarily simple) $O(h)$-hop paths from s to all other vertices in G such that for any t, $\ell(\pi_t) \le \delta^h_G(s, t)$.*

For any $v \in V$, the data structure allows:

- *accessing $\ell(\pi_v)$ and $|\pi_v|$ in $O(1)$ time,*
- *reporting the set $P_v = \{t \in V : v \in V(\pi_t)\}$ in $\widetilde{O}(|P_v|)$ time,*
- *reporting the edges of π_v in $O(|\pi_v|)$ time.*

Proof sketch. Suppose we compute shortest h-hop-bounded $s \to t$ paths p_t from s to all $t \in V$. This takes $O(mh)$ time, but storing the computed paths explicitly would require $\Theta(nh)$ space. Recall that if G has no negative cycles, then we may wlog. assume that the paths p_t are all simple. As a result, one can deterministically compute an $\widetilde{O}(n/h)$-sized hitting set H of the $\lceil h/3 \rceil$-hop infixes starting at the $(\lceil h/3 \rceil + 2)$-th hop of those of the computed p_t that satisfy $|p_t| \geq \lceil 2h/3 \rceil$. We explicitly store the paths p_y for all $y \in H$ which costs only $\widetilde{O}(|H| \cdot h) = \widetilde{O}(n)$ space.

Let G' be obtained from G be adding shortcut edges $e_y = sy$ of weight $w(e_y) = \ell(p_y)$ for all $y \in H$. Note that for all $v \in V$, $\delta_{G'}^{\lceil 2h/3 \rceil}(s, v) \leq \delta_G^h(s, v) = \ell(p_v)$ and every $\leq \lceil 2h/3 \rceil$-hop path in G' corresponds to a path in G with at most $h + \lceil h/3 \rceil$ hops.

We recursively solve the problem on the graph G' with hop-bound $h' = \lceil 2h/3 \rceil$. Let $\{\pi'_t : t \in V\}$ be the obtained set of paths. For every $t \in V$, we define π_t to be π'_t with possibly the first shortcut edge e_z expanded to the corresponding path p_z. One can easily prove by induction that $|\pi_t| = O(h)$ and $\ell(\pi_t) \leq \delta_G^h(s, t)$. The recursion depth is clearly $O(\log h)$.

Finally, each of the explicitly stored $\widetilde{O}(n/h)$ paths p_t at some level of the recursion can be imagined to point to at most one path of the previous level (corresponding to a shortcut edge) and some $O(h)$ distinct vertices of G. By keeping only the nodes reachable from the paths at the last level of the recursion in this pointer system, and storing reverse pointers, we can report the elements of each P_v so that every element gets reported $O(\log n)$ times. ◀

To reduce the space to $\widetilde{O}(n^2)$, we simply replace the Bellman-Ford like procedure run on $G - C$ in the preprocessing of Lemma 3 with the construction of Lemma 10. The total congestion of all the vertices can increase only by a constant factor then. In Section 3.5 we have assumed that the preprocessed paths $\pi_{s,t}$ were simple when hitting all $\pi_{s,t}$ satisfying $|\pi_{s,t}| \geq h/2$ with H_0. But we can as well assume that H_0 hits all $\pi_{s,t}$ with $|V(\pi_{s,t})| \geq h/2$ instead. Even though the paths represented by the data structure of Lemma 10 might be non-simple, we can compute the sizes $|V(\pi_v)|$ within the same bound easily. Moreover, the algorithm behind Lemma 8 can be implemented so that it requires only $O(n)$ additional space if it is possible to (1) iterate through the elements of individual sets of $\mathcal{Y}$ in $O(1)$ time per element, and (2) report the sets $Y \in \mathcal{Y}$ containing a given $x \in X$ in near-linear time in the number of reported sets. This is precisely what Lemma 10 enables.

3.7 Negative edges and cycles

In this section we briefly describe the modifications to the data structure needed to handle negative edge weights and possibly negative cycles.

First of all, we run in parallel a deterministic fully dynamic negative cycle detection algorithm with $\widetilde{O}(m)$ worst-case update time (see, e.g., [27]). That algorithm also maintains a feasible price function p of the current graph G. With this in hand, whenever G has a negative cycle, we refrain from running the update procedure and forbid issuing queries. Otherwise, p is also a feasible price function of $G - D$, and thus the Dijkstra-based update procedure can simply use p to ensure that all the edge and path lengths accessed are non-negative.

In the basic randomized variant of our data structure we don't need to alter the preprocessing at the beginning of a phase at all. Indeed, our analysis did not require that the paths $\pi'_{s,t}$ are simple or with no negative cycles, and h-hop-bounded shortest paths are well-defined even in presence of negative cycles. In the $O(n^2 h)$-space deterministic variant (Section 3.5), similarly as in Section 3.6, we may compute the hitting set H_0 only for those $\pi_{s,t}$ that satisfy $|V(\pi_{s,t})| \geq h/2$. Recall that if the update procedure is run, then $G - D$ has no negative cycle and hence no path $\pi_{s,t}$ containing a negative cycle survives in $G - D$ anyway.

Finally, the preprocessing algorithm behind Lemma 10 internally uses hitting-set arguments (valid for simple paths) and requires, out-of-the-box, that there are no negative cycles. We now sketch how to deal with negative cycles while using the space-saving Lemma 10.

Whenever the preprocessing in Lemma 10 for source s encounters a path p_t containing a negative cycle, we use it as the desired path π_t, but discard it when computing a hitting set and thus also in the recursive preprocessing in Lemma 10 – effectively making reporting π_t (in any way) during update or query impossible. Similarly, such a path is included as $\pi_{s,t} \in \Pi$ in Lemma 3 only implicitly and marked as *negative*, but nevertheless used for updating the congestion counters $\alpha(\cdot)$ during the preprocessing. Note that during the update procedure, if G has no negative cycles, then for each "negative" path $\pi_{s,t}$, we have $V(\pi_{s,t}) \cap D \neq \emptyset$. The used charging scheme ensures that we can afford reconstructing the path $\pi'_{s,t}$ within the $\widetilde{O}(\tau\Delta)$ bound even though we do not know which vertices of D lie on $\pi_{s,t}$.

4 Algebraic fully dynamic reachability in sparse digraphs

In this section we show how the algebraic approach to dynamic reachability [35] can be applied in the case of sparse graphs, even without resorting to fast matrix multiplication [3, 21].

Assume for simplicity that $m = |E(G)| \geq n$ at all times. We prove the following.

▶ **Theorem 2.** *Let G be a directed graph. Let $t \in [1, \sqrt{m}]$. There exist a Monte Carlo randomized data structure maintaining G subject to fully dynamic single-edge updates with $\widetilde{O}(mn/t)$ worst-case update time and supporting arbitrary-pair reachability queries in $O(t)$ time. The answers produced are correct with high probability.*

Let us first review the approach of [35]. Identify the vertices of $G = (V, E)$ with $\{1, \ldots, n\}$. Assume G has a self-loop at every vertex, i.e., $vv \in E$ for all $v \in V$; self-loops do not change the reachability relation in G. Let $A(G)$ be an adjacency matrix of G, that is, an $n \times n$ matrix with the entry $A(G)_{ij}$ equal to 1 if $ij \in E(G)$, and 0 otherwise.

Let us choose a field $\mathbb{F} = \mathbb{Z}/p\mathbb{Z}$, for a prime number $p = \Theta(n^c)$, where $c \geq 3$ is a constant. Let the matrix $\tilde{A}(G)$ be obtained from $A(G)$ by replacing each 1 with a random element from $\mathbb{F}$. Sankowski [35, Theorem 6] showed the following.

▶ **Theorem 11.** [35] *With high probability (controlled by the constant c), the matrix $\tilde{A}(G)$ is invertible over $\mathbb{F}$ and for all $u, v \in V$, $(\tilde{A}(G)^{-1})_{u,v} \neq 0$ holds if and only if there exists a $u \to v$ path in G.*

Theorem 11 reduces fully dynamic reachability to the *dynamic matrix inverse* problem. Note that a single-edge update to G translates to a single-entry matrix update on $\tilde{A}(G)$, whereas a reachability query corresponds to an element query on the inverse $\tilde{A}(G)^{-1}$.

Sankowski [35] studied update/query tradeoffs for the dynamic matrix inverse problem. One tradeoff, summarized by the following theorem, is of our particular interest.

▶ **Theorem 12.** [35] *Suppose a matrix $A \in \mathbb{F}^{n \times n}$ is subject to single-element updates that keep A non-singular at all times.*

Let $\delta \in (0, 1)$. There exists a data structure maintaining A^{-1} with $\widetilde{O}(n^{\omega(1,\delta,1)-\delta} + n^{1+\delta})$ worst-case update time and supporting element queries on A^{-1} in $O(n^\delta)$ time.

Above, $\omega(1, \delta, 1) \geq 2$ denotes the rectangular matrix multiplication exponent (see [21]), i.e., a value such that one can multiply an $n \times n^\delta$ matrix by an $n^\delta \times n$ matrix in $\widetilde{O}\left(n^{\omega(1,\delta,1)}\right)$ time. Here, the time is measured in field operations. By applying Theorem 12 with $\delta \approx 0.529$ such that $\omega(1, \delta, 1) = 1 + 2\delta$ to the matrix $\tilde{A}(G)$ (whose inverse correctly encodes the

transitive closure of G throughout $\operatorname{poly} n$ updates, w.h.p. against an adaptive adversary), Sankowski [35] obtains a Monte Carlo randomized fully dynamic reachability algorithm with $\widetilde{O}(n^{1.529})$ worst-case update and $O(n^{0.529})$ query time.

To continue, we need to discuss some of the internals of the data structure of Theorem 12 [35, Section 6]. That data structures operates in phases of n^δ updates. At the end of each phase, the inverse A^{-1} is *explicitly* recomputed from (1) the explicitly stored inverse $(A_0)^{-1}$ of the matrix A_0 from the beginning of the phase, and (2) the n^δ updates in the current phase, via rectangular matrix multiplication. This is the sole reason why the term $n^{\omega(1,\delta,1)-\delta}$ appears in the update bound. In particular, at the beginning of each phase, we could also recompute the inverse of the current matrix A *from scratch* in $O(n^\omega)$ time and thus obtain a slightly worse update bound of $\widetilde{O}(n^{\omega-\delta}+n^{1+\delta})$, which in turn leads to the $\widetilde{O}(n^{(\omega+1)/2}) = O(n^{1.687})$ update bound if optimized wrt. δ. The query time is proportional to the phase length n^δ.

Speaking more generally, if we could compute the inverse of the maintained matrix at any time in T time, then by following the approach behind Theorem 12, for any parameter $t \in [1, n]$ (denoting the phase length) we could obtain a data structure with $\widetilde{O}(T/t + nt)$ worst-case update time and $O(t)$ query time. For $T = \Omega(n)$, it only makes sense to use $t \in \left[1, \sqrt{T/n}\right]$, and the update bound then simplifies to $\widetilde{O}(T/t)$. To obtain our fully dynamic reachability algorithm for sparse digraphs, we use this observation combined with the below theorem following from a classical result on solving linear systems in parallel [26].

▶ **Theorem 13.** *Let $A \in \mathbb{F}^{n\times n}$ be a non-singular matrix with $m = \Omega(n)$ non-zero entries. Assume the finite field $\mathbb{F}$ has at least n^{2+c} elements, where $c \geq 1$ is a constant.*

There is a Las Vegas randomized algorithm that computes A^{-1} in $\widetilde{O}(nm)$ time. The success probability is at least $1 - O(n^{-c})$.

Proof sketch. Kaltofen and Pan [26, Theorem 4] show, using techniques of [41], that finding the determinant of A can be reduced, in $\widetilde{O}(n^2)$ time, to solving the following subproblems:

(a) For a given vector $v \in \mathbb{F}^{n\times 1}$, computing vectors $\tilde{A}^i \cdot v$, for $i = 0, \ldots, 2n-1$, where $\tilde{A} = A \cdot H \cdot D$, $H \in \mathbb{F}^{n\times n}$ is a Hankel matrix, and $D \in \mathbb{F}^{n\times n}$ is a diagonal matrix.

(b) For a given vector $v \in \mathbb{F}^{n\times 1}$, computing vectors $T^i \cdot v$ for $i = 0, \ldots, n-1$, where $T \in \mathbb{F}^{n\times n}$ is a Toeplitz matrix.

Then, in [26, Section 4] it is proven that if the determinant algorithm is realized using a randomized algebraic circuit, or, in other words, a straight-line program with no conditional branches, loops, etc., that possibly can divide by zero with low probability, then the Baur-Strassen theorem [7] implies that the matrix inverse can be computed within the same asymptotic bound as the determinant, even in parallel.

The subproblems (a) and (b) for general dense $n \times n$ matrices can be solved within this model in $\widetilde{O}(n^\omega)$ time using a folklore combination of repeated squaring and fast matrix multiplication (see, e.g., [29]). In our case, to obtain a desired $\widetilde{O}(mn)$-time sparse matrix inverse algorithm, it is enough to argue that the subproblems (a) and (b) admit $\widetilde{O}(mn)$ time straight-line program (SLP) solutions for matrices with m non-zero entries.

Consider the subproblem (a), since (b) is very similar. We compute each subsequent vector $\tilde{A}^{i+1} \cdot v$ as $A \cdot (H \cdot (D \cdot (\tilde{A}^i v)))$. Multiplying a vector by a matrix with $m = \Omega(n)$ non-zero entries can clearly be realized in $O(m)$ time using an SLP with no conditional statements. This justifies that multiplications by the matrices A and D can be realized in the required model. It is also well-known that multiplying a vector by a Hankel/Toeplitz matrix reduces to polynomial multiplication (see, e.g., [22]), and thus also can be realized

using an $\widetilde{O}(n)$-gate straight-line program (see, e.g., [11]). This proves that each $\tilde{A}^{i+1}v$ can be obtained from $\tilde{A}^{i}v$ in $\widetilde{O}(m)$ time in the SLP model. This implies the desired $\widetilde{O}(nm)$ SLP time bound for subproblem (a). The theorem follows. ◀

▶ **Corollary 14.** *Suppose a matrix $A \in \mathbb{F}^{n\times n}$ is subject to single-element updates that keep A non-singular at all times and the number of non-zero elements in A is always $O(m)$, where $m \geq n$. Let $t \in [1, \sqrt{m}]$. There exists a data structure maintaining A^{-1} with $\widetilde{O}(mn/t)$ worst-case update time and supporting element queries on A^{-1} in $O(t)$ time.*

The above corollary applied to the matrix $\tilde{A}(G)$ combined with Theorem 11 implies Theorem 2.

References

1. Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014*, pages 434–443. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.53.
2. Ittai Abraham, Shiri Chechik, and Sebastian Krinninger. Fully dynamic all-pairs shortest paths with worst-case update-time revisited. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 440–452. SIAM, 2017. doi:10.1137/1.9781611974782.28.
3. Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 522–539. SIAM, 2021. doi:10.1137/1.9781611976465.32.
4. Stephen Alstrup, Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup. Maintaining information in fully dynamic trees with top trees. *ACM Trans. Algorithms*, 1(2):243–264, 2005. doi:10.1145/1103963.1103966.
5. Giorgio Ausiello, Giuseppe F. Italiano, Alberto Marchetti-Spaccamela, and Umberto Nanni. Incremental algorithms for minimal length paths. *J. Algorithms*, 12(4):615–638, 1991. doi:10.1016/0196-6774(91)90036-X.
6. Surender Baswana, Ramesh Hariharan, and Sandeep Sen. Improved decremental algorithms for maintaining transitive closure and all-pairs shortest paths. *J. Algorithms*, 62(2):74–92, 2007. doi:10.1016/j.jalgor.2004.08.004.
7. Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theor. Comput. Sci.*, 22:317–330, 1983. doi:10.1016/0304-3975(83)90110-X.
8. Thiago Bergamaschi, Monika Henzinger, Maximilian Probst Gutenberg, Virginia Vassilevska Williams, and Nicole Wein. New techniques and fine-grained hardness for dynamic near-additive spanners. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 1836–1855. SIAM, 2021. doi:10.1137/1.9781611976465.110.
9. Aaron Bernstein. Maintaining shortest paths under deletions in weighted directed graphs. *SIAM J. Comput.*, 45(2):548–574, 2016. doi:10.1137/130938670.
10. Aaron Bernstein, Maximilian Probst Gutenberg, and Thatchaphol Saranurak. Deterministic decremental SSSP and approximate min-cost flow in almost-linear time. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*, pages 1000–1008. IEEE, 2021. doi:10.1109/FOCS52979.2021.00100.
11. David G. Cantor and Erich Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28(7):693–701, 1991. doi:10.1007/BF01178683.
12. Sílvia Casacuberta and Rasmus Kyng. Faster sparse matrix inversion and rank computation in finite fields. In *13th Innovations in Theoretical Computer Science Conference, ITCS 2022*, volume 215 of *LIPIcs*, pages 33:1–33:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.33.

13 Shiri Chechik and Tianyi Zhang. Faster deterministic worst-case fully dynamic all-pairs shortest paths via decremental hop-restricted shortest paths. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023*, pages 87–99. SIAM, 2023. doi:10.1137/1.9781611977554.ch4.

14 Li Chen, Gramoz Goranci, Monika Henzinger, Richard Peng, and Thatchaphol Saranurak. Fast dynamic cuts, distances and effective resistances via vertex sparsifiers. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 1135–1146. IEEE, 2020. doi:10.1109/FOCS46700.2020.00109.

15 Julia Chuzhoy. Decremental all-pairs shortest paths in deterministic near-linear time. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 626–639. ACM, 2021. doi:10.1145/3406325.3451025.

16 Camil Demetrescu and Giuseppe F. Italiano. A new approach to dynamic all pairs shortest paths. *J. ACM*, 51(6):968–992, 2004. doi:10.1145/1039488.1039492.

17 Camil Demetrescu and Giuseppe F. Italiano. Trade-offs for fully dynamic transitive closure on dags: breaking through the $O(n^2)$ barrier. *J. ACM*, 52(2):147–156, 2005. doi:10.1145/1059513.1059514.

18 Wayne Eberly, Mark Giesbrecht, Pascal Giorgi, Arne Storjohann, and Gilles Villard. Faster inversion and other black box matrix computations using efficient block projections. In *Symbolic and Algebraic Computation, International Symposium, ISSAC 2007, Proceedings*, pages 143–150. ACM, 2007. doi:10.1145/1277548.1277569.

19 Jacob Evald, Viktor Fredslund-Hansen, Maximilian Probst Gutenberg, and Christian Wulff-Nilsen. Decremental APSP in unweighted digraphs versus an adaptive adversary. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*, volume 198 of *LIPIcs*, pages 64:1–64:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.64.

20 Sebastian Forster, Yasamin Nazari, and Maximilian Probst Gutenberg. Deterministic incremental APSP with polylogarithmic update time and stretch. *CoRR*, abs/2211.04217, 2022. doi:10.48550/arXiv.2211.04217.

21 Francois Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 1029–1046. SIAM, 2018. doi:10.1137/1.9781611975031.67.

22 Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

23 Maximilian Probst Gutenberg, Virginia Vassilevska Williams, and Nicole Wein. New algorithms and hardness for incremental single-source shortest paths in directed graphs. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*, pages 153–166. ACM, 2020. doi:10.1145/3357713.3384236.

24 Maximilian Probst Gutenberg and Christian Wulff-Nilsen. Fully-dynamic all-pairs shortest paths: Improved worst-case time and space bounds. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020*, pages 2562–2574. SIAM, 2020. doi:10.1137/1.9781611975994.156.

25 Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015*, pages 21–30. ACM, 2015. doi:10.1145/2746539.2746609.

26 Erich Kaltofen and Victor Y. Pan. Processor efficient parallel solution of linear systems over an abstract field. In *Proceedings of the 3rd Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA '91*, pages 180–191. ACM, 1991. doi:10.1145/113379.113396.

27 Adam Karczmarz. Fully dynamic algorithms for minimum weight cycle and related problems. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*, volume 198 of *LIPIcs*, pages 83:1–83:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.83.

28 Adam Karczmarz, Anish Mukherjee, and Piotr Sankowski. Subquadratic dynamic path reporting in directed graphs against an adaptive adversary. In *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1643–1656. ACM, 2022. doi:10.1145/3519935.3520058.

29 Walter Keller-Gehrig. Fast algorithms for the characteristic polynomial. *Theor. Comput. Sci.*, 36:309–317, 1985. doi:10.1016/0304-3975(85)90049-0.

30 Valerie King. Fully dynamic algorithms for maintaining all-pairs shortest paths and transitive closure in digraphs. In *40th Annual Symposium on Foundations of Computer Science, FOCS 1999*, pages 81–91. IEEE Computer Society, 1999. doi:10.1109/SFFCS.1999.814580.

31 Veli Mäkinen, Alexandru I. Tomescu, Anna Kuosmanen, Topi Paavilainen, Travis Gagie, and Rayan Chikhi. Sparse dynamic programming on dags with small width. *ACM Trans. Algorithms*, 15(2):29:1–29:21, 2019. doi:10.1145/3301312.

32 Liam Roditty. A faster and simpler fully dynamic transitive closure. *ACM Trans. Algorithms*, 4(1):6:1–6:16, 2008. doi:10.1145/1328911.1328917.

33 Liam Roditty and Uri Zwick. Improved dynamic reachability algorithms for directed graphs. *SIAM J. Comput.*, 37(5):1455–1471, 2008. doi:10.1137/060650271.

34 Liam Roditty and Uri Zwick. On dynamic shortest paths problems. *Algorithmica*, 61(2):389–401, 2011. doi:10.1007/s00453-010-9401-5.

35 Piotr Sankowski. Dynamic transitive closure via dynamic matrix inverse (extended abstract). In *45th Symposium on Foundations of Computer Science FOCS 2004*, pages 509–517. IEEE Computer Society, 2004. doi:10.1109/FOCS.2004.25.

36 Mikkel Thorup. Fully-dynamic all-pairs shortest paths: Faster and allowing negative cycles. In *Algorithm Theory – SWAT 2004, 9th Scandinavian Workshop on Algorithm Theory, Proceedings*, volume 3111 of *Lecture Notes in Computer Science*, pages 384–396. Springer, 2004. doi:10.1007/978-3-540-27810-8_33.

37 Jeffrey D. Ullman and Mihalis Yannakakis. High-probability parallel transitive-closure algorithms. *SIAM J. Comput.*, 20(1):100–125, 1991. doi:10.1137/0220006.

38 Jan van den Brand, Sebastian Forster, and Yasamin Nazari. Fast deterministic fully dynamic distance approximation. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022*, pages 1011–1022. IEEE, 2022. doi:10.1109/FOCS54457.2022.00099.

39 Jan van den Brand and Danupon Nanongkai. Dynamic approximate shortest paths and beyond: Subquadratic and worst-case update time. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019*, pages 436–455. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00035.

40 Jan van den Brand, Danupon Nanongkai, and Thatchaphol Saranurak. Dynamic matrix inverse: Improved algorithms and matching conditional lower bounds. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019*, pages 456–480. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00036.

41 Douglas H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Trans. Inf. Theory*, 32(1):54–62, 1986. doi:10.1109/TIT.1986.1057137.

42 Uri Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *J. ACM*, 49(3):289–317, 2002. doi:10.1145/567112.567114.

A Further variants of the fully dynamic shortest paths data structure

A.1 Unweighted digraphs

Similarly as in the case of previous fully dynamic APSP data structures [2, 24], improved bounds can be obtained if the graph G is unweighted. This is simply because the preprocessing of Lemma 3 can be completed in $O(mn)$ time instead of $O(mnh)$ time. Indeed, in an unweighted graph, the shortest h-hop-bounded s, t path, if exists, coincides with the (globally) shortest s, t path. As a result, the Bellman-Ford-based computation can be replaced with breadth-first search. Similarly, the collection of paths Π can be represented using n BFS trees and thus one can achieve quadratic space without resorting to Lemma 10.

For unweighted graphs, the update bound becomes $\widetilde{O}(m\Delta+mn/\Delta+mn/h+nm^2h/\tau+\Delta\tau)$, whereas the query time remains $\widetilde{O}(\Delta + nmh/\tau + n/h)$. For $\Delta = h = n^{1/4}$ and $\tau = mn^{1/2}$ the update and query time bounds become $\widetilde{O}(mn^{3/4})$ and $\widetilde{O}(n^{3/4})$, respectively.

A.2 A slight tradeoff

In the basic variant of the data structure, it is not clear whether pushing the update time below $\widetilde{O}(n^{4/5})$ is possible even at the cost of increasing the query time. Here, we sketch that a slight tradeoff is indeed possible with another trick of [24, Section 4.1]: to delegate handling paths through the congested set to the data structure of [2, Section 3]. For simplicity, assume again that the edge weights are non-negative. Since that data structure, in turn, is tailored to dense graphs, we instead use the following sparse variant implicit in [27].

▶ **Lemma 15.** [2, 27] *Let $G = (V, E)$ be a directed graph and let $C \subseteq V$. Let $h \in [1, n]$. In $\widetilde{O}(|C|mh)$ time one can build a data structure supporting the following.*

For any query set $D \subseteq V$, update the data structure so that it supports queries computing the length of some $s \to t$ path of length at most $\min_{c\in C}\{\delta^h_{G-D}(s, c) + \delta^h_{G-D}(c, t)\}$ for any $s, t \in V$. The worst-case update time is $\widetilde{O}(|D|mh)$ and the query time is $O(|C|)$.

Proof sketch. For at most $2|C|$ *centers* $c_1, \ldots, c_\ell$, repeatedly find shortest $2h$-hop-bounded paths from/to c_i in $G - \{c_1, \ldots, c_{i-1}\}$. While this computation proceeds, maintain vertex congestions $\alpha(\cdot)$ as in Lemma 3. When choosing the subsequent centers c_i, alternate between picking an unused vertex from C and the most congested vertex of $V \setminus \{c_1, \ldots, c_{i-1}\}$, until all vertices of C are used. This preprocessing costs $O(\ell mh) = O(|C|mh)$ time.

Given the above preprocessing, one can prove that by proceeding as in Lemma 4, in $\widetilde{O}(|D|mh)$ time one can recompute a representation of paths $s \to c_i$ of length at most $\delta^{2h}_{G-(D\cup\{c_1,\ldots,c_{i-1}\})}(s, c_i)$ and analogous paths $c_i \to t$, for all i and $s, t \in V$.

Upon a query (s, t), in $O(\ell) = O(|C|)$ time we can find an $s \to t$ path of length at most $y^* = \min_{i=1}^{\ell}\{y_i\}$, where $y_i := \delta^{2h}_{G-(D\cup\{c_1,\ldots,c_{i-1}\})}(s, c_i) + \delta^{2h}_{G-(D\cup\{c_1,\ldots,c_{i-1}\})}(c_i, t)$. To see that this is enough, let $c^* \in C$ be such that $\delta^h_{G-D}(s, c^*) + \delta^h_{G-D}(c^*, t)$ is minimum. Let j be minimum index such that the corresponding $\leq 2h$-hop path $Q = s \to c^* \to t$ contains the center c_j. Then we have $Q \subseteq G - (D \cup \{c_1, \ldots, c_{j-1}\})$ and thus $y^* \leq y_j \leq \ell(Q)$. ◀

Note that by computing shortest-paths trees from and to a randomly sampled $\widetilde{O}(n/h)$-sized hitting set H we can in fact handle "long" shortest paths in the current graph G, and not only in $G - (C \cup D)$. As a result, we don't need to recompute full shortest paths trees from C – instead, it would be enough to consider short paths in $G - D$ through C upon query. This is what we use Lemma 15 for. Every Δ updates, when a new phase starts, a fresh congested set C is computed. We additionally initialize the data structure of Lemma 15 for the current graph G and the congested set C. This way, that data structure is always off from the current G by at most Δ updates, and thus can be updated in $\widetilde{O}(\Delta mh)$ time. Again, the data structure of Lemma 15 can be reinitialized in such a way that the additional worst-case cost incurred is $\widetilde{O}(|C|mh/\Delta)$. The full worst-case update time becomes:

$$\widetilde{O}(m\Delta + mnh/\Delta + mn/h + \Delta\tau + m^2nh^2/(\tau\Delta) + \Delta hm).$$

Balancing as before, for $\Delta = h^2$ and $\tau = mn/h^3$, we obtain the update bound $\widetilde{O}(mn/h+mh^3)$. Note that this bound is $\Omega(mn^{3/4})$ for any h.

The query bound unfortunately remains $\widetilde{O}(\Delta + |H| + mnh/\tau) = \widetilde{O}(n/h + h^4)$. If we aim at serving $\Theta(n)$ queries per update and the graph is sparse, then we get no improvement over the basic approach. However, for a desired query time of $\widetilde{O}(t)$, where $t \in [n^{4/5}, n]$, we can achieve $\widetilde{O}(mn/t^{1/4})$ worst-case update time this way.

New Additive Emulators

Shimon Kogan
Weizmann Institute of Science, Rehovot, Israel

Merav Parter
Weizmann Institute of Science, Rehovot, Israel

Abstract

For a given (possibly weighted) graph $G = (V, E)$, an additive emulator H is a weighted graph in $V \times V$ that preserves the (all pairs) G-distances up to a small additive stretch. In their breakthrough result, [Abboud and Bodwin, STOC 2016] ruled out the possibility of obtaining $o(n^{4/3})$-size emulator with $n^{o(1)}$ additive stretch. The focus of our paper is in the following question that has been explicitly stated in many of the prior work on this topic:

What is the minimal additive stretch attainable with linear size emulators?

The only known upper bound for this problem is given by an implicit construction of [Pettie, ICALP 2007] that provides a linear-size emulator with $+\widetilde{O}(n^{1/4})$ stretch. No improvement on this problem has been shown since then.

In this work we improve upon the long standing additive stretch of $\widetilde{O}(n^{1/4})$, by presenting constructions of linear-size emulators with $\widetilde{O}(n^{0.222})$ additive stretch. Our constructions improve the state-of-the-art size vs. stretch tradeoff in the entire regime. For example, for every $\epsilon > 1/7$, we provide $+n^{f(\epsilon)}$ emulators of size $\widetilde{O}(n^{1+\epsilon})$, for $f(\epsilon) = 1/5 - 3\epsilon/5$. This should be compared with the current bound of $f(\epsilon) = 1/4 - 3\epsilon/4$ by [Pettie, ICALP 2007].

The new emulators are based on an extended and optimized toolkit for computing weighted additive emulators with sublinear distance error. Our key construction provides a weighted modification of the well-known Thorup and Zwick emulators [SODA 2006]. We believe that this TZ variant might be of independent interest, especially for providing improved stretch for distant pairs.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases Spanners, Emulators, Distance Preservers

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.85

Category Track A: Algorithms, Complexity and Games

Funding This project is funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 949083), and by the Israeli Science Foundation (ISF), grant No. 2084/18.

1 Introduction

Emulators are well-studied compression schemes that approximately encode the distance metric of a (dense) undirected input graph $G = (V, E)$ by a sparse *weighted* graph $H \subseteq V \times V$. This extends the notion of spanners which are required to be subgraphs of G. Along with their spanner cousin, emulators admit a wide range of algorithmic applications, most notably in settings related to graph compression, routing schemes, distributed computing, and all pairs shortest paths approximation. The focus of this paper is in providing improved constructions for *additive emulators* which only allow for additive stretch. For a given unweighted n-vertex graph $G = (V, E)$, a graph $H \subseteq V \times V$ is an $f(d)$-emulator if $\text{dist}_G(u, v) \leq \text{dist}_H(u, v) \leq f(\text{dist}_G(u, v))$ for every $u, v \in V$. An $f(d)$-emulator for $f(d) = d + \beta$ for some fixed β is denoted as *additive* emulator. There has been a long line of work on additive emulators, both from an upper bound and lower bound perspectives, see Table 1.

The first explicit construction for this setting obtained $+4$ emulators of size $O(n^{4/3})$ by Dor, Halperin and Zwick [11]. The question of whether sparser emulators exist for any constant additive stretch has been one of the most major open problems in the area. In their breakthrough result, Abboud and Bodwin [1] refuted this possibility by demonstrating that any emulator with $O(n^{4/3-\epsilon})$ edges might induce a polynomially large additive stretch of $\Omega(n^{\delta(\epsilon)})$, for any ϵ.

On the other side of the size vs. stretch tradeoff, additive emulators of *linear* size have in particular attracted a lot of attention over the years [18, 7, 10, 9, 1, 16, 2, 14, 17]. To this date, the best additive stretch known for linear size emulators is $\widetilde{O}(n^{1/4})$, as shown (implicitly) by an earlier work of Pettie [18]. Bodwin and Vassilevska Williams [10] designed linear-size spanners and emulators with additive stretch of $+\widetilde{O}(\sqrt{n})$ (resp., $+\widetilde{O}(n^{1/3})$). In a follow-up work [9] , they cleverly improved the spanner's stretch to the state-of-the-art bound of $+\widetilde{O}(n^{3/7})$; Unfortunately, their improved spanner constructions do not seem to imply improved bounds for emulators, and Pettie's result [18] remains the state-of-the-art.

In this paper we focus on the following basic graph compression problem which despite all efforts is still fairly open:

▶ **Question 1.1.** What is the minimal additive error that can be achieved with linear space?

This question on its various forms (e.g., spanners, emulators) has been raised in many of the prior work on the topic, see e.g., [10, 3], especially in light of the "4/3 barrier" of [1]. Indeed in their seminal lower bound paper, Abboud and Bodwin [1] explicitly asked:

> *Our work shows that polynomial additive error must be suffered in order to obtain near-linear size compression of graphs. Given this, it is natural to wonder* how much *polynomial error is necessary to obtain compression in this regime.*

While not much progress has been provided on the upper bound side, there has been more movement on the lower bound aspects of the problem. Abboud and Bodwin showed that any linear size emulator must suffer $\Omega(n^{1/22})$ additive stretch, in the worst case. Huang and Pettie [16] improved this bound to $+\Omega(n^{1/18})$. This was further improved by Lu, Wein, Vassilevska Williams, and Xu [17] to $+\Omega(n^{2/29})$. Very recently, Bodwin and Hoppenworth [8] provided an $+\Omega(n^{1/7})$ stretch lower bound for linear spanners, by extending the known obstacle product framework to support also non-layered graphs.

Our new constructions are built upon modifying and extending the existing constructions for emulators with sublinear additive stretch and weighted additive spanners. While these notions have been studied before, our primary conceptual contribution is in demonstrating their usefulness for computing additive emulators of unweighted graphs. We next discuss the prior work on each of these settings.

Sublinear additive stretch. Elkin and Peleg showed that the "4/3 barrier" could be broken if one allows a $(1+\epsilon)$ multiplicative stretch, in addition to a small additive stretch [15]. Thorup and Zwick gave an elegant construction of an $O(kn^{1+1/(2^{k+1}-1)})$-size emulator H with $O(1+\epsilon, O(k/\epsilon)^{k-1})$-type stretch[1]. Their emulator has the remarkable property that its stretch bound holds for every $\epsilon > 0$ *simultaneously*, as its size bound is independent in ϵ. For any distance d, choosing $\epsilon = k/d^{1/k}$ leads to an emulator with a sublinear additive stretch function $f(d) = d + O(d^{1-1/k} + 3^k)$. As noted in [2], an interesting open question is whether one can match this size-stretch tradeoff for spanners.

[1] I.e., for every $u, v \in V$, $\mathrm{dist}_H(u,v) \leq (1+\epsilon)\mathrm{dist}_G(u,v) + O(k/\epsilon)^{k-1}$.

Table 1 Upper and lower bounds for additive emulators. New bounds are marked in blue.

Emulator Size	Additive Stretch	Remark	Citation
$O(n^{3/2})$	2		[5]
$\widetilde{O}(n^{4/3})$	4		[11]
$\Omega(n^{1+1/k})$	$2k-1$		[20]
$O(n^{4/3-\epsilon})$	$\Omega(n^{\delta(\epsilon)})$		[1]
$\widetilde{O}(n^{1+\epsilon})$	$O(n^{1/2-3\epsilon/2})$	implicit	[7]
$\widetilde{O}(n^{1+\epsilon})$	$O(n^{1/3-2\epsilon/3})$		[10]
$\widetilde{O}(n^{1+\epsilon+o(1)})$	$O(n^{3/11-9\epsilon/11})$		[9]
$\widetilde{O}(n^{1+\epsilon})$	$O(n^{1/4-3\epsilon/4})$	implicit	[18]
$\widetilde{O}(n^{1+\epsilon})$	$O(n^{1/5-3\epsilon/5})$	$\epsilon \geq 1/7$	**new**
$\widetilde{O}(n^{1+\epsilon})$	$O(n^{(25-87\epsilon)/112})$	$0 \leq \epsilon \leq 1/5$	**new**
$\widetilde{O}(n)$	$O(n^{2/9-1/1600-o(1)})$		**new**
$\widetilde{O}(n)$	$\Omega(n^{1/22})$		[1]
$\widetilde{O}(n)$	$\Omega(n^{1/18})$		[16]
$\widetilde{O}(n)$	$\Omega(n^{2/29})$		[17]

Weighted (near) additive stretch. Elkin, Gitlitz and Neiman [13] provided the first constructions of near-additive spanners for *weighted* graphs. Their algorithm extends the unweighted construction of near-additive spanners (e.g., by [15]) to provide stretch guarantees of $f(d) = (1+\epsilon)d + \beta W$ where W is the maximum edge weight. Ahmed et al. [3] extended the constructions of spanners with purely additive stretch to weighted graphs by an ingenious amortized argument (which plays a role in our constructions, as well). Consequently, they provide $+2W, +4W, +8W$ weighted spanners with $\widetilde{O}(n^{3/2})$, $\widetilde{O}(n^{7/5})$ and $\widetilde{O}(n^{4/3})$ edges, respectively. Elkin, Gitlitz and Neiman [12] improve the latter stretch bound to $(6+o(1))W$, nearly matching the unweighted result for $W = 1$. Note that the above mentioned constructions also provide a *local* stretch guarantee of $+\beta \cdot W_{s,t}$ for every s, t pair, where $W_{s,t}$ is the largest edge weight on an s-t shortest path.

1.1 New Results

We provide a positive progress for Question 1.1 by improving upon the long-standing bound of $+\widetilde{O}(n^{1/4})$ by Pettie [18] to an additive stretch of $+O(n^{0.222-o(1)})$. Our end result is:

▶ **Theorem 1.2.** *Any unweighted n-vertex graph $G = (V, E)$ admit a linear-size emulator with additive stretch $\widetilde{O}(n^{2/9-1/1600-o(1)})$.*

The main novel aspect of this result is in our approach, which draws an interesting connection between weighted additive emulators and unweighted emulators with polynomial additive stretch. The final additive bound $O(n^{0.222-o(1)})$ is obtained by taking a gradual approach, containing two major steps of optimizations.

To illustrate our new algorithmic approach, we start by presenting a very simple construction for recovering the state-of-the-art additive stretch of $+\widetilde{O}(n^{1/4})$. This construction is obtained by using in a black-box manner the recent constructions of *weighted additive*

spanners by Ahmed et al. [4] and Elkin, Gitlitz and Neiman [12] which provides an additive stretch $+\beta W$. While such an additive term might be undesirable in many settings, these constructions play a key role in providing additive emulators for *unweighted* graphs[2].

Improved Stretch vs. Size Bounds via Weighted Additive Emulators. A careful inspection of our $+\widetilde{O}(n^{1/4})$ additive construction reveals that our reduction yields in fact, a specialized *weighted* graph instance with several convenient properties. In particular, we enjoy the fact that our generated weighted graphs are in fact obtained from unweighted graphs, in the sense that the edge weights corresponds to distances, rather than being arbitrary. We then provide a designated construction of weighted additive emulators that takes advantage of these specialized weighted graph instances. This leads to a quite general construction which improves over the known bounds in the entire regime of sparsity, i.e., n to $n^{4/3}$:

▶ **Theorem 1.3.** *For any n-vertex graph $G = (V, E, \omega)$ where $\omega : E \to \{1, \ldots, W\}$ and $0 \leq \epsilon \leq \frac{1}{3}$, there exists a $+\widetilde{O}(W \cdot n^{f(\epsilon)})$ emulator H of size at most $\widetilde{O}(n^{1+\epsilon})$ where:*

$$f(\epsilon) = \begin{cases} (1-3\epsilon)/5 & \text{if } 1/7 \leq \epsilon \leq 1/3; \\ (9-31\epsilon)/40 & \text{if } 3/37 \leq \epsilon \leq 1/7; \\ (3-9\epsilon)/14 & \text{if } 1/15 \leq \epsilon \leq 3/37; \\ (25-87\epsilon)/112 & \text{if } 0 \leq \epsilon \leq 1/15. \end{cases}$$

Setting $\epsilon = 0$, provides a linear-size emulator with additive stretch $n^{25/112} \sim n^{0.223}$.

Discretization of the Thorup-Zwick (TZ) Emulator Construction. Our final emulator result of Theorem 1.2 is based on a rather involved discretization of the TZ emulator construction adapted for weighted graphs. The following (quite technically to state) result serves as the core component of the final linear-size emulator:

▶ **Theorem 1.4.** *For every n-vertex unweighted $G = (V, E)$ a constant integer $k \geq 3$ and integer $D \geq 1$, one can compute an emulator H with additive stretch $O(D^{1-1/(k-1)} \log n)$ for any distance $d = O(D \cdot \log n)$. The size of H is bounded by*

$$\widetilde{O}(n^{1+1/(2^{k+1}-1)} + n^{1+1/(2^k-1)}/(D^{(2^k-2k)/((2^k-1)k(k-1))}).$$

This should be compared with the original TZ construction that provides an additive stretch of $d^{1-1/k}$ using $n^{1+1/(2^{k+1}-1)}$ edges. Theorem 1.4 can also be shown to imply that the stretch function of the TZ emulator is optimal only for a restricted regime of distances. In particular, with a size bound of $\widetilde{O}(n^{1+1/(2^{k+1}-1)})$, one can provide pairs at distances $d \geq n^{k^2/2^k}$ an additive stretch of $O(d^{1-1/(k-1)})$, rather than $O(d^{1-1/k})$ as provided by the TZ bounds, which might be of independent interest.

1.2 Technical Overview

Our $+\widetilde{O}(n^{0.222})$-additive linear-size emulator is obtained in a sequence of two intermediate results, that gradually take advantage of several interesting degrees of freedom in the current constructions of weighted (near) additive emulators. Our technique exhibits several directions

[2] While our constructions utilizes the $+\beta W$ stretch guarantees, it is unclear if the local $+\beta W_{s,t}$ stretch guarantees can be useful in our context, as well.

of optimizations in the emulator framework of Thorup and Zwick [19], which become useful in the context of designing additive emulators with a small polynomial stretch. Note that while all our constructions are implemented in polynomial time, in this paper we put emphasis on the stretch vs. size tradeoff.

Beginner: $+\widetilde{O}(n^{1/4})$ Additive Stretch. As a warmup to our approach, we provide in Sec. 2 a new proof technique to obtain $+\widetilde{O}(n^{1/4})$ emulators of linear size, which simplifies the (implicit) state-of-the-art construction of Pettie [18]. Interestingly, our argument follows immediately by the weighted $+O(W)$ additive spanners of Ahmed et al. [4] and Gitlitz, Elkin, Neiman [12] with $\widetilde{O}(n^{4/3})$ edges, where W is the maximum edge weight of the graph. This provides the starting indication for the potential connection between weighted additive emulators and purely additive emulators for unweighted graphs.

On a high level, the construction works by computing a weighted *net* graph G' for the given (unweighted) graph G, obtained by sampling each G-vertex independently with probability of $\Theta(1/n^{1/4})$. The edges of G' connect every pair of sampled vertices u, v provided that their G-distance is at most $\Theta(n^{1/4}\log n)$. The net edges are weighted by the G-distance between their endpoints. The output emulator is union of two spanners: (i) a $O(\log n)$ multiplicative spanner for G (see Lemma 1.7), and (ii) a $+O(W)$ additive spanner for G' where $W = \Theta(n^{1/4}\log n)$. It is easy to see that the size bound is (near) linear[3]. The stretch argument for nearby pairs u, v at G-distance $O(n^{1/4}\log n)$ follows by the addition of the $O(\log n)$ multiplicative spanner. The argument for distant pairs $\Omega(n^{1/4}\log n)$ follows by using the $+O(W)$ additive spanner for G'.

Intermediate : $+\widetilde{O}(n^{0.223})$ Additive Stretch. The essence of the above mentioned construction is to employ on a weighted additive algorithm on the computed (weighted) net graph G', in a black box manner. Our starting observation, to break the current $+\Theta(n^{1/4})$ barrier, is the following: while G' is indeed a weighted graph, it is obtained from a given *unweighted* base graph G. Therefore it might be possible to treat G' better than any arbitrary input weighted graph. More specifically, by including the TZ emulator for G, one can provide a *sublinear* stretch guarantee for any neighboring pairs in G'. This, in principle, is impossible, for general weighted graphs. Since the sublinear stretch guarantees of the TZ emulators require a superlinear size bound, we cannot employ them directly on G, but rather on a subsampled *net* of G. This sub-sampling immediately converts the unweighted input instance into a weighted instance. We therefore conclude that the key task should be concerned with providing sparse constructions of *weighted additive* emulators. Our core construction computes a superlinear-size emulator for any weighted graph whose weighted stretch and size guarantees depend on the input integer parameters D, k, as follows:

▶ **Theorem 1.5.** *Any n-vertex graph $G = (V, E, \omega)$ with max weight W and integers $k \geq 2, D \geq 1$ admits a $+O(WD)$ emulator of size $\widetilde{O}(n^{1+1/(2^{k+1}-1)} + n^{4/3}/(D^{4/3} + 2/(3k)))$.*

Theorem 1.5 serves as the key technical step in providing the improved additive stretch vs. size bounds in almost the entire regime of parameters (see Theorem 1.3). In particular, by using a suitable pre-sampling of a net graph G' and applying Thm. 1.5 on G', we obtain linear-size emulator with $+\widetilde{O}(n^{25/112})$ stretch. Moreover, for any $\epsilon > 1/7$, Thm. 1.5 allows us to provide an $+n^{f(\epsilon)}$ emulator with $n^{1+\epsilon}$ edges, where $f(\epsilon) = 1/5 - 3\epsilon/5$. This improves the state-of-the-art bounds of $1/4 - 3\epsilon/4$ due to [18].

[3] One can make it linear by reducing the sampling probability an $O(\log n)$ factor.

We prove 1.5 by presenting a three-step algorithm. The first step (which takes care of the short distances) include a weighted variant of the TZ emulators which for integer stretch k provides $f(d)$-emulator with $\widetilde{O}(n^{1+1/(2^{k+1}-1)})$ edges and $f(d) = d + d^{1-1/k}W^{1/k}$ for $d \geq W$ and $f(d) = O(W)$, otherwise. This variant can be obtained by a straightforward adaptation of the TZ construction to the weighted setting. In particular, setting $W = 1$ recovers the TZ bounds (see Thm. 1.8).

The second step is based on the useful tool of *light-initialization* introduced by Ahemd et al. [3] in the context of translating the existing constructions of additive spanners for unweighted graphs into suitable constructions for weighted graphs. For a given weighted graph G and integer parameter t, the t light-initialization is a subgraph H' of G containing the t-lightest (based on edge weight) edges[4] incident to each vertex in G. Ahemd et al. [3] provided a very elegant argument that in essence achieves the same net effect as obtained in the unweighted setting (where one simply adds t arbitrary edges per vertex): Specifically, the key property is that any u-v shortest path P that has misses ℓ edges in H' must contains $\Omega(t\ell)$ vertices that are incident to the vertices of P via the edges of H'. Our algorithm employs the light-initialization tool on sampled net G' of G, for a carefully chosen parameter t. Each G-vertx is sampled into the net G' with probability $\Theta(\log n/D)$. The third and last step further sub-samples the vertices of G' and adds the complete weighted graph on this sample to the output spanner.

The stretch analysis of this scheme has the following structure. First, using the TZ emulators allows us to satisfy the stretch for pairs at distance $O(WD\log n)$ in G. The focus is then on bounding the stretch for a pair of sampled vertices $u, v \in V(G')$. The argument considers a u-v shortest path P in G' and distinguishes between two cases: $|P \setminus H| \leq q$ for some chosen parameter q, and the complementarity case where $|P \setminus H| > q$. For the first case, we use the weighted-TZ spanner of G to obtain a sublinear stretch guarantee for every edge on $P \setminus H$, taking advantage of the fact that each such edge corresponds to a path in the original graph G. The benefit that we get from the sublinear stretch bounds allows us to accumulate it for each of the t missing edges.

To handle the complementary case where $|P \setminus H| > q$, we use H' to claim that the final sampled set V'' contains a pair u'', v'' that are sufficiently close to u and v. The stretch bound is provided by the addition of the edge (u'', v'') to the final emulator.

Advanced: $+\widetilde{O}(n^{0.222})$ Additive Stretch. Our last and most involved improvement performs a *root treatment* to the TZ emulator construction. Instead of using the weighted-TZ variant in a black-box manner on our weighted sampled graph, we provide a discretization variant for this algorithm in which we replace the continuous TZ stretch function by a step function. The latter provides worse bounds for nearby pairs, with the benefit of using fewer edges. More specifically, the construction is parameterized by integers D, k, p and show:

▶ **Theorem 1.6.** *For every n-vertex $G = (V, E, \omega)$ with maximum weight W, a constant integer $k \geq 3$, integer $D \geq 1$ and $p \in (0, 1)$, one can compute an emulator H with additive stretch $O(D^{1-1/(k-1)} \cdot W \log n/p)$ for any distance $d = O(D \cdot W \log n/p)$. The size of H is bounded by*

$$|H| = \widetilde{O}\left(n^{1+\frac{1}{2^{k+1}-1}} + (n \cdot p)^{1+\frac{1}{2^k-1}} / \left((1/p)^{\frac{2^k-2}{(2^k-1)k}} \cdot D^{\frac{2^k-2k}{(2^k-1)k(k-1)}}\right)\right) .$$

[4] Each vertex sorts its incident edges in increasing edge weight, and the t first edges in this ordering are taken.

Our optimized variant of the weighted TZ emulators is fitted to the setting where the given weighted graph provided as input to Theorem 1.6 is in fact a net graph G' that corresponds to some unweighted base graph[5] G. We then aim at exploiting the fact that the edges of G' corresponds to G-paths already in the construction of the weighted TZ emulators. To present our key ideas, we briefly describe the TZ algorithm. For a subset of vertices V' and probability q, let $V'[q]$ be the set of vertices obtained by sampling each vertex $v \in V'$ independently with probability of q.

For a given parameter k, the algorithm computes a hierarchy $V = V_0 \supset V_1 \supset V_2 \supset \ldots \supset V_{k-1}$ of levels, where $V_i = V_{i-1}[q_{i-1}]$ for $q_{i-1} = |V_{i-1}|/n^{1+1/(2^k-1)}$. For every vertex $v \in V_i$, its $(i+1)^{th}$ pivot $p_{i+1}(v)$ is the closest vertex to v in V_{i+1}. The bunch $B_i(v)$ contains all vertices in V_i that are closer to v than its pivot $p_{i+1}(v)$. The algorithm adds to the emulator the edges between each $v \in V_i$ to all vertices in its bunch $B_i(v)$. The weights of the edges are the G-distance of their endpoints. This is done for every $i \in \{1, \ldots, k-2\}$. Finally, all edges in $V_{k-1} \times V_{k-1}$ are added to the emulator.

Our adaptation to weighted net graphs G' (whose edges correspond to paths in a base graph G) computes a hierarchy of $2(k-1)$ levels: $V = V_{1/2} \supseteq V_1 \supseteq V_{3/2} \supseteq V_2 \supseteq \ldots \supseteq V_{k-1/2}$, where $V_{(j+1)/2} \leftarrow V_{j/2}[q_j]$ for every $j \in \{1, \ldots, 2(k-1)\}$. Hence, we have $k-1$ integral levels $V_1, \ldots, V_{k-1}$ and k "half"-levels $V_{1/2}, V_{3/2}, \ldots, V_{k-1/2}$. Intuitively, the "half" levels represent an intermediate step that re-scales the "aggregate" benefit obtained by the existence of the precomputed emulator H_0 that takes care of the short distances in G'. The selection of the sampling probabilities are made in a careful manner that depend on the properties of H_0. Once the hierarchy is computed, we have $k-1$ steps which mimic the TZ algorithm with one main distinction, we add to the emulator edges from the *half*-level $V_{i+1/2}$ to the next *integral*-level V_{i+1}.

That is, for every $i \in \{0, \ldots, k-2\}$ and every $u \in V_{i+0.5}$, the algorithm computes a pivot $p_{i+1}(v)$ (closest vertex in V_{i+1}) and a bunch $B_{i+0.5}(u)$, which consists of all $V_{i+0.5}$ vertices that are closer to u than its pivot $p_{i+1}(u)$. The edges in $\{u\} \times B_{i+0.5}(u)$ are added to the emulator. Finally, in the last half level $k-0.5$, we add all edges in $V_{k-0.5} \times V_{k-0.5}$.

Remark. We note that our approach for computing improved linear emulators of Thm. 1.2 can also be used to improve the general tradeoff provided in Thm. 1.3. The total improvement, however, is limited to a small $o(1)$ additive term, and therefore we make this extra effort only for linear size emulators. We also note that our approach for the latter could be further optimized by considering a large number of recursive sampling steps, but again the net effect on the stretch is negligible (in particular, an additional sampling step might reduce the stretch by an 0.0001 additive term).

Notations. For a possibly weighted graph G, let $\text{dist}_G(u, v)$ be the *length* of a shortest path from u to v. The length of a shortest-path Q is measured by the sum of its weighted edges. Let $|Q|$ be the number of edges on this path. We use $\widetilde{O}(\cdot)$ notation to hide polylogarithmic factors in n. For a set of elements X and $p \in [0, 1]$, let $X[p]$ be the set obtained by sampling each X-element independently with probability p.

For a given (possibly) weighted graph G and integer t, a subgraph $H \subseteq G$ is a *t-spanner* if $\text{dist}_H(u, v) \leq t \cdot \text{dist}_G(u, v)$ for every $u, v \in V$. Our constructions use the following algorithm as a subroutine, mainly for $t = O(\log n)$.

[5] I.e., in our constructions, the graph G provided as input to Theorem 1.6 is in fact a net graph G' of some base graph G.

▶ **Lemma 1.7** ([6]). *For every n-vertex (possibly weighted) graph G and a given integer $k \geq 1$, one can compute a $(2k-1)$-spanner $H \subseteq G$ with $|H| \leq n^{1+1/k}$ edges.*

▶ **Theorem 1.8** ([19]). *For every n-vertex unweighted graph G and a given integer $k \geq 1$, one can compute an emulator H with $\widetilde{O}(n^{1+1/(2^{k+1}-1)})$ edges, such that for every $u, v \in V$, it holds that $\mathrm{dist}_G(u,v) \leq \mathrm{dist}_H(u,v) \leq \mathrm{dist}_G(u,v) + (\mathrm{dist}_G(u,v))^{1-1/k}$.*

Roadmap. In Sec. 2, we present a simple approach to recover the state-of-the-art bound of $+\widetilde{O}(n^{1/4})$ additive emulator. Sec. 3 provides an improved emulator construction for the entire regime, proving Theorem 1.5 and consequently also Thm. 1.3. Finally, in Sec. 4 we provide the proof of the key result, Thm. 1.2.

2 Warmup: $+\widetilde{O}(n^{1/4})$ Linear Emulators

We start by presenting a simple construction of linear size $+\widetilde{O}(n^{1/4})$-emulators, which uses the following theorem for weighted additive spanners by [4] (recently improved by [12]).

▶ **Theorem 2.1** (Theorem 3 in [4]). *Any n-vertex weighted graph $G = (V, E, \omega)$ with max edge weight W admits a $+8W$ additive spanner $H \subseteq G$ with $O(n^{4/3})$ edges.*

Algorithm. The algorithm for computing $+\widetilde{O}(n^{1/4})$-emulator has two steps. The first step computes a $O(\log n)$-multiplicative spanner $H_1 \subseteq G$, which as we show later handles the short distances in G. The second step computes a net graph $G' = (V', E', \omega')$ defined over a sampled subset $V' = V[p]$ for $p = \log n / n^{1/4}$. The edge set E' consists of all pairs in $V' \times V'$ whose distance in G is at most $n^{1/4}$. The weights of the E' are taken to be the G-distances. Formally, $E' = \{(x,y) \in V' \times V' \mid \mathrm{dist}_G(x,y) \leq n^{1/4}\}$, $\omega((x,y)) = \mathrm{dist}_G(x,y), \forall (x,y) \in E'$. Note that, by definition, the maximum weight W' of G' is $O(n^{1/4})$. The algorithm then applies Theorem 2.1 to compute $+8W'$ emulator H_2 for G'. The output emulator is given by $H = H_1 \cup H_2$. This completes the description of the algorithm.

The size analysis is immediate as w.h.p. $|V'| = O(n^{3/4} \log n)$ and thus by Theorem 2.1 $|H_2| = \widetilde{O}(n)$. We now consider the stretch argument. Fix $u, v \in V$. Assume first that $\mathrm{dist}_G(u,v) \leq c \cdot n^{1/4}$ for some constant c. Then, by including the $O(\log n)$-multiplicative spanner H_1, we have that $\mathrm{dist}_H(u,v) = O(n^{1/4} \log n)$, as desired.

Consider the complementary case where $\mathrm{dist}_G(u,v) > c \cdot n^{1/4}$, and let P be a u-v shortest path in G. Let P', P'' be the $n^{1/4}$-length prefix (resp., suffix) of P. By the Chernoff bound, w.h.p., we have that there exists a sampled vertex $u' \in P' \cap V'$ and $v' \in P'' \cap V'$. By the previous argument (for short distances), it remains to show that $\mathrm{dist}_H(u', v') \leq \mathrm{dist}_G(u', v') + O(n^{1/4} \cdot \log n)$.

Observe that since every $n^{1/4}$-length consecutive segment on P contains, w.h.p., a sampled vertex in V', we have that $\mathrm{dist}_{G'}(u', v') = \mathrm{dist}_G(u', v')$. By the properties of H_2, we then have that $\mathrm{dist}_{H_2}(u', v') \leq \mathrm{dist}_{G'}(u', v') + 8W' = \mathrm{dist}_G(u', v') + 8n^{1/4}$. Overall, we have

$$\begin{aligned} \mathrm{dist}_H(u,v) &\leq \mathrm{dist}_{H_1}(u,u') + \mathrm{dist}_{H_2}(u',v') + \mathrm{dist}_{H_1}(v',v) \\ &\leq O(\log n)(\mathrm{dist}_G(u,u') + \mathrm{dist}_G(v',v)) + \mathrm{dist}_G(u',v') + 8n^{1/4} \\ &= \mathrm{dist}_G(u,v) + O(n^{1/4} \cdot \log n) \ . \end{aligned}$$

3 New Weighted Additive Emulators

3.1 The Core Construction

We start by presenting the key construction which for n-vertex weighted graphs provides emulators with $+O(WD)$ stretch and with $O(n^{4/3}/f(D))$ edges, for some monotone increasing function $f(\cdot)$. These emulators serve the basis for improved emulator constructions in wide range of parameters, and in particular computing linear emulators with improved additive stretch $+n^{0.222}$. We show:

▶ **Theorem 3.1.** *There is an algorithm* SuperLinEmulator *that given any n-vertex graph $G=(V,E,\omega)$ with maximum weight W, and integers $k\geq 2, D\geq 1$ computes a $+O(WD)$ emulator H of size $\widetilde{O}\left(n^{1+1/(2^{k+1}-1)}+\frac{n^{4/3}}{D^{4/3+2/(3k)}}\right)$.*

We start by presenting the two main tools used by Algorithm SuperLinEmulator.

Tool I: Weighted Near-Additive Emulator. We used the following adaptation of the Thorup and Zwick emulators to the weighted setting. An adaptation for universal emulators has been recently provided by Elkin, Gitlitz and Neiman [13]. In the full version, we show:

▶ **Lemma 3.2.** *There is an algorithm* WeightedTZEmulator *that for any n-vertex graph $G=(V,E,\omega)$ with maximum weight W, and any fixed integer $k\geq 2$ computes an $+f(d)$ emulator H of size $O(n^{1+1/(2^{k+1}-1)})$, where $f(d)=d+O(d^{1-1/k}\cdot W^{1/k})$ for any distance $d>W$, and $f(d)=d+O(W)$ for $d\leq W$.*

Tool II: Light Initialization. A t-light initialization of a weighted graph $G=(V,E,\omega)$, introduced by Ahmed et al. [4], is a subgraph $H\subseteq G$ obtained by including the t lightest edges incident to each vertex v (or all its edges when $\deg(v)\leq t$). Edge weight ties can be broken arbitrarily; Let Initialization be the algorithm that given the graph G and a parameter t, outputs the t-light initialization subgraph H. We say that v is a t-light neighbor of u if the edge (u,v) is among the t-lightest edges incident on u.

▶ **Theorem 3.3** (Theorem 5 in [4])**.** *Let $G=(V,E,\omega)$ be an undirected weighted graph and let $H=$ Initialization(G,t) for some input integer t. Then, for every shortest path $P_{u,v}$ that is missing ℓ edges in H (i.e., $|P_{u,v}\setminus H|=\ell$), there is a set of vertices $S\subseteq V$ such that (i) $|S|=\Omega(t\cdot\ell)$ and (ii) for every $a\in S$, there is a vertex $b\in P_{u,v}$ satisfying that a is a t-light neighbor of b.*

Tool III: Algorithm Net. Given an n-vertex weighted graph $G=(V,E,\omega)$ with maximum edge weight W and a probability $p\in(0,1)$, the algorithm Net(G,p) outputs a graph $G'=(V',E',\omega')$, denoted as a *net*, defined as follows. Let $V'=V[p]$ be a random sample of V, obtained by sampling each $v\in V$ independently with probability of p. Let $E'=\{(u,v)\in V'\times V' \mid \mathrm{dist}_G(u,v)\leq\Theta(\log n/p)\cdot W\}$ and $\omega'((u,v))=\mathrm{dist}_G(u,v)$ for every $(u,v)\in E'$. We use the following observation in our constructions:

▶ **Observation 3.4.** *Let $G'=(V',E',\omega')$ be the output net graph of Alg.* Net(G,p) *where $G=(V,E,\omega)$ is an n-vertex graph with maximum edge weight W. Then w.h.p., the following holds: (i) $|V'|=O(np\log n)$, (ii) for every $u,v\in V'$, $\mathrm{dist}_{G'}(u,v)=\mathrm{dist}_G(u,v)$, and (iii) the maximum edge weight of G' is bounded by $W'=\Theta(W\log n/p)$.*

Description of Alg. SuperLinEmulator. The algorithm has three main steps, each computes an emulator graph H_1, H_2, H_3 whose union provides the desired emulator. The first emulator H_1 is obtained by computing the weighted-variant of the Thorup-Zwick emulator using Lemma 3.2. As we will see in the analysis, this would provide the desired stretch for short distances. The second emulator H_2 is obtained by computing the t-initialization of some net graph G_2 for $t = n^{1/3}/D^{(1+2/k)/3}$. The net G_2 is defined by sampling a subset of vertices $V_2 = V[p_1]$ for $p_1 = 10 \log n/D$. The edges E_2 of the net G_2 are defined by connecting each pair $(u, v) \in V_2 \times V_2$ provided that $\text{dist}_G(u, v) \leq WD$, every edge (u, v) in G_2 is then weighted by the G-distance between its endpoints. Finally, the last emulator graph H_3 is obtained by adding all the weighted edges between a sampled set $V_3 = V_2[p_2]$ for $p_2 = 10 \log n/(t \cdot D^{1/k})$. The weights are taken to be the G-distances between the endpoints. This completes the algorithmic description.

Algorithm 1 SuperLinEmulator(G, k, D).

Input: Graph $G = (V, E, \omega)$ with maximum edge weight W, integers k, D.
Output: A $+O(W \cdot D)$ emulator H of size $\widetilde{O}\left(n^{1+1/(2^{k+1}-1)} + \frac{n^{4/3}}{D^{4/3+2/(3k)}}\right)$.

1. $H_1 \leftarrow$ WeightedTZEmulator(G, k) (using Lemma 3.2).
2. Let $G_2 = (V_2, E_2, \omega_2) \leftarrow$ Net(G, p_1) for $p_1 = 10 \log n/D$.
3. $H_2 \leftarrow$ Initialization(G_2, t) for $t = n^{1/3}/D^{(1+2/k)/3}$ (using Thm. 3.3).
4. Let $V_3 \leftarrow V_2[p_2]$ for $p_2 = 10 \log n/(t \cdot D^{1/k})$;
5. Set $H_3 \leftarrow (V_3, V_3 \times V_3, \omega_3)$ where $\omega_3((u, v)) = \text{dist}_G(u, v)$ for every $(u, v) \in H_3$.
6. Output $H \leftarrow H_1 \cup H_2 \cup H_3$.

Size analysis. By Lemma 3.2, $|H_1| = O(n^{1+1/(2^{k+1}-1)})$. By the Chernoff bound, w.h.p $|V_2| = n \cdot p_1$ and $|H_2| = t \cdot |V_2| = \widetilde{O}\left(\frac{n^{4/3}}{D^{4/3+2/(3k)}}\right)$. Finally, by the Chernoff bound, w.h.p, $|V_3| = |V_2| \cdot p_2$, and as $|H_3| = |V_3|^2$, we also get that $|H_3| = \widetilde{O}\left(\frac{n^{4/3}}{D^{4/3+2/(3k)}}\right)$.

Stretch analysis. We prove the following somewhat stronger lemma.

▶ **Lemma 3.5.** *Let H' be an emulator for G with maximum edge weight W such that for any u, v pair at G-distance at most WD, it holds that $\text{dist}_{H'}(u, v) \leq \text{dist}_G(u, v) + O(WD^{1-1/k})$. Then, for* every $u, v \in V$*, it holds that $\text{dist}_{H' \cup H_2 \cup H_3}(u, v) \leq \text{dist}_G(u, v) + O(WD)$.*

By Lemma 3.2, we then have that $\text{dist}_{H_1}(u, v) \leq \text{dist}_G(u, v) + O(WD^{1-1/k})$ for every u, v pair at G-distance at most WD, hence by taking $H' = H_1$, the stretch argument holds.

Proof of Lemma 3.5. Fix a pair $u, v \in V$ and first consider the simpler case where $\text{dist}_G(u, v) \leq WD$. By the properties of H', $\text{dist}_{H'}(u, v) \leq O(WD)$. From now on, we assume that $\text{dist}_G(u, v) > WD$. Let $P_{u,v}$ be the u-v shortest path in G, and let u', v' be a sampled vertex in V_2 in the $D/4$-hop prefix (resp., suffix) of the path. By the above, we have that $\text{dist}_G(u, u') \leq WD$ and $\text{dist}_G(v', v) \leq WD$, and therefore H' provides an additive $+O(WD)$ term for each of these distances.

Our next goal is to bound the u'-v' distance in H where $u', v' \in V_2$. It is easy to see that w.h.p., $\text{dist}_{G_2}(u', v') = \text{dist}_G(u', v')$, since each D-hop segment on the $P_{u,v}$ path contains a sampled vertex in V_2. Let P' be a u'-v' shortest path in G_2. We distinguish between two cases depending on the number of edges in $P_2 \setminus H_2$.

Case 1: $|P' \setminus H_2| \leq D^{1/k}$. Each edge (x, y) in $P' \subseteq G_2$ corresponds to an x-y shortest path where $\text{dist}_G(x, y) \leq WD$. Using the H' emulator, we have that for each such edge $(x, y) \in G_2$, $\text{dist}_{H'}(x, y) \leq \text{dist}_G(u, v) + O(D^{1-1/k} \cdot W)$. Since there are at most $D^{1/k}$, the total additive stretch introduced due to these edges is bounded by $O(D^{1/k} \cdot D^{1-1/k} \cdot W) = O(WD)$, as required. This is the critical point where we exploit the fact that the weighted edges of G_2 correspond to short paths in G.

Case 2: $|P' \setminus H_2| > D^{1/k}$. We next turn to consider the case where H_2 misses many edges from P'. Here we will exploit the expansion property guaranteed by the addition of the t-light initialization. Let P_1 (resp., P_2) be a prefix (resp., suffix) of P' for which H_2 misses exactly $D^{1/k}/2$. I.e., $|P_i \setminus H_2| = D^{1/k}/2$ for $i \in \{1, 2\}$. By Theorem 3.3 the following claims holds for every $i \in \{1, 2\}$: There exists a subset $S_i \subseteq V_2$ such that (i) $|S_i| = \Omega(t \cdot D^{1/k})$ and (ii) for every $a \in S_i$, there is a vertex $b_i \in P_i$ such that a is t-light neighbor of b_i. By the value of p_2, we get that w.h.p., there exists $s_i \in S_i \cap V_3$ for every $i \in \{1, 2\}$. Therefore, the emulator H_3 contains the edge (s_1, s_2) with weight $\text{dist}_G(s_1, s_2)$.

Since the maximum edge weight in G_2 is at most $W_2 = WD$, and $(b_i, s_i) \in H_2$, we have that $\text{dist}_H(b_1, s_1) + \text{dist}_H(s_2, b_2) = O(WD)$. By the triangle inequality,

$$\text{dist}_G(s_1, s_2) \leq \text{dist}_G(b_1, b_2) + O(WD) \ . \tag{3.1}$$

Since the segments $P'[u, b_1]$ and $P'[b_2, v]$, each has at most $D^{1/k}$ missing edges in H. Therefore, by applying the argument for Case 1, we have:

$$\text{dist}_H(u, b_i) \leq \text{dist}_G(u, b_i) + O(WD), \text{ for } i \in \{1, 2\} \ . \tag{3.2}$$

We are now ready to complete the stretch argument by showing:

$$\begin{aligned} \text{dist}_H(u', v') &\leq \text{dist}_H(u', b_1) + \text{dist}_H(b_1, s_1) + \text{dist}_H(s_1, s_2) + \text{dist}_H(s_2, b_2) + \text{dist}_H(b_2, v') \\ &\leq \text{dist}_G(u', b_1) + \text{dist}_G(b_1, b_2) + \text{dist}_G(b_2, v') + O(WD) \ , \end{aligned}$$

where the inequalities follow by plugging Eq. (3.1,3.2), and using the fact that as $(s_1, s_2) \in H_3$, by the triangle inequality $\text{dist}_H(s_1, s_2) = \text{dist}_G(s_1, s_2) \leq \text{dist}_G(b_1, b_2) + O(WD)$. ◀

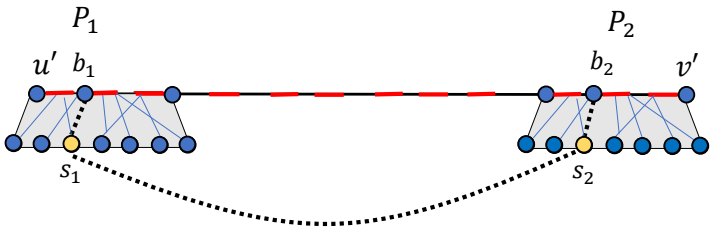


Figure 1 An illustration for the stretch argument of Alg. SuperLinEmulator. Shown is a u'-v' shortest path $P' \subseteq G_2$, the segments P_1, P_2 each containing $D^{1/k}/2$ missing edges w.r.t H_2. By the properties of the t-initialization procedure, each these segments contains a vertex b_1, b_2 with at least one sampled t-light neighbor, s_1, s_2. The added weighted edge (s_1, s_2) establishes the stretch guarantees.

3.2 Improved Additive Emulators

Our improved additive stretch bounds are provided by using Theorem 3.1 with two sparsity bounds determined by $k = 2, 3$. We have:

▶ **Corollary 3.6.** *For any n-vertex graph $G = (V, E)$ with max weight W, there exists a:*
1. *$+O(W \cdot n^{4/35})$ emulator of size $\widetilde{O}(n^{8/7})$, and*
2. *$+O(W \cdot n^{6/35})$ emulator of size $\widetilde{O}(n^{16/15})$.*

Proof. (1) follows by setting $k = 2$ and $D = n^{4/35}$ in Theorem 3.1, and (2) follows by setting $k = 3$ and $D = n^{6/35}$ in Theorem 3.1. ◀

Using these two emulator constructions, we show an improved stretch vs. size tradeoff in almost the entire regime of interest.

Proof of Thm. 1.3 for $0 \leq \epsilon \leq 1/15$ and $3/37 \leq \epsilon \leq 1/7$. We describe Algorithm ImprovedEmulator which given $G = (V, E, \omega)$ and $\epsilon \in [0, 1/15] \cup (3/37, 1/7]$, computes the desired emulator. The algorithm starts by computing a $O(\log n)$ multiplicative spanner H_0, which as always, takes care of the short distances in G. Next, the algorithm computes a net graph G' whose bounds depends on the value of ϵ, as follows. Define:

$$k_\epsilon = \begin{cases} 3, & \text{for } \epsilon \in [0, 1/15], \\ 2, & \text{for } \epsilon \in (3/37, 1/7] \ . \end{cases} \tag{3.3}$$

Let $n_\epsilon = n^{(1-1/2^{k_\epsilon+1})(1+\epsilon)}$ and $q = n/n_\epsilon$. Then, the net graph G' is obtained by applying Alg. $\mathsf{Net}(G, p)$ for $p = 10 \log n/q$. Finally, it applies Alg. SuperLinEmulator with the input G', k_ϵ and $D = (n_\epsilon)^{2k_\epsilon/35}$. This results in the emulator H_1. The output emulator is given by $H = H_0 \cup H_1$.

Algorithm 2 ImprovedEmulator(G, ϵ).

Input: Graph $G = (V, E, \omega)$ with maximum weight W, $\epsilon \in [0, 1/15] \cup (3/37, 1/7]$.
Output: $+\widetilde{O}(W \cdot n^{f(\epsilon)})$ emulator H of size $\widetilde{O}(n^{1+\epsilon})$.
1. $H_0 \leftarrow \mathsf{MultSpanner}(G, O(\log n))$.
2. Let $n_\epsilon = n^{(1-1/2^{k_\epsilon+1})(1+\epsilon)}$ and $q = n/n_\epsilon$ (see Eq. (3.3)).
3. $(G' = (V', E', \omega')) \leftarrow \mathsf{Net}(G, p)$ for $p = 10 \log n/q$.
4. $H_1 \leftarrow \mathsf{SuperLinEmulator}(G', k_\epsilon, D)$ for $D = (n_\epsilon)^{2k_\epsilon/35}$.
5. Output $H_0 \cup H_1$.

Analysis. We start with a stretch argument for a fixed pair $u, v \in V$. First, assume the more interesting case where $u', v' \in V'$. By the properties of H_1, the additive stretch is:

$$\widetilde{O}(D \cdot q \cdot W) = \widetilde{O}(n_\epsilon)^{2k_\epsilon/35} \cdot n^{1/2^{k_\epsilon+1}} \cdot n^{\epsilon(1/2^{k_\epsilon+1}-1)} \cdot W) = \widetilde{O}(W \cdot n^{f(\epsilon)}) \ . \tag{3.4}$$

Next assume that $\mathrm{dist}_G(u, v) \leq W \cdot q$. By adding the multiplicative spanner H_0, we have $\mathrm{dist}_{H_0}(u, v) \leq O(Wq \log n)$. Finally, assume that $\mathrm{dist}_G(u, v) \geq Wq$ and let $u', v' \in V'$ be the closest sampled vertex to u (resp., v) on the u-v shortest path. W.h.p., $\mathrm{dist}_G(u, u'), \mathrm{dist}_G(v, v') \leq Wq$ and therefore, $\mathrm{dist}_{H_0}(u, u'), \mathrm{dist}_{H_0}(v, v') \leq O(Wq \log n)$. Since w.h.p. $\mathrm{dist}_G(u', v') = \mathrm{dist}_{G'}(u', v')$, the stretch argument is completed by Eq. (3.4). The size bound follows by plugging $|H_0| = \widetilde{O}(n)$, and moreover, $|H_1| = \widetilde{O}(n^{1+\epsilon})$ by Corollary 3.6. We are now ready to complete the proof for the missing regimes.

Complete proof of Thm. 1.3. For the range $1/7 \leq \epsilon \leq 1/3$, the proof follows by letting $H = \mathsf{SuperLinEmulator}(G, k = 2, D)$ for $D = n^{(1-3\epsilon)/5}$. For the range $1/15 \leq \epsilon \leq 3/37$, the proof follows by letting $H = \mathsf{SuperLinEmulator}(G, k = 3, D)$ for $D = n^{(3-9\epsilon)/14}$. ◀

4 $+n^{0.222}$ Emulator of Linear Size

4.1 An Optimized Weighted Thorup-Zwick Emulator

In this section, we present an optimized variant of Thorup-Zwick that plays a key role in the construction of our linear additive emulator. We prove the following theorem which in particular implies Thm. 1.4.

▶ **Theorem 4.1.** *For every n-vertex $G = (V, E, \omega)$ with maximum weight W, a constant integer $k \geq 3$, integer $D \geq 1$ and $p \in (0, 1)$, there is an Algorithm* ImprovedTZEmulator *for computing an emulator H with additive stretch $O(D^{1-1/(k-1)} \cdot W \log n/p)$ for any distance $d = O(D \cdot W \log n/p)$. The size of H is bounded by*

$$|H| = \widetilde{O}\left(n^{1+\frac{1}{2^{k+1}-1}} + (n \cdot p)^{1+\frac{1}{2^k-1}} / \left((1/p)^{\frac{2^k-2}{(2^k-1)k}} \cdot D^{\frac{2^k-2k}{(2^k-1)k(k-1)}}\right)\right) .$$

We can also show interesting corollaries of Thm. 4.1 which demonstrate the sub-optimality of the TZ construction for a wide-range of distances. For example, the following holds:

▶ **Corollary 4.2.** *Every n-vertex unweighted graph G and given integer $k \geq 1$ admits an emulator H of size $\widetilde{O}(n^{1+1/(2^{k+1}-1)})$ such that pairs at distances $d \geq n^{k^2/2^k}$ have additive stretch of $O(d^{1-1/(k-1)})$.*

This should be compared with the additive stretch of $O(d^{1-1/k})$ provided by the TZ emulator (which also marks the state-of-the-art bounds). Thus, while the original TZ emulator is optimal for small distances as proven in [2], this optimality holds in a restricted range of distances, especially for non-constant values of k, e.g., $k = O(\log \log n)$. We now turn to prove Thm. 4.1 which constitutes the key technical contribution in the linear emulator construction.

Algorithm ImprovedTZEmulator. The algorithm starts by applying our weighted-variant of the Thorup-Zwick emulator to obtain $H_1 \leftarrow \mathsf{WeightedTZEmulator}(G, k)$, see Lemma 3.2. Next, it computes a net $G' = \mathsf{Net}(G, p)$ obtained by sampling each vertex in V into the net G' independently with probability p. By Obs. 3.4, we have that the maximum edge weight G' is bounded by $W' = \Theta(\log n \cdot W/p)$. In addition, w.h.p. it also holds that the G'-distances equal to the G-distances. The key technically involved step is in the computation of an additional emulator that we denoted by H_2 for G'. This emulator is computed by applying a new variant of the TZ emulator which takes advantageous of the fact that each weighted edge in G' corresponds to some path in a prior graph G, and more specifically, that there is a precomputed emulator (namely, H_1) that handles short distances in G'.

The construction builds a hierarchy $V = V_{1/2} \supseteq V_1 \supseteq V_{3/2} \supseteq V_2 \supseteq \ldots \supseteq V_{k-1/2}$, where $V_{(j+1)/2} \leftarrow V_{j/2}[q_j]$ for every $j \in \{1, \ldots, 2(k-1)\}$. Note that in contrast to the classic TZ emulator construction, our hierarchy has $2k-1$ levels: $k-1$ integral levels $V_1, \ldots, V_{k-1}$ and k "half"-levels $V_{1/2}, V_{3/2}, \ldots, V_{k-1/2}$. Intuitively, the "half" levels represent an intermediate

step that re-scales the extra-benefit obtained by the existence of the emulator H_1 (that takes care of short distances in G'). The definition of the sampling probabilities q_j is somewhat more involved compared to that of the classic construction. To define these probabilities, we need the following function definitions, for every integer $0 \leq i \leq k$:

$$h(i) = 1 - \frac{2^i - 1}{2^k - 1}, \qquad f(i) = \frac{2(2^k-1)i - 2k(2^i-1) - 2^k + 2^i}{(2^k-1)k} \quad \text{and} \quad g(i) = \frac{2^k - 2^i}{(2^k-1)k} \ . \tag{4.1}$$

The probabilities q_j for $j \in \{2i, 2i+1\}$ are chosen in order to satisfy the following, w.h.p., for every $i \in \{1, \ldots, k-1\}$:

$$|V_i| = \widetilde{O}\left(n^{h(i)} \cdot \Delta^{f(i)} \cdot \left(\frac{W'}{W}\right)^{g(i)}\right) \quad \text{and} \quad |V_{i+1/2}| = \widetilde{O}\left(|V_i| / \left(\Delta^{(i-1)/k} \cdot \left(\frac{W'}{W}\right)^{1/k}\right)\right) \ . \tag{4.2}$$

The sampling probabilities $q_{j=2i}$ for every $i \in \{1, \ldots, k-1\}$ have a simple to state expression:

$$q_{2i} = \Theta\left(\frac{\log n}{\Delta^{(i-1)/k} \cdot \left(\frac{W'}{W}\right)^{1/k}}\right) \ . \tag{4.3}$$

Let us give a concrete example for $k = 4$: For ease of notation, let $\widehat{W} = W'/W$.

1. $|V_{0.5}| = n$.
2. $|V_1| = \widetilde{O}\left(n^{14/15} \cdot \Delta^{2/15} \cdot \left(\widehat{W}\right)^{7/30}\right)$.
3. $|V_{1.5}| = \widetilde{O}\left(|V_1|/(\widehat{W})^{1/4}\right) = \widetilde{O}(n^{14/15} \cdot \Delta^{2/15} \cdot \left(\widehat{W}\right)^{7/30-1/4}\right)$.
4. $|V_2| = \widetilde{O}\left(n^{12/15} \cdot \Delta^{6/15} \cdot \left(\widehat{W}\right)^{1/5}\right)$.
5. $|V_{2.5}| = \widetilde{O}\left(|V_2|/(\Delta \cdot \widehat{W})^{1/4}\right) = \widetilde{O}\left(n^{12/15} \cdot \Delta^{6/15-1/4} \cdot \left(\widehat{W}\right)^{1/5-1/4}\right)$.
6. $|V_3| = \widetilde{O}\left(n^{8/15} \cdot \Delta^{13/30} \cdot \left(\widehat{W}\right)^{2/15}\right)$.
7. $|V_{3.5}| = \widetilde{O}\left(|V_3|/(\Delta^2 \cdot \widehat{W})^{1/4}\right) = \widetilde{O}\left(n^{8/15} \cdot \Delta^{13/30-1/2} \cdot \left(\widehat{W}\right)^{2/15-1/4}\right)$.

Given the q_j's probabilities, the algorithm proceeds in a very similar manner to the TZ emulator algorithm, with one main emphasis: There are $k-1$ phases in which we add to the emulator edges from the *half*-level $V_{i+1/2}$ to the next *integral*-level V_{i+1}. That is, no edges are added between V_{i+1} to $V_{i+1.5}$. For every $i \in \{0, \ldots, k-2\}$ and every $u \in V_{i+0.5}$, the algorithm computes a pivot $p_{i+1}(u)$ and a bunch $B_{i+0.5}(u)$, as follows. The pivot $p_{i+1}(u)$ is the closest[6] vertex to u in the next integral-level, V_{i+1}. The bunch $B_{i+0.5}(u)$ consists of all vertices in $V_{i+0.5}$ that are strictly closer to u than its pivot $p_{i+1}(u)$. The edges in $\{u\} \times B_{i+0.5}[u]$ are added to the emulator H_2, weighted by their G'-distances (which by Obs. 3.4(ii) also equal to the G-distances). Finally, all edges between the vertices in the last-half level $V_{k-0.5}$ are also added to H_2. The output emulator is given by $H_1 \cup H_2$.

The analysis is deferred to the full version.

[6] As usual, we can assume that the shortest-paths are unique.

Algorithm 3 ImprovedTZEmulator(G, k, p, D).

Input: Graph $G = (V, E, \omega)$ with maximum weight W and parameters $k \geq 3, D \geq 1$, $p \in (0, 1)$.
Output: $+O(D^{1-1/(k-1)} \cdot W \log n/p)$ emulator H for pairs at distance $d = O(D \cdot W \log n/p)$

1. $H_1 \leftarrow$ WeightedTZEmulator(G, k) (using Lemma 3.2), $H_2 \leftarrow \emptyset$.
2. $(G' = (V', E', \omega')) \leftarrow$ Net(G, p).
3. Set $\Delta = D^{1/(k-1)}$ and $V_{0.5} = V$.
4. For $j \in \{1, \ldots, 2(k-1)\}$ do: $V_{(j+1)/2} \leftarrow V_{j/2}[q_j]$, where q_j is defined based on Eq. (4.2).
5. For $i = 0$ to $k-2$ do:
   - For every $u \in V_{i+0.5}$ do:
     a. $p_{i+1}(u) = CLOSEST(V_{i+1}, u)$.
     b. $B_{i+0.5}(u) \leftarrow \{v \in V_{i+0.5} \mid \text{dist}_{G'}(u, v) < \text{dist}(u, p_{i+1}(u))\}$.
     c. $B_{i+0.5}[u] \leftarrow B_{i+0.5}(u) \cup \{p_{i+1}(u)\}$.
     d. $H_2 \leftarrow H_2 \cup (\{u\} \times B_{i+0.5}[u])$.
6. $H_2 \leftarrow H_2 \cup (V_{k-0.5} \times V_{k-0.5})$.
7. $H = H_1 \cup H_2$.

4.2 Improved Linear Emulators

This section is devoted to the proof of Theorem 1.2. Let ModifiedSuperLinEmulator be the same algorithm as SuperLinEmulator only that we omit the computation of H_1 in Step (1). We next present Algorithm ImprovedLinearEmulator that computes the desired emulators, as follows:

Algorithm 4 ImprovedLinearEmulator.

Input: An *unweighted* graph $G = (V, E)$ on n vertices.
Output: $+\widetilde{O}(n^{2/9-1/1600})$ emulator H of size $\widetilde{O}(n)$.
1. $H_0 \leftarrow$ MultSpanner$(G, k = O(\log n))$.
2. Set $p_1 = 10 \log n/n^{1/32}$, $p_2 = 10 \log n/n^{21/1060}$ and $D = n^{723/4240}$.
3. $(G_1 = (V_1, E_1, \omega_1)) \leftarrow$ Net(G, p_1).
4. $H_1' \leftarrow$ ImprovedTZEmulator$(G_1, k = 4, p_2, D = n^{723/4240})$.
5. $(G_2 = (V_2, E_2, \omega_2)) \leftarrow$ Net(G_1, p_2).
6. $H_2 \leftarrow$ ModifiedSuperLinEmulator$(G_2, k = 3, D)$.
7. Output $H_0 \cup H_1' \cup H_2$.

Algorithm 5 ModifiedSuperLinEmulator(G, k, D).

1. Let $G_2 = (V_2, E_2, \omega_2) \leftarrow$ Net(G, p_1) for $p_1 = 10 \log n/D$.
2. $H_2 \leftarrow$ Initialization(G_2, t) for $t = n^{1/3}/D^{(1+2/k)/3}$ (using Thm. 3.3).
3. Let $V_3 \leftarrow V_2[p_2]$ for $p_2 = 10 \log n/(t \cdot D^{1/k})$;
4. Set $H_3 \leftarrow (V_3, V_3 \times V_3, \omega_3)$ where $\omega_3((u, v)) = \text{dist}_G(u, v)$ for every $(u, v) \in H_3$.
5. Output $H \leftarrow H_2 \cup H_3$.

Size analysis. Clearly, $|H_0| = \widetilde{O}(n)$. By Theorem 4.1, we have that:

$$|H'_1| = \widetilde{O}\left((n \cdot p_1)^{1+\frac{1}{2^{4+1}-1}} + (n \cdot p_1 \cdot p_2)^{1+\frac{1}{2^4-1}} / \left((1/p_2)^{\frac{2^4-2}{(2^4-1)4}} \cdot D^{\frac{2^4-2\cdot 4}{(2^4-1)4(4-1)}}\right)\right) .$$

Therefore, $|H'_1| = \widetilde{O}(n^{(1-1/32-21/1060)(16/15)-(21/1060)\cdot(14/60)-(723/4240)\cdot(8/(15\cdot 4\cdot 3))}) = \widetilde{O}(n^1)$. Finally, by the proof of Thm. 3.1, it holds that $|H_2| = O((|V_2|)^{4/3}/D^{4/3+2/(3k)})$. Therefore,

$$|H_2| = (n \cdot p_1 \cdot p_2)^{4/3}/D^{4/3+2/9} = \widetilde{O}(n^{(1-1/32-21/1060)\cdot(4/3)-(4/3+2/9)\cdot(723/4240)}) = \widetilde{O}(n) .$$

Stretch analysis. Let W_1 (resp. W_2) be the maximum edge weight of G_1 (reps., G_2). By Obs. 3.4, we have that $W_1 = \widetilde{O}(1/p_1)$, $W_2 = \widetilde{O}(W_1 \cdot (1/p_2))$. We show that the additive stretch is $O(W_2 \cdot D) = O(n^{2/9-1/1600})$. By Obs. 3.4, the G_1-distances and G_2-distances, w.h.p., equal to the G-distances.

Case 1: Consider first a vertex pair $u', v' \in V_2$, we shall compute the stretch argument for the pair u', v'. By Lemma 3.5 with $H' = H'_1$ and $W = W_2$ we have that the additive stretch of the pair u', v' is given by:

$$\widetilde{O}(D \cdot W_2) = O(n^{2/9-1/1600}) . \tag{4.4}$$

Case 2: $\mathrm{dist}_G(u, v) \leq W_2$. By adding the multiplicative spanner H_0, $\mathrm{dist}_{H_0}(u, v) \leq O(W_2 \log n)$.

Case 3: $\mathrm{dist}_G(u, v) > W_2$. Let $u', v' \in V_2$ be the closest sampled vertex to u (resp., v) on the u-v shortest path in G. By the Chernoff bound, w.h.p., $\mathrm{dist}_G(u, u'), \mathrm{dist}_G(v, v') \leq W_2$ and therefore, by Case 2, $\mathrm{dist}_{H_0}(u, u'), \mathrm{dist}_{H_0}(v, v') \leq O(W_2 \log n)$. By Obs. 3.4, w.h.p., $\mathrm{dist}_G(u', v') = \mathrm{dist}_{G_2}(u', v')$, and the stretch argument is completed by Eq. (4.4) of Case 1.

References

1 Amir Abboud and Greg Bodwin. The 4/3 additive spanner exponent is tight. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 351–361. ACM, 2016.

2 Amir Abboud, Greg Bodwin, and Seth Pettie. A hierarchy of lower bounds for sublinear additive spanners. *SIAM J. Comput.*, 47(6):2203–2236, 2018.

3 Abu Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen G. Kobourov, and Richard Spence. Graph spanners: A tutorial review. *Comput. Sci. Rev.*, 37:100253, 2020.

4 Abu Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Stephen G. Kobourov, and Richard Spence. Weighted additive spanners. In Isolde Adler and Haiko Müller, editors, *Graph-Theoretic Concepts in Computer Science – 46th International Workshop, WG 2020, Leeds, UK, June 24-26, 2020, Revised Selected Papers*, volume 12301 of *Lecture Notes in Computer Science*, pages 401–413. Springer, 2020.

5 Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM J. Comput.*, 28(4):1167–1181, 1999.

6 Ingo Althöfer, Gautam Das, David P. Dobkin, and Deborah Joseph. Generating sparse spanners for weighted graphs. In John R. Gilbert and Rolf G. Karlsson, editors, *SWAT 90, 2nd Scandinavian Workshop on Algorithm Theory, Bergen, Norway, July 11-14, 1990, Proceedings*, volume 447 of *Lecture Notes in Computer Science*, pages 26–37. Springer, 1990.

7 Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. Additive spanners and (alpha, beta)-spanners. *ACM Trans. Algorithms*, 7(1):5:1–5:26, 2010.

8 Greg Bodwin and Gary Hoppenworth. New additive spanner lower bounds by an unlayered obstacle product. *CoRR*, abs/2207.11832, 2022. doi:10.48550/arXiv.2207.11832.

9 Greg Bodwin and Virginia Vassilevska Williams. Better distance preservers and additive spanners. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 855–872. SIAM, 2016.

10 Gregory Bodwin and Virginia Vassilevska Williams. Very sparse additive spanners and emulators. In Tim Roughgarden, editor, *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 377–382. ACM, 2015.

11 Dorit Dor, Shay Halperin, and Uri Zwick. All pairs almost shortest paths. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 452–461. IEEE Computer Society, 1996.

12 Michael Elkin, Yuval Gitlitz, and Ofer Neiman. Improved weighted additive spanners. In Seth Gilbert, editor, *35th International Symposium on Distributed Computing, DISC 2021, October 4-8, 2021, Freiburg, Germany (Virtual Conference)*, volume 209 of *LIPIcs*, pages 21:1–21:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

13 Michael Elkin, Yuval Gitlitz, and Ofer Neiman. Almost shortest paths with near-additive error in weighted graphs. In Artur Czumaj and Qin Xin, editors, *18th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2022, June 27-29, 2022, Tórshavn, Faroe Islands*, volume 227 of *LIPIcs*, pages 23:1–23:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

14 Michael Elkin and Shaked Matar. Ultra-sparse near-additive emulators. In Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen, editors, *PODC '21: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, July 26-30, 2021*, pages 235–246. ACM, 2021.

15 Michael Elkin and David Peleg. (1+epsilon, beta)-spanner constructions for general graphs. In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 173–182. ACM, 2001.

16 Shang-En Huang and Seth Pettie. Lower bounds on sparse spanners, emulators, and diameter-reducing shortcuts. *SIAM J. Discret. Math.*, 35(3):2129–2144, 2021. doi:10.1137/19M1306154.

17 Kevin Lu, Virginia Vassilevska Williams, Nicole Wein, and Zixuan Xu. Better lower bounds for shortcut sets and additive spanners via an improved alternation product. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9–12, 2022*, pages 3311–3331. SIAM, 2022.

18 Seth Pettie. Low distortion spanners. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, volume 4596 of *Lecture Notes in Computer Science*, pages 78–89. Springer, 2007.

19 Mikkel Thorup and Uri Zwick. Spanners and emulators with sublinear distance errors. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 802–809. ACM Press, 2006.

20 David P. Woodruff. Lower bounds for additive spanners, emulators, and more. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 389–398. IEEE Computer Society, 2006.

Nearly-Linear Time LP Solvers and Rounding Algorithms for Scheduling Problems

Shi Li
State Key Laboratory for Novel Software Technology, Nanjing University, China

Abstract

We study nearly-linear time approximation algorithms for non-preemptive scheduling problems in two settings: the unrelated machine setting, and the identical machine with job precedence constraints setting, under the well-studied objectives such as makespan and weighted completion time. For many problems, we develop nearly-linear time approximation algorithms with approximation ratios matching the current best ones achieved in polynomial time.

Our main technique is linear programming relaxation. For the unrelated machine setting, we formulate mixed packing and covering LP relaxations of nearly-linear size, and solve them approximately using the nearly-linear time solver of Young. For the makespan objective, we develop a rounding algorithm with $(2+\epsilon)$-approximation ratio. For the weighted completion time objective, we prove the LP is as strong as the rectangle LP used by Im and Li, leading to a nearly-linear time $(1.45+\epsilon)$-approximation for the problem.

For problems in the identical machine with precedence constraints setting, the precedence constraints can not be formulated as packing or covering constraints. To achieve the nearly-linear running time, we define a polytope for the constraints, and leverage the multiplicative weight update (MWU) method with an oracle which always returns solutions in the polytope.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases Nearly-Linear Time, Sheduling, Approximation Algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.86

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2111.04897`

Funding *Shi Li*: Part of the work was supported by NSF-grant CCF-1844890.

1 Introduction

Scheduling theory is an important sub-area of combinatorial optimization, operations research and approximation algorithms. Over the past few decades, advanced techniques have been developed to design approximation algorithms for numerous scheduling problems, among which mathematical relaxation is a prominent one. The algorithms based on the technique follow a two-step framework: solve some linear/convex/semi-definite programming relaxation for the problem to obtain a fractional schedule, and round it into an integral one. The main focus of the algorithm design in the literature has been the best approximation ratios that can be achieved in polynomial time. Many of the LPs used have size much larger than that of the input, and a general convex/semi-definite program requires a large polynomial time to solve, making these algorithms impractical.

To overcome the running time issue, we design approximate LP-based scheduling algorithms that run in *nearly-linear* time. We focus on two well-studied non-preemptive scheduling settings:

1. **Unrelated machine setting.** We are given a set J of n jobs, a set M of m machines, a bipartite graph $G=(M,J,E)$ between M and J, and a processing time $p_{ij}\in\mathbb{Z}_{>0}$ for every $ij\in E$, indicating the time it takes to process job j on machine i. If $ij\notin E$, then

the job j can not be processed on machine i. The output of a problem in this setting is an assignment $\sigma \in M^J$ of jobs to machines so that $\sigma_j j \in E$ for every $j \in J$. This indicates that we process the job j on machine σ_j.

2. **Identical machine with job precedence constraints setting.** In this setting, we are given a set J of n jobs, each job $j \in J$ with a processing time $p_j \in \mathbb{Z}_{\geq 0}$, and the number $m \geq 1$ of identical machines. There are precedence constraints of the form $j \prec j'$, indicating that the job j' can only start after job j completes. The output of a problem in the setting is a completion time vector $(C_j)_{j \in J} \in \mathbb{Z}^J_{\geq 0}$, meaning that a job $j \in J$ is processed during the time interval $(C_j - p_j, C_j]$. We need $C_j \geq p_j$ for every $j \in J$, $C_j \leq C_{j'} - p_{j'}$ for every $j \prec j'$, and every integer $t \geq 1$ is contained in $(C_j - p_j, C_j]$ for at most m jobs $j \in J$.[1]

The main objective function we focus on is *weighted completion time*: We are additionally given a weight $w_j \in \mathbb{Z}_{>0}$ for every job $j \in J$, and the goal of the problem is to minimize $\sum_{j \in J} w_j C_j$, where C_j is the completion time of j on its assigned machine. For the second setting, this is explicitly given by the output. For the first setting, given the assignment $\sigma \in M^J$ of jobs to machines, it is well-known that the Smith's rule[2] gives the optimum order on each machine i. For the first setting, we also consider the objective of minimizing the *makespan*, which is defined as $\max_i \sum_{j \in \sigma^{-1}(i)} p_{ij}$, i.e., the maximum load over all machines.

It is convenient for us to use the classic three-field notation $\alpha|\beta|\gamma$ in [19] to denote scheduling problems studied in this paper.[3] The makespan and weighted completion time minimization problems in the unrelated machine setting are denoted as $R||C_{\max}$ and $R||\sum_j w_j C_j$ respectively. The problem to minimize weighted completion time in the identical machine with job precedence constraint setting is denoted as $P|\text{prec}|\sum_j w_j C_j$. We will also consider special cases of the problem, and give their notations when we discuss them.

There is a rich literature on designing approximation algorithms for these problems. For the unrelated makespan minimization problem, i.e., $R||C_{\max}$, the classic result of Lenstra, Shmoys and Tardos [32] gives a 2-approximation, which remains the state-of-the-art result. The problem is NP-hard to approximate within a factor of better than 1.5. Plotkin, Shmoys and Tardos [39] studied fast approximation algorithms for the problem, as an application of their packing and covering LP solver. They developed a randomized $(2+\epsilon)$-approximation algorithm in time $\tilde{O}_\epsilon(mn)$.[4] So their algorithm is nearly-linear if $|E| = \Theta(mn)$. Much work on the problem has focused on a special setting called the restricted assignment setting [49, 24, 25], where there is an intrinsic size $p_j \in \mathbb{Z}_{>0}$ for every $j \in J$, and for every $ij \in E$ we have $p_{ij} = p_j$.

For the unrelated machine weighted completion time problem, i.e., $R||\sum_j w_j C_j$, many independent rounding algorithms achieve an approximation ratio of 1.5 [42, 47, 43, 35]. Bansal, Svensson and Srinivasan [5] showed that the barrier of 1.5 is inherent for this type of algorithms. To overcome the barrier, they developed a novel dependent rounding scheme

[1] It is a folklore that if the last property is satisfied, we can assign $\{(C_j - p_j], j \in [J]\}$ to m machines so that the intervals assigned to each machine are disjoint.

[2] By this rule, we schedule jobs j assigned to a machine i using non-decreasing order of p_{ij}/w_j.

[3] In the notation, α indicates the machine model, β gives the set of additional constraints, and γ is the objective. $\alpha = R$ and $\alpha = P$ denote the unrelated and identical machine settings respectively, and $\text{prec} \in \beta$ indicates that jobs have precedence constraints. $\gamma = C_{\max}$ and $\gamma = \sum_j w_j C_j$ denote the makespan and weighted completion time objectives respectively.

[4] In this paper, we use $\tilde{O}_\epsilon(\cdot)$ to hide a factor that is poly-logarithmic in the input size of the instance being considered, which will be clear from the context, and polynomial in $1/\epsilon$, where ϵ is a precision parameter. An algorithm is nearly-linear if its running time is $\tilde{O}_\epsilon(\text{input size})$.

and a lifted SDP relaxation for the problem, leading to a $(1.5 - 1/2160000)$-approximation algorithm. The ratio has been improved to $1.5 - 1/6000$ by Li [35], to 1.488 by Im and Shadloo [23] and to the current best ratio of 1.45 by Im and Li [22]. The three subsequent works are based on the rectangle LP relaxation for the problem.

There is a vast literature on the problem of minimizing weighted completion time in the identical machine with job precedence constraints setting, i.e., the problem $P|\text{prec}|\sum_j w_j C_j$. A special case of the problem where there is only one machine (i.e., $m = 1$), denoted as $1|\text{prec}|\sum_j w_j C_j$, is already non-trivial. Hall et al. [20] developed a 2-approximation for the problem, which is the best possible under some stronger version of the unique game conjecture introduced by Bansal and Khot [4]. Another special case that is considered moderately in the literature is when all jobs have unit-size, denoted as $P|\text{prec}, p_j = 1|\sum_j w_j C_j$. Munier, Queyranne and Schulz [37] gave approximation ratios of 3 and 4 for the special case and the general problem $P|\text{prec}|\sum_j w_j C_j$ respectively. The ratios were improved to $1 + \sqrt{2}$ and $2 + 2\ln 2$ by Li [35]. Most algorithms [20, 37, 41, 35] for $P|\text{prec}|\sum_j w_j C_j$ and the two special cases use the following framework: Solve some linear/convex program to obtain an order of the jobs respecting the precedence constraints. For every job in this order, schedule it as early as possible, without violating the precedence and m-machine constraints.

Most of the results we discussed focused on optimizing the approximation ratios with polynomial time algorithms. Albeit being polynomial, the running times in these results are often very large. For LP-based algorithms, this may be caused by two factors. First, the size of an LP might already be large w.r.t the input size. Consider a typical time-indexed LP relaxation in the unrelated machine setting, one need a variable for every triple ijs with $ij \in E$ and s being the starting time. Assuming the number of possible starting times is linear in n, the number of variables in the LP is already $\Theta(n|E|)$; the size of the LP can only be bigger. Second, these algorithms often use a general LP solver, which has a large running time w.r.t the size of the LP. There is a vast literature in recent years on designing exact and approximate general LP solvers. Here we could only include a few representative results. To solve a linear program with $\bar{n}$ variables, $\bar{m}$ constraints and $\bar{N}$ non-zero coefficients up to a precision of ϵ, Lee and Sidford [29] developed an algorithm with running time $\tilde{O}\big((\bar{N} + \bar{m}^2)\sqrt{\bar{m}}\log\frac{1}{\epsilon}\big)$. Lee, Song and Zhang [30] gave an algorithm with running time $\tilde{O}(\bar{n}^\omega \log\frac{1}{\epsilon})$,[5] where $\omega \approx 2.373$ is the current best exponent for matrix multiplication. Brand, Lee, Sidford and Song [8] provided a $\tilde{O}(\bar{m}\bar{n} + \bar{n}^3)$ time randomized algorithm that solves the LP exactly with high probability; the running time is nearly linear if the constraint matrix is dense and tall. However, to solve general linear programs, these running times are at least quadratic, even if the LP has a linear size. Convex or semi-definite programming based algorithms need to solve the CP/SDP using the interior point or ellipsoid methods, which are often time-consuming.

1.1 Our Results

To overcome the above issue, we design approximation algorithms for scheduling problems, that run in *nearly-linear* time, i.e., in time $\tilde{O}_\epsilon(\text{input size})$. So, up to a $\text{poly}(\log n, 1/\epsilon)$-factor, our running times are the best possible. Some of the algorithms we developed have been studied empirically [2]. In the unrelated machine setting, $G = (M, J, E)$ denotes the bipartite graph between M and J, and a nearly-linear time is of order $\tilde{O}_\epsilon(|E|)$. For the identical machine with precedence constraints setting, we use κ to denote the number of precedence

[5] The result requires that the LP does not have redundant constraints.

constraints. A nearly-linear time algorithm runs in time $\tilde{O}_\epsilon(n+\kappa)$. Unlike the polynomial running time scenario, we can not assume $\prec$ is transitive, as it may dramatically increase the number of precedence constraints to quadratic. Moreover, the best known algorithm computing the transitive closure of the precedence constraints takes $O(n\kappa)$ time [40].

For many problems, including $R||C_{\max}, R||\sum_j w_jC_j, 1|\text{prec}|\sum_j w_jC_j$ and $P|\text{prec}, p_j = 1|\sum_j w_jC_j$, our nearly-linear time algorithms achieve the correspondent best known polynomial-time approximation ratios, due to Lenstra, Shmoys and Tardos [32], Im and Li [22], Hall et al. [20], and Li [35] respectively.

▶ **Theorem 1.1.** *For any $\epsilon > 0$, there is a $\tilde{O}_\epsilon(|E|)$-time $(2+\epsilon)$-approximation algorithm for $R||C_{\max}$, i.e., the makespan minimization problem on unrelated machines.*

For the problem $R||\sum_j w_jC_j$, we believe that showing that the rectangle LP can be approximated in nearly-linear time is interesting on its own. So we give two theorems for the problem. Refer to LP(6) for the formal description of the rectangle LP for the problem.

▶ **Theorem 1.2.** *Consider an instance of $R||\sum_j w_jC_j$ and the rectangle LP (6) for the instance. Let $\epsilon > 0$ and $\mathsf{lp}_{(6)}$ be the value of the LP. Then in $\tilde{O}_\epsilon(|E|)$ time, we can construct a solution $\mathbf{z}$ to the LP such that:*

- *$\mathbf{z}$ satisfies all the constraints in the LP, except that the constraint at most one job is processed on any machine at any time may be violated by a factor of $1+\epsilon$. (Formally, Constraint (8) is only satisfied with the right-side replaced by $1+\epsilon$.)*
- *The value of $\mathbf{z}$ to the LP is at most $(1+\epsilon)\mathsf{lp}_{(6)}$.*

In the theorem, our $\mathbf{z}$ will be represented by the list of non-zero coordinates and their values. Then, we show that the rounding algorithm of Im and Li [22] can indeed run in time nearly-linear on the support size of the LP solution. This gives the following theorem.

▶ **Theorem 1.3.** *For any $\epsilon > 0$, there is a $\tilde{O}_\epsilon(|E|)$-time $(1.45+\epsilon)$-approximation algorithm for $R||\sum_j w_jC_j$, i.e., the weighted completion time minimization problem on unrelated machines.*

The following two theorems are for $1|\text{prec}|\sum_j w_jC_j$ and $P|\text{prec}, p_j = 1|\sum_j w_jC_j$.

▶ **Theorem 1.4.** *For any $\epsilon > 0$, there is a $\tilde{O}_\epsilon((n+\kappa)\log p_{\max})$-time $(2+\epsilon)$-approximation algorithm for $1|\text{prec}|\sum_j w_jC_j$, i.e., the weighted completion time problem on a single machine with precedence constraints, where $p_{\max} := \max_{j\in J} p_j$ is the maximum job size.*

So the algorithm runs in nearly-linear time only when $p_{\max}$ is polynomially bounded.

▶ **Theorem 1.5.** *For any $\epsilon > 0$, there is a $\tilde{O}_\epsilon(n+\kappa)$-time $(1+\sqrt{2}+\epsilon)$-approximation algorithm for $P|\text{prec}, p_j = 1|\sum_j w_jC_j$, i.e., the weighted completion time problem on identical machines with unit-size jobs and precedence constraints.*

Along the way of algorithm design for the identical machine with precedence constraints setting, we developed a nearly-linear time $(1+\epsilon)$-approximation algorithm for the single commodity network flow problem in directed acyclic graphs, with bounded supplies and demands on sources and sinks, but infinite capacities on edges.

Recently there has been a lot of progress on solving maximum flow problem on undirected and directed graphs. For undirected graphs, the problem can be approximated within a factor of $1+\epsilon$ in nearly-linear time [26, 38, 45], and solved exactly with a slightly weaker running time of $m^{1+o(1)}$ (this is called *almost-linear* time) [7]. It was open whether an almost-linear

running time can be achieved for solving maximum flow on directed graphs.[6] This was resolved in the affirmative by a recent breakthrough due to Chen et al. [14]: They developed an algorithm that computes exact maximum flows on directed graphs with polynomially bounded integral capacities in $m^{1+o(1)}$ time. Thus, we could use the result as a black-box for our problem, if we allow the running time to be almost-linear. Nevertheless as our theme is to design *nearly-linear* time algorithms, we include in the full version of the paper our approximate maximum-flow algorithm for the special case with this running time. To the best of our knowledge, this was not known before.

For the general precedence-constrained scheduling problem $P|\text{prec}|\sum_j w_j C_j$ (on multiple machines with variant job lengths), we achieve an $O(1)$-approximation algorithm in nearly-linear time. However, the approximation ratio of the algorithm is $6+\epsilon$, which is worse than the best polynomial-time ratio of $2+2\ln 2$ due to Li [35].

▶ **Theorem 1.6.** *For any $\epsilon > 0$, there is a $\tilde{O}_\epsilon((n+\kappa)\log p_{\max})$-time $(6+\epsilon)$-approximation algorithm for $P|\text{prec}|\sum_j w_j C_j$, i.e., the weighted completion time minimization problem on identical machines with precedence constraints, where $p_{\max} := \max_{j\in J} p_j$ is the maximum job size.*

1.2 Our Techniques

All of our algorithms are based on linear programming: We design an LP relaxation of nearly-linear size, solve it in nearly-linear time to obtain a $(1+\epsilon)$-approximate solution, and round the solution into an integral schedule in nearly-linear time.

For $R||C_{\max}$, the natural LP relaxation has $O(|E|)$ size, and the mixed packing and covering form. Thus it can be solved within a factor of $1+\epsilon$ by the algorithm of Young [51] in $\tilde{O}_\epsilon(|E|)$ time. In particular, the algorithm outputs a $(1+\epsilon)$-approximate solution that violates the constraints by a factor of $1\pm\epsilon$, in $O\left(\frac{\bar{N}\log\bar{m}}{\epsilon^2}\right) = \tilde{O}_\epsilon(\bar{N})$ time, where $\bar{m}$ and $\bar{N}$ are the number of constraints and non-zero coefficients in the LP respectively. To round the fractional solution, we apply the grouping technique of [46] for the so called generalized assignment problem, but with a $(1+\epsilon)$-slack. This gives us a bipartite graph $H = (V, J, E_H)$ satisfying $|N_H(J')| \geq (1+\epsilon)|J'|$ for every $J' \subseteq J$, where $N_H(J')$ is the set of neighbors of J' in H. This allows us to find a matching in H that covers J in nearly-linear time, which leads to a $(2+\epsilon)$-approximate solution, matching the current best approximation of 2 in [32]. We remark that the $\tilde{O}_\epsilon(mn)$-running time of [39] comes from both solving the LP, and rounding the LP solution. So even with the nearly-linear time mixed covering and packing LP solver, the algorithm of [39] still requires $\tilde{O}_\epsilon(mn)$ time.

For the problem $R||\sum_j w_j C_j$, we give a nearly-linear size mixed packing and covering LP that (up to a factor of $1+O(\epsilon)$) is equivalent to the rectangle LP used by Li [35], Im and Shadloo [23], Im and Li [22]. In the rectangle LP, there is a variable x_{ijs} indicating if a job j is scheduled on the machine i and has starting time s, and constraints that at most one job is processed at any time on any machine. To reduce the size of the LP to $\tilde{O}_\epsilon(|E|)$, we partition the time horizon into *windows*, with lengths geometrically increasing by a factor of $1+\epsilon$. We distinguish between two types of scheduling intervals: If a job is scheduled within a window on some machine i (we call this an inside-window interval), then we do not need to capture the precise location of the scheduling interval. On the other hand, if the job

[6] By repeatedly solving maximum flow instances on residual graphs, one can convert an approximate maximum flow algorithm on directed graphs to an exact algorithm, without much loss on the running time. So for directed graphs, allowing $(1+\epsilon)$-approximation does not give much advantage.

starts and ends at two different windows (we call the interval an cross-window interval), we will approximately capture its starting and ending times. To do so, we divide each window into $1/\epsilon$ sub-windows, and let the LP variables capture the two sub-windows containing the starting and completion times. In the LP, we require all the cross-window intervals incur a congestion of 1: any point t is covered by at most 1 fraction of cross-window intervals. Then we require the total volume of jobs processed inside each window is at most its length. We show that up to a factor of $1 + O(\epsilon)$, a solution to the LP can be converted to one for the rectangle LP with no large cost. Roughly speaking, the width of window is small compared to its position and so we do not need to know the precise location of an inside-window-interval. For a cross-window-interval, we may incur an error on its length that is about ϵ times the total length of its starting window and ending window. As a sub-window has a small length, and a cross-window-interval covers some window-boundary, the total error incurred will also be small.

We proceed to our techniques for the weighted completion time problems in the identical machine with precedence constraints setting, i.e., the problem $P|\text{prec}|\sum_j w_j C_j$ and its special cases. Due to the precedence constraints, the LP relaxations do not have the mixed packing and covering form anymore. Nevertheless, the multiplicative weight update (MWU) framework can still be applied. We enclose the precedence constraints in a polytope $\mathcal{Q}$. In each iteration of the MWU framework, we guarantee that all these constraints are satisfied, i.e., the vector we obtain is in $\mathcal{Q}$. Other than the precedence constraints, we have $\tilde{O}_\epsilon(\log p_{\max})$ packing inequalities correspondent the m-machine constraint. This is due to that we can round completion times to integer powers of $1 + \epsilon$.

The number of iterations the MWU framework takes is $\tilde{O}_\epsilon(\bar{m})$, where $\bar{m}$ is the number of packing constraints in the LP, without counting the constraints for $\mathcal{Q}$. Fortunately we have $\bar{m} = \tilde{O}_\epsilon(\log p_{\max})$. To obtain the claimed $\tilde{O}_\epsilon((n + \kappa) \log p_{\max})$ time, we need to run each iteration of MWU in nearly-linear time. The bottleneck comes from finding a vector in $\mathcal{Q}$ satisfying one aggregated packing constraint, that maximizes a linear objective with non-negative coefficients.

A key technical contribution of our paper is an oracle for the problem. For an appropriately defined directed acyclic graph $G = (V, E)$, the polytope $\mathcal{Q}$ can be formulated as $\{\mathbf{y} \in [0, 1]^V : y_v \leq y_u, \forall vu \in E\}$. For two given row vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^V_{\geq 0}$, the aggregated LP in each iteration of MWU is: $\max \mathbf{ay}$ subject to $\mathbf{y} \in \mathcal{Q}$ and $\mathbf{by} \leq 1$. Using LP duality, the problem is reduced to the special single commodity maximum flow problem we introduced: We have bounded supplies and demands on sources and sinks, but infinite capacities on edges. When allowing a $(1 + \epsilon)$-approximation for the scheduling problem, we need to find a flow whose value is at least the maximum value for the instance with sink capacities scaled by $\frac{1}{1+\epsilon}$. This is done by our nearly-linear time maximum-flow algorithm for the special case.

1.3 Other Related Work

The makespan minimization problem in the identical machine setting with precedence constraints, i.e., the problem $P|\text{prec}|C_{\max}$, is another classic problem in scheduling theory. The seminal work of Graham [18] gives a simple greedy algorithm that achieves a 2-approximation. On the negative side, Lenstra and Rinnooy Kan [31] proved a $(4/3 - \epsilon)$-hardness for the problem. Under the stronger version of the Unique Game Conjecture (UGC) introduced by Bansal and Khot [4], Svensson [48] showed that the problem is hard to approximate within a factor of $2 - \epsilon$ for any $\epsilon > 0$. Much work has focused on the special case where $m = O(1)$ and all jobs have size 1 [33, 17, 34], for which obtaining a PTAS is a long-standing open problem.

The multiplicative weight update (MWU) method for solving linear programs has played an important role in a wide range of applications. Some of its foundational work can be found in a beautiful survey by Arora, Hazan and Kale [3]. There has been a vast literature on solving packing, covering, and mixed packing and covering LPs approximately to a factor of $1+\epsilon$ using iterative methods [44, 39, 36, 50, 16, 28, 27, 51, 1, 13]. In particular, to solve a mixed packing and covering LP with $\bar{n}$ variables, $\bar{m}$ constraints and $\bar{N}$ non-zero coefficients, the algorithm of Young [51] returns $(1+\epsilon)$-approximation deterministically in $O\left(\frac{\bar{N}\ln\bar{m}}{\epsilon^2}\right)$ time. The dependence on ϵ has been improved slightly by Chekuri and Quanrud [13], who gave a randomized algorithm with running time $\tilde{O}\left(\frac{\bar{N}}{\epsilon}+\frac{\bar{m}}{\epsilon^2}+\frac{\bar{n}}{\epsilon^3}\right)$, where $\tilde{O}(\cdot)$ hides a poly-logarithmic factor.

There has been a recent surge of interest in designing fast or nearly-linear time approximation algorithms for combinatorial optimization problems [11, 12, 9, 15, 34, 6].

Organization. The rest of the paper is organized as follows. In Section 2, we define some elementary notations used across the paper, and describe the result of Young [51] on solving mixed packing and covering LPs, and a template solver for packing LPs over an "easy" polytope. In Sections 3 and 4, we present our results for $R||C_{\max}$ and $R||\sum_j w_jC_j$. Due to the page limit, we leave our algorithms for $P|\text{prec}|\sum_j w_jC_j$ and the two special cases $1|\text{prec}|\sum_j w_jC_j$ and $P|\text{prec}, p_j=1|\sum_j w_jC_j$ to the full version of the paper. The full version also contains other technicalities, such as how to handle the case where input integers are not polynomially bounded, how to reduce problems to the promise versions and how to use the self-balancing binary search tree data structure to run a list scheduling algorithm.

2 Preliminaries

We use bold lowercase letters to denote vectors, and their correspondent italic letters to denote their coordinates. We use bold uppercase letters to denote matrices. $\mathbf{0}$ and $\mathbf{1}$ are used to denote the all-0 and all-1 vectors whose domain can be inferred from the context. Given a template vector $\mathbf{v}$ over some finite domain, and a subset S of the domain, let $v(S):=\sum_{e\in S}v_e$ be the sum of v-values over elements in S.

Given an (undirected) graph $H=(V_H,E_H)$, we use $\delta_H(v), N_H(v), \delta_H(U), N_H(U)$ to respectively denote the sets of incident edges of $v\in V_H$, neighbors of v, edges between the set $U\subseteq V_H$ and $V_H\setminus U$, and vertices in $V_H\setminus U$ with at least one neighbor in U, in the graph H. Given a directed graph $H=(V_H,E_H)$, for every $v\in V_H$, we use $\delta^+_H(v)$ and $\delta^-_H(v)$ to denote the sets of outgoing and incoming edges of v respectively. For every $U\subseteq V_H$, let $\delta^+_H(U):=\{uv\in E_H: u\in U, v\notin U\}$ and $\delta^-_H(U):=\{uv\in E_H: u\notin U, v\in U\}$ be the sets of edges from U to $V_H\setminus U$ and from $V_H\setminus U$ to U respectively. When $H=G$ for the graph G in the context (which can be undirected or directed), we omit the subscript H in the notations.

For cleanness of exposition, we use $\tilde{O}_\epsilon(\cdot)$ to hide factors that are polynomial in $\frac{1}{\epsilon}$ and poly-logarithmic in the size of the input. As we gave the first nearly-linear time algorithms for the studied problems, the hidden factors are small compared to the improvements we make. The final approximation ratios we get have an additive factor of $O(\epsilon)$ (instead of ϵ); but it can be reduced to ϵ if we start from a smaller ϵ. By default, for an (undirected or directed) graph $H=(V_H,E_H)$ we deal with, we assume every vertex is incident to at least one edge so $|E_H|=\Omega(V_H)$. For any $a\in\mathbb{R}$, we define $(a)_+$ as $\max\{a,0\}$.

2.1 Nearly-Linear Time Mixed Packing and Covering LP Solver

A mixed packing and covering LP is an LP of the following form:

$$\text{find } \mathbf{x} \quad \text{such that} \quad \mathbf{x} \geq 0, \quad \mathbf{Px} \leq \mathbf{1} \quad \text{and} \quad \mathbf{Cx} \geq \mathbf{1}, \tag{MPC}$$

where $\mathbf{P} \in \mathbb{R}_{\geq 0}^{\bar{m}_{\mathbf{P}} \times \bar{n}}$ and $\mathbf{C} \in \mathbb{R}_{\geq 0}^{\bar{m}_{\mathbf{C}} \times \bar{n}}$ for some positive integers $\bar{n}, \bar{m}_{\mathbf{P}}, \bar{m}_{\mathbf{C}}$. Let $\bar{m} = \bar{m}_{\mathbf{P}} + \bar{m}_{\mathbf{C}}$ and $\bar{N}$ be the total number of non-zeros in $\mathbf{P}$ and $\mathbf{C}$. Young [51] developed a nearly-linear time algorithm that solves (MPC) approximately:

▶ **Theorem 2.1** ([51]). *Given an instance of* (MPC) *and $\epsilon > 0$, there is an $O\left(\frac{\bar{N} \log \bar{m}}{\epsilon^2}\right)$-time algorithm that either claims* (MPC) *is infeasible, or outputs an $\mathbf{x} \in \mathbb{R}_{\geq 0}^{\bar{n}}$ such that $\mathbf{Px} \leq (1+\epsilon)\mathbf{1}$ and $\mathbf{Cx} \geq \frac{\mathbf{1}}{1+\epsilon}$.*

2.2 Template Packing LP Solver over a Simple Polytope

In this section, we describe a template MWU-based LP solver for a packing linear program with an additional requirement that the solution is inside an "easy" polytope $\mathcal{Q}$. The framework we describe here is introduced in [10] and later reformulated in [11].

Let $\mathbf{P} \in \mathbb{R}_{\geq 0}^{\bar{m} \times \bar{n}}$ be a non-negative matrix, with $\bar{N}$ non-zero entries. Let $\mathbf{a} \in \mathbb{R}_{\geq 0}^{\bar{n}}$ be a row vector, and $\mathcal{Q} \subseteq \mathbb{R}_{\geq 0}^{\bar{n}}$ be a polytope which is defined by "easy" constraints. We focus on the following linear program:

$$\max \ \mathbf{ax} \quad \text{subject to} \quad \mathbf{x} \in \mathcal{Q} \quad \text{and} \quad \mathbf{Px} \leq \mathbf{1}. \tag{$\mathrm{P}_{\mathcal{Q}}$}$$

Throughout the paper, we make sure all instances of ($\mathrm{P}_{\mathcal{Q}}$) we deal with are feasible.

▶ **Definition 2.2.** *Let $\epsilon \in (0,1), \phi > 0$ be two parameters. An (ϵ, ϕ)-approximate solution to* ($\mathrm{P}_{\mathcal{Q}}$) *is a vector $\mathbf{x} \in \mathcal{Q}$ satisfying $\mathbf{Px} \leq (1+\epsilon)\mathbf{1}$ and $\mathbf{ax} \geq \mathbf{ax}^* - \phi$, where $\mathbf{x}^* \in \mathcal{Q}$ is the optimum solution to* ($\mathrm{P}_{\mathcal{Q}}$).

As a hindsight, we only allow a loss of an additive factor ϕ in the objective function of the LP for $P|\text{prec}|\sum_j w_j C_j$, which will be set to be a polynomially small term. As is typical in a MWU framework, we need to solve the following LP where the constraints $\mathbf{Px} \leq \mathbf{1}$ are aggregated into one constraint $\mathbf{by} \leq 1$, where $\mathbf{b} \in \mathbb{R}_{\geq 0}^{\bar{n}}$ is a row vector:

$$\max \ \mathbf{ay} \quad \text{subject to} \quad \mathbf{y} \in \mathcal{Q} \quad \text{and} \quad \mathbf{by} \leq 1. \tag{1}$$

Again we guarantee all instances of (1) we encounter are feasible.

▶ **Definition 2.3.** *Let $\epsilon \in (0,1), \phi > 0$ be two parameters. An (ϵ, ϕ)-approximate solution to* (1) *is a vector $\mathbf{y} \in \mathcal{Q}$ satisfying $\mathbf{by} \leq 1 + \epsilon$ and $\mathbf{ay} \geq \mathbf{ay}^* - \phi$, where $\mathbf{y}^*$ is the optimum solution to the LP. An (ϵ, ϕ)-oracle for* (1) *is an algorithm that, given an instance of* (1)*, and $\epsilon \in (0,1), \phi > 0$, outputs an (ϵ, ϕ)-approximate solution $\mathbf{y}$ to* (1).

The template LP solver is described in Algorithm 1, where we use $\mathbf{P}_i$ to denote the i-th row vector of $\mathbf{P}$. By our assumption that ($\mathrm{P}_{\mathcal{Q}}$) is feasible, the instance of (1) defined in every execution of Step 3 is also feasible. The performance of the algorithm is summarized in the following theorem.

▶ **Theorem 2.4.** *Algorithm 1 will return an $(O(\epsilon), \phi)$-approximate solution $\mathbf{x}$ to* ($\mathrm{P}_{\mathcal{Q}}$)*, within $O(\frac{\bar{m} \log \bar{m}}{\epsilon^2})$ iterations of Loop 2.*

Algorithm 1 LP Solver for $(\mathrm{P}_{\mathcal{Q}})$.

Input: an instance of $(\mathrm{P}_{\mathcal{Q}})$, $\epsilon \in (0,1), \phi > 0$, and (ϵ,ϕ)-oracle $\mathcal{O}$ for (1)
Output: an $(O(\epsilon),\phi)$-approximate solution $\mathbf{x}$ for $(\mathrm{P}_{\mathcal{Q}})$

1: $t \leftarrow 0, \rho \leftarrow \frac{\ln \bar{m}}{\epsilon^2}, \mathbf{x}^{(0)} \leftarrow \mathbf{0} \in \mathbb{R}^{\bar{n}}_{\geq 0}, \mathbf{u}^{(0)} \leftarrow \mathbf{1} \in \mathbb{R}^{\bar{m}}_{\geq 0}$
▷ $\mathbf{x}^{(t)}$'s are column vectors and $\mathbf{u}^{(t)}$'s are row vectors
2: **while** $t < 1$ **do**
3: define $\mathbf{b} := \frac{\mathbf{u}^{(t)}}{|\mathbf{u}^{(t)}|}\mathbf{P}$, and run the oracle $\mathcal{O}$ for (1) to obtain an (ϵ,ϕ)-approximate solution $\mathbf{y}$ for (1)
4: $\delta \leftarrow \min\left\{\min_{i\in[\bar{m}]} \frac{1}{\rho \cdot \mathbf{P}_i\mathbf{y}}, 1-t\right\}$
5: **for** every $i \in [\bar{m}]$ **do** $u_i^{(t+\delta)} \leftarrow u_i^{(t)} \cdot \exp\left(\delta\epsilon\rho \cdot \mathbf{P}_i\mathbf{y}\right)$
6: $\mathbf{x}^{(t+\delta)} \leftarrow \mathbf{x}^{(t)} + \delta\mathbf{y}, t \leftarrow t+\delta$
7: **return** $\mathbf{x} := \mathbf{x}^{(1)}$

Proof. Focus on one iteration of Loop 2. Let t be the value of t at the beginning of the iteration, $\mathbf{y}$ and δ be the $\mathbf{y}$ and δ obtained in Step 3 and 4 in the iteration respectively. Then we have

$$|\mathbf{u}^{(t+\delta)}| = \sum_{i\in[\bar{m}]} u_i^{(t+\delta)} = \sum_{i\in[\bar{m}]} u_i^{(t)} \exp(\delta\epsilon\rho \cdot \mathbf{P}_i\mathbf{y}) \leq \sum_{i\in[\bar{m}]} u_i^{(t)}(1+(1+\epsilon)\epsilon \cdot \delta\rho \cdot \mathbf{P}_i\mathbf{y})$$
$$= |\mathbf{u}^{(t)}| + (1+\epsilon)\epsilon\delta\rho \cdot \mathbf{u}^{(t)}\mathbf{P}\mathbf{y} \leq |\mathbf{u}^{(t)}| + (1+\epsilon)^2\epsilon\delta\rho \cdot |\mathbf{u}^{(t)}| \leq |\mathbf{u}^{(t)}| \exp((1+\epsilon)^2\epsilon\delta\rho).$$

The inequality in the first line is by that $\delta\rho \cdot \mathbf{P}_i\mathbf{y} \in [0,1]$ for every $i \in [\bar{m}]$ and $e^{\epsilon\theta} \leq 1+\epsilon\theta+(\epsilon\theta)^2 \leq 1+\epsilon\theta+\epsilon^2\theta$ for every $\epsilon \in [0,1]$ and $\theta \in [0,1]$. The first inequality in the second line is by that $\frac{\mathbf{u}^{(t)}}{|\mathbf{u}^{(t)}|}\mathbf{P}\mathbf{y} = \mathbf{b}\mathbf{y} \leq 1+\epsilon$.

Combining the inequality over all iterations, we have

$$|\mathbf{u}^{(1)}| \leq |\mathbf{u}^{(0)}| \exp\left((1+\epsilon)^2\epsilon\rho\right) = \bar{m} \cdot \exp\left((1+\epsilon)^2\epsilon\rho\right). \tag{2}$$

For every $i \in [\bar{m}]$, we have $u_i^{(1)} = \exp\left(\epsilon\rho \cdot \mathbf{P}_i\mathbf{x}\right)$, where $\mathbf{x} := \mathbf{x}^{(1)}$ is the returned solution. So, by (2), we have $\exp(\epsilon\rho \cdot \mathbf{P}_i\mathbf{x}) \leq \bar{m} \cdot \exp((1+\epsilon)^2\epsilon\rho)$, which implies $\mathbf{P}_i\mathbf{x} \leq \frac{\ln \bar{m}}{\epsilon\rho} + (1+\epsilon)^2 \leq (1+\epsilon)^2 + \epsilon = 1+O(\epsilon)$.

In the end $\mathbf{x} = \mathbf{x}^{(1)}$ is a convex combination of vectors $\mathbf{y}$ obtained in all iterations. As each $\mathbf{y}$ is in $\mathcal{Q}$, we have $\mathbf{x} \in \mathcal{Q}$. Moreover, for the instance of (1) in any iteration, $\mathbf{x}^*$ is a valid solution. So, the optimum solution $\mathbf{y}^*$ to the instance of (1) has $\mathbf{a}\mathbf{y}^* \geq \mathbf{a}\mathbf{x}^*$, and the $\mathbf{y}$ returned by the oracle has $\mathbf{a}\mathbf{y} \geq \mathbf{a}\mathbf{y}^* - \phi \geq \mathbf{a}\mathbf{x}^* - \phi$. This implies our final $\mathbf{x}$ has $\mathbf{a}\mathbf{x} \geq \mathbf{a}\mathbf{x}^* - \phi$. Therefore, $\mathbf{x}$ is a $(O(\epsilon),\phi)$-approximate solution to $(\mathrm{P}_{\mathcal{Q}})$.

It remains to bound the number of iterations that Loop 2 can take. In every iteration of loop 2 except for the last one, some i has $\frac{1}{\rho\cdot\mathbf{P}_i\mathbf{y}} = \delta$, i.e., $\delta\epsilon\rho \cdot \mathbf{P}_i\mathbf{y} = \epsilon$. We say u_i is increased fully in the iteration. Notice by (2), each u_i can be increased fully in at most $\frac{\ln\left(\bar{m}\exp((1+\epsilon)^2\epsilon\rho)\right)}{\epsilon} = \frac{\ln\bar{m}+(1+\epsilon)^2\epsilon\rho}{\epsilon} = O\left(\frac{\ln\bar{m}}{\epsilon^2}\right)$ iterations. This bounds the number of iterations by $O\left(\frac{\bar{m}\log\bar{m}}{\epsilon^2}\right)$ as there are $\bar{m}$ different values of i. ◀

For each iteration of loop 2, the steps other than Step 3 takes $O(\bar{N})$ time. Therefore, the running time of Algorithm 1 is $O\left(\frac{\bar{m}\log\bar{m}\cdot\bar{N}}{\epsilon^2}\right)$, plus the time for running the oracle $O\left(\frac{\bar{m}\log\bar{m}}{\epsilon^2}\right)$ times.

3 Unrelated Machine Makespan Minimization

In this section, we give the nearly-linear time $(2+\epsilon)$-approximation algorithm for the unrelated machine makespan minimization problem, i.e, the problem $R||C_{\max}$. Recall that we are given a bipartite graph $G=(M,J,E)$ and a $p_{ij}\in\mathbb{Z}_{>0}$ for every $ij\in E$. Recall that $N(j), N(i), \delta(j)$ and $\delta(i)$ denote the set of neighbors or incident edges of a job $j\in J$ or a machine $i\in M$, in the graph G.

Via a standard technique described in the full version of the paper, we can focus on the following promise version:

- We are given a number $P\geq \mathsf{opt}$, where opt is the optimal makespan of the instance, and our goal is to construct an assignment of makespan at most $(2+O(\epsilon))P$.

For some $ij\in E$ with $p_{ij}>P$, we remove ij from E, as the optimum solution does not use the edge. The following is the natural LP relaxation for the problem:

$$\sum_{j\in N(i)} p_{ij}x_{ij}\leq P, \forall i\in M \quad (3) \qquad \sum_{i\in N(j)} x_{ij}\geq 1, \forall j\in J \quad (4) \qquad x_{ij}\geq 0, \forall ij\in E \quad (5)$$

In the correspondent integer program, $x_{ij}\in\{0,1\}$ for every $ij\in E$ indicates whether the job j is assigned to machine i. (3) requires that the makespan of the schedule to be at most P, (4) requires every job to be scheduled. In the linear program, we replace the requirement that $x_{ij}\in\{0,1\}$ with the non-negativity constraint (5).

By the promise that $P\geq\mathsf{opt}$, the LP is feasible. Therefore, applying Theorem 2.1, we can solve the LP in $\tilde{O}_\epsilon(|E|)$ time to obtain an approximate solution $\mathbf{x}\in[0,1]^E$. By scaling, we can assume (4) holds with equalities, and (3) holds with right side replaced by $(1+O(\epsilon))P$.

To round the solution to an integral assignment in $\tilde{O}_\epsilon(|E|)$-time, we use the grouping idea from [46]: For each machine $i\in M$, we break the fractional jobs assigned to i into groups, each containing $\frac{1}{1+\epsilon}$ fractional jobs. This gives us a bipartite graph H between jobs and groups. Any perfect matching (i.e., a matching covering all jobs J) will give a $(2+O(\epsilon))$-approximation for the makespan problem. In H, every subset $J'\subseteq J$ of jobs has at least $(1+\epsilon)|J'|$ neighbors. The $(1+\epsilon)$-factor allows us to design a $\tilde{O}_\epsilon(|E|)$-time algorithm to find a matching covering all jobs J, as stated in the following lemma:

▶ **Lemma 3.1.** *Assume we are given a bipartite graph $H=(S,T,E_H)$ and $\epsilon>0$ such that $|N_H(S')|\geq(1+\epsilon)|S'|$ for every $S'\subseteq S$. In $O\left(\frac{|E_H|}{\epsilon}\log|S|\right)$-time, we can find a matching in H covering all vertices in S.*

Proof. Let $L=\lfloor\log_{1+\epsilon}|S|\rfloor+1>\log_{1+\epsilon}|S|$. Then we use the shortest-augmenting path algorithm of Hopcroft and Karp [21] to find a matching for which there is no augmenting path of length at most $2L+1$. The running time of the algorithm can be made to $O(|E_H|L)=O(\frac{|E_H|}{\epsilon}\log|S|)$. It remains to show the following lemma:

▶ **Lemma 3.2.** *Let F be a matching in H for which there is no augmenting path of length at most $2L+1$. Then all vertices in S are matched in the matching F.*

Proof. Let $\vec{H}$ be the residual graph of H w.r.t the F: $\vec{H}$ is a directed graph over $S\cup T$, for every edge $st\in E_H$, we have $st\in\vec{H}$, and for every $st\in F$, we have $ts\in\vec{H}$. We say a vertex in S is free if it is unmatched in F. For every integer $\ell\in[0,L]$, define S^ℓ (T^ℓ resp.) to be the set of vertices in S (T, resp.) to which there exists a path in $\vec{H}$ of length *at most* 2ℓ ($2\ell+1$, resp.) from a free vertex. So, we have $S^0\subseteq S^1\subseteq S^2\subseteq\cdots\subseteq S^L$ and $T^0\subseteq T^1\subseteq T^2\subseteq\cdots\subseteq T^L$.

Notice that $T^\ell = N_H(S^\ell)$ for every $\ell \in [0, L]$. So for every $\ell \in [0, L]$, we have $(1+\epsilon)|S^\ell| \le |T^\ell|$ by the condition of the lemma. All vertices in T^L are matched by our assumption that there are no augmenting paths of length at most $2L+1$. So for every $\ell \in [0, L-1]$, we have $|T^\ell| \le |S^{\ell+1}|$ as all vertices in T^ℓ are matched to $S^{\ell+1}$.

Combining the two statements gives us $(1+\epsilon)|S^\ell| \le |S^{\ell+1}|$ for every $\ell \in [0, L-1]$. Thus $|S^L| \ge (1+\epsilon)^L |S^0|$, which contradicts the definition of L and that $|S^0| \ge 1, |S^L| \le |S|$. ◀

This finishes the proof of Lemma 3.1. ◀

With the lemma, we prove the following theorem using the grouping technique from [39]:

▶ **Theorem 3.3.** *Given $\mathbf{x} \in [0,1]^E$ satisfying $x(\delta(j)) = 1$ for every $j \in J$, and $\epsilon \in (0,1)$, there is an $O\left(\frac{|E|}{\epsilon} \log n\right)$-time algorithm that outputs an assignment $\sigma \in M^J$ of jobs to machines such that $\sigma_j j \in E$ and $x_{\sigma_j j} > 0$ for every $j \in J$, and for every $i \in M$, we have*

$$\sum_{j \in \sigma^{-1}(i)} p_{ij} \le (1+\epsilon) \sum_{j \in N(i)} p_{ij} x_{ij} + \max_{j \in \sigma^{-1}(i)} p_{ij}. \quad \textit{(Assume the maximum over } \emptyset \textit{ is 0.)}$$

Proof. We construct a bipartite graph $H = (V, J, E_H)$, starting with $V = \emptyset$ and $E_H = \emptyset$. For every machine $i \in M$, we run the following procedure. See Figure 1 for an illustration. (The notations defined in the paragraph depend on i; if a notation does not contain i in the subscript, it will only be used locally, in this paragraph.) Let D_i be the number of jobs j with positive x_{ij} values. Let $j_1, j_2, \cdots, j_{D_i}$ be these jobs j, sorted in non-increasing order of p_{ij}; that is, we have $p_{ij_1} \ge p_{ij_2} \ge \cdots \ge p_{ij_{D_i}}$. For every integer $d \in [0, D_i]$, we define $Z_d = \sum_{d'=1}^{d} x_{ij_{d'}}$. Let $R_i = \lceil (1+\epsilon) Z_{D_i} \rceil = \lceil (1+\epsilon) x(\delta(i)) \rceil$. For every $r = 1, 2, 3, \cdots, R_i$, we create a vertex ir and add it to V. We add to E_H an edge between $ir, r \in [R_i]$ and j_d, $d \in [D_i]$ if $(\frac{r-1}{1+\epsilon}, \frac{r}{1+\epsilon}) \cap (Z_{d-1}, Z_d) \ne \emptyset$, and we define $y_{(ir)j_d}$ to be the length of the interval. This finishes the construction of $H = (V, J, E_H)$, along with a vector $\mathbf{y} \in \left(0, \frac{1}{1+\epsilon}\right]^{E_H}$.

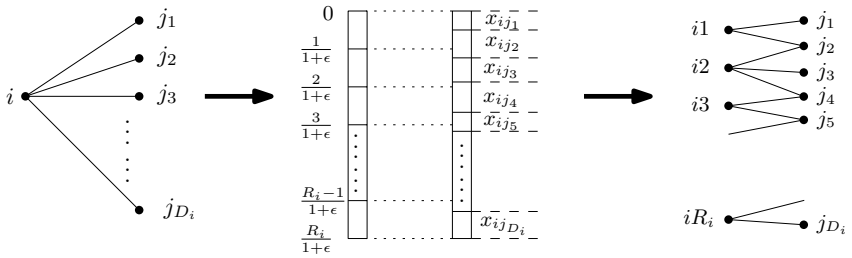


Figure 1 Construction of the H for the machine $i \in M$. In the bipartite graph between $\{i1, i2, \cdots, iD_i\}$ and $\{j_1, j_2, \cdots, j_{D_i}\}$ and there is an edge between j_d and (ir) iff the interval correspondent to j_d intersects the interval $(\frac{r-1}{1+\epsilon}, \frac{r}{1+\epsilon})$.

The number of edges in H for each i is at most $D_i + R_i - 1 \le |\delta(i)| + (1+\epsilon)x(\delta(i))$. Therefore the total number of edges we created in H is at most $|E| + (1+\epsilon)|J| = O(|E|)$. For every $ij \in E$, we have $\sum_{r:(ir)j \in E_H} y_{(ir)j} = x_{ij}$. This implies that for every $j \in J$, we have $y(\delta_H(j)) = 1$. For every $ir \in V$, we have $y\big(\delta_H(ir)\big) \le \frac{1}{1+\epsilon}$, and the inequality holds with equality except when $r = R_i$.

For every set $J' \subseteq J$, we have $|N_H(J')| \ge (1+\epsilon)|J'|$, as we can view $\mathbf{y}$ as a fractional matching in H where every $j \in J$ is matched to an extent of 1 and every $ir \in V$ is matched to an extent of at most $\frac{1}{1+\epsilon}$. Then we can use Lemma 3.1 [7] to find a matching in H that

[7] We need to switch the left and right sides when going from the bipartite graph H in Theorem 3.3 to that in Lemma 3.1. That is, we set $S = J$ and $T = V$.

covers all jobs J. The running time of the algorithm is $O\left(\frac{|E_H|}{\epsilon}\log n\right) = O\left(\frac{|E|}{\epsilon}\log n\right)$. The matching gives an assignment $\sigma \in M^J$: If j is matched to ir, then define $\sigma_j = i$. Fix some $i \in M$ with $\sigma^{-1}(i) \neq \emptyset$; we upper bound $\sum_{j\in\sigma^{-1}(i)} p_{ij}$:

$$\begin{aligned}\sum_{j\in\sigma^{-1}(i)} p_{ij} \leq & \max_{j\in\sigma^{-1}(i)} p_{ij} + \sum_{r=2}^{R_i} \max_{j\in N_H(ir)} p_{ij} \\ \leq & \max_{j\in\sigma^{-1}(i)} p_{ij} + (1+\epsilon)\sum_{r=2}^{R_i} \sum_{j\in N_H(i(r-1))} p_{ij} y_{(i(r-1))j} \\ \leq & \max_{j\in\sigma^{-1}(i)} p_{ij} + (1+\epsilon)\sum_{r=1}^{R_i} \sum_{j\in N_H(ir)} p_{ij} y_{(ir)j} \;=\; \max_{j\in\sigma^{-1}(i)} p_{ij} + (1+\epsilon)\sum_{j\in N(i)} p_{ij} x_{ij}.\end{aligned}$$

To see the first inequality, notice that the job j' matched to $i1$ (if it exists) has $p_{ij'} \leq \max_{j\in\sigma^{-1}(i)} p_{ij}$, and the job j' matched to each ir, $r \in [2, R_i]$, has $p_{ij'} \leq \max_{j\in\delta_H((ir))} p_{ij}$. Consider the second inequality. For every $r \in [2, R_i]$, any $j \in \delta_H(ir)$ and any $j' \in \delta_H(i(r-1))$, we have $p_{ij} \leq p_{ij'}$. Moreover, for every $r \in [2, R_i]$, we have $y\big(\delta_H(i(r-1))\big) = \frac{1}{1+\epsilon}$. The inequality in the third line follows from replacing r with $r+1$. The equality holds since for every $ij \in E$ we have $\sum_{r:(ir)j\in E_H} y_{(ir)j} = x_{ij}$. ◀

We can then apply Theorem 3.3 with the solution $\mathbf{x}$ we obtained from solving LP(3-5). Clearly we have $\max_{j\in\sigma^{-1}(i)} p_{ij} \leq P$ for every $i \in M$. So, the total load on any machine i is at most $P + (1+\epsilon)\cdot\sum_{j\in N(i)} p_{ij}x_{ij} \leq P + (1+\epsilon)\cdot(1+O(\epsilon))P = (2+O(\epsilon))P$, as (3) is satisfied with right side replaced by $(1+O(\epsilon))P$. This finishes the analysis of the algorithm for $R||C_{\max}$ and proves Theorem 1.1.

4 Unrelated Machine Weighted Completion Time Minimization

In this section, we give our nearly-linear time algorithm for $R||\sum_j w_j C_j$, with an approximation ratio of $1.45+\epsilon$, matching the current best ratio of Im and Li [22] achieved in polynomial time. Our result is based on formulating an LP relaxation that is equivalent to the rectangle LP introduced by Li [35]. The new LP relaxation has a nearly-linear size and the mixed packing and covering form; thus it can be solved in nearly-linear time using Theorem 2.1. We describe the rectangle LP (LP(6)), our new LP relaxation (LP(11)) and show their equivalence in Sections 4.1, 4.2 and 4.3 respectively.

In the full version of the paper we show how to construct a solution to LP(6) from one to LP(11) in nearly-linear time, finishing the proof of Theorem 1.2. We also show in the full version that the rounding algorithm of Im and Li can run in nearly-linear time; this finishes the proof of Theorem 1.3. Throughout the section, we assume all processing times are integers bounded by a polynomial of n. The general case is handled in the full version.

4.1 Rectangle LP Relaxation

We describe the rectangle LP relaxation for $R||\sum_j w_j C_j$ introduced by Li [35]. Let $T = \sum_{j\in J}\max_{i\in N(j)} p_{ij}$ so that any schedule will complete by time T. The following is the rectangle LP:

$$\min \quad \sum_{j\in J} w_j \sum_{i\in N(j), s\in[0,T)} z_{ijs}(s+p_{ij}) \tag{6}$$

$$\sum_{i\in N(j),s\in[0,T)} z_{ijs} \geq 1 \quad \forall j \in J \qquad (7)$$

$$\sum_{j\in N(i),s\in[t-p_{ij},t)} z_{ijs} \leq 1 \quad \forall i \in M, t \in [T] \qquad (8)$$

$$z_{ijs} = 0 \quad \forall ij \in E, s > T - p_{ij} \qquad (9)$$

$$z_{ijs} \geq 0 \quad \forall ij \in E, s \in [0,T) \qquad (10)$$

In the correspondent integer program, z_{ijs} for every $ij \in E$ and integer $s \in [0,T)$ indicates if job j is scheduled on machine i, with starting time s. The objective gives the weighted completion time of the schedule. (7) requires that every job j is scheduled. (8) requires that at any time on machine i, at most one job is being processed. (9) ensures that no jobs complete after time T. (10) is the non-negativity constraint. Im and Li [22] showed that given a solution $\mathbf{z}$ to LP(6), one can round it to an integral schedule, whose weighted completion time in expectation is at most 1.45 times the value of $\mathbf{z}$.

4.2 A Nearly-Linear Size LP Relaxation

In this section we formulate the relaxation that can be solved in nearly-linear time, and prove its equivalence to LP(6) in Section 4.3. We create a list of time points as follows: $T_0 = 0$, $T_d = \lfloor (1+\epsilon)T_{d-1} \rfloor + 1$ for every integer $d \geq 1$. Define $D = O(\frac{\log n}{\epsilon})$ to be the smallest integer so that $T_D \geq T$. We call $(T_{d-1}, T_d]$ the d-th *window*, and the time points $T_0, T_1, \cdots, T_D$ *window boundaries* (or simply boundaries). Define $\Delta_d = T_d - T_{d-1}$ to be the length of the d-th window.

Let $\eta_d := \lceil \epsilon \Delta_d \rceil$. We partition $(T_{d-1}, T_d]$ into *sub-windows* of length η_d, except that the last sub-window may be shorter. Then $q_d := \left\lceil \frac{\Delta_d}{\eta_d} \right\rceil \leq \frac{1}{\epsilon}$ is the number of sub-windows of $(T_{d-1}, T_d]$. Let $\tau_0^{(d)} = T_{d-1}, \tau_1^{(d)}, \tau_2^{(d)}, \cdots, \tau_{q_d}^{(d)} = T_d$ be the boundaries of the q_d sub-windows.

We describe the variables in the LP. For every $ij \in E$ and $d \in [D]$ with $p_{ij} \leq \Delta_d$, we introduce a variable x_{ijd}, indicating if j is scheduled on i inside the d-th window. Let S_j and C_j be the starting and completion time of j in the target optimum schedule (which the algorithm does not know). For every $ij \in E, 1 \leq d \leq e < D$, integers $u \in [0, q_d), v \in [0, q_{e+1})$, we *may* introduce a variable y_{ijdeuv} indicating if j is scheduled on i, $S_j \in [\tau_u^{(d)}, \tau_{u+1}^{(d)})$ and $C_j \in (\tau_v^{(e+1)}, \tau_{v+1}^{(e+1)}]$. That means, the scheduling interval $(S_j, C_j]$ of j contains the d'-th window for every $d' \in [d+1, e]$, and a non-empty part of the d-th and $(e+1)$-th windows. u and v approximately give the volumes of j processed in the two windows. It is disjoint from all other windows. As a hindsight, a sub-window is short enough and we can afford to incur an error equaling its length for every window. We only introduce a y-variable if the correspondent event can happen. That is, the following conditions need to be satisfied for the existence of y_{ijdeuv}: $\tau_v^{(e+1)} - \tau_{u+1}^{(d)} + 2 \leq p_{ij} \leq \tau_{v+1}^{(e+1)} - \tau_u^{(d)}$. Notice when $y_{ijdeuv} = 1$, then the scheduling interval $(S_j, C_j]$ of j intersects at least two windows.

For a variable y_{ijdeuv} and an integer $d' \in [D]$, we define

$$Q_{ijdeuvd'} := \begin{cases} 0 & \text{if } d' \leq d-1 \text{ or } d' \geq e+2 \\ \Delta_{d'} & \text{if } d+1 \leq d' \leq e \\ T_d - \tau_{u+1}^{(d)} + 1 & \text{if } d' = d \\ \tau_v^{(e+1)} - T_e + 1 & \text{if } d' = e+1 \end{cases}.$$

Assuming j starts at time $\tau_{u+1}^{(d)} - 1$ and ends at time $\tau_v^{(e+1)} + 1$ on machine i, we have that $Q_{ijdeuvd'}$ is the volume of job j processed in the d'-th window. So, if $y_{ijdeuv} = 1$ in a schedule, then $Q_{ijdeuvd'}$ gives a lower bound on the volume.

We say the quadruple $deuv$ left-covers the pair $d'u'$ if the sub-window $(\tau^{(d')}_{u'-1}, \tau^{(d')}_{u'}]$ is between the sub-windows $(\tau^{(d)}_{u}, \tau^{(d)}_{u+1}]$ (exclusive) and $(\tau^{(e+1)}_{v}, \tau^{(e+1)}_{v+1}]$ (inclusive) in the time horizon, or if $(\tau^{(d')}_{u'-1}, \tau^{(d')}_{u'}] = (\tau^{(d)}_{u}, \tau^{(d)}_{u+1}]$ and $\tau^{(d')}_{u'} - \tau^{(d')}_{u'-1} = 1$. So if $deuv$ left-covers $d'u'$ and $y_{ijdeuv} = 1$, then the scheduling interval of j will surely cover the left-most time unit of the sub-window $(\tau^{(d')}_{u'-1}, \tau^{(d')}_{u'}]$.

With the necessary definitions, we can formulate the LP relaxation as LP(11). For the sake of convenience, we assume if a variable does not exist, then it is not included in a summation.

$$\min \quad \sum_{ijd} w_j \cdot (T_{d-1} + 1) \cdot x_{ijd} + \sum_{ijdeuv} w_j \cdot (T_e + 1) \cdot y_{ijdeuv} \tag{11}$$

$$\sum_{id} x_{ijd} + \sum_{ideuv} y_{ijdeuv} \geq 1 \qquad \forall j \in J \tag{12}$$

$$\sum_{\substack{jdeuv \,:\, deuv \\ \text{left-covers } d'u'}} y_{ijdeuv} \leq 1 \qquad \forall i \in M, d' \in [D], u' \in [q_{d'}] \tag{13}$$

$$\sum_{j} p_{ij} \cdot x_{ijd'} + \sum_{jdeuv} Q_{ijdeuvd'} \cdot y_{ijdeuv} \leq \Delta_{d'} \qquad \forall i \in M, d' \in [D] \tag{14}$$

$$\text{all variables are non-negative} \tag{15}$$

Consider the correspondent integer program and an integral schedule. If $x_{ijd} = 1$, then the completion time of j is in $(T_{d-1}, T_d]$. If $y_{ijdeuv} = 1$, then it is in $(T_e, T_{e+1}]$. So, the objective (11) approximates and underestimates the total weighted completion time of the schedule.[8] (12) requires that every job is scheduled: either the scheduling interval of a job j is inside some window, or it overlaps with at least two windows. (13) follows from the definition of $deuv$ left-covering $d'u'$. If $x_{ijd'} = 1$, then the p_{ij} units of job j is processed in the d'-th window on machine i. If $y_{ijdeuv} = 1$, then at least $Q_{ijdeuvd'}$ units is processed. So (14) is valid as the volume of the jobs processed in the d'-th window is at most $\Delta_{d'}$. Therefore, LP(11) is valid, and its value is at most the weighted completion time of the optimum schedule for the instance.

There are at most $D|E|$ x-variables. We then count the number of tuples $ijdeuv$ such that y_{ijdeuv} is a variable in the LP. For fixed $ij \in E, d \in D$ and $u \in [0, q_d)$, there are at most $O(1)$ possibilities for (e, v), as the lengths of sub-windows do not decrease from left to right in the time horizon, except for the last sub-window of each window. Hence the number of y-variables is at most $O\big(\frac{D|E|}{\epsilon}\big) = O\big(\frac{|E| \log n}{\epsilon^2}\big)$.[9] The number of constraints is $O(n + \frac{m|D|}{\epsilon}) = O\big(n + \frac{m \log n}{\epsilon^2}\big)$. The number of non-zeros is at most $O\big(\frac{|E| \log^2 n}{\epsilon^4}\big)$ as each variable appears in at most $O(\frac{D}{\epsilon})$ constraints.

Therefore, by Theorem 2.1, in $O\left(\frac{|E| \log^3 n}{\epsilon^6}\right) = \tilde{O}_\epsilon(|E|)$ time, we can find a solution $(\mathbf{x}, \mathbf{y})$ satisfying the following conditions: Its cost is at most $1 + \epsilon$ times that of the optimum solution to the LP, all variables are non-negative, (12) holds with equalities, and (13) and

[8] A more precise estimation for the case $y_{ijdeuv} = 1$ is $\tau^{(e+1)}_{v} + 1$. But the estimation $T_e + 1$ is good enough.

[9] By cutting job lengths p_{ij} by a factor of ϵ, one can reduce the number of y variables to $O\Big(|E|\big(\frac{1}{\epsilon^2} + \frac{\log n}{\epsilon}\big)\Big)$. But we prioritize on giving a clean algorithm, rather than optimizing the poly$(\log n, \frac{1}{\epsilon})$-factor in the running time.

(14) hold with right sides replaced by $1+\epsilon$ and $(1+\epsilon)\Delta_{d'}$ respectively. For convenience, we call such a solution a $(1+\epsilon)$-approximate solution to LP(11); but keep in mind that it may violate (13) and (14) by a factor of $1+\epsilon$.

4.3 Equivalence of LP(11) and LP(6)

We use $\mathsf{lp}_{(6)}$ and $\mathsf{lp}_{(11)}$ to denote the values of LP(6) and LP(11) respectively. It is easy to show that $\mathsf{lp}_{(11)} \leq \mathsf{lp}_{(6)}$, as one can convert a solution to LP(6) into one to LP(11) with smaller or equal value. The following theorem gives the other direction, proving the equivalence of the two LPs up to a $1+O(\epsilon)$ factor:

▶ **Theorem 4.1** (Equivalence of LP(11) and LP(6))**.** *Let $(\mathbf{x},\mathbf{y})$ be a $(1+\epsilon)$-approximate solution to LP(11). Then in $\tilde{O}_\epsilon(|E|)$-time we can find a solution $\mathbf{z}$ to LP(6) except that (8) is only satisfied with the right-side replaced by $1+\epsilon$, such that the following is true for an absolute constant $c \geq 1$, every $ij \in E$ and integer $t \geq 0$:*

$$\sum_{s+p_{ij}>(1+c\epsilon)t} z_{ijs} \leq \sum_{d:T_{d-1}+1>t} x_{ijd} + \sum_{deuv:T_e+1>t} y_{ijdeuv}.$$

In words, for every $ij \in E$, and every time $t \geq 0$, the fraction of job j scheduled on i with completion time after $(1+c\epsilon)t$ in $\mathbf{z}$ is at most the fraction with completion time after t in $(\mathbf{x},\mathbf{y})$. Then, the following corollary is immediate:

▶ **Corollary 4.2.** *Let $(\mathbf{x},\mathbf{y})$ and $\mathbf{z}$ be defined as in Theorem 4.1. Then the value of $\mathbf{z}$ to LP(6) is at most $1+c\epsilon$ times that of $(\mathbf{x},\mathbf{y})$ to LP(11). This implies that the value of $\mathbf{z}$ to LP(6) is at most $(1+c\epsilon)(1+\epsilon)\cdot\mathsf{lp}_{(11)} \leq (1+O(\epsilon))\cdot\mathsf{lp}_{(6)}$.*

To better present the ideas behind the proof, we only show the existence of such a vector $\mathbf{z}$ in this section. That is, we are not concerned with the running time of the algorithm that constructs $\mathbf{z}$. In the full version of the paper we show how $\mathbf{z}$ can be constructed in nearly-linear time.

So the rest of this section is dedicated to proving the existence of $\mathbf{z}$ satisfying the conditions in Theorem 4.1. Till the end, we fix the solution $(\mathbf{x},\mathbf{y})$. We assume all variables in $(\mathbf{x},\mathbf{y})$ have values being integer multiplies of $1/\Phi$, and $(1+\epsilon)\Phi$ is an integer, for a large enough integer $\Phi > 0$. We fix a machine $i \in M$ and show how to construct the $\mathbf{z}$ values for this i. We create $(1+\epsilon)\Phi$ *mini-machines*, each serving as $1/\Phi$ fraction of the machine i. We create two types of *mini-jobs*:

- For every variable x_{ijd} with positive value, we create Φx_{ijd} mini-jobs of length p_{ij}; we call them *inside-mini-jobs*. Each such inside-mini-job has an *intended completion time* of $T_{d-1}+1$; this is the estimation used in the objective (11).
- For every variable y_{ijdeuv} with positive value, we create Φy_{ijdeuv} mini-jobs of length $\tau_v^{(e+1)} - \tau_{u+1}^{(d)} + 2$; we call them *cross-mini-jobs*. Notice the length may be smaller than p_{ij}. Similarly, the cross-mini-jobs have an *intended completion time* of T_e+1. We define the *blocking interval* of these mini-jobs to be the union of the sub-windows $(\tau_{u'-1}^{(d')}, \tau_{u'}^{(d')}]$ such that *deuv* left-covers $d'u'$. This is indeed an interval. As (13) holds with right side replaced by $1+\epsilon$, every time point is covered by blocking intervals of at most $(1+\epsilon)\Phi$ cross-mini-jobs.

Our goal becomes to schedule all the mini-jobs on the $(1+\epsilon)\Phi$ mini-machines integrally, guaranteeing that the completion time of each mini-job is at most $1+5\epsilon$ times its intended completion time (after we extend the lengths of cross-mini-jobs). The construction of the

schedule is given in Algorithm 2; recall that we are not concerned with the running time in this proof. The solution $\mathbf{z}$ to LP(6) will be the integral schedule scaled by a factor of $\frac{1}{\Phi}$: z_{ijs} is $\frac{1}{\Phi}$ times the number of mini-jobs for j that start at time s in the schedule.

Algorithm 2 Scheduling of mini-jobs on mini-machines for a machine $i \in M$.

1: define a vector σ : cross-mini-jobs $\to$ mini-machines, so that for every mini-machine h, the blocking intervals of $\sigma^{-1}(h)$ are disjoint.
2: **for** $d' \leftarrow 1$ to D **do**
3: **for** every cross-mini-job k for some variable y_{ijdeuv} with $d \le d' \le e+1$ **do**
4: $\mathsf{load}_{\sigma_k} \leftarrow \mathsf{load}_{\sigma_k} + Q_{ijdeuvd'}$
5: **if** $d' = e+1$ **then** append k to the mini-machine σ_k
6: **for** every inside-mini-job k for some variable $x_{ijd'}$ **do**
7: let h be the mini-machine with the smallest load_h
8: $\mathsf{load}_h \leftarrow \mathsf{load}_h + p_{ij}$, append k to the mini-machine h
9: extend the length of each cross-mini-job for a variable y_{ijdeuv} to p_{ij} in the constructed schedule

Step 1 of Algorithm 2 is possible since each point is covered by at most $(1+\epsilon)\Phi$ blocking intervals. When we schedule an inside-mini-job k on a mini-machine h, we increase load_h by the length of k (Step 8). The scheduling of a cross-mini-job k for some variable y_{ijdeuv} is done differently. First the mini-machine σ_k for k is pre-defined. Second, we append k to σ_k only in iteration $d' = e+1$ (Step 5), but we add the length of k to load_{σ_k} piece by piece: In iterations $d' = d, d+1, \cdots, e+1$, we increase the load by $Q_{ijdeuvd'}$ (Step 4). Still we ensure that the load to σ_k contributed by k is equal to the length of k. A mini-job for a variable y_{ijdeuv} may have length smaller than the desired length p_{ij}, so in Step 9 we extend these mini-jobs.

▶ **Lemma 4.3.** *At the end of iteration d' of Loop 2, every mini-machine has a load of at most $T_{d'} - 1 + \Delta_{d'}$.*

Proof. There are two types of loads added to mini-machines during iteration d' of Loop 2: those from cross-mini-jobs, and those from inside-mini-jobs. The total load (from both cross- and inside-mini-jobs) added to all mini-machines is at most $(1+\epsilon)\Phi\Delta_{d'}$: it is precisely Φ times the left-side of (14) for the machine i and d', which is at most $(1+\epsilon)\Phi\Delta_{d'}$ as the constraint is violated only by a factor of $1+\epsilon$.

The total load from cross-mini-jobs added to a mini-machine h in iteration d' is at most $\Delta_{d'}$ as the blocking intervals of all mini-jobs in $\sigma^{-1}(h)$ are disjoint. We need to check the case when one mini-job $k \in \sigma^{-1}(h)$ has blocking interval ending at $\tau_{u'}^{(d')}$ and another mini-job $k' \in \sigma^{-1}(h)$ has blocking interval starting at the time. If the length of the sub-window $(\tau_{u'-1}^{(d')}, \tau_{u'}^{(d')}]$ is at least 2, then the statement holds as we only gave 1 unit length to k and k' in this sub-window. If the length is 1, then because we handled the case in a special way in the definition of left-covering, we did not give any length to k' for the sub-window.

With the observations, we can prove the lemma. Before we add an inside-mini-job k for $x_{ijd'}$ to a mini-machine h in iteration d', the total load of all mini-machines is strictly less than $(1+\epsilon)\Phi\sum_{d''=1}^{d'}\Delta_{d''} = (1+\epsilon)\Phi T_{d'}$ (as the length of k has not been added to the loads yet). Therefore $\mathsf{load}_h < T_{d'}$ before we append k to h. After that, we have $\mathsf{load}_h \le T_{d'} - 1 + p_{ij} \le T_{d'} - 1 + \Delta_{d'}$.

Assume towards the contradiction that the lemma does not hold and consider the first time when the condition is violated. Assume this is at iteration d', and some mini-machine has a load of at least $T_{d'} + \Delta_{d'}$. This must be due to that we add the loads from cross-mini-jobs to

the machine. By our assumption, every mini-machine has a load of at most $T_{d'-1} - 1 + \Delta_{d'-1}$ at the end of iteration $d' - 1$. (A special case is when $d' = 1$; but this can be handled trivially.) As we argued, we add a load of at most $\Delta_{d'}$ from cross-mini-jobs to each mini-machine i in iteration d'. Therefore after we add the loads, every mini-machine has a load of at most $T_{d'-1} - 1 + \Delta_{d'-1} + \Delta_{d'} = T_{d'} - 1 + \Delta_{d'-1} \leq T_{d'} - 1 + \Delta_{d'}$, a contradiction. ◀

Now we consider how Step 9 changes the completion times. The length of a cross-mini-job for a variable y_{ijdeuv} is increased by at most $\eta_d - 1 + \eta_{e+1} - 1 \leq \epsilon\Delta_d + \epsilon\Delta_{e+1} \leq 2\epsilon(\Delta_d + \Delta_e)$. For all cross-mini-jobs assigned to the same mini-machine h, the correspondent intervals $\{d, d+1, \cdots, e\}$ are disjoint. Therefore, a mini-job scheduled in iteration d' of Loop 2 is delayed by at most $2\epsilon(\Delta_1 + \Delta_2 + \cdots + \Delta_{d'}) = 2\epsilon T_{d'}$ units time. In the final schedule constructed by Algorithm 2 the completion time of a mini-job scheduled in iteration d is at most

$$\begin{aligned} &T_d - 1 + \Delta_d + 2\epsilon T_d \leq T_d - 1 + ((1+\epsilon)T_{d-1} + 1) - T_{d-1} + 2\epsilon T_d \\ &= T_d + \epsilon T_{d-1} + 2\epsilon T_d \leq (1 + 5\epsilon)(T_{d-1} + 1). \end{aligned}$$

Setting $c = 5$, Theorem 4.1 follows from that $T_{d-1} + 1$ is the intended completion time of the mini-job.

References

1. Zeyuan Allen-Zhu and Lorenzo Orecchia. Nearly-linear time positive LP solver with faster convergence rate. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing (STOC 2015)*, pages 229–236, 2015.
2. Måns Alskog. Implementation of a fast approximation algorithm for precedence constrained scheduling. Master's thesis, Linköping University, Applied Mathematics, Faculty of Science and Engineering, 2022.
3. Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012. doi:10.4086/toc.2012.v008a006.
4. Nikhil Bansal and Subhash Khot. Optimal long code test with one free bit. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2009)*, pages 453–462, 2009.
5. Nikhil Bansal, Aravind Srinivasan, and Ola Svensson. Lift-and-round to improve weighted completion time on unrelated machines. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing (STOC 2016)*, pages 156–167, 2016.
6. Yair Bartal and Lee-Ad Gottlieb. Near-linear time approximation schemes for steiner tree and forest in low-dimensional spaces. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2021)*, pages 1028–1041, 2021. doi:10.1145/3406325.3451063.
7. A. Bernstein, M. Gutenberg, and T. Saranurak. Deterministic decremental sssp and approximate min-cost flow in almost-linear time. In *Proceedings of the 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2021)*, pages 1000–1008, 2021. doi:10.1109/FOCS52979.2021.00100.
8. Jan van den Brand, Yin Tat Lee, Aaron Sidford, and Zhao Song. Solving tall dense linear programs in nearly linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2020)*, pages 775–788, 2020. doi:10.1145/3357713.3384309.
9. Chandra Chekuri, Sariel Har-Peled, and Kent Quanrud. Fast lp-based approximations for geometric packing and covering problems. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2020)*, pages 1019–1038, 2020.

10 Chandra Chekuri, T.S. Jayram, and Jan Vondrak. On multiplicative weight updates for concave and submodular function maximization. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science (ITCS 2015)*, pages 201–210, 2015.

11 Chandra Chekuri and Kent Quanrud. Near-linear time approximation schemes for some implicit fractional packing problems. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 801–820, 2017.

12 Chandra Chekuri and Kent Quanrud. Fast approximations for metric-tsp via linear programming. *arXiv*, abs/1802.01242, 2018. arXiv:1802.01242.

13 Chandra Chekuri and Kent Quanrud. Randomized MWU for positive LPs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2018)*, pages 358–377, 2018.

14 Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *Proceedings of the 63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2022)*, pages 612–623, 2022. doi:10.1109/FOCS54457.2022.00064.

15 Julia Chuzhoy, Yu Gao, Jason Li, Danupon Nanongkai, Richard Peng, and Thatchaphol Saranurak. A deterministic algorithm for balanced cut with applications to dynamic connectivity, flows, and beyond. In Sandy Irani, editor, *Proceedings of the 61st Annual IEEE Annual Symposium on Foundations of Computer Science (FOCS 2020)*, pages 1158–1167, 2020. doi:10.1109/FOCS46700.2020.00111.

16 Naveen Garg and Jochen Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM Journal on Computing*, 37(2):630–652, 2007. doi:10.1137/S0097539704446232.

17 Shashwat Garg. Quasi-PTAS for scheduling with precedences using LP hierarchies. In *Proceedings of 45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, pages 59:1–59:13, 2018.

18 R. L. Graham. Bounds on multiprocessing timing anomalies. *Siam Journal on Applied Mathematics*, 17(2):416–429, 1969.

19 R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. Discrete Math.*, 4:287–326, 1979.

20 Leslie A. Hall, Andreas S. Schulz, David B. Shmoys, and Joel Wein. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Math. Oper. Res.*, 22(3):513–544, August 1997.

21 John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973. doi:10.1137/0202019.

22 Sungjin Im and Shi Li. Improved approximations for unrelated machine scheduling. In *Proceedings of the Thirty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2023)*, pages 2917–2946, 2023. doi:10.1137/1.9781611977554.ch111.

23 Sungjin Im and Maryam Shadloo. Weighted completion time minimization for unrelated machines via iterative fair contention resolution [extended abstract]. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2020)*, pages 2790–2809, 2020.

24 Klaus Jansen and Lars Rohwedder. On the configuration-LP of the restricted assignment problem. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 2670–2678, 2017.

25 Klaus Jansen and Lars Rohwedder. A quasi-polynomial approximation for the restricted assignment problem. *SIAM Journal on Computing*, 49(6):1083–1108, 2020.

26 Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2014)*, pages 217–226, 2014.

27 Christos Koufogiannakis and N. Young. A nearly linear-time ptas for explicit fractional packing and covering linear programs. *Algorithmica*, 70:648–674, 2013.

28 Christos Koufogiannakis and Neal E. Young. Beating simplex for fractional packing and covering linear programs. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007)*, pages 494–504, 2007. doi:10.1109/FOCS.2007.62.

29 Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS 2015)*, pages 230–249, 2015.

30 Yin Tat Lee, Zhao Song, and Qiuyi Zhang. Solving empirical risk minimization in the current matrix multiplication time. In *Proceedings of the Thirty-Second Conference on Learning Theory (COLT 2019)*, pages 2140–2157, 2019.

31 J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Oper. Res.*, 26(1):22–35, 1978.

32 Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.

33 Elaine Levey and Thomas Rothvo. A $(1+\epsilon)$-approximation for makespan scheduling with precedence constraints using lp hierarchies. *SIAM Journal on Computing*, 50(3):STOC16–201–STOC16–217, 2021.

34 Jason Li. Deterministic mincut in almost-linear time. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2021)*, pages 384–395, 2021. doi:10.1145/3406325.3451114.

35 Shi Li. Scheduling to minimize total weighted completion time via time-indexed linear programming relaxations. *SIAM Journal on Computing*, 49(4):FOCS17–409–FOCS17–440, 2020.

36 Michael Luby and Noam Nisan. A parallel approximation algorithm for positive linear programming. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing (STOC 1993)*, pages 448–457, 1993. doi:10.1145/167088.167211.

37 Alix Munier, Maurice Queyranne, and Andreas S. Schulz. Approximation bounds for a general class of precedence constrained parallel machine scheduling problems. In *Integer Programming and Combinatorial Optimization (IPCO 1998)*, pages 367–382, 1998.

38 Richard Peng. Approximate undirected maximum flows in $O(m\text{polylog}(n))$ time. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016)*, pages 1862–1867, 2016.

39 Serge A. Plotkin, David B. Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20(2):257–301, 1995. doi:10.1287/moor.20.2.257.

40 Paul Purdom. A transitive closure algorithm. *BIT Numerical Mathematics*, 10:76–94, 1970.

41 Maurice Queyranne and Andreas S. Schulz. Approximation bounds for a general class of precedence constrained parallel machine scheduling problems. *SIAM J. Comput.*, 35(5):1241–1253, May 2006.

42 Andreas S. Schulz and Martin Skutella. Scheduling unrelated machines by randomized rounding. *SIAM J. Discret. Math.*, 15(4):450–469, April 2002.

43 Jay Sethuraman and Mark S. Squillante. Optimal scheduling of multiclass parallel machines. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1999)*, pages 963–964, 1999.

44 Farhad Shahrokhi and D. W. Matula. The maximum concurrent flow problem. *J. ACM*, 37(2):318–334, April 1990. doi:10.1145/77600.77620.

45 Jonah Sherman. Area-convexity, ℓ_∞ regularization, and undirected multicommodity flow. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2017)*, pages 452–460, 2017. doi:10.1145/3055399.3055501.

46 David B. Shmoys and Éva Tardos. An approximation algorithm for the generalized assignment problem. *Math. Program.*, 62(1–3):461–474, February 1993.

47 Martin Skutella. Convex quadratic and semidefinite programming relaxations in scheduling. *J. ACM*, 48(2):206–242, March 2001.

48 Ola Svensson. Conditional hardness of precedence constrained scheduling on identical machines. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing (STOC 2010)*, pages 745–754, 2010.

49 Ola Svensson. Santa Claus schedules jobs on unrelated machines. *SIAM Journal on Computing*, 41(5):1318–1341, 2012.

50 Neal E. Young. Randomized rounding without solving the linear program. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1995)*, pages 170–178, 1995.

51 Neal E. Young. Nearly linear-work algorithms for mixed packing/covering and facility-location linear programs, 2014. `arXiv:1407.3015`.

Simulating Markovian Open Quantum Systems Using Higher-Order Series Expansion

Xiantao Li
Department of Mathematics, Pennsylvania State University, University Park, PA, USA

Chunhao Wang
Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA

Abstract

We present an efficient quantum algorithm for simulating the dynamics of Markovian open quantum systems. The performance of our algorithm is similar to the previous state-of-the-art quantum algorithm, i.e., it scales linearly in evolution time and poly-logarithmically in inverse precision. However, our algorithm is conceptually cleaner, and it only uses simple quantum primitives without compressed encoding. Our approach is based on a novel mathematical treatment of the evolution map, which involves a higher-order series expansion based on Duhamel's principle and approximating multiple integrals using scaled Gaussian quadrature. Our method easily generalizes to simulating quantum dynamics with time-dependent Lindbladians. Furthermore, our method of approximating multiple integrals using scaled Gaussian quadrature could potentially be used to produce a more efficient approximation of time-ordered integrals, and therefore can simplify existing quantum algorithms for simulating time-dependent Hamiltonians based on a truncated Dyson series.

2012 ACM Subject Classification Theory of computation → Quantum computation theory

Keywords and phrases Quantum algorithms, open quantum systems, Lindblad simulation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.87

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2212.02051` [43]

Funding Both XL and CW were supported by a seed grant from the Institute of Computational and Data Science (ICDS).
Xiantao Li: XL's research is supported by the National Science Foundation Grants DMS-2111221.
Chunhao Wang: CW acknowledges support from National Science Foundation grant CCF-2238766 (CAREER).

Acknowledgements We thank the anonymous reviewers for the valuable comments.

1 Introduction

The last few decades have witnessed the exciting progress in quantum information science to understand and utilize systems that exhibit quantum properties. In the meantime, quantum algorithms for simulating quantum dynamics have received extensive attention. This is because such simulation algorithms are critical tools in many physics applications, and they have the potential to become the first application (if it is not factoring integers!) once large-scale fault-tolerant quantum computers are available. In fact, simulating quantum dynamics was one of the original motivations Feynman proposed quantum computers [13], who realized the unfavorable scaling for classical algorithms for this task and foresaw the power of quantum computation back in 1982.

Up to now, the majority of the research for simulating quantum dynamics is focused on the "Hamiltonian regime", where the system is governed by Schrödinger evolution and it has no interaction with the environment. Such idealized systems are often referred to as *closed systems*. However, if one believes that the universe is a closed system, then it is reasonable to assume that every quantum system, as a subsystem of the whole universe, is an *open system* because every realistic system is coupled to an uncontrollable environment in a non-negligible way. For example, we always model quantum gates as unitary matrices, while their implementations are always subject to noise induced by the environment no matter how hard one pushes the technology forward.

A key challenge in simulating the dynamics of open quantum systems is the lack of a microscopic description of the dynamics influenced by the physical law of the environment. Even if such a description exists, the degrees of freedom will involve numerous information, which would exceed the capacity of quantum computers. Fortunately, for a special class of open quantum systems, their dynamics can be *fully* described by operators acting on the system. This special class captures the scenario when the system is weakly coupled to the environment and the dynamics of the environment occur at a much faster rate than the system. Intuitively, the environment is fast enough so that the information only flows from the system to the environment and there is no information flowing back. Precisely due to such Markovianity, these open systems are called *Markovian open quantum systems*. Specifically, such dynamics are described in terms of the density matrix ρ by the differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\rho = \mathcal{L}(\rho) \coloneqq -i[H,\rho] + \sum_{j=1}^{m}\left(L_j\rho L_j^\dagger - \frac{1}{2}\left\{L_j^\dagger L_j, \rho\right\}\right) \tag{1}$$

which is referred to as the *Lindblad equation* [27, 15]. The superoperator $\mathcal{L}$ is called the *Lindbladian*, and the L_j's are often called the *jump operators*. The solution to the Lindblad equation is given by

$$\rho_t = e^{\mathcal{L}t}(\rho_0). \tag{2}$$

Here, the superoperator $e^{\mathcal{L}t}$ is a quantum channel for all $t \geq 0$.

It turns out that Markovian open quantum systems are general enough to model many realistic quantum systems in quantum physics [24, 42], quantum chemistry [30, 32], and quantum biology [12, 16, 31]. Computationally, such systems also arise in the context of entanglement preparation [23, 18, 36], thermal state preparation [17], quantum state engineering [41], and modeling the noise of quantum circuits [29, 33, 38].

The first quantum algorithm for simulating Markovian open quantum systems was presented by Kliesch et al. in [20] in 2011 and the complexity has scaling $O(t^2/\epsilon)$ for evolution time t and precision ϵ. In 2017, Childs and Li [7] constructed an improved algorithm with cost $O(t^{1.5}/\sqrt{\epsilon})$. Cleve and Wang [9] pushed the study further by reducing the complexity to nearly optimal: $O(t\,\mathrm{polylog}(t/\epsilon))$, which was the first to achieve the complexity that scales linearly in t and poly-logarithmically in $1/\epsilon$ – that is exponentially better than previous approaches. Recently, researchers have explored these simulation algorithms in various scenarios such as simulating heavy-ion collisions [11], simulating the non-equilibrium dynamics in the Hubbard model [40], simulating the non-equilibrium dynamics in the Schwinger model [10], and preparing thermal states [35].

The quantum algorithm by Cleve and Wang [9] is based on the first-order approximation of $e^{\mathcal{L}t}$, which can be further approximated by a completely positive map whose Kraus operators involve H and L_j's. Due to the inefficiency of the first-order approximation, the building blocks (the implementation of linear combination unitaries) of [9] need to be repeated many

times to simulate a constant-time evolution, which tends to break the poly-logarithmic dependence on $1/\epsilon$. However, it was shown in [9] that the state of the control qubits of the building blocks concentrates to low-Hamming weight states. Thus a compression scheme had to be employed in [9] to exponentially reduce the uses of the building blocks.

In the literature of Hamiltonian simulation, there is an elegant quantum algorithm that uses a truncated Taylor series [3]. This algorithm is conceptually much simpler than its first-order approximation predecessor [4] while keeping the same efficiency. Thus a natural question arises: Can we generalize the truncated Taylor series approach to simulating Lindblad evolution to obtain a much simpler algorithm? It was not known how to achieve this due to the obstacle that higher powers of the Lindbladian are too intricate to keep track of its completely positiveness, which is the key to implementing a superoperator. To demonstrate this challenge, consider the expression of $\mathcal{L}^2$. For simplicity, let us assume $H = 0$ and $m = 1$. We have

$$\begin{aligned}\mathcal{L}^2(\rho) = L^2\rho L^{\dagger 2} - \frac{1}{2}LL^\dagger L\rho L^\dagger - \frac{1}{2}L\rho L^\dagger LL^\dagger - \frac{1}{2}L^\dagger L^2\rho L^\dagger + \frac{1}{4}L^\dagger LL^\dagger L\rho \\ - \frac{1}{2}L\rho L^{\dagger 2}L + \frac{1}{2}L^\dagger L\rho L^\dagger L + \frac{1}{4}\rho L^\dagger LL^\dagger L.\end{aligned} \tag{3}$$

As the above shows, it is highly nontrivial to maintain the completely positive structure in the Taylor series $e^{\mathcal{L}} \approx \sum_{\ell=0}^{K} \frac{\mathcal{L}^\ell}{\ell!}$ even for this overly simplified case where $H = 0$ and $m = 1$.

In this paper, we present a quantum algorithm that takes advantage of a higher-order series expansion based on Duhamel's principle (this principle is briefly discussed in Section 2.1 for readers not familiar with this subject). Our quantum algorithm is conceptually simple and it only contains straightforward applications of simple quantum primitives such as oblivious amplitude amplification for isometries and linear combinations of unitaries (LCU) for channels (presented in the language of block-encodings) [7]. Our approach is inspired by a classical algorithm by Cao and Lu [6] based on the Duhamel's principle. The basic idea is to separate the Lindblad generator into two groups, the first group of which can be expressed as a matrix exponential that immediately induces a completely positive map. By applying the Duhamel's principle repeatedly, a series expansion with arbitrary order of accuracy can be obtained. We prove a rigorous error bound for this truncation. This procedure exhibits some level of resemblance to the time-dependent Hamiltonian simulation method using Dyson series [19]. However, an important focus in the simulation of Lindblad dynamics, due to the presence of jump operators, is to maintain the completely positive property. In addition, we apply Gaussian quadrature, which for any fixed number of quadrature points, has the optimal order of accuracy, to treat the multiple integrals in the series expansion. This approach, compared with the rectangle rule used in [19] and the mid-point rule used in [6], compressed drastically the number of terms in the Kraus form of the completely positive maps. The other added advantage is that it completely eliminates the need for time clocking, which requires either a compression scheme or a quantum sorting network to implement.

We consider a very general model of input, namely, the operators are given by their block-encodings. Informally, a unitary U_A is the block-encoding of A with normalizing factor α if the top-left block of U_A is A/α. This input model abstract but general enough to assume for most realistic physical models. In fact, traditional input models such as local Hamiltonians, sparse Hamiltonians, and a linear combination of tensor product of Paulis can all be efficiently converted to block-encodings. Suppose the operators H, and L_j's are given as block-encodings with corresponding normalizing factors α_0, α_j's, respectively. We define the following norm for the purpose of normalizing the magnitude of the Lindbladian in Equation (1).

$$\|\mathcal{L}\|_{\mathrm{be}} \coloneqq \alpha_0 + \sum_{j=1}^{m} \alpha_j^2. \tag{4}$$

Note that a similar norm $\|\mathcal{L}\|_{\mathrm{pauli}}$ was defined in [7], which is a special case of $\|\mathcal{L}\|_{\mathrm{be}}$ in the context of linear combination of unitaries input model. Our main result is stated as follows.

▶ **Theorem 1** (Informal version of Theorem 11). *For a Lindbladian $\mathcal{L}$ with m jump operators. Suppose we are given a block-encoding U_H of H, and a block-encoding U_{L_j} for each L_j. For all $t, \epsilon > 0$, there exists a quantum algorithm that approximates $e^{\mathcal{L}t}$ in terms of the diamond norm using $O(\tau \operatorname{polylog}(t/\epsilon))$ queries to U_H and U_{L_j} and $O(m\tau \operatorname{polylog}(t/\epsilon))$ additional 1- and 2-qubit gates, where $\tau \coloneqq t\|\mathcal{L}\|_{\mathrm{be}}$.*

Our approach trades off mathematical simplicity for technical conciseness. In fact, the majority of the analysis is devoted to proving the bound of the truncated series, and the accuracy for using a scaled Gaussian quadrature to approximate each layer of a multiple integral. Once the mathematical treatment is established, we obtain an approximate map for $e^{\mathcal{L}t}$ that is completely positive, represented in terms of Kraus operators. Then, it is straightforward to use simple quantum primitives including LCU for channels and oblivious amplitude amplification for isometries to implement this completely positive map. Moreover, it is more direct to obtain the gate complexity that scales linearly in m, for which the dependence was $O(m^2)$ as presented in [9] [1].

In this paper, we focus on time-independent Lindbladians. It is worth noting that our approach easily generalizes to time-dependent Lindbladians. We sketch this generalization in Appendix A.

The rest of the paper is structured as follows. We introduce the preliminaries, including an introduction to Duhamel's principle and the algorithmic tools in Section 2. In Section 3, we present the series expansion based on Duhamel's principle and prove the error bound. In Section 4, we show how to use scaled Gaussian quadrature to approximate multiple integrals. The main theorem is proved in Section 5 and our simulation algorithm is presented in the proof. Finally, we sketch how to generalize our method to simulating time-dependent Lindbladians in Appendix A.

2 Preliminaries

In this paper, we use $\|A\|$ to denote the *spectral norm* of a square matrix A, and we use $\|A\|_1$ to denote its *trace norm*. We use I to denote the identity matrix and we leave its dimension implicitly when it is clear from the context. We use calligraphic font, such as $\mathcal{L}$, $\mathcal{J}$ to denote *superoperators*. In particular, we use $\mathcal{I}$ to denote the *identity map*. We use $\mathcal{K}[A]$ to denote the completely positive map induced by the *Kraus operator* A, i.e.,

$$\mathcal{K}[A](\rho) \coloneqq A\rho A^\dagger \tag{5}$$

for all ρ. The *induced trace norm* of a superoperator $\mathcal{M}$, denoted by $\|\mathcal{M}\|_1$, is defined as $\|\mathcal{M}\|_1 \coloneqq \max\{\|\mathcal{M}(A)\|_1 : \|A\| \le 1\}$. The *diamond norm* of a superoperator $\mathcal{M}$, denoted by $\|\mathcal{M}\|_\diamond$, is defined as $\|\mathcal{M}\|_\diamond \coloneqq \|\mathcal{M} \otimes \mathcal{I}\|_1$, where the identity map $\mathcal{I}$ acts on a Hilbert space with the same dimension as the space $\mathcal{M}$ is acting on.

[1] We realized that it is possible to improve the dependence to $O(m)$ in [9] by a more careful construction of the encoding gate in their compression scheme.

We use *block-encodings* as the efficient description of the operators. Intuitively, we say a unitary U_A block-encodes a matrix A if A appears in the upper-left block of A, i.e.,

$$U_A = \begin{pmatrix} A/\alpha & \cdot \\ \cdot & \cdot \end{pmatrix}, \tag{6}$$

where α is the *normalizing factor*. More formally, an $(n+b)$-qubit unitary U_A is an (α, b, ϵ)-block-encoding of an n-qubit operator A if

$$\left\| A - \alpha(\langle 0^{\otimes b}| \otimes I) U_A (|0^{\otimes b}\rangle \otimes I) \right\| \leq \epsilon, \tag{7}$$

where the identity operator is acting on n qubits.

2.1 Duhamel's principle

For a differential equation written in the form of,

$$u'(t) = Lu + f\big(t, u(t)\big), \quad u(0) = u_0, \tag{8}$$

where L is a linear operator, but f can in principle be a nonlinear function of u.

The Duhamel's principle allows to separate the contribution to the solution from the initial condition and the contribution from the non-homogeneous term. Specifically, we can write the solution as

$$u = v + w, \tag{9}$$

where v satisfies the equation without f,

$$v'(t) = Lv, \quad v(0) = u_0, \tag{10}$$

while w follows the equation

$$w'(t) = Lw + f\big(t, u(t)\big), \quad w(0) = 0, \tag{11}$$

The solution v, due to the fact that Equation (10) is linear and homogeneous, can be simply written as $v(t) = e^{tL}u_0$. On the other hand, the equation for w can be rewritten as $(w(t)e^{-tL})' = e^{-tL} f\big(t, u(t)\big)$, from which a direct integration yields,

$$w(t) = \int_0^t g(t,s)\mathrm{d}s, \quad g(t,s) \coloneqq e^{(t-s)L} f\big(s, u(s)\big). \tag{12}$$

Notice that when s is held fixed, the function $g(t,s)$ also follows a homogeneous equation similar to Equation (10),

$$\frac{d}{dt} g(t,s) = Lg(t,s), \quad \lim_{t \to s} g(t,s) = f\big(s, u(s)\big), \tag{13}$$

which is typically how the Duhamel's principle is expressed.

The derivation of our algorithm will heavily involve the Duhamel's principle, which can be summarized into the following formula,

$$u(t) = e^{tL}u_0 + \int_0^t e^{(t-s)L} f\big(s, u(s)\big)\mathrm{d}s. \tag{14}$$

2.2 Algorithmic tools

Given a completely positive map whose Kraus operators are given as block-encodings, we use the following lemma to probabilistically implement this complete positive map. Note that this lemma is a reformulation of the LCU for channels [7] in the language of block-encodings.

▶ **Lemma 2** (Implementing completely positive maps via block-encodings of Kraus operators [26]). *Let $A_1, \ldots, A_m \in \mathbb{C}^{2^n}$ be the Kraus operators of a completely positive map. Let $U_1, \ldots, U_m \in \mathbb{C}^{2^{n+n'}}$ be their corresponding (s_j, n', ϵ)-block-encodings, i.e.,*

$$\|A_j - s_j(\langle 0| \otimes I)U_j |0\rangle \otimes I)\| \leq \epsilon, \quad \text{for all } 1 \leq j \leq m. \tag{15}$$

Let $|\mu\rangle \coloneqq \frac{1}{\sqrt{\sum_{j=1}^m s_j^2}} \sum_{j=1}^m s_j |j\rangle$. Then $(\sum_{j=1}^m |j\rangle\langle j| \otimes U_j) |\mu\rangle |0\rangle \otimes I$ implements this completely positive map in the sense that

$$\left\| I \otimes \langle 0| \otimes I \left(\sum_{j=1}^m |j\rangle\langle j| \otimes U_j \right) |\mu\rangle |0\rangle |\psi\rangle - \frac{1}{\sqrt{\sum_{j=1}^m s_j^2}} \sum_{j=1}^m |j\rangle A_j |\psi\rangle \right\| \leq \frac{m\epsilon}{\sqrt{\sum_{j=1}^m s_j^2}} \tag{16}$$

for all $|\psi\rangle$.

The following lemma shows how to construct the block-encoding as a linear combination of block-encodings.

▶ **Lemma 3** (Block-encoding of a sum of block-encodings [26]). *Suppose $A \coloneqq \sum_{j=1}^m y_j A_j \in \mathbb{C}^{2^n \times 2^n}$, where $A_j \in \mathbb{C}^{2^n \times 2^n}$ and $y_j > 0$ for all $j \in \{1, \ldots m\}$. Let U_j be an (α_j, a, ϵ)-block-encoding of A_j, and B be a unitary acting on b qubits (with $m \leq 2^b - 1$) such that $B |0\rangle = \sum_{j=0}^{2^b-1} \sqrt{\alpha_j y_j / s} |j\rangle$, where $s = \sum_{j=1}^m y_j \alpha_j$. Then a $(\sum_j y_j \alpha_j, a+b, \sum_j y_j \alpha_j \epsilon)$-block-encoding of $\sum_{j=1}^m y_j A_j$ can be implemented with a single use of $\sum_{j=0}^{m-1} |j\rangle\langle j| \otimes U_j + ((I - \sum_{j=0}^{m-1} |j\rangle\langle j|) \otimes I_{\mathbb{C}^{2^a}} \otimes I_{\mathbb{C}^{2^n}})$ plus twice the cost for implementing B.*

Finally, we need the oblivious amplitude amplification for isometries.

▶ **Lemma 4** (Oblivious amplitude amplification for isometries [9]). *For all $a, b \in \mathbb{N}_+$, let $|\hat{0}\rangle \coloneqq |0\rangle^{\otimes a}$ and $|\mu\rangle$ be arbitrary b-qubit state. For any n-qubit state $|\psi\rangle$, let $|\hat{\psi}\rangle \coloneqq |\hat{0}\rangle |\mu\rangle |\psi\rangle$. Also define $|\hat{\phi}\rangle \coloneqq |\hat{0}\rangle |\phi\rangle$, where $|\phi\rangle$ is a $(b+n)$-qubit state. Let $P_0 \coloneqq |\hat{0}\rangle\langle\hat{0}| \otimes I_{2^b} \otimes I_{2^n}$ and $P_1 \coloneqq |\hat{0}\rangle\langle\hat{0}| \otimes |\hat{\mu}\rangle\langle\hat{\mu}| \otimes I_{2^n}$ be two projectors. Suppose there exists an operator W satisfying*

$$W |\hat{\phi}\rangle = \frac{1}{2} |\hat{\phi}\rangle + \sqrt{\frac{3}{4}} |\hat{\phi}^\perp\rangle, \tag{17}$$

where $|\hat{\phi}\rangle$ satisfies $P_0 |\hat{\phi}^\perp\rangle = 0$. Then it holds that

$$-W(I - 2P_1)W^\dagger(I - 2P_0)W |\hat{\psi}\rangle = |\hat{\phi}\rangle. \tag{18}$$

3 Higher order series expansion based on Duhamel's principle

The goal of our quantum algorithm is to simulate the Lindblad equation defined in Equation (1). In the context of quantum trajectory theory [34], we view the commutator and the anti-commutator terms as the *drifting part*, and the $L_j \rho L_j^\dagger$ terms are regarded as *jump part*. Accordingly, motivated by the numerical method in [6]. We decompose $\mathcal{L}$ into two superoperators, the *drifting part* $\mathcal{L}_{\mathrm{D}}$ and the *jump part* $\mathcal{L}_{\mathrm{J}}$. Namely,

$$\mathcal{L} = \mathcal{L}_{\mathrm{D}} + \mathcal{L}_{\mathrm{J}}, \quad \text{and} \tag{19}$$

$$\mathcal{L}_{\mathrm{D}}(\rho) \coloneqq J\rho + \rho J^{\dagger}, \quad \mathcal{L}_{\mathrm{J}}(\rho) \coloneqq \sum_{j=1}^{m} L_j \rho L_j^{\dagger}. \tag{20}$$

Here we define J as

$$J \coloneqq -iH_{\mathrm{eff}}. \tag{21}$$

where an *effective Hamiltonian* H_{eff} is given by,

$$H_{\mathrm{eff}} \coloneqq H + \frac{1}{2i}\sum_{j=1}^{m} L_j^{\dagger} L_j. \tag{22}$$

Thus $\mathcal{L}_{\mathrm{D}}$ can be viewed as a generalized anti-commutator.

By treating the term with $\mathcal{L}_{\mathrm{D}}$ as a non-homogeneous term, the Duhamel's principle in Equation (14) can be applied, and we get,

$$\rho_t = e^{\mathcal{L}t}(\rho_0) = e^{\mathcal{L}_{\mathrm{D}}t}(\rho_0) + \int_0^t e^{\mathcal{L}_{\mathrm{D}}(t-s)}(\mathcal{L}_{\mathrm{J}}\rho_s)\mathrm{d}s. \tag{23}$$

Note that the solution ρ_s is still involved in the integral on the right hand side. Therefore, this equation does not provide an explicit formula for the solution; Rather, it offers an integral representation of the Lindblad equation. Nevertheless, one can apply Equation (14) again to ρ_s in the integral. After K such iterations, we arrive at

$$\begin{aligned}\rho_t = {} & e^{\mathcal{L}_{\mathrm{D}}t}(\rho_0) \\ & + \sum_{k=1}^{K} \int_{0\le s_1\le\cdots\le s_k\le t} e^{\mathcal{L}_{\mathrm{D}}(t-s_k)}\mathcal{L}_{\mathrm{J}} e^{\mathcal{L}_{\mathrm{D}}(s_k-s_{k-1})}\mathcal{L}_{\mathrm{J}}\cdots e^{\mathcal{L}_{\mathrm{D}}(s_2-s_1)}\mathcal{L}_{\mathrm{J}} e^{\mathcal{L}_{\mathrm{D}}(s_1)}(\rho_0)\mathrm{d}s_1\cdots\mathrm{d}s_k \\ & + \int_{0\le s_1\le\cdots\le s_{K+1}\le t} e^{\mathcal{L}_{\mathrm{D}}(t-s_{K+1})}\mathcal{L}_{\mathrm{J}} e^{\mathcal{L}_{\mathrm{D}}(s_{K+1}-s_K)}\mathcal{L}_{\mathrm{J}}\cdots e^{\mathcal{L}_{\mathrm{D}}(s_2-s_1)}\mathcal{L}_{\mathrm{J}}(\rho_{s_1})\mathrm{d}s_1\cdots\mathrm{d}s_{K+1}.\end{aligned} \tag{24}$$

Now notice that the first two terms on the right hand side only depend on the initial density matrix, and thus they are amenable to numerical approximations. Meanwhile, the last term will be regarded as a truncation error, which will be bounded later.

We first derive the Kraus representation of $e^{\mathcal{L}_{\mathrm{D}}t}$, which is the first term in the expansion, but also appears in each integral. The Kraus form can be obtained from a Taylor series. To see this, let us consider the case where $\mathcal{L}_{\mathrm{D}}$ is acting on a pure state $|\psi\rangle$:

$$\frac{\mathrm{d}}{\mathrm{d}t}|\psi\rangle = J|\psi\rangle. \tag{25}$$

Using the chain rule, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}|\psi\rangle\langle\psi| = J|\psi\rangle\langle\psi| + |\psi\rangle\langle\psi|J^{\dagger} = \mathcal{L}_{\mathrm{D}}(|\psi\rangle\langle\psi|). \tag{26}$$

The above two equations also hold for general states ρ by linearity. Hence, solving Equation (25) and Equation (26) yields

$$e^{\mathcal{L}_{\mathrm{D}}t} = \mathcal{K}\left[e^{tJ}\right]. \tag{27}$$

Now, we adopt the notation from [6],

$$\mathcal{F}_k(s_k,\dots,s_1) \coloneqq \mathcal{K}[e^{J(t-s_k)}]\mathcal{L}_{\mathrm{J}}\mathcal{K}[e^{J(s_k-s_{k-1})}]\mathcal{L}_{\mathrm{J}}\cdots\mathcal{K}[e^{J(s_2-s_1)}]\mathcal{L}_{\mathrm{J}}\mathcal{K}[e^{J(s_1-0)}]. \tag{28}$$

We further define

$$\mathcal{G}_K(t) \coloneqq \mathcal{K}[e^{Jt}] + \sum_{k=1}^{K}\int_{0\le s_1\le\cdots\le s_k\le t}\mathcal{F}_k(s_k,\dots,s_1)\,\mathrm{d}s_1\cdots\mathrm{d}s_k. \tag{29}$$

At this point, the problem is reduced to approximating $e^{\mathcal{L}t}$ by $\mathcal{G}_M(t)$. We first prove an error bound.

▶ **Lemma 5.** *It holds that*

$$\left\|e^{\mathcal{L}t} - \mathcal{G}_K(t)\right\|_\diamond \le \frac{(2\|\mathcal{L}\|_{\mathrm{be}}t)^{K+1}}{(K+1)!}. \tag{30}$$

The proof of Lemma 5 is shown in the full version of this paper [43].

Eventually, we need to approximate the Kraus operator e^{Jt} in our quantum algorithm. This can be done by a truncated Taylor series. For notational simplicity, we define

$$\mathcal{J}_{K'} = \mathcal{K}\left[\sum_{\ell=0}^{K'}\frac{J^\ell t^\ell}{\ell!}\right]. \tag{31}$$

We quantify the error of this approximation in the following lemma.

▶ **Lemma 6.** *Suppose that $k\in\mathbb{N}$ such that $(k+1)!\ge 2\|J\|^{k+1}t^{k+1}$. Let $\mathcal{J}_k$ be defined in Equation* (31)*. It holds that*

$$\left\|\mathcal{K}[e^{Jt}] - \mathcal{J}_k(t)\right\|_\diamond \le \frac{8e^{\|\mathcal{L}\|_{\mathrm{be}}t}\|\mathcal{L}\|_{\mathrm{be}}^{k+1}t^{k+1}}{(k+1)!}. \tag{32}$$

The proof of Lemma 6 is shown in the full version of this paper [43].

We also provide the following useful lemma, which will be used in the final analysis of our algorithm

▶ **Lemma 7.** *Suppose that $k,K'\in\mathbb{N}$ such that $(K'+1)!\ge 8e^{\|\mathcal{L}\|_{\mathrm{be}}t}\|\mathcal{L}\|_{\mathrm{be}}^{K'+1}t^{K'+1}$. Let $\mathcal{J}_{K'}$ be defined in Equation* (31)*, and $\mathcal{F}_k$ be defined in Equation* (28)*. It holds that*

$$\begin{aligned}&\|\mathcal{J}_{K'}(t-s_m)\mathcal{L}_{\mathrm{J}}\mathcal{J}_{K'}(s_m-s_{m-1})\mathcal{L}_{\mathrm{J}}\cdots\mathcal{J}_{K'}(s_2-s_1)\mathcal{L}_{\mathrm{J}}\mathcal{J}_{K'}(s_1) - \mathcal{F}_k(s_k,\dots,s_1)\|_\diamond\\ &\qquad\le \frac{8e^{\|\mathcal{L}\|_{\mathrm{be}}t}\|\mathcal{L}\|_{\mathrm{be}}^{K'+1}}{(K'+1)!}(2\|\mathcal{L}\|_{\mathrm{be}})^k 2^k t^{K'+1}.\end{aligned} \tag{33}$$

The proof of Lemma 7 is shown in the full version of this paper [43].

4 Approximating multiple integrals using scaled Gaussian quadrature

To obtain an algorithm that can be directly implemented, we apply Gaussian quadrature formulas to approximate the multiple integrals in Equation (29). Due to their optimal accuracy, the number of terms in the approximation is significantly compressed. Typically, quadrature error depends on the smoothness of the function. For this purpose, we first bound the derivatives of $\mathcal{F}_k$.

▶ **Lemma 8.** *For all $k' \in [k]$, it holds that*

$$\left\| \frac{\mathrm{d}^{k'}}{\mathrm{d}s_j^{k'}} \mathcal{F}_k \right\|_\diamond \leq 2^{2k'+k} \|\mathcal{L}\|_{\mathrm{be}}^k \|J\|^{k'}. \tag{34}$$

The proof of Lemma 8 is shown in the full version of this paper [43].

We now discuss the quadrature approximation for the integral in Equation (29):

$$\int_{0 \leq s_1 \leq \cdots \leq s_k \leq t} \mathcal{L}_{\mathrm{J}} \mathcal{K}[e^{J(s_m - s_{m-1})}] \mathcal{L}_{\mathrm{J}} \cdots \mathcal{K}[e^{J(s_2 - s_1)}] \mathcal{L}_{\mathrm{J}} \mathcal{K}[e^{J(s_1 - 0)}] \, \mathrm{d}s_1 \cdots \mathrm{d}s_k. \tag{35}$$

For optimal accuracy, we use Gaussian quadrature. In the Gaussian quadrature rule, the interpolation nodes $0 \leq \hat{s}_1 \leq \cdots \leq \hat{s}_q \leq t$ and the weights $w_1, \ldots, w_q$ are independent of the function and can be pre-computed. More specifically, let $\{\mathscr{P}_i(x)\}_i$ be the standard Legendre polynomials. They are an orthonormal family of polynomials in the sense that

$$\int_{-1}^{1} \mathscr{P}_r(x) \mathscr{P}_s(x) \, \mathrm{d}x = \begin{cases} 0 & r \neq s, \\ 1 & r = s. \end{cases} \tag{36}$$

By a simple scaling,

$$\hat{\mathscr{P}}(x) \coloneqq \mathscr{P}\left(\frac{2x}{t} - 1\right). \tag{37}$$

we obtain the functions $\hat{\mathscr{P}}$ that are orthogonal for the interval $[0, t]$. Let $\{\hat{s}_i\}_{i=1}^n$ be the roots of the n-th degree polynomial $\hat{\mathscr{P}}_q$. We also define

$$\pi_q(x) \coloneqq (x - \hat{s}_1)(x - \hat{s}_2) \cdots (x - \hat{s}_q). \tag{38}$$

Then the i-th Lagrange polynomial for points $\hat{s}_1, \ldots, \hat{s}_q$ is

$$\mathscr{L}_{q-1,i}(x) = \frac{\pi_q(x)}{(x - \hat{s}_i)\pi_q'(\hat{s}_i)}. \tag{39}$$

Once the quadrature points are selected, the weight of the Gaussian quadrature can be expressed as

$$w_i = \int_0^t \mathscr{L}_{q-1,i}(x) \, \mathrm{d}x, \tag{40}$$

which can be direct deduced from a polynomial interpolation.

We refer to $\hat{s}_1, \ldots, \hat{s}_q$ as the *canonical quadrature points* and $w_1, \ldots, w_q$ as the *canonical weights*. In approximating $\int_0^t f(x) \, \mathrm{d}x$ using $\sum_{j=1}^q f(\hat{s}_j) w_j$, the error follows the standard bound,

$$E_q[f] = \int_0^t f(x) \, \mathrm{d}x - \sum_{j=1}^q f(\hat{s}_j) w_j = \frac{f^{(2q)}(\xi)}{(2q)!} \int_0^t \pi_q(x)^2 \mathrm{d}x, \tag{41}$$

for some $\xi \in [0, t]$.

To estimate the integral term in the error, we notice that,

$$\pi_q(x) = \frac{t^q q!}{2^q (2q-1)!!} \mathscr{P}_q\left(\frac{2x}{t} - 1\right), \tag{42}$$

The coefficient on the right hand side is determined by observing that $\pi_q(x)$ is a monic polynomial, and the leading coefficient of the standard Legendre polynomial is $(2q-1)!!/q!$. The notation !! indicates a double factorial, i.e., $(2q-1)!! = (2q-1)(2q-3)\cdots 1$ and we use the convention that $(-1)!! = 1$.

Combining these formulas, we arrive at an explicit bound

$$|E_q[f]| = \frac{\left|f^{(2q)}\right|(\xi)t^{2q+1}(q!)^2}{(2q)!2^{2q}(2q-1)!!(2q+1)!!} \leq \frac{\left|f^{(2q)}\right|(\xi)t^{2q+1}q}{(2q)!2^{4q-1}} \tag{43}$$

for some $\xi \in [0,t]$, where the inequality follows from the fact that $q!! = 2^{q/2}(q/2)!$ for even q. Here we also used the identity,

$$\int_{-1}^{1} \mathscr{P}_q^2(x)\,\mathrm{d}x = \frac{2}{2q+1}.$$

We hold t fixed. For an interval $[0, s_k]$ with $s_k \leq t$, we use a scaled canonical quadrature points and weights: $s_k\hat{s_1}/t, \ldots, s_k\hat{s}_q/t$, and $s_k w_1/t, \ldots, s_k w_q/t$. Then, $\int_0^{s_k} f(x)\,\mathrm{d}x$ can be approximated by the scaled quadrature points and weights with the same error bound:

$$\int_0^{s_k} f(x)\,\mathrm{d}x = \sum_j^n f\left(\frac{s_k\hat{s}_j}{t}\right)\frac{s_k w_j}{t} + \mathcal{O}\left(\frac{\left\|f^{(2n)}\right\|_\infty s_k^{2n+1} n}{(2n)!2^{4n-1}}\right). \tag{44}$$

For each $j \in [n]$, define the functions u_k and v_k as

$$u_j(x) \coloneqq x\hat{s}_j/t \tag{45}$$
$$v_j(x) \coloneqq x w_j/t. \tag{46}$$

Note that for any s_ℓ, $\{u_j(s_\ell)\}_{j=1}^q$ are the scaled canonical quadrature point and $\{v_j(s_\ell)\}_{j=1}^q$ are the scaled weights.

To simplify the notation, we extend Equation (45) and define

$$\hat{x}_{(j_k)} \coloneqq \hat{s}_{j_k}, \quad \text{and} \quad \hat{x}_{(j_k,\ldots,j_{k-\ell})} \coloneqq u_{j_{k-\ell}} \circ \cdots \circ u_{j_{k-1}}(\hat{s}_{j_k}) \text{ for all } 1 \leq \ell \leq k-1, \tag{47}$$
$$\hat{w}_{(j_k)} \coloneqq w_{j_k}, \quad \text{and} \quad \hat{w}_{(j_k,\ldots,j_{k-\ell})} \coloneqq v_{j_{k-\ell}}(\hat{x}_{(j_k,\ldots,j_{k-\ell+1})}) \text{ for all } 1 \leq \ell \leq k-1. \tag{48}$$

With these notations, the approximation of the integral in Equation (35) becomes,

$$\sum_{j_1=1}^{q} \cdots \sum_{j_k=1}^{q} \mathcal{F}_k\left(\hat{x}_{(j_k)}, , \ldots, \hat{x}_{(j_k,\ldots,j_1)}\right)\hat{w}_{(j_k)} \cdots \hat{w}_{(j_k,\ldots,j_1)}. \tag{49}$$

We first show some useful properties of the quadrature weights.

▶ **Lemma 9.** *for all $\ell \in \{0,\ldots,q\}$, it holds that*

$$\sum_{j=1}^{q} w_j \hat{s_j}^\ell = \frac{t^{\ell+1}}{\ell+1}. \tag{50}$$

In particular, when $\ell = 0$

$$\sum_{j=1}^{q} w_j = t. \tag{51}$$

For all positive integers k, ℓ with $\ell < k$, it also holds that

$$\sum_{j_k=1}^{q} \cdots \sum_{j_{k-\ell}=1}^{q} \hat{w}_{(j_k)} \cdots \hat{w}_{(j_k,\ldots,j_{k-\ell})} = \frac{t^{\ell+1}}{(\ell+1)!}. \tag{52}$$

In particular, when $\ell = k-1$, it holds that

$$\sum_{j_k=1}^{q} \cdots \sum_{j_1=1}^{q} \hat{w}_{(j_k)} \cdots \hat{w}_{(j_k,\ldots,j_1)} = \frac{t^{k}}{(k+1)!}. \tag{53}$$

The proof of Lemma 9 is shown in the full version of this paper [43].

With the bound on the derivatives of the integrand in Lemma 8 and the Gaussian quadrature error Equation (43), we can estimate the overall quadrature error, as stated in the following lemma,

▶ **Lemma 10.** *It holds that*

$$\begin{aligned} &\left\| \int_{0\le s_1\le\cdots\le s_k\le t} \mathcal{F}_k(s_k,\ldots,s_1)\,\mathrm{d}s_1\cdots\mathrm{d}s_k \right. \\ &\quad \left. - \sum_{j_1=1}^{q}\cdots\sum_{j_k=1}^{q} \mathcal{F}_k\left(\hat{x}_{(j_k)},,\ldots,\hat{x}_{(j_k,\ldots,j_1)}\right)\hat{w}_{(j_k)}\cdots\hat{w}_{(j_k,\ldots,j_1)} \right\|_\diamond \\ &= O\left(\frac{(2t)^{k-1}2^{k+1}\|\mathcal{L}\|_{\mathrm{be}}^{k}\|J\|^{(2q)}t^{2q+1}q}{(k-1)!(2q)!}\right). \end{aligned} \tag{54}$$

The proof of Lemma 10 is shown in the full version of this paper [43].

5 Quantum algorithm and the proof of the main theorem

In this section, we prove the main theorem and describe the algorithm in the proof. Our algorithm constructs a segment for constant evolution time, i.e., $t\|\mathcal{L}\|_{\mathrm{be}} = \Theta(1)$. For arbitrary evolution time, we just repeat the simulation segment $O(t\|\mathcal{L}\|_{\mathrm{be}})$ times with a scaled precision parameter.

We first present the higher order approximation of $e^{\mathcal{L}}$ as a completely positive map with explicit Kraus operators. Then we use Lemma 2 to implement this completely positive map with success probability 1/4, which can be calculated by analyzing the normalizing constants of the block-encodings of the Kraus operators. Then we show that the special state $|\mu\rangle$ required by Lemma 2 can be efficiently prepared. Finally, we analyze the error introduced by using a truncated Taylor series to approximate e^{Jt}, which is part of the Kraus operators.

▶ **Theorem 11.** *Suppose we are given an (α_0, a, ϵ')-block-encoding U_H of H, and an (α_j, a, ϵ')-block-encoding U_{L_j} for each L_j. For all $t, \epsilon \ge 0$ with $\epsilon' \le \epsilon/(t(\|\mathcal{L}\|_{\mathrm{be}})$, there exists a quantum algorithm for simulating $e^{\mathcal{L}t}$ using*

$$O\left(t\|\mathcal{L}\|_{\mathrm{be}}\frac{\log(t\|\mathcal{L}\|_{\mathrm{be}}/\epsilon)}{\log\log(t\|\mathcal{L}\|_{\mathrm{be}}/\epsilon)}\right) \tag{55}$$

queries to U_H and U_{L_j} and

$$O\left(mt\|\mathcal{L}\|_{\mathrm{be}}\left(\frac{\log(t\|\mathcal{L}\|_{\mathrm{be}}/\epsilon)}{\log\log(t\|\mathcal{L}\|_{\mathrm{be}}/\epsilon)}\right)^2\right) \tag{56}$$

additional 1- and 2-qubit gates.

Proof. We describe our simulation algorithm and prove the theorem as follows.

Completely-positive approximation. The final approximation to $e^{\mathcal{L}t}$ is

$$\mathcal{K}[e^{Jt}] + \sum_{k=1}^{K} \sum_{j_1=1}^{q} \cdots \sum_{j_k=1}^{q} \mathcal{F}_K\left(\hat{x}_{(j_k)}, , \ldots, \hat{x}_{(j_k,\ldots,j_1)}\right) \hat{w}_{(j_k)} \cdots \hat{w}_{(j_k,\ldots,j_1)}, \tag{57}$$

which is a completely positive map and the block-encodings of its Kraus operators can be easily obtained by a product of the block-encoding U_H of H, and the block-encodings U_{L_j} of L_j as well as positive factors determined by Taylor's expansion and the Gaussian quadrature weights. More specifically, define the index sets $\mathscr{I}$ as

$$\mathscr{I} \coloneqq \{k, \ell_1, \ldots, \ell_k, j_1, \ldots, j_k : k \in [K], \ell_1, \ldots, \ell_k \in [m], j_1 \ldots, j_k \in [q]\}. \tag{58}$$

The completely positive map in Equation (57) can be written as

$$A_0 \rho A_0^\dagger + \sum_{j \in \mathscr{I}} A_j \rho A_j^\dagger, \tag{59}$$

with $A_0 \coloneqq e^{Jt}$, and

$$A_j = \sqrt{\hat{w}_{(j_k)} \cdots \hat{w}_{(j_k,\ldots,j_1)}} e^{J(t-\hat{x}_{(j_k)})} L_{\ell_k} \cdots e^{J(\hat{x}_{(j_k,\ldots,j_2)} - \hat{x}_{(j_k,\ldots,j_1)})} L_{\ell_1} e^{J(\hat{x}_{(j_k,\ldots,j_1)})}, \tag{60}$$

for $j = (k, \ell_1, \ldots, \ell_{k-1}, j_1, \ldots, j_k) \in \mathscr{I}$.

Setting parameters for 1/4 success probability. We use Lemma 2 to implement the above map, and the success probability is determined by the sum-of-squares of the normalizing constants of the Kraus operators.

We first consider the Kraus operators of

$$\mathcal{F}_k(s_k, \ldots, s_1) = \mathcal{K}[e^{J(t-s_k)}] \mathcal{L}_{\mathrm{J}} \mathcal{K}[e^{J(s_k - s_{k-1})}] \mathcal{L}_{\mathrm{J}} \cdots \mathcal{K}[e^{J(s_2-s_1)}] \mathcal{L}_{\mathrm{J}} \mathcal{K}[e^{J(s_1-0)}], \tag{61}$$

for any $s_1 \leq \cdots \leq s_k$. For each $\mathcal{K}[e^{Js}]$, we use Lemma 3 to approximate its block-encoding as a truncated Taylor series. For the convenience of analysis, let us for now assume an infinite Taylor series is implemented. The normalizing constant of the block-encoding for e^{Js} is then

$$\sum_{\ell=0}^{\infty} \frac{s^\ell (\alpha_0 + \frac{1}{2}\sum_{j=1}^{m} \alpha_j^2)}{\ell!} = e^{s\|\mathcal{L}\|_{\mathrm{be}}}. \tag{62}$$

As a result, the sum-of-squares of the normalizing constants of the Kraus operators of $\mathcal{F}_k(s_k, \ldots, s_1)$ is

$$\sum_{j_1,\ldots,j_k=0}^{m} e^{2\|L\|_{\mathrm{be}}(t-s_k)} e^{2\|L\|_{\mathrm{be}}(s_k-s_{k-1})} \cdots e^{2\|L\|_{\mathrm{be}}(s_1-0)} \alpha_{j_1}^2 \cdots \alpha_{j_k}^2 = e^{2\|L\|_{\mathrm{be}} t} \left(\sum_{j=1}^{m} \alpha_j^2\right)^k \tag{63}$$

For the approximation of the integral,

$$\sum_{j_1=1}^{q} \cdots \sum_{j_k=1}^{q} \mathcal{F}_k\left(\hat{x}_{(j_k)}, , \ldots, \hat{x}_{(j_k,\ldots,j_1)}\right) \hat{w}_{(j_k)} \cdots \hat{w}_{(j_k,\ldots,j_1)}, \tag{64}$$

the sum-of-squares of the normalizing constants of its Kraus operators is

$$e^{2\|\mathcal{L}\|_{\text{be}}t}\left(\sum_{j=1}^{m}\alpha_j^2\right)^k\sum_{j_1=1}^{q}\cdots\sum_{j_k=1}^{q}\hat{w}_{(j_k)}\cdots\hat{w}_{(j_k,\ldots,j_1)}=\frac{t^k}{(k-1)!}e^{2\|\mathcal{L}\|_{\text{be}}t}\left(\sum_{j=1}^{m}\alpha_j^2\right)^k. \tag{65}$$

Finally, for the approximation

$$\mathcal{K}[e^t]+\sum_{k=1}^{K}\sum_{j_1=1}^{q}\cdots\sum_{j_k=1}^{q}\mathcal{F}_k\left(\hat{x}_{(j_k)},,\ldots,\hat{x}_{(j_k,\ldots,j_1)}\right)\hat{w}_{(j_k)}\cdots\hat{w}_{(j_k,\ldots,j_1)}, \tag{66}$$

the sum-of-squares of the normalizing constants of its Kraus operators is

$$e^{2\|\mathcal{L}\|_{\text{be}}t}+\sum_{k=1}^{K}\frac{t^k}{(k-1)!}e^{2\|\mathcal{L}\|_{\text{be}}t}\left(\sum_{j=1}^{m}\alpha_j^2\right)^k \tag{67}$$

$$=e^{2\|\mathcal{L}\|_{\text{be}}t}+te^{2\|\mathcal{L}\|_{\text{be}}t}\sum_j\alpha_j^2\sum_{k=1}^{K}\frac{t^{k-1}(\sum_j\alpha_j^2)^{k-1}}{(k-1)!} \tag{68}$$

$$\leq e^{2\|\mathcal{L}\|_{\text{be}}t}+t\sum_j\alpha_j^2e^{2\|\mathcal{L}\|_{\text{be}}t}e^{t\sum_j\alpha_j^2}. \tag{69}$$

Note that the inequality above provides an upper bound for the sum-of-squares of the normalizing constants. There is a closed-form expression for this quantity. By setting the right hand side to 2, and solve the equation, the above upper bound implies that t must satisfy

$$t\|\mathcal{L}\|_{\text{be}}=\Theta(1). \tag{70}$$

Then we use Lemma 4 to boost the success probability to 1 with only three application of the circuit.

Determining truncation orders. Now, we analyze the error by setting $t\|\mathcal{L}\|_{\text{be}}=\Theta(1)$. By Lemma 5 and Lemma 10, the total approximation error can be bounded by the following.

$$\left\|e^{\mathcal{L}t}-\mathcal{K}[e^{Jt}]-\sum_{k=1}^{K}\sum_{j_1=1}^{q}\cdots\sum_{j_k=1}^{q}\mathcal{F}_k\left(\hat{x}_{(j_k)},,\ldots,\hat{x}_{(j_k,\ldots,j_1)}\right)\hat{w}_{(j_k)}\cdots\hat{w}_{(j_k,\ldots,j_1)}\right\|_\diamond \tag{71}$$

$$\leq\frac{(2\|\mathcal{L}\|_{\text{be}})^{K+1}}{(K+1)!}+O\left(\sum_{k=1}^{K}\frac{(2t)^{k-1}2^{k+1}\|\mathcal{L}\|_{\text{be}}^k\|J\|^{(2q)}t^{2q+1}q}{(k-1)!(2q)!}\right) \tag{72}$$

$$=\frac{(2\|\mathcal{L}\|_{\text{be}})^{K+1}}{(K+1)!}+\frac{\|J\|^{2q}t^{2q+1}q}{(2q)!}O\left(\sum_{k=1}^{K}\frac{(2t)^{k-1}2^{k+1}\|\mathcal{L}\|_{\text{be}}^k}{(k-1)!}\right) \tag{73}$$

$$=\frac{(2\|\mathcal{L}\|_{\text{be}})^{K+1}}{(K+1)!}+\frac{\|J\|^{2q}t^{2q+1}q}{(2q)!}O\left(e^{4t\|\mathcal{L}\|_{\text{be}}}\right) \tag{74}$$

$$\leq\frac{(2\|\mathcal{L}\|_{\text{be}})^{K+1}}{(K+1)!}+\frac{\|\mathcal{L}\|_{\text{be}}^{2q}t^{2q+1}q}{(2q)!}O\left(e^{4t\|\mathcal{L}\|_{\text{be}}}\right), \tag{75}$$

where the last inequality follows from

$$\|J\|\leq\|H\|+\frac{1}{2}\sum_j\|L_j\|^2\leq\alpha_0+\frac{1}{2}\sum_j\alpha_j^2=\|\mathcal{L}\|_{\text{be}}. \tag{76}$$

With $t\|\mathcal{L}\|_{\text{be}} = \Theta(1)$, it suffices to set

$$K, q = O\left(\frac{\log(1/\epsilon)}{\log\log(1/\epsilon)}\right) \tag{77}$$

to make the approximation error at most $\epsilon/2$.

Applying the main algorithmic tool (Lemma 2). In Lemma 2, we need to prepare the special state $|\mu\rangle$ which encodes a superposition of normalizing constants of all Kraus operators. Now we show how to efficiently prepare this state. First observe that the normalizing constant for A_j in Equation (60) is

$$\sqrt{\hat{w}_{(j_k)}\cdots\hat{w}_{(j_k,\ldots,j_1)}}e^{\|\mathcal{L}\|_{\text{be}}(t-\hat{x}_{(j_k)})}\alpha_{\ell_k}\cdots e^{\|\mathcal{L}\|_{\text{be}}(\hat{x}_{(j_k,\ldots,j_2)}-\hat{x}_{(j_k,\ldots,j_1)})}\alpha_{\ell_1}e^{\|\mathcal{L}\|_{\text{be}}(\hat{x}_{(j_k,\ldots,j_1)})} \tag{78}$$

$$= \sqrt{\hat{w}_{(j_k)}\cdots\hat{w}_{(j_k,\ldots,j_1)}}e^{\|\mathcal{L}\|_{\text{be}}t}\alpha_{\ell_k}\cdots\alpha_{\ell_1} \tag{79}$$

$$= \frac{1}{\sqrt{t^{k(k-1)/2}}}\sqrt{w_{j_k}\hat{s}_{j_k}^{k-1}w_{j_{k-1}}\hat{s}_{j_{k-1}}^{k-2}\cdots w_{j_2}\hat{s}_{j_2}w_{j_1}}e^{\|\mathcal{L}\|_{\text{be}}t}\alpha_{\ell_k}\cdots\alpha_{\ell_1}, \tag{80}$$

and the normalizing constant for A_0 is $e^{\|\mathcal{L}\|_{\text{be}}t}$. Note that $e^{\|\mathcal{L}\|_{\text{be}}t}$ appears in every amplitude of $|\mu\rangle$ and therefore can be ignored. We use three registers in $|\mu\rangle$. The first register contains K qubits and it encodes k in unary representation, i.e., we use $\left|1^k0^{K-1}\right\rangle$ to represent k. The second register contains k subregisters of $\log m$ qubits to represent $\ell_1,\ldots,\ell_k$. The third register contains k subregisters of $\log q$ qubits to represent $j_1,\ldots,j_k$. We first prepare the normalized version of the state $\sum_{k=0}^{K}\frac{1}{\sqrt{t^{k(k-1)/2}}}\left|1^k0^{K-1}\right\rangle$, which can be done using $O(K)$ gates: we apply a rotation on the first qubit, and then apply a rotation on each subsequent qubit controlled by the previous qubit. For the second register, in each subregister we prepare the normalized version of $\sum_{j=1}^{m}\alpha_j|j\rangle$. The total gate cost for the second register is $O(Km)$. For the ℓ-th subregister of the third register, we prepare the normalized version of $\sum_{j=1}^{q}\sqrt{w_j s_j^{\ell-1}}|j\rangle$. The total gate cost for the third register is $O(Kq)$. Note that each gate acting on the ℓ-th subregister of the second and the third register is further controlled on the ℓ-th qubit of the first register, which effects the truncation. Therefore, the total gate cost for preparing $|\mu\rangle$ is $O(K(m+q))$.

Now, we show how to use Lemma 3 to approximate the block-encoding U_s of e^{Js} for any $0 < s \le t$, where s is provided in a time register containing $|s\rangle$. Here we use K' to denote the Taylor series truncation error. So we need to use Lemma 3 to implement a block-encoding of $\sum_{k=0}^{K'}s^kJ^k/k!$. Recall that $J = -iH - \frac{1}{2}\sum_{j=1}^{m}L_j^\dagger L_j$. In Lemma 3, we need to implement the B gate for preparing a superposition of coefficients. We use $K'+1$ control registers: the first register contains K qubits which encode k in unary; each subsequent register contains $O(\log(m))$ qubits. The B gate is implemented as follows. Controlled by the time register $|s\rangle$, we implement the normalized version of the state $\sum_{k=0}^{K'}\sqrt{s^k/k!}\left|1^k0^{K'-k}\right\rangle$ on the first control register, which can be done with $O(K)$ controlled-rotations. For each subsequent control resister, we implement the normalized version of the state $|0\rangle + \sum_{j=1}^{m}\sqrt{\alpha_j^2/2}|j\rangle$, which costs $O(m)$ gates. The controlled operation $\sum_j|j\rangle\langle j|\otimes A_j$ can be implemented by the controlled-U_H and controlled-U_{L_j} controlled by the $K+1$ control registers. Therefore, the total gate cost for implementing B is $O(K'm)$. The controlled rotations on the first control register controlled by the time register costs $O(\text{poly}(b))$ gates where b is the bits used to represent s. It suffices to set $b = O(\log(1/\epsilon))$ for a precise representation of s within ϵ. As a result, the cost $O(\text{poly}(b))$ is not dominating. As a result, the total gate cost for implementing $\sum_s|s\rangle\langle s|\otimes U_s$ is $O(Km)$.

Additional approximation. It is important to note that by a direct application of Lemma 3, the error of the block-encoding we implement is $O(\epsilon' e^{s\|\mathcal{L}\|_{\text{be}}})$. However, a more careful analysis shows a much better error bound: first assume we had implemented the infinity Taylor series. Then the error ϵ' of each block-encoding will cause error for the implementation that is bounded by[2] $\|e^{Js} - e^{\tilde{J}s}\| \leq \left\|J - \tilde{J}\right\| s \leq \epsilon' s \|\mathcal{L}\|_{\text{be}}$. Further, Lemma 6 implies that the error caused by the truncation is $\frac{(2e^{s\|\mathcal{L}\|_{\text{be}}})^{K'+1}}{(K'+1)'}$. By assuming $\epsilon' \leq \epsilon/(t(\|\mathcal{L}\|_{\text{be}})$, the truncation error will dominate by our choice of K'.

In Lemma 2, we need $|j\rangle\langle j| \otimes A_j$, where A_j is defined in Equation (60). This can be implemented by a sequence of at most K controlled-U_s and at most K controlled-U_{L_j}. Note that the time register required for implementing U_s can be extracted from the index j, and then uncomputed. Therefore, the gate cost for this is $O(KK'm)$. Therefore, the additional 1- and 2-qubits for this implementation is dominated by $O(KK'm)$.

Next, we analyze how the truncation of e^{Js} at order K' affects the total error. By Lemma 7, we have

$$\begin{aligned}&\|\mathcal{J}_{K'}(t-s_m)\mathcal{L}_{\text{J}}\mathcal{J}_{K'}(s_m-s_{m-1})\mathcal{L}_{\text{J}}\cdots\mathcal{J}_{K'}(s_2-s_1)\mathcal{L}_{\text{J}}\mathcal{J}_{K'}(s_1)-\mathcal{F}_k(s_k,\ldots,s_1)\|_\diamond \\ &\qquad\leq \frac{8e^{\|\mathcal{L}\|_{\text{be}}t}\|\mathcal{L}\|_{\text{be}}^{K'+1}}{(K'+1)!}(2\|\mathcal{L}\|_{\text{be}})^k 2^k t^{K'+1}.\end{aligned} \tag{81}$$

Taking the weighted sum for quadrature points, the error is at most

$$\begin{aligned}&\frac{8e^{\|\mathcal{L}\|_{\text{be}}t}\|\mathcal{L}\|_{\text{be}}^{K'+1}}{(K'+1)!}(2\|\mathcal{L}\|_{\text{be}})^k 2^k t^{K'+1}\sum_{j_1=1}^{q}\cdots\sum_{j_k=1}^{q}\hat{w}_{(j_k)}\cdots\hat{w}_{(j_k,\ldots,j_1)} \\ &\qquad= \frac{8t^k e^{\|\mathcal{L}\|_{\text{be}}t}\|\mathcal{L}\|_{\text{be}}^{K'+1}}{(k-1)!(K'+1)!}(2\|\mathcal{L}\|_{\text{be}})^k 2^k t^{K'+1}.\end{aligned} \tag{82}$$

Therefore, the total error is

$$\sum_{k=1}^{K}\frac{8t^k e^{\|\mathcal{L}\|_{\text{be}}t}\|\mathcal{L}\|_{\text{be}}^{K'+1}}{(k-1)!(K'+1)!}(2\|\mathcal{L}\|_{\text{be}})^k 2^k t^{K'+1} \leq \frac{32e^{5\|\mathcal{L}\|_{\text{be}}t}\|\mathcal{L}\|_{\text{be}}^{K'+2}t^{K'+2}}{(K'+1)!}. \tag{83}$$

With $t\|\mathcal{L}\|_{\text{be}} = \Theta(1)$, it suffices to set

$$K' = O\left(\frac{\log(1/\epsilon)}{\log\log(1/\epsilon)}\right) \tag{84}$$

to make this error $\leq \epsilon/2$. Therefore, the total error is bounded by ϵ.

Multiple simulation blocks. For arbitrary evolution time t, we divide it into $O(t\|\mathcal{L}\|_{\text{be}})$ segments and set

$$K, K', q = O\left(\frac{\log(t\|\mathcal{L}\|_{\text{be}}/\epsilon)}{\log\log(t\|\mathcal{L}\|_{\text{be}}/\epsilon)}\right) \tag{85}$$

so that the total error of the $O(t\|\mathcal{L}\|_{\text{be}})$ segments is within ϵ. For the remaining smaller segment of this division, the normalizing constant is smaller which yields a larger success probability. However, the amplitude amplification will overshoot. We use standard technique by adding an ancillary qubit and use a rotation to dilute the success probability to 1/4. ◀

[2] The inequality $\|e^{Js} - e^{\tilde{J}s}\| \leq \left\|J - \tilde{J}\right\| s$ does not hold for general matrices J. However, in our case it holds because J is dissipative and hence $\|e^{Js}\| \leq 1$ for all $s \geq 0$.

6 Conclusion and open questions

In this paper, we presented a quantum algorithm for simulating Lindblad evolution, which captures the dynamics of Markovian open quantum systems. The algorithm can be used to forecast the dynamics of a quantum system interacting with an environment. Informally, the complexity of our algorithm scales as $\mathcal{O}(t\,\mathrm{polylog}(t/\epsilon))$, which matches the previous state-of-the-art algorithm. Our algorithm is based on a conceptually novel mathematical treatment of evolution channel to preserve its complete positivity: we use a higher-order series expansion based on Duhamel's principle, and we approximate the integrals by scaled Gaussian quadrature, which exponentially reduces the number of terms in the summation. Our mathematical treatment trades off mathematical simplicity for technical conciseness, and it yields a much simpler algorithm based on linear combination of unitaries. We also outlined how our algorithm can be generalized to simulate time-dependent Lindbladians. Moreover, our approximation of multiple integrals using scaled Gaussian quadrature can be potentially used to produce a more efficient approximation of time-ordered integrals, which will simplify existing quantum algorithms for simulating time-dependent Hamiltonians based on a truncated Dyson series, e.g., [18].

The open questions of this work are summarized as follows.

- Can we achieve the additive complexity, i.e., $\mathcal{O}(t+\mathrm{polylog}(1/\epsilon))$? This additive complexity has been achieved for simulating Hamiltonian evolution by quantum signal processing [28] and quantum singular transformation [14], and it is proved to be optimal [5]. As Hamiltonian evolution is a special case of Lindblad evolution, the complexity for simulating the latter is at least $\Omega(t + \mathrm{polylog}(1/\epsilon))$. It is yet unknown how to generalize the techniques of quantum signal processing and quantum singular value transformation to superoperators.
- What are the practical performances of our algorithm? For Hamiltonian simulation, although LCU-based algorithms have a better asymptotic scaling, it was reported in [8] that Trotter-based algorithms surprisingly perform just as well in practice. Regarding simulating Lindblad evolution, do LCU-based algorithms, i.e., the algorithms presented in this paper and [9], have a practical advantage compared with Trotter-based simulation algorithms, e.g., [20, 7]? An empirical study on the performances of quantum algorithms for simulating open quantum systems would be beneficial.

References

1. Robert Alicki. The quantum open system as a model of the heat engine. *Journal of Physics A: Mathematical and General*, 12(5):L103, 1979.
2. Robert Alicki and David Gelbwaser-Klimovsky. Non-equilibrium quantum heat machines. *New Journal of Physics*, 17(11):115012, 2015.
3. Dominic W Berry, Andrew M Childs, Richard Cleve, Robin Kothari, and Rolando D Somma. Simulating Hamiltonian dynamics with a truncated Taylor series. *Physical Review Letters*, 114(9), 2015. doi:10.1103/physrevlett.114.090502.
4. Dominic W Berry, Andrew M Childs, Richard Cleve, Robin Kothari, and Rolando D Somma. Exponential improvement in precision for simulating sparse Hamiltonians. *Forum of Mathematics, Sigma*, 5, 2017. doi:10.1017/fms.2017.2.
5. Dominic W Berry, Andrew M Childs, and Robin Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2015)*, 2015.
6. Yu Cao and Jianfeng Lu. Structure-preserving numerical schemes for Lindblad equations. *arXiv:2103.01194 [quant-ph]*, March 2021. arXiv:2103.01194.

7 Andrew M Childs and Tongyang Li. Efficient simulation of sparse Markovian quantum dynamics. *Quantum Information & Computation*, 17(11&12):0901–0947, 2017.

8 Andrew M Childs, Dmitri Maslov, Yunseong Nam, Neil J Ross, and Yuan Su. Toward the first quantum simulation with quantum speedup. *Proceedings of the National Academy of Sciences*, 115(38):9456–9461, 2018.

9 Richard Cleve and Chunhao Wang. Efficient quantum algorithms for simulating Lindblad evolution. In *44th International Colloquium on Automata, Languages, and Programming, (ICALP 2017)*, pages 17:1–17:14, 2017.

10 Wibe A de Jong, Kyle Lee, James Mulligan, Mateusz Płoskoń, Felix Ringer, and Xiaojun Yao. Quantum simulation of nonequilibrium dynamics and thermalization in the Schwinger model. *Physical Review D*, 106(5):054508, 2022.

11 Wibe A de Jong, Mekena Metcalf, James Mulligan, Mateusz Płoskoń, Felix Ringer, and Xiaojun Yao. Quantum simulation of open quantum systems in heavy-ion collisions. *Physical Review D*, 104(5):L051501, 2021.

12 Ross Dorner, John Goold, and Vlatko Vedral. Towards quantum simulations of biological information flow. *Interface Focus*, 2(4):522–528, 2012. doi:10.1098/rsfs.2011.0109.

13 Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, 1982. doi:10.1007/bf02650179.

14 András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 193–204. ACM, 2019. arXiv:1806.01838.

15 Vittorio Gorini, Andrzej Kossakowski, and Ennackal Chandy George Sudarshan. Completely positive dynamical semigroups of n-level systems. *Journal of Mathematical Physics*, 17(5):821–825, 1976.

16 Susana F. Huelga and Martin B. Plenio. Vibrations, quanta and biology. *Contemporary Physics*, 54(4):181–207, 2013. doi:10.1080/00405000.2013.829687.

17 Michael J. Kastoryano and Fernando G. S. L. Brandão. Quantum Gibbs samplers: the commuting case. *Communications in Mathematical Physics*, 344(3):915–957, 2016. doi:10.1007/s00220-016-2641-8.

18 Michael J. Kastoryano, Florentin Reiter, and Anders S. Sørensen. Dissipative preparation of entanglement in optical cavities. *Physical Review Letters*, 106(9), 2011. doi:10.1103/physrevlett.106.090502.

19 Mária Kieferová, Artur Scherer, and Dominic W Berry. Simulating the dynamics of time-dependent hamiltonians with a truncated dyson series. *Physical Review A*, 99(4):042314, 2019.

20 Martin Kliesch, Thomas Barthel, Christian Gogolin, Michael J Kastoryano, and Jens Eisert. Dissipative quantum Church-Turing theorem. *Physical Review Letters*, 107(12), 2011. doi:10.1103/physrevlett.107.120501.

21 Christiane P Koch. Controlling open quantum systems: tools, achievements, and limitations. *Journal of Physics: Condensed Matter*, 28(21):213001, 2016.

22 Ronnie Kosloff and Amikam Levy. Quantum heat engines and refrigerators: Continuous devices. *Annual Review of Physical Chemistry*, 65:365–393, 2014.

23 Barbara Kraus, Hans P. Büchler, Sebastian Diehl, Adrian Kantian, Andrea Micheli, and Peter Zoller. Preparation of entangled states by quantum Markov processes. *Physical Review A*, 78(4), 2008. doi:10.1103/physreva.78.042307.

24 Anthony J. Leggett, Sudip Chakravarty, Alan T. Dorsey, Matthew P. A. Fisher, Anupam Garg, and Wilhelm Zwerger. Dynamics of the dissipative two-state system. *Reviews of Modern Physics*, 59(1):1–85, 1987. doi:10.1103/revmodphys.59.1.

25 Amikam Levy, Anthony Kiely, Juan Gonzalo Muga, Ronnie Kosloff, and Erik Torrontegui. Noise resistant quantum control using dynamical invariants. *New Journal of Physics*, 20(2):025006, 2018.

26 Xiantao Li and Chunhao Wang. Succinct description and efficient simulation of non-markovian open quantum systems. *arXiv preprint*, 2021. arXiv:2111.03240.

27 Goran Lindblad. On the generators of quantum dynamical semigroups. *Communications in Mathematical Physics*, 48(2):119–130, 1976.

28 Guang Hao Low and Isaac L Chuang. Optimal Hamiltonian simulation by quantum signal processing. *Physical Review Letters*, 118(1), 2017. doi:10.1103/physrevlett.118.010501.

29 Easwar Magesan, Daniel Puzzuoli, Christopher E. Granade, and David G. Cory. Modeling quantum noise for efficient testing of fault-tolerant circuits. *Physical Review A*, 87(1), 2013. doi:10.1103/physreva.87.012324.

30 Volkhard May and Oliver Kühn. *Charge and Energy Transfer Dynamics in Molecular Systems*. John Wiley & Sons, 2008.

31 Sarah Mostame, Patrick Rebentrost, Alexander Eisfeld, Andrew J Kerman, Dimitris I. Tsomokos, and Alán Aspuru-Guzik. Quantum simulator of an open quantum system using superconducting qubits: exciton transport in photosynthetic complexes. *New Journal of Physics*, 14(10):105013, 2012. doi:10.1088/1367-2630/14/10/105013.

32 Abraham Nitzan. *Chemical Dynamics in Condensed Phases: Relaxation, Transfer and Reactions in Condensed Molecular Systems*. Oxford University Press, 2006.

33 Matthew Otten and Stephen K Gray. Accounting for errors in quantum algorithms via individual error reduction. *npj Quantum Information*, 5(1):1–6, 2019.

34 Martin B Plenio and Peter L Knight. The quantum-jump approach to dissipative dynamics in quantum optics. *Reviews of Modern Physics*, 70(1):101, 1998.

35 Patrick Rall, Chunhao Wang, and Pawel Wocjan. Thermal state preparation via rounding promises. *arXiv preprint*, 2022. arXiv:2210.01670.

36 Florentin Reiter, David Reeb, and Anders S Sørensen. Scalable dissipative preparation of many-body entanglement. *Physical Review Letters*, 117(4), 2016. doi:10.1103/physrevlett.117.040501.

37 Yair Rezek and Ronnie Kosloff. Irreversible performance of a quantum harmonic heat engine. *New Journal of Physics*, 8(5):83, 2006.

38 Jinzhao Sun, Xiao Yuan, Takahiro Tsunoda, Vlatko Vedral, Simon C Benjamin, and Suguru Endo. Mitigating realistic noise in practical noisy intermediate-scale quantum devices. *Physical Review Applied*, 15(3):034026, 2021.

39 Nishchay Suri, Felix C Binder, Bhaskaran Muralidharan, and Sai Vinjanampathy. Speeding up thermalisation via open quantum system variational optimisation. *The European Physical Journal Special Topics*, 227(3):203–216, 2018.

40 Sabine Tornow, Wolfgang Gehrke, and Udo Helmbrecht. Non-equilibrium dynamics of a dissipative two-site Hubbard model simulated on IBM quantum computers. *Journal of Physics A: Mathematical and Theoretical*, 55(24):245302, 2022.

41 Frank Verstraete, Michael M. Wolf, and Juan Ignacio Cirac. Quantum computation and quantum-state engineering driven by dissipation. *Nature Physics*, 5(9):633–636, 2009. doi:10.1038/nphys1342.

42 Ulrich Weiss. *Quantum Dissipative Systems*. World Scientific, 2012.

43 Chunhao Wang Xiantao Li. Simulating Markovian open quantum systems using higher-order series expansion. *arXiv preprint*, 2022. arXiv:2212.02051.

A Simulating time-dependent Lindbladians

There exist natural generalizations to Lindblad equations. One such generalization is time-dependent Markovian open quantum systems, which arises in the context of quantum heat engine [1, 22, 2, 37] and controlling open quantum systems [21, 25, 39]. In this section, we sketch how our simulation techniques can be generalized to the case of time-dependent Lindbladians. More specifically, consider a time-dependent version of Equation (1):

$$\frac{\mathrm{d}}{\mathrm{d}t}\rho = \mathcal{L}(t)(\rho) \coloneqq -i[H(t),\rho] + \sum_{j=1}^{m}\left(L_j(t)\rho L_j^{\dagger(t)} - \frac{1}{2}\left\{L_j(t)^\dagger L_j(t),\rho\right\}\right). \tag{86}$$

Now, $H(t)$ and $L_j(t)$ are time-dependent. We decompose this time-dependent Lindbladian into drift terms and jump terms as:

$$\mathcal{L}(t) = \mathcal{L}_{\mathrm{D}}(t) + \mathcal{L}_{\mathrm{J}}(t), \quad \text{and} \tag{87}$$

$$\mathcal{L}_{\mathrm{D}}(t)(\rho) \coloneqq J(t)\rho + \rho J(t)^\dagger, \quad \mathcal{L}_{\mathrm{J}}(t)(\rho) = \sum_{j=1}^{m} L_j(t)\rho L_j(t)^\dagger. \tag{88}$$

We express the evolution driven by $\mathcal{L}_{\mathrm{D}}$ as,

$$\rho_t = \mathcal{V}(0,t) \coloneqq V(0,t)\rho_0 V(0,t)^\dagger, \tag{89}$$

where $V(s,t)$ satisfies the equation,

$$\frac{d}{dt}V(s,t) = J(t)V(s,t), \quad \text{and} \quad V(s,s) = I. \tag{90}$$

One can express the unitary $V(0,t)$ using time-ordered evolution operators,

$$V(s,t) = \mathcal{T}e^{\int_s^t J(\tau)d\tau}. \tag{91}$$

Further, for Equation (1), the Duhamel's principle implies a generalization of Equation (23),

$$\rho_t = \mathcal{V}(0,t)(\rho_0) + \int_0^t \mathcal{V}(s,t)(\mathcal{L}_{\mathrm{J}}(s)(\rho_s))\mathrm{d}s. \tag{92}$$

In the Hamiltonian simulation [19], Such an operator is approximated by Dyson series,

$$V(0,t) = \sum_{k=0}^{K}\frac{t^k}{M^k k!}\sum_{j_1,j_2,\ldots,j_k=0}^{M-1}\mathcal{T}J(t_k)\cdots J(t_1) + \mathcal{O}\left(\frac{(\|J\|_{\max}t)^{K+1}}{(K+1)!} + \frac{t^2\|\dot{J}\|_{\max}}{M}\right), \tag{93}$$

where $\mathcal{T}$ indicates a strict time-ordering $t_1 \le t_2 \le \cdots \le t_k$ in the product. The formula here approximates the evolution from 0 to t. This can be easily extended to another interval, due to the observation that,

$$V(s,t) = \mathcal{T}e^{\int_s^t J(\tau)d\tau} = \mathcal{T}e^{\int_0^{t-s} J(s+\tau)d\tau}, \tag{94}$$

which leads to

$$V(s,t) = \sum_{k=0}^{K}\frac{t^k}{M^k k!}\sum_{j_1,j_2,\ldots,j_k=s}^{M-1}\mathcal{T}J(t_k)\cdots J(t_1) + \mathcal{O}\left(\frac{(\|J\|_{\max}t)^{K+1}}{(K+1)!} + \frac{t^2\|\dot{J}\|_{\max}}{M}\right). \tag{95}$$

This suggests that, by repeatedly applying Equation (23), we can adapt our series expansion in Equation (29) to

$$\mathcal{G}_K(t) \coloneqq \mathcal{K}[V(0,t)] + \sum_{k=1}^{K}\int_{0\le s_1\le\cdots\le s_k\le t}\mathcal{F}_k(s_k,\ldots,s_1)\,\mathrm{d}s_1\cdots\mathrm{d}s_k, \tag{96}$$

where

$$\begin{aligned}&\mathcal{F}_k(s_k,\ldots,s_1)\\&\coloneqq \mathcal{K}[V(s_k,t)]\mathcal{L}_{\mathrm{J}}(s_k)\mathcal{K}[V(s_{k-1},s_k)]\mathcal{L}_{\mathrm{J}}(s_{k-1})\cdots\mathcal{K}[V(s_1,s_2)]\mathcal{L}_{\mathrm{J}}(s_1)\mathcal{K}[V(0,s_1)].\end{aligned} \tag{97}$$

Then, we can approximate the integral using scaled Gaussian quadrature as in Section 4, and implement the completely positive map using the techniques presented in Section 5. Further note that we use a truncated Dyson series

$$\mathcal{J}_K = \mathcal{K}\left[\sum_{k=0}^{K}\frac{t^k}{M^k k!}\sum_{j_1,j_2,\cdots,j_k=s}^{M-1} J(t_k)\cdots J(t_1)\right]. \tag{98}$$

to approximate $V(s,t)$.

Space-Efficient Interior Point Method, with Applications to Linear Programming and Maximum Weight Bipartite Matching

S. Cliff Liu
Carnegie Mellon University, Pittsburgh, PA, USA

Zhao Song
Adobe Research, San Jose, CA, USA

Hengjie Zhang
Columbia University, New York, NY, USA

Lichen Zhang
Massachusetts Institute of Technology, Cambridge, MA, USA

Tianyi Zhou
University of California San Diego, CA, USA

Abstract

We study the problem of solving linear program in the streaming model. Given a constraint matrix $A \in \mathbb{R}^{m \times n}$ and vectors $b \in \mathbb{R}^m, c \in \mathbb{R}^n$, we develop a space-efficient interior point method that optimizes solely on the dual program. To this end, we obtain efficient algorithms for various different problems:

- For general linear programs, we can solve them in $\widetilde{O}(\sqrt{n}\log(1/\epsilon))$ passes and $\widetilde{O}(n^2)$ space for an ϵ-approximate solution. To the best of our knowledge, this is the most efficient LP solver in streaming with no polynomial dependence on m for both space and passes.
- For bipartite graphs, we can solve the minimum vertex cover and maximum weight matching problem in $\widetilde{O}(\sqrt{m})$ passes and $\widetilde{O}(n)$ space.

In addition to our space-efficient IPM, we also give algorithms for solving SDD systems and isolation lemma in $\widetilde{O}(n)$ spaces, which are the cornerstones for our graph results.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Linear programming

Keywords and phrases Convex optimization, interior point method, streaming algorithm

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.88

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/pdf/2009.06106.pdf`

Funding *Lichen Zhang*: Supported by NSF grant No. CCF-1955217 and NSF grant No. CCF-2022448.

Acknowledgements The authors would like to thank Jonathan Kelner for many helpful discussions.

1 Introduction

Given a constraint matrix $A \in \mathbb{R}^{m \times n}$, vectors $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$, the linear program problem asks us to solve the primal program (P) or its dual (D):

$$(P) = \max_{A^\top y \leq c, y \geq 0} b^\top y \text{ and } (D) = \min_{Ax \geq b} c^\top x \tag{1}$$

is one of the most fundamental problems in computer science and operational research. Many efforts have been dedicated to develop time-efficient linear program solvers in the past half a century, such as the simplex method [23], ellipsoid method [44] and interior point method [41]. In the last few years, speeding up linear program solve via interior point method (IPM) has been heavily studied [20, 55, 13, 35, 65, 25, 71]. The state-of-the-art IPM has the runtime of $O(m^{2+1/18} + m^{\omega})$ when $m \approx n$ and $O(mn + n^3)$ when $m \gg n$. To achieve these impressive improvements, most of these algorithms utilize randomized and dynamic data structures to maintain the primal and dual solutions simultaneously. While these algorithms are time-efficient, it is highly unlikely that they can be implemented in a *space-efficient* manner: maintaining the primal-dual formulation requires $\Omega(m + n^2)$ space, which is particularly unsatisfactory when $m \gg n$.

In this paper, we study the problem of solving a linear program in the streaming model: At each pass, we can query the i-th row of A and the corresponding of the b. The goal is to design an LP solver that is both space and pass-efficient. By efficient, our objective is to obtain an algorithm with no polynomial dependence on m, or more concretely, we present a robust IPM framework that uses only $\widetilde{O}(n^2)$ space and $\widetilde{O}(\sqrt{n}\log(1/\epsilon))$ passes.[1] To the best of our knowledge, this is the most efficient streaming LP algorithm that achieves a space and pass independent of m. Current best streaming algorithms for LP either require $\Omega(n)$ passes or $\Omega(n^2 + m^2)$ space for $O(\sqrt{n})$ passes. For the regime of tall dense LP ($m \gg n$), our algorithm achieves the best space and passes.

The key ingredient for obtaining these LP algorithms is a paradigm shift from the time-efficient primal-dual IPM to a less time-efficient dual-only IPM [64]. From a time perspective, dual-only IPM requires $\widetilde{O}(\sqrt{n}\log(1/\epsilon))$ iterations, with each iteration can be computed in $\widetilde{O}(mn + \mathrm{poly}(n))$ time. However, it is much more space-efficient than that of primal-dual approach. Specifically, we show that per iteration, it suffices to maintain an $n \times n$ Hessian matrix in place. To obtain $\widetilde{O}(\sqrt{n}\log(1/\epsilon))$ passes, we show that non-trivial quantities such as the Lewis weights [56, 21] can be computed recursively, in an in-place fashion with only $\widetilde{O}(n^2)$ space.

Now that we have a space and pass-efficient IPM for general LP in the streaming model, we instantiate it with applications for graph problems in the semi-streaming model. In the semi-streaming model, each edge is revealed along with its weight in an online fashion and might subject to an adversarial order, and the algorithm is allowed to make multiples passes over the stream in $\widetilde{O}(n)$ space.[2] We particularly focus on the *maximum weight bipartite matching* problem, in which the edges with weights are streamed to us, and the goal is to find a matching that maximizes the total weights in it. While there is a long line of research ([2, 36, 24, 3, 9] to name a few) on this problem, most algorithms can only compute an approximate matching, meaning that the weight is at least $(1 - \epsilon)$ of the maximum weight. For the case of exact matching, a recent work [6] provides an algorithm that takes $n^{4/3+o(1)}$ passes in $\widetilde{O}(n)$ space for computing a maximum *cardinality* matching. It remains an open question to compute an exact maximum *weight* bipartite matching in semi-streaming model, with $o(n)$ passes.

We answer this question by presenting a semi-streaming algorithm that uses $\widetilde{O}(n)$ space and $\widetilde{O}(\sqrt{m})$ passes, this means that as long as the graph is relatively sparse, i.e., $m = o(n^2)$, we achieve $o(n)$ passes. To obtain an $\widetilde{O}(n)$ space algorithm for *any graph*, we require

[1] We use $\widetilde{O}(\cdot)$ notation to hide polylogarithmic dependence on n and m.

[2] Some authors define the space in the streaming model to be the number of cells, where each cell can hold $O(\log n)$ bits or even a number with infinite precision. Our bounds remain unchanged even if each cell only holds $O(1)$ bits, i.e., when arithmetic only applies to $O(1)$-bits operands.

additional machinery; more specifically, for each iteration of our dual-only IPM, we need to compute the Newton step via a symmetric diagonal dominant (SDD) solve in $\widetilde{O}(n)$ space. Since the seminal work of Spielman and Teng [67], many efforts have been dedicated in designing a time-efficient SDD system solver [45, 46, 43, 19]. This solvers run in $\widetilde{O}(m)$ time with improved dependence on the logarithmic terms. However, all of them require $\widetilde{\Theta}(m)$ space. To achieve $\widetilde{O}(n)$ space, we make use of small-space spectral sparsifiers [38] as preconditioners to solve the system in a space and pass-efficient manner.

Finally, we note that with $\widetilde{O}(n)$ space, we essentially solve the dual problem, which is the *generalized minimum vertex cover* on bipartite graph. To turn a solution on vertices to a solution on edges, we utilize the isolation lemma [60] and implement it in $\widetilde{O}(n)$ bits via a construction due to [17].

1.1 Our contribution

In this section, we showcase three main results of this paper and discuss their consequences.

The first result regards solving a general linear program in the streaming model with $\widetilde{O}(n^2)$ space and $\widetilde{O}(\sqrt{n}\log(1/\epsilon))$ passes.

▶ **Theorem 1** (General LP, informal version of Theorem 7.4 in Full version [57]). *Given a linear program with m constraints and n variables and $m \geq n$ in the streaming model, there exists an algorithm that outputs an ϵ-approximate solution to the dual program (Eq. (1)) in $\widetilde{O}(n^2)$ space and $\widetilde{O}(\sqrt{n}\log(1/\epsilon))$ passes.*

By ϵ-approximate solution, we mean that the algorithm finds $x \in \mathbb{R}^n$ such that $c^\top x - c^\top x^* \leq \epsilon$, where x^* is the optimal solution. The key to obtain our result is a small space implementation of leverage score and Lewis weights, so that we can utilize the Lee-Sidford barrier [51], with the number of passes depending on the smaller dimension.

In conjunction with an SDD solver in $\widetilde{O}(n)$ space, our next result shows that in the semi-streaming model, we can solve the minimum vertex cover problem on a bipartite graph with $\widetilde{O}(\sqrt{m})$ passes.

▶ **Theorem 2** (Minimum vertex cover, informal version of Theorem 9.7 in Full version [57]). *Given a bipartite graph G with n vertices and m edges, there exists a streaming algorithm that computes a minimum vertex cover of G in $\widetilde{O}(\sqrt{m})$ passes and $\widetilde{O}(n)$ space with probability $1 - 1/\operatorname{poly}(n)$.*[3]

The reason we end up with $\widetilde{O}(\sqrt{m})$ passes instead of $\widetilde{O}(\sqrt{n})$ passes is that to compute some fundamental quantities such as leverage scores or Lewis weights, we need to solve $\Theta(m)$ SDD systems and result in a total of $\widetilde{O}(m\sqrt{n})$ passes. By using the logarithmic barrier, we only need to solve $O(1)$ SDD systems per iteration, which gives the $\widetilde{O}(\sqrt{m})$ passes.

We are now ready to present our result for bipartite matching in $\widetilde{O}(\sqrt{m})$ passes, which solves the longstanding problem of whether maximum weight matching can be solved in $o(n)$ passes for any $m = n^{2-c}$ with $c > 0$.

▶ **Theorem 3** (Maximum weight bipartite matching, informal version of Theorem 10.1 in Full version [57]). *Given a bipartite graph G with n vertices and m edges, there exists a streaming algorithm that computes an (exact) maximum weight matching of G in $\widetilde{O}(\sqrt{m})$ passes and $\widetilde{O}(n)$ space with probability $1 - 1/\operatorname{poly}(n)$.*

[3] We can actually solve a *generalized* version of the minimum vertex cover problem in bipartite graph: each edge e needs to be *covered* for at least $b_e \in \mathbb{Z}^+$ times, where the case of $b = \mathbf{1}_m$ is the classic minimum vertex cover.

Our matching result relies on turning the solution to the dual minimum vertex cover problem, to a primal solution for the maximum weight matching. We achieve so by an $\widetilde{O}(n)$ space implementation of the isolation lemma [60, 17].

1.2 Related work

Interior point method for solving LP. The interior point method was originally proposed by Karmarkar [41] for solving linear program. Since then, there is a long line of work on speeding up interior point method for solving classical optimization problems, e.g., linear program [68, 64, 69, 61, 22, 50, 51, 52, 20, 55, 53, 12, 13, 71, 35, 25, 65, 33]. In 1987, the running time of LP solver becomes $O(n^3)$ [68, 64]. In 1989, Vaidya proposed an $O(n^{2.5})$ LP solver based on a specific implementation of IPMs, known as the *central path algorithm* [68, 69]. Lee and Sidford show how to solve LP in $\sqrt{n}(\text{nnz}(A) + n^\omega)$ time [49, 50, 51], where ω is the exponent of matrix multiplication [70, 48, 4][4] (the first $\sqrt{n}$-iteration IPM). In 2019, [20] show how to solve LP in $n^\omega + n^{2.5-\alpha} + n^{2+1/6}$, where α is the dual exponent of matrix multiplication [31][5]. This is the first breakthrough result improving $O(n^{2.5})$ from 30 years ago. Later, [35] improved that running time to $n^\omega + n^{2.5-\alpha} + n^{2+1/18}$ by maintaining two layers of data-structure instead of one layer of data-structure as [20]'s algorithm. In 2020, [13] improved the running time of LP solver on tall matrices to mn when $m \geq \text{poly}(n)$. Another line of work focuses on solving linear program with small treewidth [25, 71] in time $\widetilde{O}(m\tau^2)$.

Small space algorithms for solving LP. Simplex algorithm is another popular approach to solve linear programs. It has an even better compatibility with streaming algorithms. For instance, [15] shows that the non-recursive implementation of Clarkson's algorithm [18] gives a streaming LP solver that uses $O(n)$ passes and $\widetilde{O}(n\sqrt{m})$ space. They also show that the recursive implementation gives a streaming LP solver that uses $n^{O(1/\delta)}$ passes and $(n^2 + m^\delta)\,\text{poly}(1/\delta)$ space. [7] proposes a streaming algorithm for solving n-dimensional LP that uses $O(nr)$ pass and $O(m^{1/r})\,\text{poly}(n, \log m)$ space, where $r \geq 1$ is a parameter. All above algorithms needs space depending on m.

Streaming algorithms for approximate matching. Maximum matching has been extensively studied in the streaming model for decades, where almost all of them fall into the category of approximation algorithms. For algorithms that only make one pass over the edges stream, researchers make continuous progress on pushing the constant approximation ratio above $1/2$, which is under the assumption that the edges are arrived in a uniform random order [37, 5, 27, 11]. The random-order assumption makes the problem easier (at least algorithmically). A more general setting is multi-pass streaming with adversarial edge arriving. Under this setting, the first streaming algorithm that beats the $1/2$-approximation of bipartite cardinality matching is [29], giving a $2/3 \cdot (1-\epsilon)$-approximation in $1/\epsilon \cdot \log(1/\epsilon)$ passes. The first to achieve a $(1-\epsilon)$-approximation is [59], which takes $(1/\epsilon)^{1/\epsilon}$ passes.[6] Since then, there is a long line of research in proving upper bounds and lower bounds on the number of passes to compute a maximum matching in the streaming model [2, 26, 32, 26, 36, 24, 3, 8, 10, 9] (see next subsection for more details). Notably, [2, 3] use linear programming and duality theory (see the next subsection for more details).

[4] Currently, $\omega \approx 2.37$.
[5] Currently, $\alpha \approx 0.31$.
[6] For the weighted case, there is a $(1/2-\epsilon)$-approximation algorithm that only takes one pass [62].

However, all the algorithms above can only compute an approximate maximum matching: to compute a matching whose size is at least $(1-\epsilon)$ times the optimal, one needs to spend $\text{poly}(1/\epsilon)$ passes (see [24, 3] and the references therein). For readers who are interested in the previous techniques for solving matching, we refer to Section A in full version [57] which contains a brief summary.

Recent developments for exact matching. Recently, [6] proposes an algorithm that computes a $(1-\epsilon)$-approximate maximum *cardinality* matching in $O(\epsilon^{-1}\log n\log\epsilon^{-1})$ passes and $\widetilde{O}(n)$ space. Their method leverages recent advances in ℓ_1-regression with several ideas for implementing it in small space, leading to a streaming algorithm with no dependence on ϵ in the space usage, and thus improving over [3]. The resulted semi-streaming algorithm computes an exact maximum *cardinality* matching (not for weighted) in $n^{3/4+o(1)}$ passes.

Streaming spectral sparsifer. Initialized by the study of cut sparsifier in the streaming model [1], a simple one-pass semi-streaming algorithm for computing a spectral sparsifier of any weighted graph is given in [42], which suffices for our applications in this paper. The problem becomes more challenging in a dynamic setting, i.e., both insertion and deletion of edges from the graph are allowed. Using the idea of linear sketching, [38] gives a single-pass semi-streaming algorithm for computing the spectral sparsifier in the dynamic setting. However, their brute-force approach to recover the sparsifier from the sketching uses $\Omega(n^2)$ time. An improved recover time is given in [39] but requires more spaces, e.g., $\epsilon^{-2}n^{1.5}\log^{O(1)}n$. Finally, [40] proposes a single-pass semi-streaming algorithm that uses $\epsilon^{-2}n\log^{O(1)}n$ space and $\epsilon^{-2}n\log^{O(1)}n$ recover time to compute an ϵ-spectral sparsifier which has $O(\epsilon^{-2}n\log n)$ edges. Note that $\Omega(\epsilon^{-2}n\log n)$ space is necessary for this problem [14].

SDD solver. There is a long line of work focusing on fast SDD solvers [66, 45, 46, 43, 19, 63, 47]. Spielman and Teng give the first nearly-linear time SDD solver, which is simplified with a better running time in later works. The current fastest SDD solver runs in $O(m\log^{1/2}n\,\text{poly}(\log\log n)\log(1/\epsilon))$ time [19]. All of them require $\widetilde{\Theta}(m)$ space.

2 Technical overview

We start with an overview of our IPM framework. We first note that many recent fast IPM algorithms do not fit into $\widetilde{O}(n^2)$ space. Algorithms such as [51, 35, 13] need to maintain *both* primal and dual solutions, thus require $\Omega(m)$ space. In fact, any algorithms that rely on the primal formulation will need $\Omega(m)$ space to maintain the solution. To bypass this issue, we draw inspiration from the state-of-the-art SDP solver [34]: in their setting, $m=\Omega(n^2)$, which means any operation on the dimension m will be too expensive to perform. They instead resort to the *dual-only* formulation. The dual formulation provides a more straightforward optimization framework on small dimension and makes it harder to maintain key quantities. This is exactly what we want: an algorithm that operates on the smaller dimension, removing the polynomial dependence on m. While efficient maintenance is the key to design time-efficient IPM, it is less a concern for us since our constraining resource is space, not time. To this end, we show that Renegar's IPM algorithm [64] can be implemented in a streaming fashion with only $\widetilde{O}(n^2)$ space. As the number of passes of an IPM crucially depends on the barrier function being used, the [64] algorithm only gives a pass bound of $\widetilde{O}(\sqrt{m}\log(1/\epsilon))$. To further improve the number of passes required, we show that the nearly-universal barrier of Lee and Sidford [51, 53] can also be implemented in $\widetilde{O}(n^2)$ space.

This involves computing Lewis weights in an extremely space-efficient manner. We present a recursive algorithm with $\widetilde{O}(1)$ depth, based on [28], that uses only $\widetilde{O}(n^2)$ space. This gives the desired $\widetilde{O}(\sqrt{n}\log(1/\epsilon))$ passes.

We now turn to our graph results, which is a novel combination of the space-efficient IPM, SDD solvers, duality and the isolation lemma. Note that for both graph problems only allow $\widetilde{O}(n)$ space, so it won't suffice to directly apply our IPM algorithms .

To give a better illustration of the $\widetilde{O}(n)$ space constraint, note that storing a matching already takes $\widetilde{\Theta}(n)$ space, meaning that we have only a polylogarithmic space overhead per vertex to store auxiliary information. The conventional way of solving maximum bipartite matching using an IPM solver would get stuck at the very beginning - maintaining the solution of the relaxed linear program, which is a fractional matching, already requires $\Omega(m)$ space for storing all LP constraints, which seems inevitable.

Our key insight is to show that solving the *dual* form of the above LP, which corresponds to the generalized (fractional) minimum vertex cover problem, is sufficient, and therefore only $\widetilde{O}(n)$ space is needed for maintaining a fractional solution. We use several techniques to establish this argument. The first idea is to use complementary slackness for the dual solution to learn which n edges will be in the final maximum matching and therefore reduce the size of the graph from m to n. However, this is not always the case: For instance, in a bipartite graph that admits a perfect matching, all left vertices form a minimum vertex cover, but the complementary slackness theorem gives no information on which edges are in the perfect matching. To circumvent this problem, we need to slightly perturb the weight on every edge, so that the minimum vertex cover (which is now unique) indeed provides enough information. We use the isolation lemma [60] to realize this objective.

It is then instructive to implement the isolation lemma in limited space. Perturbing the weight on every edge requires storing $\widetilde{O}(m)$ bits of randomness, since the perturbation should remain identical across two different passes. We bypass this issue by using the generalized isolation lemma proposed by [17], in which only $O(\log(Z))$ bits of randomness is needed, where Z is the number of candidates. In our case, $Z \leq n^n$ is the number of all possible matchings. So $\widetilde{O}(n)$ space usage perfectly fits into the semi-streaming model. We design an oracle that stores $\widetilde{O}(n)$ random bits and outputs the same perturbations for all edges in all passes.

Now that we can focus on solving the minimum vertex cover problem in $\widetilde{O}(n)$ space. When the constraint matrix is an incidence matrix, each iteration of our IPM can be implemented as an SDD (or Laplacian) solver, so it suffices to show how to solve SDD system in the semi-streaming model, which, to the best of our knowledge, has not been done prior to our work.

In the following subsections we elaborate on each of the above components:

- In Section 2.1, we provide a high-level picture of how our dual-only interior point method works.
- In Section 2.2, we show evidences that our interior point method can run in space independent of m for all of the three different barriers.
- In Section 2.3, we describe our contribution on our implementations of SDD solver, IPM, and the isolation lemma in the streaming model. We show a novel application of the isolation lemma to turn dual into primal.

2.1 Dual-only robust IPM

The cornerstone of our results is to design a robust IPM framework that works only on the dual formulation of the linear program. The framework fits in barriers including the logarithmic barrier, hybrid barrier and Lee-Sidford barrier. It is also robust enough as it can tolerate approximation errors in many quantities, while preserving the convergence behavior.

Algorithm 1 is a simplified version of our dual-only IPM. The earlier works of Renegar's algorithm [64] require the Newton's direction be computed exactly as $\Delta x = -H(x)^{-1}\nabla f_t(x)$, in order to get double exponential convergence rate of Newton's method. To strengthen its guarantee, we develop a more robust framework for this IPM. Specifically, we show that the Hessian of the barrier functions, the gradient and the Newton's direction can all be approximated. This requires a much more refined error analysis. Below, we carefully bound the compound errors caused by three layers of approximations.

First, from Δx to δ_x (Line 9), we allow our Hessian to be spectrally approximated within any small constant factor. This provides us enough leeway to implement the Hessian of barrier functions in a space-efficient manner. For example, the Hessian of the volumetric barrier is $H(x) = A_x^\top(3\Sigma_x - 2P_x^{(2)})A_x$, where Σ_x is a diagonal matrix and $P_x^{(2)}$ is taking entry-wise square of a dense projection matrix. But $\widetilde{H}(x) = A_x^\top \Sigma_x A_x$ is a 5-approximation of $H(x)$ and we can compute it in the same space as computing leverage scores.

Second, from δ_x (Line 9) to δ'_x (Line 11), we allow approximation on the gradient in the sense that it has small local norm with respect to the true gradient, i.e., $\|\nabla f_t(x) - \widetilde{\nabla} f_t(x)\|_{H(x)^{-1}} \leq 0.1$.[7] To give a concrete example, let $\sigma \in \mathbb{R}^m$ denote the leverage score vector, and suppose the Hessian matrix is in the form of $H(x) = A^\top \Sigma A$ and the gradient is $\nabla f(x) = A^\top \sigma$. The leverage score σ can then be approximated in an entry-wise fashion: each entry can tolerate a multiplicative $(1 \pm O(1/\sqrt{n}))$ error. This is because

$$\begin{aligned}
\|\nabla f_t(x) - \widetilde{\nabla} f_t(x)\|^2_{H(x)^{-1}} &= \mathbf{1}_m^\top(\Delta\Sigma)A(A^\top\Sigma A)^{-1}A^\top(\Delta\Sigma)\mathbf{1}_m \\
&= \mathbf{1}_m^\top(\Delta\Sigma)\Sigma^{-1/2}\Sigma^{1/2}A(A^\top\Sigma A)^{-1}A^\top\Sigma^{1/2}\Sigma^{-1/2}(\Delta\Sigma)\mathbf{1}_m \\
&\leq \mathbf{1}_m^\top(\Delta\Sigma)\Sigma^{-1}(\Delta\Sigma)\mathbf{1}_m \\
&= \sum_{i=1}^m \frac{(\sigma_i - \widetilde{\sigma}_i)^2}{\sigma_i} \\
&\leq \frac{0.01}{n} \cdot n = 0.01,
\end{aligned}$$

where the first inequality follows from property of projection matrix (for any projection matrix P, we have $P \preceq I$. Then we know $x^\top P x \leq x^\top x$ for all vector x), the last inequality follows from $\sum_{i=1}^m \sigma_i = n$.

Third, from δ'_x (Line 11) to $\widetilde{\delta}$ (Line 12), we can tolerate the approximation error on the Newton's direction $\|\widetilde{\delta} - \delta'_x\|_{H(x)} \leq 0.1$. This is crucial for our graph applications, since we need to use small space SDD solver to approximate the Newton's direction.

2.2 Solve LP in small space

In this section, we show how to implement our IPM in space not polynomially dependent on m for different barrier functions.

[7] For a vector y and a positive semidefinite matrix A, we define $\|y\|_A := \sqrt{y^\top A y}$.

Algorithm 1 A simplified version of our algorithm.

procedure OurAlgorithm($A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n$)
  Choose $F(x) \in \mathbb{R}^n \to \mathbb{R}$ to be any θ^2-self concordant barrier function
  Let $f_t(x) := t \cdot c + \nabla F(x) \in \mathbb{R}^n$
  Let $H(x) := \nabla^2 F(x) \in \mathbb{R}^{n\times n}$
  Let T be the number of iterations
  Initialize x, t
  for $k \leftarrow 1$ to T **do**
    Let $\widetilde{H}(x)$ be any PSD matrix that $\frac{1}{\log m}\widetilde{H}(x) \preceq H(x) \preceq \widetilde{H}(x)$
    Let $\delta_x := -\widetilde{H}(x)^{-1} \cdot \nabla f_t(x)$
    Let $\widetilde{\nabla} f_t(x)$ be that $\|\widetilde{\nabla} f_t(x) - \nabla f_t(x)\|_{H(x)^{-1}} \leq 0.1$
    Let $\delta'_x := -\widetilde{H}(x)^{-1} \cdot \widetilde{\nabla} f_t(x)$
    Let $\widetilde{\delta}_x$ be any vector that $\|\widetilde{\delta}_x - \delta'_x\|_{H(x)} \leq 0.1$
    $x \leftarrow x + \widetilde{\delta}_x$
    $t \leftarrow t \cdot (1 + \theta^{-1})$
  end for
  Output x
end procedure

For three barriers (logarithmic, hybrid and Lee-Sidford), all of their Hessians take the form of $A^\top HA \in \mathbb{R}^{n\times n}$ for an $m \times m$ non-negative diagonal matrix H. For logarithmic barrier, $H_{i,i} = s_i(x)^{-2}$, as $s_i(x)$ can be computed in $O(1)$ space, it is not hard to see that the Gram matrix can be computed as $\sum_{i\in[m]} H_{i,i} \cdot a_i a_i^\top$ in $O(n^2)$ space.

The more interesting case is to consider the hybrid barrier and Lee-Sidford barrier. The gradient and Hessian of the hybrid barrier requires us to compute m leverage scores defined as $\mathrm{diag}(\sqrt{H}A(A^\top HA)^{-1}A^\top\sqrt{H})$. Forming this projection matrix will require a prohibitive m^2 space. To implement it in n^2 space, we rely on an observation that $\sigma_i = H_{i,i} \cdot a_i^\top (A^\top HA)^{-1} a_i$, thus, if we can manage to maintain $(A^\top HA)^{-1}$ in $O(n^2)$ space, then we can compute the leverage score. Similar to the logarithmic barrier scenario, $A^\top HA$ can be computed in 1 pass and $O(n^2)$ space, then the inverse can be computed in $O(n^2)$ space. Thus, we can supply the i-th leverage score in $O(n^2)$ space, and compute the gradient and Hessian in designated space constraint.

Given an oracle that can compute the i-th leverage score in $O(n^2)$ space, we can even implement the $\ell_{\log m}$ Lewis weights in $\widetilde{O}(n^2)$ space. To do so, we rely on an iterative scheme introduced in [28]. Unfortunately, as we are only allowed a space budget of $O(n^2)$, we cannot store the intermediate Lewis weights. To circumvent this issue, we develop a recursive algorithm to query Lewis weights from prior iterations. Each recursion takes $O(n^2)$ space, and the algorithm uses at most $O(\mathrm{poly}(\log m))$ iterations, therefore, we can compute the Lewis weights in $\widetilde{O}(\sqrt{n})$ space.

2.3 Semi-streaming maximum weight bipartite matching in $\widetilde{O}(\sqrt{m})$ passes

Recall that in the semi-streaming model, we are only allowed with $\widetilde{O}(n)$ space. For the IPMs we've developed before, we can not meet such space constraint. For general graphs, we have to invent more machinery to realize the $\widetilde{O}(n)$ space.

For matching, we start by noting that the constraint matrix $A \in \mathbb{R}^{m \times n}$ is a graph incidence matrix. This means that for logarithmic barrier, the Hessian matrix $A^\top S^{-2} A$ can be treated as a Laplacian matrix with edge weight s_i^{-2}. Therefore, computing the Newton direction reduces to perform an SDD solve in $\widetilde{O}(n)$ space.

SDD solver in the semi-streaming model. Though solving SDD system can be done in an extremely time-efficient manner, it is unclear how to compute them when only $\widetilde{O}(n)$ space is allowed. To circumvent this problem, we rely on two crucial observations. Let L_G denote the SDD matrix corresponding to the Hessian.

- Solving a system $L_G \cdot x = b$ will require $\widetilde{\Omega}(m)$ space, but multiplying L_G with a vector $v \in \mathbb{R}^n$ can be done in $O(n)$ space: as $L_G = \sum_{i \in [m]} \frac{a_i a_i^\top}{s_i^2}$, $L_G \cdot v$ can be computed as $a_i (a_i^\top v)/s_i^2$ in $O(n)$ space, and accumulate the sum over a pass of the graph.
- Suppose we have a sparse graph H with only $\widetilde{O}(n)$ edges, then the system $L_H \cdot x = b$ can be solved in $\widetilde{O}(n)$ space.

It turns out that these two observations are enough for us to solve a general SDD system in $\widetilde{O}(n)$ space. Given the graph G, we first compute a $(1 \pm \delta)$-spectral sparsifier with only $\widetilde{O}(\delta^{-2} n \log^{O(1)} n)$ edges in a single pass [40]. Let H denote this sparsifier, we then use L_H^{-1} as a preconditioner for solving our designated SDD system. More concretely, let $r_t := b - L_G \cdot x_t$ denote the residual at t-th iteration, we solve the system $L_H \cdot y_t = r_t$. As $y_t = L_H^{-1} \cdot b - L_H^{-1} L_G \cdot x_t$, we can then update the solution via the preconditioned-solution $x_{t+1} = x_t + y_t$. The residual is then $r_{t+1} = b - L_G \cdot x_{t+1} = b - L_G \cdot x_t - L_G \cdot y_t = r_t - L_G \cdot y_t$, i.e., we only need to implement one matrix-vector product with L_G. After $\widetilde{O}(1)$ iterations, we have refined an accurate enough solution for the SDD system.

From dual to primal. Though we can solve the dual in $\widetilde{O}(n)$ space, it only produces a solution to the minimum vertex cover and we need to transform it to a solution to maximum weight matching.

Turning an optimal dual solution to an optimal primal solution for general LP requires at least solving a linear system, which takes $O(n^\omega)$ time and $O(mn)$ space (Lee, Sidford and Wong [54]), which is unknown to be implemented in the semi-streaming model even for bipartite matching LP.[8] We bypass this issue by using the complementary slackness theorem to highlight n tight dual constraints and therefore sparsify the original graph from m edges to n edges without losing the optimal matching. However, this is only true if the solution to the primal LP is **unique**.

To give a better illustration, let us consider a simple example. Suppose the graph has a (maximum weight) perfect matching (see Figure 1 for example). Then the following trivial solution is optimal to the dual LP: choosing all vertices in V_L. Let us show what happens when applying the complementary slackness theorem. The complementary slackness theorem says that when y is a feasible primal solution and x is a feasible dual solution, then y is optimal to the primal and x is optimal to the dual **if and only if**

$$\langle y, Ax - \mathbf{1}_m \rangle = 0 \quad \text{and} \quad \langle x, \mathbf{1}_n - A^\top y \rangle = 0. \tag{2}$$

From the above case, we have $Ax - \mathbf{1}_m = 0$, so the first equality $\langle y, Ax - \mathbf{1}_m \rangle = 0$ puts no constraint on y. Therefore, any solution $y \geq \mathbf{0}_m$ to the linear system $a_i^\top y_i = 1,\ \forall i \in V_L$ is an optimal solution, where a_i is the i-th column of A. Note that this linear system has m variables and $|V_L|$ equations, which is still hard to find a solution in $\widetilde{O}(n)$ space.

[8] In general, the inverse of a sparse matrix can be dense, which means the standard Gaussian elimination method for linear system solving does not apply in the semi-streaming model.

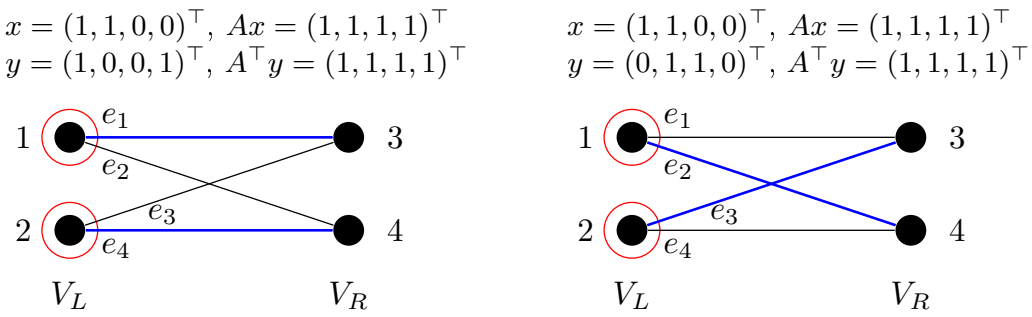


Figure 1 The red circle is a minimum vertex cover, which is an optimal dual solution. The blue edge is a maximum matching, which is an optimal primal solution. In both examples, the primal and dual satisfy complementary slackness Eq. (2).

Now consider perturbing the primal objective function by some vector $b \in \mathbb{R}^m$ such that the optimal solution to the following primal LP is **unique**:

$$\textbf{Primal} \quad \max_{y \in \mathbb{R}^m} b^\top y, \qquad \text{s.t. } A^\top y \leq \mathbf{1}_n \text{ and } y \geq \mathbf{0}_m.$$

Suppose we find the optimal solution x in the dual LP and we want to recover the optimal solution y in the primal LP. Again by plugging in the complementary slackness theorem, we get at most n equations from the second part $\langle x, \mathbf{1}_n - A^\top y \rangle = 0$. Since the optimal y is unique and y has dimension m, the first part $\langle y, Ax - \mathbf{1}_m \rangle$ must contribute to at least $m - n$ equations. Note that these equations have the form

$$y_i = 0, \ \forall i \in [m] \text{ s.t. } (Ax)_i - 1 > 0.$$

This means that the corresponding edges are *unnecessary* in order to get one maximum matching. As a result, we can reduce the number of edges from m to n, then compute a maximum matching in $\widetilde{O}(n)$ space without reading the stream.

Isolation lemma in the semi-streaming model. It remains to show how to perturb the objective so that the primal solution is unique. As the perturbation is over all edges, one natural idea is to randomly perturb them using $\widetilde{O}(m)$ bits of randomness. This becomes troublesome when the random bits need to be stored since the perturbation should remain consistent across different passes. We resolve this problem via the isolation lemma.

Let us recall the definition of the isolation lemma (see Section C in full version [57] for details).

▶ **Definition 4** (Isolation lemma). *Given a set system $(S, \mathcal{F})$ where $\mathcal{F} \subseteq \{0,1\}^S$. Given weight w_i to each element i in S, the weight of a set F in $\mathcal{F}$ is defined as $\sum_{i \in F} w_i$. The isolation lemma says there exists a scheme that can assign weight oblivious to $\mathcal{F}$, such that there is a unique set in $\mathcal{F}$ that has the minimum (maximum) weight under this assignment.*

The isolation lemma says that if we randomly choose weights, then with a good probability the uniqueness is ensured. However, this does not apply to the streaming setting since the weight vector is over all edges, which require $\Omega(m)$ space.

To apply isolation lemma for bipartite matching, we note that the set S is all the edges and the family $\mathcal{F}$ contains all possible matchings. The total number of possible matchings is at most $(n+1)^n$, as each vertex can choose none or one of the vertices to match. We

leverage this parameterization and make use of [17], which requires $\log(|F|)$ random bits. For matching, we only need $O(n \log n)$ bits, which suits in our space budget. To the best of our knowledge, this is the first use of isolation lemma in the streaming model.

2.4 Discussions

For matching, improving $\sqrt{m}$ passes to $\sqrt{n}$ passes will require us to compute fundamental quantities such as leverage scores and Lewis weights by solving $\widetilde{O}(1)$ SDD systems. As reachability [58] and single source shortest path [30, 16] can be solved in $n^{1/2+o(1)}$ passes in the semi-streaming model, we believe it is an important open problem to close the gap between bipartite matching and these problems.

References

1 Kook Jin Ahn and Sudipto Guha. Graph sparsification in the semi-streaming model. In *International Colloquium on Automata, Languages, and Programming*, pages 328–338. Springer, 2009.

2 Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. In *ICALP*, pages 526–538. Springer, 2011.

3 Kook Jin Ahn and Sudipto Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. *ACM Transactions on Parallel Computing (TOPC)*, 4(4):1–40, 2018.

4 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *SODA*, 2021.

5 Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab Mirrokni, and Cliff Stein. Coresets meet EDCS: algorithms for matching and vertex cover on massive graphs. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1616–1635. SIAM, 2019.

6 Sepehr Assadi, Arun Jambulapati, Yujia Jin, Aaron Sidford, and Kevin Tian. Semi-streaming bipartite matching in fewer passes and optimal space. In *SODA*. arXiv preprint, 2022. `arXiv:2011.03495`.

7 Sepehr Assadi, Nikolai Karpov, and Qin Zhang. Distributed and streaming linear programming in low dimensions. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*, pages 236–253, 2019.

8 Sepehr Assadi, Gillat Kol, Raghuvansh R. Saxena, and Huacheng Yu. Multi-pass graph streaming lower bounds for cycle counting, max-cut, matching size, and other problems. In *FOCS*, 2020.

9 Sepehr Assadi, S. Cliff Liu, and Robert E. Tarjan. An auction algorithm for bipartite matching in streaming and massively parallel computation models. In *The SIAM Symposium on Simplicity in Algorithms (SOSA@SODA'21)*, 2021.

10 Sepehr Assadi and Ran Raz. Near-quadratic lower bounds for two-pass graph streaming algorithms. In *FOCS*, 2020.

11 Aaron Bernstein. Improved bounds for matching in random-order streams. In *ICALP*, 2020.

12 Jan van den Brand. A deterministic linear program solver in current matrix multiplication time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 259–278. SIAM, 2020.

13 Jan van den Brand, Yin Tat Lee, Aaron Sidford, and Zhao Song. Solving tall dense linear programs in nearly linear time. In *STOC*, 2020.

14 Charles Carlson, Alexandra Kolla, Nikhil Srivastava, and Luca Trevisan. Optimal lower bounds for sketching graph cuts. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2565–2569. SIAM, 2019.

15 Timothy M Chan and Eric Y Chen. Multi-pass geometric algorithms. *Discrete & Computational Geometry*, 37(1):79–102, 2007.

16 Yi-Jun Chang, Martin Farach-Colton, Tsan-Sheng Hsu, and Meng-Tsung Tsai. Streaming complexity of spanning tree computation. In *37th international symposium on theoretical aspects of computer science (STACS)*, 2020.

17 Suresh Chari, Pankaj Rohatgi, and Aravind Srinivasan. Randomness-optimal unique element isolation with applications to perfect matching and related problems. *SIAM Journal on Computing*, 24(5):1036–1050, 1995.

18 Kenneth L Clarkson. Las vegas algorithms for linear and integer programming when the dimension is small. *Journal of the ACM (JACM)*, 42(2):488–499, 1995.

19 Michael B Cohen, Rasmus Kyng, Gary L Miller, Jakub W Pachocki, Richard Peng, Anup B Rao, and Shen Chen Xu. Solving sdd linear systems in nearly $m \log^{1/2} n$ time. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing(STOC)*, pages 343–352, 2014.

20 Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, 2019.

21 Michael B. Cohen and Richard Peng. Lp row sampling by lewis weights. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, 2015.

22 Samuel I Daitch and Daniel A Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *Proceedings of the fortieth annual ACM symposium on Theory of computing (STOC)*, pages 451–460, 2008.

23 George B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. In *Activity Analysis of Production and Allocation*, Cowles Commission Monograph No. 13. ., 1951.

24 Shahar Dobzinski, Noam Nisan, and Sigal Oren. Economic efficiency requires interaction. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing (STOC)*, pages 233–242, 2014.

25 Sally Dong, Yin Tat Lee, and Guanghao Ye. A nearly-linear time algorithm for linear programs with small treewidth: A multiscale representation of robust central path. In *STOC*, 2021. `arXiv:2011.05365`.

26 Sebastian Eggert, Lasse Kliemann, Peter Munstermann, and Anand Srivastav. Bipartite matching in the semi-streaming model. *Algorithmica*, 63(1-2):490–508, 2012.

27 Alireza Farhadi, Mohammad Taghi Hajiaghayi, Tung Mah, Anup Rao, and Ryan A Rossi. Approximate maximum matching in random streams. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1773–1785. SIAM, 2020.

28 Maryam Fazel, Yin Tat Lee, Swati Padmanabhan, and Aaron Sidford. Computing lewis weights to high precision. In *SODA*, 2021.

29 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. In *ICALP*, pages 531–543. Springer, 2004.

30 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the data-stream model. *SIAM Journal on Computing*, 38(5):1709–1727, 2009.

31 Francois Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms(SODA)*, pages 1029–1046, 2018.

32 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms (SODA)*, pages 468–485. SIAM, 2012.

33 Yuzhou Gu and Zhao Song. A faster small treewidth sdp solver. *arXiv preprint*, 2022. `arXiv:2211.06033`.

34 Baihe Huang, Shunhua Jiang, Zhao Song, Runzhou Tao, and Ruizhe Zhang. Solving tall dense sdps in the current matrix multiplication time. In *FOCS*, 2022.

35 Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. Faster dynamic matrix inverse for faster lps. In *STOC*, 2021. arXiv:2004.07470.

36 Michael Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1679–1697. SIAM, 2013.

37 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 734–751. SIAM, 2014.

38 Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. *SIAM J. Comput.*, 46(1):456–477, 2017.

39 Michael Kapralov, Aida Mousavifar, Cameron Musco, Christopher Musco, Navid Nouri, Aaron Sidford, and Jakab Tardos. Fast and space efficient spectral sparsification in dynamic streams. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1814–1833. SIAM, 2020.

40 Michael Kapralov, Navid Nouri, Aaron Sidford, and Jakab Tardos. Dynamic streaming spectral sparsification in nearly linear time and space. In *arXiv preprint*, 2019.

41 Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing(STOC)*, pages 302–311. ACM, 1984.

42 Jonathan A Kelner and Alex Levin. Spectral sparsification in the semi-streaming setting. In *STACS*, 2011.

43 Jonathan A Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu. A simple, combinatorial algorithm for solving sdd systems in nearly-linear time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing (STOC)*, pages 911–920, 2013.

44 Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

45 Ioannis Koutis, Gary L. Miller, and Richard Peng. Approaching optimality for solving sdd linear systems. In *FOCS*, pages 235–244, 2010.

46 Ioannis Koutis, Gary L Miller, and Richard Peng. A nearly-$m \log n$ time solver for sdd linear systems. In *52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 590–598, 2011.

47 Rasmus Kyng and Sushant Sachdeva. Approximate gaussian elimination for laplacians-fast, sparse, and simple. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 573–582. IEEE, 2016.

48 François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th international symposium on symbolic and algebraic computation (ISSAC)*, pages 296–303. ACM, 2014.

49 Yin Tat Lee and Aaron Sidford. Path finding i: Solving linear programs with $\widetilde{O}(\sqrt{rank})$ linear system solves. *arXiv preprint*, 2013. arXiv:1312.6677.

50 Yin Tat Lee and Aaron Sidford. Path finding ii: An $\widetilde{O}(m\sqrt{n})$ algorithm for the minimum cost flow problem. *arXiv preprint*, 2013. arXiv:1312.6713.

51 Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $O(\sqrt{rank})$ iterations and faster algorithms for maximum flow. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 424–433. IEEE, 2014.

52 Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 230–249. IEEE, 2015.

53 Yin Tat Lee and Aaron Sidford. Solving linear programs with sqrt (rank) linear system solves. *arXiv preprint*, 2019. arXiv:1910.08033.

54 Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2015.

55 Yin Tat Lee, Zhao Song, and Qiuyi Zhang. Solving empirical risk minimization in the current matrix multiplication time. In *COLT*, 2019.

56 D. Lewis. Finite dimensional subspaces of l_p. *Studia Mathematica*, 1978.

57 S Cliff Liu, Zhao Song, Hengjie Zhang, Lichen Zhang, and Tianyi Zhou. Space-efficient interior point method, with applications to linear programming and maximum weight bipartite matching. *arXiv*, 2020. `arXiv:2009.06106`.

58 Yang P Liu, Arun Jambulapati, and Aaron Sidford. Parallel reachability in almost linear work and square root depth. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1664–1686. IEEE, 2019.

59 Andrew McGregor. Finding graph matchings in data streams. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 170–181. Springer, 2005.

60 Ketan Mulmuley, Umesh V Vazirani, and Vijay V Vazirani. Matching is as easy as matrix inversion. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing (STOC)*, pages 345–354, 1987.

61 Yu Nesterov and Arkadi Nemirovsky. Self-concordant functions and polynomial-time methods in convex programming. *Report, Central Economic and Mathematic Institute, USSR Acad. Sci*, 1989.

62 Ami Paz and Gregory Schwartzman. A $(2+\epsilon)$-approximation for maximum weight matching in the semi-streaming model. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2153–2161, 2017.

63 Richard Peng and Daniel A Spielman. An efficient parallel solver for sdd linear systems. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing (STOC)*, pages 333–342, 2014.

64 James Renegar. A polynomial-time algorithm, based on newton's method, for linear programming. *Mathematical Programming*, 40(1-3):59–93, 1988.

65 Zhao Song and Zheng Yu. Oblivious sketching-based central path method for solving linear programming. In *ICML*, 2021.

66 Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 81–90. `arXiv:cs/0310051`, divided into `arXiv:0809.3232`, `arXiv:0808.4134`, `arXiv:cs/0607105`, 2004.

67 Daniel A. Spielman and Shang-Hua Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM J. Matrix Analysis Applications*, 35(3):835–885, 2014.

68 Pravin M Vaidya. An algorithm for linear programming which requires $O(((m+n)n^2+(m+n)^{1.5}n)L)$arithmetic operations. In *FOCS*. IEEE, 1987.

69 Pravin M Vaidya. Speeding-up linear programming using fast matrix multiplication. In *FOCS*. IEEE, 1989.

70 Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing (STOC)*, pages 887–898. ACM, 2012.

71 Guanghao Ye. *Fast Algorithm for Solving Structured Convex Programs*. Bachelor's thesis, University of Washington, 2021.

List Decoding of Rank-Metric Codes with Row-To-Column Ratio Bigger Than $\frac{1}{2}$

Shu Liu
The National Key Laboratory on Wireless Communications,
University of Electronic Science and Technology of China, Chengdu, China

Chaoping Xing
School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China

Chen Yuan
School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China

Abstract

Despite numerous results about the list decoding of Hamming-metric codes, development of list decoding on rank-metric codes is not as rapid as its counterpart. The bound of list decoding obeys the Gilbert-Varshamov bound in both the metrics. In the case of the Hamming-metric, the Gilbert-Varshamov bound is a trade-off among rate, decoding radius and alphabet size, while in the case of the rank-metric, the Gilbert-Varshamov bound is a trade-off among rate, decoding radius and column-to-row ratio (i.e., the ratio between the numbers of columns and rows). Hence, alphabet size and column-to-row ratio play a similar role for list decodability in each metric. In the case of the Hamming-metric, it is more challenging to list decode codes over smaller alphabets. In contrast, in the case of the rank-metric, it is more difficult to list decode codes with large column-to-row ratio. In particular, it is extremely difficult to list decode square matrix rank-metric codes (i.e., the column-to-row ratio is equal to 1).

The main purpose of this paper is to explicitly construct a class of rank-metric codes $\mathcal{C}$ of rate R with the column-to-row ratio up to $2/3$ and efficiently list decode these codes with decoding radius beyond the decoding radius $(1-R)/2$ (note that $(1-R)/2$ is at least half of relative minimum distance δ). In literature, the largest column-to-row ratio of rank-metric codes that can be efficiently list decoded beyond half of minimum distance is $1/2$. Thus, it is greatly desired to efficiently design list decoding algorithms for rank-metric codes with the column-to-row ratio bigger than $1/2$ or even close to 1. Our key idea is to compress an element of the field $\mathbb{F}_{q^n}$ into a smaller $\mathbb{F}_q$-subspace via a linearized polynomial. Thus, the column-to-row ratio gets increased at the price of reducing the code rate. Our result shows that the compression technique is powerful and it has not been employed in the topic of list decoding of both the Hamming and rank metrics. Apart from the above algebraic technique, we follow some standard techniques to prune down the list. The algebraic idea enables us to pin down the message into a structured subspace of dimension linear in the number n of columns. This "periodic" structure allows us to pre-encode the message to prune down the list.

2012 ACM Subject Classification Mathematics of computing → Coding theory

Keywords and phrases Coding theory, List-decoding, Rank-metric codes

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.89

Category Track A: Algorithms, Complexity and Games

Funding *Shu Liu*: Research supported in part by the National Key R&D Program of China under Grant 2022YFA1004900, in part by the National Natural Science Foundation of China under Grant 12271084, in part by the National Key Laboratory of Science and Technology on Communications under Contract G02214307.
Chaoping Xing: Research supported in part by the National Key Research and Development Projects under Grant 2021YFE0109900 and in part by the National Natural Science Foundation of China under Grant 12031011.
Chen Yuan: Research supported in part by the Natural Science Foundation of China under Grant 12101403.

1 Introduction

Rank-metric codes were first introduced by Delsarte in [1] and have found various applications [14, 16]. A rank-metric code $\mathcal{C}$ of rate R and relative minimum distance δ must obey the Singleton bound $1 - R \geqslant \delta$ (see Subsection 2.1). The equality $1 - R = \delta$ holds if the code $\mathcal{C}$ is maximum rank distance (MRD for short). As for every alphabet size q and ratio ρ, one can always construct an MRD code. Therefore, we view decoding radius $(1 - R)/2$ as the half of minimum distance decoding radius or unique decoding radius.

The unique decoding algorithms for rank-metric codes have been extensively studied [3, 14]. However, the list decoding algorithm of the rank-metric codes are not understood very well. Despite of many results about the list decoding of Hamming-metric codes in literature, very few were known about the list decoding of rank-metric codes. In particular, for the column-to-row ratio bigger than $\frac{1}{2}$, there are no known explicit constructions of rank-metric codes that can be list decoded beyond half the minimum distance decoding radius. On the other hand, with high probability, a square random rank-metric code of rate R can be list decoded up to its decoding radius $1 - \sqrt{R}$ (see [2]). Note that $1 - \sqrt{R}$ is always bigger than $(1 - R)/2$. This means that with high probability, a random square rank-metric code can be list decoded beyond the half of minimum distance decoding radius.

In the the Hamming-metric case, it is more challenging to list decode codes over small alphabets. As we will see in the next subsection, in contrast, it becomes more difficult to list decode codes with large column-to-row ratio (i.e., the ratio between the numbers of rows and columns). In particular, it is extremely difficult to list decoding of square matrix rank-metric codes (i.e., the column-to-row ratio is equal to 1). Therefore, it is a great challenge to design efficient algorithms to list decode rank-metric codes with the column-to-row ratio close to 1 and decoding radius beyond $(1 - R)/2$.

1.1 Known results

Let us fix some notations before stating known results. Denote by $\mathbb{F}_q^{t\times n}$ the collection of $t \times n$ matrices over $\mathbb{F}_q$. We may assume that $n \leqslant t$. Otherwise, we can consider transpose of matrices. One can define the rank-metric within $\mathbb{F}_q^{t\times n}$ (see the detailed definition in Subsection 2.1). A subset $\mathcal{C}$ of $\mathbb{F}_q^{t\times n}$ equipped with rank-metric is called a rank-metric code. Unlike Hamming-metric codes, apart from rate and minimum distance there is an important parameter $\rho(\mathcal{C}) := \frac{n}{t}$ which is called the column-to-row ratio.

▶ **Definition 1.** Let $\tau \in (0, 1)$ and $L \geqslant 1$ be an integer. A rank-metric code $\mathcal{C}$ is (τ, L)-list decodable if for every $X \in \mathbb{F}_q^{t\times n}$

$$|\mathcal{B}_R(X, \tau n) \cap \mathcal{C}| \leq L,$$

where $\mathcal{B}_R(X, \tau n)$ is a rank-metric ball defined in Subsection 2.1.

Limit to list decodability of rank-metric codes and list decodability of random rank-metric codes are known [2, 15]. More precisely, we have the following result (see [2]):

- **(i)** If the ratio n/t tends to a fixed real ρ, a rank-metric code $\mathcal{C} \subseteq \mathbb{F}_q^{t\times n}$ of rate R that is (τ, L)-list decodable with $L = \text{poly}(n)$ must obey the Gilbert-Varshamov bound, i.e., $R \leqslant (1 - \tau)(1 - \rho\tau)$.
- **(ii)** With high probability a random rank-metric code can be list decoded up to the Gilbert-Varshamov bound, i.e., a random rank-metric code of rate R in $\mathbb{F}_q^{t\times n}$ is $(\tau, O(1/\varepsilon))$-list decodable with $R = (1 - \tau)(1 - \rho\tau) - \varepsilon$ for any small real $\varepsilon > 0$. In particular, if the

ratio n/t is a small constant ε, then with high probability a random rank-metric code of rate R in $\mathbb{F}_q^{t\times n}$ is $(1-R-\varepsilon, O(1/\varepsilon))$-list decodable.

Let us introduce the state-of-art results by comparing list decoding of Hamming-metric and rank-metric codes. First of all, we note that both Hamming-metric and rank-metric codes obey the Gilbert-Varshamov bounds in each metric for list decoddability. In the case of the Hamming-metric, the Gilbert-Varshamov bound is a trade-off among rate, decoding radius and alphabet size, while in the case of the rank-metric, the Gilbert-Varshamov bound is a trade-off among rate, decoding radius and column-to-row ratio. Hence, alphabet size and column-to-row ratio play the similar role for list decodability in each metric. We will see from the following paragraph that the small column-to-row ratio for list decoding of rank-metric codes is compatible with large alphabet size for list decoding of Hamming-metric codes and vice versa.

Recall that in the case of the Hamming-metric, the limit on list decodability is the Hamming-metric Gilbert-Varshamov bound $1-H_q(\tau)$, where $H_q(x)$ is the entropy function, and a random code can be list decoded up to the Hamming-metric Gilbert-Varshamov bound [5]. When alphabet size $q=\exp(\Omega(\frac{1}{\varepsilon}))$, the Hamming-metric Gilbert-Varshamov bound $1-H_q(\tau)$ tends to the Singleton bound $1-R-\varepsilon$. Currently, for list decoding of Hamming-metric codes over large alphabet q, the best result is that, for $q=O(\frac{1}{\varepsilon^2})$, by making use of folded algebraic geometry codes or algebraic geometry codes with evaluation points in subfields (for convenience let us call them subfield algebraic geometry codes) one can list decode Hamming-metric codes up to the Singleton bound $1-R-\varepsilon$ (see [10, 11, 12]). Thus, for list-decoding of Hamming-metric codes over large alphabet q, it remains an open problem to design efficient algorithm to list decode up to $1-R-\varepsilon$ for q-ary codes with $q=\Omega(\frac{1}{\varepsilon})$. On the other hand, for sufficiently small column-to-row ratio, say $\rho=O(\varepsilon)$, the rank-metric Gilbert-Varshamov also tends the Singleton bound $1-R-\varepsilon$. Furthermore, when the column-to-row ratio $\rho=O(\varepsilon^2)$, an efficient list decoding of rank-metric codes up to the Singleton bound $1-R-\varepsilon$ was introduced in [11, 12] by making use of subfield Gabidulin codes. Hence again, it remains an open problem to design efficient algorithm to list decode rank-metric codes up to $1-R-\varepsilon$ for with column-to-row ratio $\rho=\Omega(\varepsilon)$.

For the regime of small alphabets q such as $q=2$, there is not much work on efficient list decoding algorithms for Hamming-metric codes except for the concatenation techniques. Precisely speaking, by making use of concatenation technique, one can list decode binary Hamming-metric codes up to the Blokh-Zyablov bound [7]. Similarly, in the case of rank-metric codes, not much work has been done for large column-to-row ratio, in particular, for ratio $\rho=1$, i.e., the square matrix case. The largest column-to-row ratio ρ is $\frac{1}{2}$ for which the list decoding bound lies beyond the unique decoding radius. In [17], by making use of folded Gabidulin codes, one can list decode beyond the unique decoding radius $(1-R)/2$ with the column-to-row ratio ρ arbitrarily close to $1/2$.

1.2 Our result

We propose a compression technique which is the key to construct list decodable rank-metric codes with the ratio ρ up to $\frac{2}{3}$. This moves one step further towards the ratio $\rho=1$. Our list decodable rank-metric codes are obtained by compressing folded Gabidulin codes. The following theorem summarizes our main result.

▶ **Main Theorem 1.** *For every constant finite field $\mathbb{F}_q$, any small real $\varepsilon>0$ and integer $s>1$, there exists an explicit constriction of $\mathbb{F}_q$-linear rank metric codes with the ratio ρ and rate R that are $\left(\frac{1-sR}{\rho(s+1)}-\varepsilon, q^{O((s-1)^2/\varepsilon)}\right)$-list-decodable. The algorithm runs in time*

poly(n, q). *Furthermore, if* $\rho < \frac{2(1-R)}{(s+1)(1-sR)}$, *then the decoding radius* $\tau = \frac{1-sR}{\rho(s+1)} - \varepsilon$ *exceeds the unique decoding radius* $\frac{1-R}{2}$.

▶ Remark 2. If we take $s = 2$, then we get rank-metric codes of the ratio ρ and rate R that are $\left(\frac{1-2R}{3\rho} - \varepsilon, q^{O(1/\varepsilon)}\right)$-list decodable. In particular, if $\rho < \frac{2(1-R)}{3(1-2R)}$, then the list decoding radius is bigger than $(1-R)/2$. Furthermore, when the rate R tends to 0, there exists an explicit construction of rank-metric codes with any ratio $\rho < \frac{2}{3}$. Note that our decoding radius depends on the ratio ρ, while the unique decoding radius is independent of the ratio ρ. Let us draw a diagram to illustrate our main result.

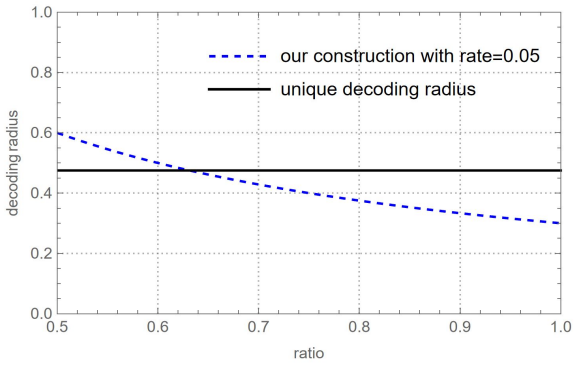


Figure 1 Comparison of our decoding radius with the unique decoding radius for different ratios.

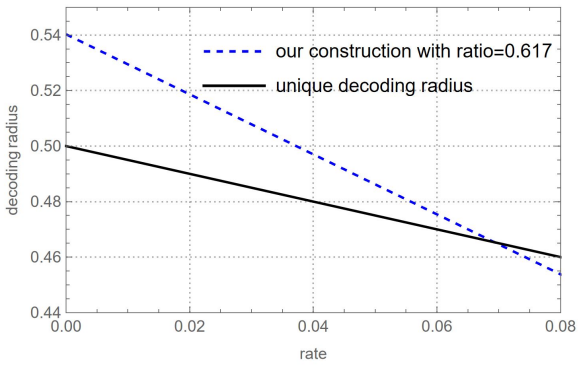


Figure 2 Comparison of our decoding radius with the unique decoding radius for different rates.

1.3 Our Techniques

In the topics of list decoding, folded codes and subfield codes are used to increase decoding radius. In the case of Hamming-metric, folding codes or taking evaluation points from a subfield increases code alphabet size, while in the case of rank-metric, folding codes or taking evaluation points from a subfield would reduce the column-to-row ratio. In the Subsection 1.1, we reviewed some techniques employed in the explicit constructions of list decodable rank-metric codes. We start from a family of list decodable rank-metric codes, i.e., folded Gabidulin codes. The list decoding algorithm for this family was already known

[11, 12, 17]. In this paper, we introduce a new technique, namely the compression technique. By combing our compression technique with the existing techniques of folded codes, we are able to increase the column-to-row ratio.

Let us illustrate our idea by combining the compression technique with the techniques of folded codes. The folded technique for list decoding of rank-metric codes was introduced in [17]. Similar to list decoding of folded Reed-Solomon codes, one can list decode folded Gabidulin codes. However, when Gabidulin codes are folded, the number of columns increases. This means that the column-to-row ratio decreases. In order to make the column-to-row ratio larger for folded Gabidulin codes, one can take a linear map that sends every element of $\mathbb{F}_{q^n}$ to a smaller $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^n}$. Thus, the number of columns shrinks. At the meantime, we still want a large list decoding radius or at least a list decoding radius exceeding $(1-R)/2$. We can choose the compression map to be a linearized polynomial and use the existing linear algebra list decoding technique [8, 10, 11, 12] to achieve our goal.

1.4 Organization of the paper

In Section 2, we introduce some preliminaries including definitions of rank-metric codes and rank-metric balls, Gabidulin codes, subspace design and periodic subspaces. In Section 3, we compress folded rank-metric code and design an efficient list decoding algorithm.

2 Preliminaries

2.1 Rank-metric codes

We first introduce some basic notations and properties about rank-metric codes. Denote by $\mathbb{F}_q^{t\times n}$ the collection of all $t\times n$ matrices over $\mathbb{F}_q$. Without loss of generality, we assume that $n\leq t$ in this paper or otherwise we can consider transpose of matrices. A rank-metric code is a subset of $\mathbb{F}_q^{t\times n}$. Denote by ρ the column-to-row ratio, i.e, $\rho=\frac{n}{t}$, then we always have $\rho\leq 1$. For any $X,Y\in\mathbb{F}_q^{t\times n}$, the rank distance between X and Y is defined by

$$d_R(X,Y):=\mathrm{rank}(X-Y),$$

where rank denotes the rank of matrices. It is straightforward to verify that d_R is indeed a distance. Similar to classical block codes, we can define minimum rank distance and rate for a rank-metric code $\mathcal{C}$ by

$$d_R(\mathcal{C})=\min_{X\neq Y\in\mathcal{C}}\{\mathrm{rank}(X-Y)\}\quad \textit{and}\quad R(\mathcal{C})=\frac{\log_q|\mathcal{C}|}{nt}.$$

A rank-metric code in $\mathbb{F}_q^{t\times n}$ with $n\leqslant t$ must obey the following Singleton bound

$$d_R(\mathcal{C})\leqslant n-R(\mathcal{C})n+1. \tag{1}$$

The rank-metric ball, an analog to the Hamming ball in classical block codes, is used to count the number of matrices within a given rank distance. The formal definition is given as follows.

▶ **Definition 3.** For a real $\tau\in[0,1]$, the rank-metric ball with center $X\in\mathbb{F}_q^{t\times n}$ and distance τn is defined by

$$\mathcal{B}_R(X,\tau n):=\{Y\in\mathbb{F}_q^{t\times n}:\ d_R(X,Y)\leq\tau n\}.$$

The size of a rank-metric ball is independent of the center.

For convenience, a vector of length t over $\mathbb{F}_q$ is identified with a column vector of $\mathbb{F}_{q^t}$ under a fixed basis. Thus, a row vector in $\mathbb{F}_{q^t}^n$ can be viewed as an $t\times n$ matrix over $\mathbb{F}_q$. We denote by $d_R(\mathbf{x},\mathbf{y})$ the rank distance $d_R(X,Y)$, where $\mathbf{x},\mathbf{y}$ are vectors in $\mathbb{F}_{q^n}^t$ corresponding to X,Y, respectively.

2.2 Gabidulin codes

A code achieving the Singleton bound (1) is called Maximal Rank Distance (or MRD for short) code. The most famous MRD codes are Gabidulin codes which are defined by using polynomial evaluations.

To better understand our codes, we briefly review the construction of Gabidulin codes [4]. A polynomial of the form $f(x) = \sum_{i=0}^{\ell} a_i x^{q^i}$ is called q-linearized, where coefficients a_i belong to the algebraic closure of $\mathbb{F}_q$. The q-degree of $f(x)$, denoted by $\deg_q(f)$, is defined to be ℓ if $a_\ell \neq 0$. Denote by $\mathcal{L}_q(n,k)$ the subset

$$\mathcal{L}_q(n,k) := \left\{ \sum_{i=0}^{k-1} a_i x^{q^i} \,:\, a_i \in \mathbb{F}_{q^n} \right\}. \tag{2}$$

Then $\mathcal{L}_q(n,k)$ is an $\mathbb{F}_{q^n}$-vector space of dimension k and it is also an $\mathbb{F}_q$-vector space of dimension kn. Denote by $\mathcal{L}_q(n)$ the union $\cup_{k=1}^{\infty}\mathcal{L}_q(n,k)$, i.e., $\mathcal{L}_q(n)$ is the collection of q-linearized polynomials over $\mathbb{F}_{q^n}$.

Fix an $\mathbb{F}_q$-linearly independent set $\{\alpha_1, \ldots, \alpha_n\}$ of $\mathbb{F}_{q^t}$. For every q-linearized polynomial $f \in \mathbb{F}_{q^t}[X]$ of q-degree at most $k-1$ with $1 \leqslant k \leqslant n$, we can encode f by the row vector $\big(f(\alpha_1), \ldots, f(\alpha_n)\big)$ over $\mathbb{F}_{q^t}$. By fixing a basis of $\mathbb{F}_{q^t}$ over $\mathbb{F}_q$, we can also think of this row vector as an $t \times n$ matrix over $\mathbb{F}_q$. This yields the Gabidulin code

$$\mathcal{G}_q(n,k) := \{\big(f(\alpha_1), \ldots, f(\alpha_n)\big) \in \mathbb{F}_q^{t\times n} \,:\, f \in \mathcal{L}_q(n,k)\}. \tag{3}$$

The Gabidulin code $\mathcal{G}_q(n,k)$ is an MRD code with rate $\frac{k}{n}$ and minimum rank distance $n-k+1$.

2.3 Subspace design

Subspace design was introduced in [11] to reduce list size from a structured list. Let us recall the definition.

▶ **Definition 4.** A collection $\mathcal{S}$ of $\mathbb{F}_q$-subspaces $H_1, \ldots, H_M \subseteq \mathbb{F}_q^n$ is called an $(s, \ell, n)_q$-subspace design if for every $\mathbb{F}_q$-linear space $W \subset \mathbb{F}_q^n$ of dimension s, one has $\sum_{i=1}^{M} \dim_{\mathbb{F}_q}(H_i \cap W) \leq \ell$.

Random subspace designs are studied in [11]. Guruswami and Kopparty [6] gives an explicit subspace design based on Wronskian determinant.

▶ **Lemma 5.** *For $\varepsilon \in (0,1)$, any prime power q and positive integers s, n with $s < \varepsilon n/4$, there is an explicit collection of $M = q^{\Omega(\varepsilon n/s)}$ subspaces in $\mathbb{F}_q^n$, each of codimension at most εn and form an $(s, 2s^2/\varepsilon, n)_q$-subspace design. Moreover, bases for $N \leqslant M$ elements of this collection can be computed in time* $\mathrm{poly}(N, n, q)$.

▶ Remark 6.

(i) If $q > n$, one can improve the intersection size from $2s^2/\varepsilon$ to $2s/\varepsilon$ by applying the subspace design based on the folded Reed-Solomon directly. For $q < n$, the approach in [6] first constructed a weak subspace design and then turn this weak subspace design to a subspace design given in Definition 4. Such transformation yields a $(s, 2s^2/\varepsilon, n)$-subspace design instead of $(s, 2s/\varepsilon, n)$-subspace design.

(ii) If $s = \Omega(\log_q n)$, then a construction of subspace designs with better parameters was given in [13]. For our applications, we are interested in the case where s is a constant.

2.4 Periodic Subspaces

The periodic subspace was introduced in [10] to characterize the list of candidates outputted by the list decodable codes. By exploiting the structure of periodic subspace, they manage to cut down the list size to polynomial size at cost of losing arbitrary small rate.

For a vector $\mathbf{a} = (a_1, a_2, \ldots, a_N) \in \mathbb{F}_q^N$ and positive integers $t_1 \leqslant t_2 \leqslant m$, we denote by $\mathrm{proj}_{[t_1,t_2]}(\mathbf{a}) \in \mathbb{F}_q^{t_2-t_1+1}$ its projection onto coordinates t_1 through t_2, i.e., $\mathrm{proj}_{[t_1,t_2]}(\mathbf{a}) = (a_{t_1}, a_{t_1+1}, \ldots, a_{t_2})$. When $t_1 = 1$, we use $\mathrm{proj}_t(\mathbf{a})$ to denote $\mathrm{proj}_{[1,t]}(\mathbf{a})$. These notions are extended to subsets of strings in the obvious way: $\mathrm{proj}_{[t_1,t_2]}(S) = \{\mathrm{proj}_{[t_1,t_2]}(\mathbf{x}) : \mathbf{x} \in S\}$.

▶ **Definition 7.** For positive integers s, b, n, an affine subspace $H \subset \mathbb{F}_q^{nb}$ is $(s, n, b)_q$-periodic if there exists a subspace $W \subseteq \mathbb{F}_q^n$ of dimension at most s such that for every $j = 1, 2, \ldots, b$, and every "prefix" $\mathbf{a} \in \mathbb{F}_q^{(j-1)n}$, the projected affine subspace of $\mathbb{F}_q^n$ defined as

$$\{\mathrm{proj}_{[(j-1)n+1,jn]}(\mathbf{x}) : \mathbf{x} \in H \text{ and } \mathrm{proj}_{(j-1)n}(\mathbf{x}) = \mathbf{a}\}$$

is contained in an affine subspace of $\mathbb{F}_q^n$ given by $W + \mathbf{v_a}$ for some vector $\mathbf{v_a} \in \mathbb{F}^n$ dependent on $\mathbf{a}$.

By combining subspace design and periodic affine spaces, we can pin down list of massages in Sections 3 and 4. The detailed result is shown below and was given in [11].

▶ **Lemma 8.** *Suppose $H_1, H_2, \ldots, H_b$ is an (s, ℓ, n)-subspace design in $\mathbb{F}_q^n$, and T is a (s, n, b)-periodic affine subspace of $\mathbb{F}_q^{nb}$. Then the set $\mathcal{T} = \{(\mathbf{f_1}, \mathbf{f_2}, \ldots, \mathbf{f_b}) \in T : \mathbf{f_j} \in H_j \text{ for } j = 1, 2, \ldots, b\}$ is an affine subspace of $\mathbb{F}_q^{nb}$ of dimension at most ℓ.*

3 Compressing the folded Gabidulin codes

In this section, we introduce the compression technique and combine this technique with folded Gabidulin codes in order to increase the ratio of folded Gabidulin codes.

3.1 Encoding Algorithm

The encoding algorithm consists of two steps. The first step is to encode a linearized polynomial $f(x)$ to a codeword. In this step, we use $\alpha_1, \ldots, \alpha_n$ as the $\mathbb{F}_q$-basis of $\mathbb{F}_{q^n}$ and evaluate $f(x)$ as $(f(\alpha_1), \ldots, f(\alpha_n))$. The second step is to choose a linearized polynomial $g(x)$ whose kernel is a σn-dimensional subspace of $\mathbb{F}_{q^n}$ for some $\sigma \in (0, 1)$ when $g(x)$ is viewed as an $\mathbb{F}_q$-linear map from $\mathbb{F}_{q^n}$ to itself, then the vector

$$\Big(g(f)(\alpha_1), g(f)(\alpha_2), \ldots, g(f)(\alpha_n)\Big)$$

belongs to a smaller subspace $\mathrm{Im}(g)$, where $\mathrm{Im}(g)$ stands for the image of $g(x)$, i.e., $g(\mathbb{F}_{q^n})$. As $\mathbb{F}_{q^{(1-\sigma)n}} \cong \mathrm{Im}(g)$, the vector $(g(f)(\alpha_1), g(f)(\alpha_2), \ldots, g(f)(\alpha_n))$ can be viewed as a matrix in $\mathbb{F}_q^{(1-\sigma)n \times n}$.

The choice of $g(x)$ can be done as follows. Choose an $\mathbb{F}_q$-subspace $V \subseteq \mathbb{F}_{q^n}$ of dimension σn and define the linearized polynomial $g(x) = \prod_{v \in V}(x - v)$ over $\mathbb{F}_{q^n}$. It follows that $\dim_{\mathbb{F}_q}(\ker(g)) = \sigma n$ and $\dim_{\mathbb{F}_q}(\mathrm{Im}(g)) = (1 - \sigma)n$. For a q-linearized polynomial $a(x) = \sum_{i=0}^{\ell} a_i x^{q^i} \in \mathbb{F}_{q^n}[x]$ and $j \geqslant 0$, we denote by $a^{(j)}(x)$ the polynomial $\sum_{i=0}^{\ell} a_i^{q^j} x^{q^i}$, i.e., $a^{(j)}(x)$ is obtained from $a(x)$ by raising each coefficient to its q^j-th power.

Denote by W_j the image space of $g^{(j)}(x)$. It is clear that W_j is of dimension $(1 - \sigma)n$ as well. Therefore, one can define the $\mathbb{F}_q$-linear isomorphism $\phi_j : W_j \to \mathbb{F}_q^{(1-\sigma)n}$. Let $\mathcal{F}_k(g) := \{g(f(x)) \in \mathcal{L}_q(n) : \deg_q(f) < k\}$. The following lemma shows that if k is not too large, the elements in $\mathcal{F}_k(g)$ are distinct.

▶ **Lemma 9.** *Let $f_1(x), f_2(x)$ be linearized polynomial of q-degree at most $k-1$. If $k+\sigma n \leqslant n$, then $g(f_1(x)) = g(f_2(x))$ if and only if $f_1(x) = f_2(x)$.*

Proof. Assume that $g(f_1(x)) = g(f_2(x))$. Suppose that $f_1(x) \neq f_2(x)$. Then as a linear of map from $\mathbb{F}_{q^n}$ to $\mathbb{F}_{q^n}$, the kernel of $f_1 - f_2$ has dimension at most $k-1$. Thus, the image of $f_1 - f_2$ has dimension at least $n-k+1$. Since $g(x)$ is a q-linearized polynomial, we have

$$g(f_1(x) - f_2(x)) = g(f_1(x)) - g(f_2(x)) = 0.$$

This means that $g(f_1(x) - f_2(x))$ send every element of $\mathbb{F}_{q^n}$ to 0. Hence, $g(x)$ maps every element in the the image of $f_1 - f_2$ to 0. This implies that the image of $f_1 - f_2$ is contained in the kernel of $g(x)$. On the other hand, the dimension of the kernel of $g(x)$ is at most the q-degree of $g(x)$ which is σn. This gives that $\sigma n \geqslant n - k + 1$, i.e., $k + \sigma n \geqslant n + 1$. This contradiction shows that $f_1(x) = f_2(x)$.

The other direction is clear. The proof is completed. ◀

Given linearized polynomials $g(x)$ and $f(x)$, we denote by g_f the linearized polynomial $g(f(x))$. It is easy to see that $g_f^{(i)}(x) = g^{(i)}\big(f^{(i)}(x)\big)$. We encode $g(f(x)) \in \mathcal{F}_k(g)$ to the codeword as follows:

$$M_s(g,f) := \begin{pmatrix} \phi_0\big(g_f(\alpha_1)\big) & \phi_0\big(g_f(\alpha_2)\big) & \cdots & \phi_0\big(g_f(\alpha_n)\big) \\ \phi_1\big(g_f^{(1)}(\alpha_1)\big) & \phi_1\big(g_f^{(1)}(\alpha_2)\big) & \cdots & \phi_1\big(g_f^{(1)}(\alpha_n)\big) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{s-1}\big(g_f^{(s-1)}(\alpha_1)\big) & \phi_{s-1}\big(g_f^{(s-1)}(\alpha_2)\big) & \cdots & \phi_{s-1}\big(g_f^{(s-1)}(\alpha_n)\big) \end{pmatrix} \in \mathbb{F}_q^{(1-\sigma)sn \times n},$$

where ϕ_j is a fixed $\mathbb{F}_q$-linear isomorphism from W_j to $\mathbb{F}_q^{(1-\sigma)n}$. Therefore, $M_s(g,f)$ has $(1-\sigma)sn$ rows and n columns. Each entry in the above matrix is viewed as a row vector of $\mathbb{F}_q^{(1-\sigma)n}$.

Fix a q-linearized polynomial $g(x) \in \mathcal{L}_q(n, \sigma n)$ with the kernel of dimension σn, let $\mathcal{C}_q(n,k;s,\sigma)$ be the collection of $M_s(g,f)$ for all $f(x) \in \mathcal{L}_q(n,k)$ defined in (2).

▶ **Lemma 10.** *If $k + \sigma n \leqslant n$, then the ratio, distance and rate of $\mathcal{C}_q(n,k;s,\sigma)$ satisfy*

$$\rho = \frac{1}{(1-\sigma)s}, \quad d_R(\mathcal{C}_q(n,k;s,\sigma)) \geq n - k - \sigma n + 1, \quad \text{and} \quad R(\mathcal{C}_q(n,k;s,\sigma)) = \frac{k}{s(1-\sigma)n},$$

respectively.

Proof. The ratio is clear. Given a nonzero linearized polynomial $f(x)$, suppose that $M_s(g,f)$ has rank less than $n - k - \sigma n + 1$. The solution space U of $M_s(g,f)x^T = 0$ has dimension at least $k + \sigma n$. Then, $g_f(x)$ has at least $q^{k+\sigma n}$ roots. This implies that g_f is a linearized polynomial of q-degree at least $k + \sigma n$. However, the q-degree of g_f is upper bounded by $k + \sigma n - 1$ as the q-degree of g is σn and the q-degree of f is at most $k-1$. This is a contradiction. It is easy to see that the map $f \mapsto M_s(g,f)$ is $\mathbb{F}_q$-linear and injective, our rank-metric codes are $\mathbb{F}_q$-linear space and its size is q^{kn}. Hence, the rate of this code is $\frac{\log_q(\mathcal{C}_q(n,k,\sigma))}{s(1-\sigma)n^2} = \frac{k}{s(1-\sigma)n}$. ◀

3.2 List Decoding Algorithm

The list decoding algorithm consists of two subroutine algorithms. The first algorithm is an interpolation algorithm which outputs the interpolation polynomial that passes through all points in the vector space of the transmitted matrix. The second algorithm is a root-finding

algorithm which finds out all roots to the interpolation algorithm that belong to the message space $\mathcal{L}_q(n,k)$. However, if our message space is the whole space of $\mathcal{L}_q(n,k)$, the output of this list decoding algorithm may be exponentially large. To reduce the list size, we make use of the subspace design [6] to "re-encode" our rank-metric codes. As far as we know, this technique was the only known method to construct the explicit list-decodable rank-metric codes [17, 9]. The resulting rank-metric code is a subcode of the original rank-metric code with ε rate loss. The list size of our resulting rank-metric code is reduced to a constant $q^{O(\frac{1}{\varepsilon})}$.

Fix a positive integer $e \leq n-s$. Suppose that a codeword $M_s(g,f)$ is transmitted and $M_y = (y_{i,j})_{0\leq i\leq s-1, 1\leq j\leq n}$ is received with at most e errors, i.e., $\mathrm{rank}(M_s(g,f) - M_y) \leq e$. Our goal is to recover the linearized polynomial $f(x)$ from M_y. Note that ϕ_j is an $\mathbb{F}_q$-isomorphism for $j = 0, \ldots, s-1$. We define the matrix $M_z = (z_{i,j})_{0\leq i\leq s-1, 1\leq j\leq n}$, where $z_{i,j} = \phi_i^{-1}(y_{i,j})$. That is, we apply the inverse maps $\phi_0^{-1}, \ldots, \phi_{s-1}^{-1}$ to M_y to retrieve $sn \times n$ matrix M_z over $\mathbb{F}_q$. Define the matrix

$$M'_s(g,f) := \begin{pmatrix} g_f(\alpha_1) & g_f(\alpha_2) & \cdots & g_f(\alpha_n) \\ g_f^{(1)}(\alpha_1) & g_f^{(1)}(\alpha_2) & \cdots & g_f^{(2)}(\alpha_n) \\ \vdots & \vdots & \ddots & \vdots \\ g_f^{(s-1)}(\alpha_1) & g_f^{(s-1)}(\alpha_2) & \cdots & g_f^{(s-1)}(\alpha_n) \end{pmatrix}$$

The following lemma shows that the error $\mathrm{rank}(M_s(g,f) - M_y)$ does not amplify under the inverse maps $\phi_0^{-1}, \ldots, \phi_{s-1}^{-1}$.

▶ **Lemma 11.** *If* $\mathrm{rank}(M_s(g,f) - M_y) \leq e$, *then* $\mathrm{rank}(M'_s(g,f) - M_z) \leq e$.

Proof. Since $\mathrm{rank}(M_s(g,f) - M_y) \leq e$, the solution space $U \subseteq \mathbb{F}_q^n$ of $(M_s(g,f) - M_y)\mathbf{x}^T = \mathbf{0}$ has dimension at least $n-e$, i.e, for every $(c_1, c_2, \ldots, c_n) \in U$ and $i = 0, 1, \ldots, s-1$,

$$\phi_i\left(g_f^{(i)}\left(\sum_{j=1}^n c_j\alpha_j\right)\right) = \sum_{j=1}^n c_j\phi_i\left(g_f^{(i)}(\alpha_j)\right) = \sum_{j=1}^n c_j y_{i,j}.$$

By taking ϕ_i^{-1} on the both sides of the above identity, we get

$$g_f^{(i)}\left(\sum_{j=1}^n c_j\alpha_j\right) = \sum_{j=1}^n \phi_i^{-1}(c_j y_{i,j}) = \sum_{j=1}^n c_j\phi_i^{-1}(y_{i,j}) = \sum_{j=1}^n c_j z_{i,j}.$$

Since it holds for every $(c_1, c_2, \ldots, c_n) \in U$, we come to the conclusion that $\mathrm{rank}(M'_s(g,f) - M_z) \leq e$. ◀

Assuming $\mathrm{rank}(M_s(g,f) - M_z) \leq e$, we will show how to list decode M_z. To begin with, we introduce the interpolation polynomials.

▶ **Definition 12.** *Let* $\mathcal{L}$ *be the space of polynomials* $Q \in \mathbb{F}_{q^n}[X, Z_1, Z_2, \ldots, Z_s]$ *of the form* $Q(X, Z_1, \ldots, Z_s) = A_0(X) + A_1(Z_1) + \cdots + A_s(Z_s)$ *with each* $A_0 \in \mathcal{L}_q(n, D + k + \sigma n)$ *and* $A_i \in \mathcal{L}_q(n, D)$ *for* $i = 1, \ldots, s$.

The interpolation polynomial $Q(X, Z_1, \ldots, Z_s)$ was used to interpolate the points $(\alpha_j, z_{0,j}, \ldots, z_{s-1,j})$ for $j = 1, \ldots, n$. Since our interpolation polynomial is q-linearized, it means Q pass all points in the subspace spanned by $(\alpha_j, z_{0,j}, \ldots, z_{s-1,j})$.

▶ **Lemma 13.** *Assume that* $D > \frac{1}{s+1}(n-k-\sigma n)$. *There exists a nonzero polynomial* $Q \in \mathcal{L}$ *such that* $Q(\alpha_i, z_{0,i}, \ldots, z_{s-1,i}) = 0$ *for* $i = 1, \ldots, n$. *Furthermore,* Q *can be found in time* $poly(n, \log q)$.

Proof. We view coefficients of $A_i(X)$ as variables. Since there are n equations and $(s+1)D + k + \sigma n - 1$ unknowns in $Q(X, Z_1, \ldots, Z_s)$, we require that $(s+1)D + k + \sigma n - 1 > n$ or equivalently $D > \frac{1}{s+1}(n-k+1-\sigma n)$. Note that the n constraints amount to n linear equations. This implies that we can interpolate polynomial Q in running time $O(n^3)$ by Gauss elimination. Moreover, as long as the number of unknowns is bigger than the number of equations, there exists a nonzero polynomial Q satisfying all these n constraints. ◀

We next prove that those codewords with small distance from M_z are the roots of Q. Then, it remains to design a root-finding algorithm to find all the roots of Q.

▶ **Lemma 14.** *Let* $g_f \in \mathcal{F}_k(g)$ *be a* q*-linearized polynomial. If* $\mathrm{rank}(M'_s(g,f) - M_z) \le e$ *and* $D + k + \sigma n - 1 < n - e$, *then* $Q(x, g_f(x), g_f^{(1)}(x), \ldots, g_f^{(s-1)}(x)) = 0$.

Proof. The condition that $\mathrm{rank}(M'_s(g,f) - M_z) \le e$ implies that there exists an $\mathbb{F}_q$-linear subspace U of dimension at least $n-e$ such that for every $(c_1, c_2, \ldots, c_n) \in U$, we have $\sum_{j=1}^n c_j z_{i,j} = \sum_{j=1}^n c_j g_f^{(i)}(\alpha_j)$ for all $i = 0, 1, \ldots, s-1$. This gives

$$\begin{aligned} 0 &= \sum_{j=1}^n c_j Q(\alpha_j, z_{0,j}, \ldots, z_{s-1,j}) = Q\left(\sum_{j=1}^n c_j\alpha_j, \sum_{j=1}^n c_j z_{0,j}, \ldots, \sum_{j=1}^n c_j z_{s-1,j}\right) \\ &= Q\left(\sum_{j=1}^n c_j\alpha_j, \sum_{j=1}^n c_j g_f(\alpha_j), \ldots, \sum_{j=1}^n c_j g_f^{(s-1)}(\alpha_j)\right) \\ &= Q\left(\sum_{j=1}^n c_j\alpha_j, g_f\left(\sum_{j=1}^n c_j\alpha_j\right), \ldots, g_f^{(s-1)}\left(\sum_{j=1}^n c_j\alpha_j\right)\right) \end{aligned}$$

Note that g_f is a linearized polynomial of q-degree at most $k + \sigma n - 1$. Then, $Q(x, g_f(x), \ldots, g_f^{(s-1)}x)$ is a q-linearized polynomial of q-degree at most $D + k + \sigma n - 1$ which is less than the dimension $n - e$ of the kernel. It must be the case that $Q(x, g_f(x), \ldots, g_f^{(s-1)}(x)) = 0$. ◀

▶ **Theorem 15.** *If* $e \le \frac{s}{s+1}((1-\sigma)n - k)$, *then* $Q(x, g_f(x), \ldots, g_f^{(s-1)}x) = 0$ *holds for all linearized polynomials* $g_f(x)$ *with* $rank(M'_{g_f} - M_z) \le e$.

Proof. Set $D = \left\lfloor \frac{1}{s+1}(n-k-\sigma n) + 1 \right\rfloor$. Then Lemma 14 ensures existence of a polynomial $Q(X, Z_1, \ldots, Z_s)$ passing through points $(\alpha_i, z_{0,i}, \ldots, z_{s-1,i})$ for $i = 1, \ldots, n$. Furthermore, Lemma 13 ensures that all linearized polynomials $g_f(x)$ with $\mathrm{rank}(M'_s(g,f) - M_z) \le e$ is a solution to $Q(x, g_f(x), \ldots, g_f^{(s-1)}x) = 0$. This completes the proof. ◀

Recall that the rate of $\mathcal{C}_q(n, k; s, \sigma)$ is $R := \frac{k}{s(1-\sigma)n}$. Plugging $k = sR(1-\sigma)n$ into the expression of $e \le \frac{s}{s+1}((1-\sigma)n - k)$, we obtain the list decoding radius $\tau = \frac{s}{s+1}(1-\sigma)(1-sR)$. As the ratio $\rho = \frac{1}{(1-\sigma)s}$, τ can be expressed as $\frac{1-sR}{\rho(s+1)}$ in terms of the ratio ρ. If we want that the list decoding radius τ exceeds the unique decoding, i.e., $\tau > \frac{1-R}{2}$, then the rate R must satisfy $R < \frac{2-(s+1)\rho}{2s-(s+1)\rho}$. This implies that $\rho < \frac{2}{s+1}$.

If we set $s = 2$, then for any ratio $\rho \in (0, \frac{2}{3})$, we obtain a list decodable rank-metric code of the ratio ρ that exceeds the unique decoding radius $\frac{1-R}{2}$. However, we still need to make sure that the list size of this code is at most polynomial in q, n and there exists explicit list

decoding algorithm to find all candidates. The following lemma tells us the structure of the solutions to $Q(x, g_f(x), g_f^{(1)}(x), \ldots, g_f^{(s-1)}(x)) = 0$. We follow the idea given in [9] to show how to obtain the structure of $g(f(x))$ from Q via the root-finding algorithm.

▶ **Lemma 16.** *Let* $a(x) = \sum_{i=0}^{\sigma n+k-1} a_i x^{q^i} \in \mathcal{L}_q(n)$. *Then the set of solutions* $(a_0, a_1, \ldots, a_{\sigma n+k-1})$ *to the equation*

$$Q(x, a(x), a^{(1)}(x), \ldots, a^{(s-1)}(x)) = 0 \tag{4}$$

forms an $(s-1, n, \sigma n + k - 1)$*-periodic subspace.*

Proof. Let $D = \left\lfloor \frac{1}{s+1}(n-k-\sigma n) + 1 \right\rfloor$. Note that we have already recovered $A_0(x), \ldots, A_s(x)$ by interpolation that satisfy the indentity

$$Q(x, a(x), a^{(1)}(x), \ldots, a^{(s-1)}(x)) = A_0(x) + A_1(a(x)) + \cdots + A_s(a^{(s-1)}(x)) = 0. \tag{5}$$

Assume that $A_0(X) = \sum_{i=0}^{D+k+\sigma n-1} b_{0,i} x^{q^i}$ and $A_j(x) = \sum_{i=0}^{D-1} b_{j,i} x^{q^i}$. If $b_{0,0}, \ldots, b_{s,0}$ are all zero, then (5) gives a new identity $(A_0'(x) + A_1'(a(x)) + \cdots + A_s'(a^{(s-1)}(x)))^q = 0$, i.e.,

$$A_0'(x) + A_1'(a(x)) + \cdots + A_s'(a^{(s-1)}(x)) = 0 \tag{6}$$

with $\deg_q(A_i) \geqslant \deg_q(A_i')$ for all $i = 0, 1, \ldots, s$. Moreover, not all A_i' are zero polynomials. Thus, without loss of generality, we may assume that at least one of $b_{0,0}, \ldots, b_{s,0}$ is nonzero.

Let $a(x) = \sum_{i=0}^{k+\sigma n-1} a_i x^{q^i}$, where $a_i \in \mathbb{F}_{q^n}$ are variables. Plugging the expression of $a(x)$ into (5) and compareing the coefficient of x on both sides give

$$b_{0,0} + \sum_{i=0}^{s-1} b_{i+1,0} a_0^{q^i} = 0. \tag{7}$$

The solution a_0 to $b_{0,j} + \sum_{i=0}^{s} b_{i,j} a_0^{q^{i-1}} = 0$ is an affine subspace of dimension at most $s-1$. For $i = 0, \ldots, k + \sigma n - 1$, define the linearized polynomial

$$B_i(x) = \sum_{j=1}^{s-1} b_{j,i} x^{q^j}.$$

Our assumption shows $B_0(x) \neq 0$. The solutions $\beta \in \mathbb{F}_{q^n}$ to $B_0(x)$ forms a subspace W of dimension at most $s-1$. Fix $i \in \{0, \ldots, k + \sigma - 1\}$. By comparing the coefficient of x^{q^i} in Equation (5), we get

$$b_{0,i} + B_i(a_0^{q^i}) + B_{i-1}(a_1^{q^{i-1}}) + \cdots + B_1(a_{i-1}^q) + B_0(a_i) = 0.$$

This implies $a_i \in W + \theta_i$ for some $\theta_i \in \mathbb{F}_{q^n}$ that is determined by $a_0, \ldots, a_i$. Thus, each choice of a_{i-1} is contained in the coset of W. The proof is completed. ◀

▶ Remark 17. For each a_i, we may have q^{s-1} solutions. Thus, the list of candidate $g_f(x)$ could be exponentially large. To cut down the list size, we pick a subspace of $\mathcal{L}_q(n,k)$ by subspace design. By imposing some constraints on our codeword, we can prune the list to a constant size. We leave it to the next subsection.

Assume that we are given a solution $a(x)$ to $Q(x, a(x), a^{(1)}(x), \ldots, a^{(s-1)}(x))$. Next lemma shows how to obtain $f(x)$ from $a(x)$. Note that not all solutions to

$$Q(x, a(x), a^{(1)}(X), \ldots, a^{(s-1)}(x)) = 0$$

are of the form $g(f(x))$.

▶ **Lemma 18.** *Given a linearized polynomial $a(x)$ of q-degree at most $k+\sigma n-1$, we can find in time $O(n^2)$ whether there exists an unique linearized polynomial $f(x)$ of q-degree at most $k-1$ such that $a(x)=g(f(x))$. Furthermore, $f(x)$ can be uniquely determined if it exists.*

Proof. Let $a(x)=\sum_{i=0}^{k+\sigma n-1} a_i x^{q^i}$, $f(x)=\sum_{i=0}^{k-1} f_i x^{q^i}$ and $g(x)=\sum_{i=0}^{\sigma n} g_i x^{q^i}$. Suppose that $a(x)=g(f(x))$. It follows that

$$a(x)=\sum_{i=0}^{\sigma n} g_i f(x)^{q^i}.$$

Comparing the coefficient of x on both sides, we get $f_0 g_0 = a_0$. Recall that the roots of $g(x)$ form a σn-dimensional subspace which implies $g(x)$ has $q^{\sigma n}$ different roots including 0. This implies g_0 is nonzero and thus f_0 is uniquely determined. Assume that $f_0,\dots,f_{i-1}$ are determined. We compare the coefficient of x^{q^i} on both sides

$$a_i = g_0 f_i + g_1 f_{i-1} + g_2 f_{i-2} + \cdots + g_i f_0.$$

Thus, f_i is uniquely determined. After all coefficients of $f(x)$ are determined, we check whether $a(x)=g(f(x))$. If the equation holds, $f(x)$ is the unique solution. Otherwise, there do not exist any solutions. It is easy to see that all operations run in time $O(n^2)$. ◀

3.3 Prune the list

We follow the standard list decoding procedure introduced in [10, 11, 12] to pre-encode and prune the list size.

▶ **Theorem 19.** *For every finite field $\mathbb{F}_q$, small real $\gamma>0$ and integer $s>1$, there exists an explicit constriction of $\mathbb{F}_q$-linear rank metric codes with the column-to-row ratio ρ and rate R that are $\left(\frac{(1-sR)}{\rho(s+1)}-\gamma, q^{O((s-1)^2/\gamma)}\right)$-list-decodable. The algorithm runs in time poly(n,q). Furthermore, if $\rho<\frac{2}{s+1}$, then the decoding radius $\tau=\frac{(1-sR)}{\rho(s+1)}-\gamma$ exceeds the unique decoding radius $\frac{1-R}{2}$.*

Proof. Note that the message space of our rank metric code is $\mathcal{F}_k(g)=\{g(f):\ f\in\mathcal{L}_q(n,k)\}$. Lemma 5 says that there exists an explicit construction of $((s-1),2(s-1)^2/\varepsilon,n)_q$-subspace design $H_0,\dots,H_{\sigma n+k-1}\subseteq\mathbb{F}_{q^n}$, each has the $\mathbb{F}_q$-dimension $n(1-\varepsilon)$. Define the polynomial set $\mathcal{S}=\{h(x)=\sum_{i=0}^{\sigma n+k-1} h_i x^{q^i} : h_i\in H_i\}$. Our new message space is $\mathcal{F}'_k(g)=\mathcal{F}_k(g)\cap\mathcal{S}$. Note that

$$\begin{aligned}
&\dim_{\mathbb{F}_q}(\mathcal{F}'_k(g)) = \dim_{\mathbb{F}_q}(\mathcal{F}_k(g)) + \dim_{\mathbb{F}_q}(\mathcal{S}) - \dim_{\mathbb{F}_q}(\mathcal{F}_k(g)+\mathcal{S})\\
&\geqslant \dim_{\mathbb{F}_q}(\mathcal{F}_k(g)) + \dim_{\mathbb{F}_q}(\mathcal{S}) - \dim_{\mathbb{F}_q}(\mathbb{F}_{q^n}^{\sigma n+k})\\
&= kn + (\sigma n+k)(1-\varepsilon)n - (\sigma n+k)n = n(k-\varepsilon(\sigma n+k))
\end{aligned}$$

Given a linearized polynomial $g(f(x))\in\mathcal{F}'_k(g)$, we encode it into the codeword $M_s(g,f)$. The new rank-metric code becomes $\mathcal{C}'(n,k;s,\sigma)$. The rate of this code is $R=\frac{n(k-\varepsilon(\sigma n+k))}{n^2 s(1-\sigma)}=\frac{1}{1-\sigma}\left(\frac{k}{n}-\frac{\varepsilon}{s}\left(\sigma+\frac{k}{n}\right)\right)\geqslant\frac{1}{1-\sigma}\left(\frac{k}{n}-\varepsilon\right)=R'-\frac{\varepsilon}{1-\sigma}$ where R' is the rate of $\mathcal{C}(n,k;s,\sigma)$ in Lemma 10.

Since our new code is a subcode of the rank metric code proposed in the Subsection 3.1. The same encoding and list decoding algorithm can be applied to this code. Assume that there are at most $\tau n=\frac{(1-sR')n}{\rho(s+1)}$ rank errors, Lemma 16 says that all candidates $(a_0,\dots,a_{\sigma n+k-1})\in\mathcal{F}_k(g)$ are contained in an $(s-1,n,\sigma n+k)$-periodic subspace. This implies that the collection of such candidates $(a_0,\dots,a_{\sigma n+k-1})\in\mathcal{F}'_k(g)$ is contained in an

affine space of dimension at most $(s-1)^2/\varepsilon$ followed by the property of subspace design Lemma 8. This implies there are at most $q^{(s-1)^2/\varepsilon}$ codewords in the list. Put $\gamma = \frac{2\varepsilon s}{\rho(1-\sigma)(s+1)}$, then $\tau = \frac{(1-sR')}{\rho(s+1)} = \frac{(1-sR)}{\rho(s+1)} - \gamma$.

It takes at most $O(n^3 q^{(s-1)^2/\gamma})$ time to find all candidates. Thus, this list decoding algorithm runs in polynomial time. Our proof is completed. ◀

References

1. Philippe Delsarte. Bilinear forms over a finite field, with applications to coding theory. *J. Comb. Theory, Ser. A*, 25(3):226–241, 1978. doi:10.1016/0097-3165(78)90015-8.
2. Yang Ding. On list-decodability of random rank metric codes and subspace codes. *IEEE Trans. Inf. Theory*, 61(1):51–59, 2015. doi:10.1109/TIT.2014.2371915.
3. Michael A. Forbes and Amir Shpilka. On identity testing of tensors, low-rank recovery and compressed sensing. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 163–172. ACM, 2012. doi:10.1145/2213977.2213995.
4. Ernst Gabidulin. Theory of codes with maximum rank distance (translation). *Problems of Information Transmission*, 21:1–12, January 1985.
5. Venkatesan Guruswami. *List decoding of error correcting codes*. PhD thesis, Massachusetts Institute of Technology, 2001. URL: http://dspace.mit.edu/handle/1721.1/8700.
6. Venkatesan Guruswami and Swastik Kopparty. Explicit subspace designs. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 608–617. IEEE Computer Society, 2013. doi:10.1109/FOCS.2013.71.
7. Venkatesan Guruswami and Atri Rudra. Better binary list decodable codes via multilevel concatenation. *IEEE Trans. Inf. Theory*, 55(1):19–26, 2009. doi:10.1109/TIT.2008.2008124.
8. Venkatesan Guruswami and Carol Wang. Linear-algebraic list decoding for variants of reed-solomon codes. *IEEE Trans. Inf. Theory*, 59(6):3257–3268, 2013. doi:10.1109/TIT.2013.2246813.
9. Venkatesan Guruswami, Carol Wang, and Chaoping Xing. Explicit list-decodable rank-metric and subspace codes via subspace designs. *IEEE Trans. Inf. Theory*, 62(5):2707–2718, 2016. doi:10.1109/TIT.2016.2544347.
10. Venkatesan Guruswami and Chaoping Xing. Folded codes from function field towers and improved optimal rate list decoding. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 339–350. ACM, 2012. doi:10.1145/2213977.2214009.
11. Venkatesan Guruswami and Chaoping Xing. List decoding reed-solomon, algebraic-geometric, and gabidulin subcodes up to the singleton bound. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 843–852. ACM, 2013. doi:10.1145/2488608.2488715.
12. Venkatesan Guruswami and Chaoping Xing. Optimal rate list decoding over bounded alphabets using algebraic-geometric codes. *J. ACM*, 69(2):10:1–10:48, 2022. doi:10.1145/3506668.
13. Venkatesan Guruswami, Chaoping Xing, and Chen Yuan. Subspace designs based on algebraic function fields. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 86:1–86:10. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.86.
14. Ralf Koetter and Frank R. Kschischang. Coding for errors and erasures in random network coding. *IEEE Trans. Inf. Theory*, 54(8):3579–3591, 2008. doi:10.1109/TIT.2008.926449.
15. Antonia Wachter-Zeh. Bounds on list decoding of rank-metric codes. *IEEE Trans. Inf. Theory*, 59(11):7268–7277, 2013. doi:10.1109/TIT.2013.2274653.

16 Huaxiong Wang, Chaoping Xing, and Reihaneh Safavi-Naini. Linear authentication codes: bounds and constructions. *IEEE Trans. Inf. Theory*, 49(4):866–872, 2003. doi:10.1109/TIT.2003.809567.

17 Chaoping Xing and Chen Yuan. A new class of rank-metric codes and their list decoding beyond the unique decoding radius. *IEEE Trans. Inf. Theory*, 64(5):3394–3402, 2018. doi:10.1109/TIT.2017.2780848.

Breaking the All Subsets Barrier for Min k-Cut

Daniel Lokshtanov
University of California Santa Barbara, CA, USA

Saket Saurabh
The Institute of Mathematical Sciences, HBNI, Chennai, India
University of Bergen, Norway

Vaishali Surianarayanan
University of California Santa Barbara, CA, USA

Abstract

In the Min k-Cut problem, the input is a graph G and an integer k. The task is to find a partition of the vertex set of G into k parts, while minimizing the number of edges that go between different parts of the partition. The problem is NP-complete, and admits a simple $3^n \cdot n^{\mathcal{O}(1)}$ time dynamic programming algorithm, which can be improved to a $2^n \cdot n^{\mathcal{O}(1)}$ time algorithm using the fast subset convolution framework by Björklund et al. [STOC'07]. In this paper we give an algorithm for Min k-Cut with running time $\mathcal{O}((2-\varepsilon)^n)$, for $\varepsilon > 10^{-50}$. This is the first algorithm for Min k-Cut with running time $\mathcal{O}(c^n)$ for $c < 2$.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Exact algorithms, min k-cut, exponential algorithms, graph algorithms, k-way cut

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.90

Category Track A: Algorithms, Complexity and Games

Funding *Daniel Lokshtanov*: NSF award CCF-2008838.
Saket Saurabh: European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 819416), and Swarnajayanti Fellowship grant DST/SJF/MSA01/2017-18.
Vaishali Surianarayanan: NSF award CCF-2008838.

1 Introduction

A *k-cut* of a graph G is a partition of the vertex set $V(G)$ into k non-empty parts. The *weight* of a k-cut is the total number of edges with endpoints in different parts of the partition. In the Min k-Cut problem the input is a graph G on n vertices and m edges, and an integer k. The task is to find a k-cut of G of minimum weight.

The problem is known to be NP-complete [23] and is extremely well studied from the perspective of approximation algorithms [42, 47, 48], parameterized algorithms [35, 13, 15], extremal combinatorics [33, 27, 28], and more recently parameterized approximation [26, 25, 34, 40]. For all of the above perspectives the best known algorithmic results come quite close to asymtotically matching existing combinatorial or complexity theoretic lower bounds: On one hand, there are several $2(1-\frac{1}{k})$-approximation algorithms that run in time polynomial in n and k [42, 47, 48]. On the other hand, this approximation ratio cannot be improved assuming the Small Set Expansion Hypothesis (SSE) [41]. A $(1+\epsilon)$-approximation algorithm with running time $(k/\epsilon)^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ was recently obtained by Lokshtanov et al. [40], the running time of this algorithm cannot be substantially improved without violating the Exponential Time Hypothesis (ETH).

The fastest known algorithm with parameter k by Gupta et al. [25] runs in time $\mathcal{O}(f(k)n^{k+o(1)})$. At the same time, an algorithm with running time $\mathcal{O}(f(k)n^{(1-\epsilon)\frac{\omega}{3}+o(1)})$ for $\epsilon > 0$, where $\omega < 2.373$ [2] is the matrix multiplication exponent, would imply an improved algorithm for k-Clique. An $\mathcal{O}(f(k)n^{(1-\epsilon)k+o(1)})$ time algorithm for $\epsilon > 0$ for the edge weighted version of Min k-Cut would imply an improved algorithm for Max-Weight k-Clique. This has been conjectured not to exist [1, 3]. The algorithm of Gupta et al. [27, 28] also proves that the number of minimum weight k-cuts in a graph is upper bounded by $\mathcal{O}(f(k)n^{k+o(1)})$, coming very close to matching the $(n/k)^k$ lower bound obtained from the complete k-partite graph.

While exact algorithms for Min k-Cut for small values of k have received considerable attention since the early 1990's [23], exact exponential time algorithms whose running time is measured in terms of the number n of vertices only remain largely unexplored.

Naive Algorithm. A simple $3^n n^{\mathcal{O}(1)}$ time algorithm can be obtained by reduction to finding a minimum weight path on exactly k edges in the following directed acyclic graph S_G: The graph S_G has a vertex for every subset X of $V(G)$. For every pair X, Y of vertex subsets, S_G has an edge from X to Y if X is a subset of Y. The weight of this edge is the number of edges in G with one endpoint in X and the other in $Y \setminus X$. It is easy to see that S_G is a directed acyclic graph, and that there is a one to one correspondence between k-cuts in G and paths from $\emptyset$ to $V(G)$ in S_G using exactly k edges. The graph S_G has 2^n vertices and 3^n edges because every edge XY in S_G corresponds to a partition of $V(G)$ into 3 parts, namely $(X, Y \setminus X, V(G) \setminus Y)$. Finding a minimum weight path using exactly k edges in a DAG can be done in time $\mathcal{O}(k(|V(S_G)| + |E(S_G)|)) = 3^n n^{\mathcal{O}(1)}$. Note that this beats $\mathcal{O}(n^k)$ time algorithms for instances where $k \geq \frac{2n}{\log n}$.

In 2009 Björklund et al. [9] gave a general purpose approach, based on the inclusion-exclusion principle, to solve partitioning problems in $2^n n^{\mathcal{O}(1)}$ time. Their applications (see [9, Proposition 6]) include a $2^n n^{O(1)}$ time algorithm for Min k-Cut. Prior to this work the algorithm of Björklund et al. [9] has remained the state of the art.

When the best algorithm for a well-studied problem is stuck at 2^n for a long time it is prudent to ask whether it is possible to do better at all, or whether 2^n really is the best possible. On one hand, better than 2^n time algorithms have been found for a number of problems, including, among many others k-SAT [49, 46] and the satisfiability problem for a number of circuit classes [10, 39, 38], Hamiltonian Cycle [4], k-Coloring for $k \in \{3, 4, 5, 6\}$ [19, 57], Bin Packing [45], Max-Cut [51], Chordal Vertex Deletion [11], Treewidth [21] and Scheduling Partially Ordered Jobs [17]. On the other hand the failure to find better than 2^n time algorithms for the satisfiability of CNF formulas and Set Cover has led to conjectures [14, 31] that no substantially better algorithms for these problems are possible, driving the field of fine-grained complexity [53, 54, 52]. For problems such as Directed Hamiltonicity, Travelling Salesman and Chromatic Number, obtaining better than 2^n time algorithms are outstanding open problems [20, 44, 55, 56]. In this paper we give the first algorithm for Min k-Cut with running time $\mathcal{O}((2-\varepsilon)^n)$ for $\varepsilon > 0$.

▶ **Theorem 1.** *There exists an algorithm that takes as input an unweighted simple graph G on n vertices, an integer $k > 0$ and returns the min k-cut of G in time $\mathcal{O}(2-\varepsilon)^n$ for some $\varepsilon > 0$.*

Observe that Theorem 1 only considers *unweighted simple graphs*. For Min k-Cut on *weighted* graphs, previous to our work, the best algorithm was the $\mathcal{O}(2^n W^{O(1)})$ time algorithm of Björklund et al. [9]. Our naive $\mathcal{O}(3^n)$ time algorithm described above can

easily be made to work for weighted graphs with running time $3^n(\log W)^{\mathcal{O}(1)}$. As part of the proof of Theorem 1 we make an alternative algorithm for weighted graphs that runs in time $2^{n+o(n)}(\log W)^{\mathcal{O}(1)}$.

▶ **Theorem 2.** *There exists an algorithm that takes as input an edge-weighted graph G on n vertices having weights on edges from $[0, W]$, an integer $k > 0$ and returns the min k-cut of G in time $2^{n+o(n)}(\log W)^{\mathcal{O}(1)}$.*

Theorem 2 is the current best algorithm for Min k-Cut for the weighted case. A natural question that arises here is whether one can improve the running time to $(2-\epsilon)^n(\log W)^{\mathcal{O}(1)}$. The answer seems to be leaning towards a no, or at the very least towards a "that's a pretty difficult question". From the perspective of $(2-\epsilon)^n$ time algorithms it is folklore that (Edge Weighted) Densest Subgraph is at least as hard as Edge Weighted Clique. In the latter problem the goal is to find a maximum weight clique in an edge weighted graph having edge weights that may be positive, negative, or even exponential in n. The best known algorithms for Edge Weighted Clique are $2^n \cdot (\log W)^{\mathcal{O}(1)}$ and $2^{\frac{\omega}{3}n}W^{\mathcal{O}(1)}$, where W is the maximum edge weight and ω is the matrix multiplication exponent. Edge Weighted Clique is a basic problem - n^k hardness of the problem is a conjecture that is sometimes used as a basis for hardness in fine grained complexity [1, 3]. From here it is not too big of a leap to conjecture that it is hard to get $(2-\epsilon)^n \cdot (\log W)^{O(1)}$ time algorithms as well. So it makes a lot more sense to focus on a $(2-\epsilon)^n \cdot (\log W)^{O(1)}$ time algorithm for Edge Weighted Clique before attempting such an algorithm for (Edge Weighted) Min k-Cut.

At a very high level our algorithm for Min k-Cut is based on a dynamic programming algorithm operating on a pruned state space having less than 2^n states. Based on how the input instance and optimal solution look, we use different methods to prune the state space. We show that the cases for which such pruning strategies don't work are "Densest t-Subgraph instances in disguise" with the optimal solution having only one non-singleton part. Here we apply the $\mathcal{O}(2^{\frac{\omega n}{3}}n^{\mathcal{O}(1)})$ time matrix multiplication based algorithm for Densest t-Subgraph by Chang et.al. [12]. The final running time is just the maximum over the running time of the algorithms we design for the various cases we consider. Our algorithm carefully balances various tools and techniques from past work on Min k-Cut, graph theory and the exact algorithms world. This includes DP table sparsification, Thorup tree packing, split and list, graph sparsification, as well as the $\binom{p+q}{p}$ combinatorial bound for the number of connected vertex sets of size p with q neighbors.

Admittedly our algorithm chops the input space into various pieces based on the properties of the input instance and the optimal solution and thus uses quite a handful of cases. One might ask to which degree this amount of case work is necessary. At the very least, the same reduction[1] that showed that *weighted* Min k-Cut is at least as hard as *weighted* Densest t-Subgraph shows that *unweighted* Min k-Cut is at least as hard as *unweighted* Densest t-Subgraph. The only known $(2-\epsilon)^n$ time algorithm [12] for Densest t-Subgraph is based on the Split and List method of Williams [51]. Therefore, the set of inputs to Min k-Cut contains both "Densest t-Subgraph instances in disguise", as well as instances that behave very differently. For an example instances with a few large cliques with few edges between each other behave much more like instances of classic "cut and separation" or clustering problems, and do not appear to be solvable by Split and List based approaches. Hence, a $(2-\epsilon)^n$ algorithm for unweighted Min k-Cut either has to invent a new method for Densest t-Subgraph (which is a very interesting research goal in and of itself) or somehow

[1] To reduce from Densest t-Subgraph to Min k-Cut, add a universal vertex and set $k = n + 2 - t$.

separate the instances of Min k-Cut into ones that are "Densest t-Subgraph instances in disguise" and the ones that can be handled by other means. Of course this only justifies the split into two cases, as opposed to our rather large case tree. It is a nice open problem whether it is possible to handle all of the non-Densest t-Subgraph instances in a more uniform way. This would likely also result in a better running time bound.

We remark that the improvement over 2^n obtained in Theorem 1 is small and holds only for unweighted simple graphs. But importantly it shows that 2^n is not the boundary for Min k-Cut. We hope that this work will initiate a line of research on exact algorithms for Min k-Cut just as the $(2-\epsilon)$-approximation by Gupta et.al [26] kick started a series of work [25, 34, 40] on FPT approximation for Min k-Cut, or the $\mathcal{O}(k^{\mathcal{O}(k)}n^{(\frac{2\omega}{3}+o(1))k})$ time algorithm of Gupta et al. [25] led to a chain of improvements [37, 27, 29, 30] for $n^{\mathcal{O}(k)}$ time algorithms for the problem.

1.1 Algorithm Overview

The first substantial hurdle in designing an algorithm for Min k-Cut that beats the bound of 2^n, is that even getting an algorithm with running time 2^n is non-trivial. The Min k-Cut problem is a graph partitioning problem, and thus the first approach to design an algorithm for Min k-Cut is to explore methods developed for such problems. All known general methods for set partitioning problems, such as those developed by Björklund et al. [9, 5], rely on enumerating all vertex subsets. Known approaches to speed up such methods rely on the input graph being sufficiently sparse, such as having bounded degree or bounded average degree [6, 7, 8, 16, 24]. As we cannot make any such assumptions here we first design an alternative stand-alone $2^{n+o(n)}$ time algorithm that does not make use of the methods for set partitioning problems of Björklund et al. [9].

Our new $2^{n+o(n)}$ time algorithm is based on the fact that Min k-Cut is solvable in time $n^{\mathcal{O}(k)}$. Note that the running time of the naive algorithm, based on a reduction to finding a minimum weight path on exactly k edges in the state graph S_G, is 3^n because S_G has many edges. If we restrict ourselves to only using edges of S_G that correspond to parts of size at most $\mathcal{O}(\frac{n}{\log n})$, then the out-degree of every vertex in the state graph drops to $2^{o(n)}$, leading to an algorithm of running time $2^{n+o(n)}$, that computes for every subset $S \subseteq V(G)$ and integer $k \leq n$ the best way of partitioning S into k parts, each of size at most $\frac{n}{\log n}$. After doing this, in order to find the best partition of $V(G)$ that may or may not use large parts (i.e parts bigger than $\frac{n}{\log n}$), we go over all choices of S and find the best partition of $V(G) \setminus S$ using the $n^{\mathcal{O}(k)}$ time algorithm of Thorup [50]. Since, all the parts in $V(G) \setminus S$ are large we only need to invoke the $n^{\mathcal{O}(k)}$ time algorithm with $k = \mathcal{O}(\log n)$. In the remainder of this outline, when we talk about the state graph S_G we will refer to the *sparsified* state graph with only $2^{n+o(n)}$ edges whose paths correspond to cuts with parts of size at most $\frac{n}{\log n}$

To design an algorithm for Min k-Cut with running time $\mathcal{O}((2-\varepsilon)^n)$, for some fixed $\varepsilon > 0$, we consider many special cases and design faster algorithms for each individual case. Some of the cases are handled by standard methods, while some require interesting new insights. In Figure 1 we provide a case tree depicting the various cases our algorithm branches into - we suggest the reader to refer to the case tree while reading the overview to get a complete picture. The first case we consider is when $k < 0.24n$. Here, k is so small that one would expect ideas from the $n^{\mathcal{O}(k)}$ time algorithm to apply, but too large to use them as black boxes because this leads to a running time of $n^{\mathcal{O}(n)}$. We use the main object behind the deterministic $n^{\mathcal{O}(k)}$ time algorithm, namely Thorup trees [50].

Thorup's algorithm is based on a tree-packing theorem which allows to efficiently find a tree T that crosses the optimal k-cut at most $2k-2$ times. Using T one can design a $\binom{n}{2k}3^{2k}$ time algorithm [50]. If $k < 0.24n$ then we can pick a random edge e from the tree and

declare that it is not part of the cut. We know that we fail (i.e. that we picked an edge from the optimal solution) with probability at most $\frac{2k}{n} \leq 0.48$. However, if we succeed we can contract the edge e, leading to an instance with $n-1$ vertices. Standard running-time/success probability trade-offs now yield that the running time of our algorithm is upper bounded by the recurrence $T(n) \leq \frac{T(n-1)}{0.52}$, which solves to $\mathcal{O}(1.93^n)$. Since we aim for a deterministic algorithm, instead of picking edges at random we directly use a pseudo-random construction from [18] instead.

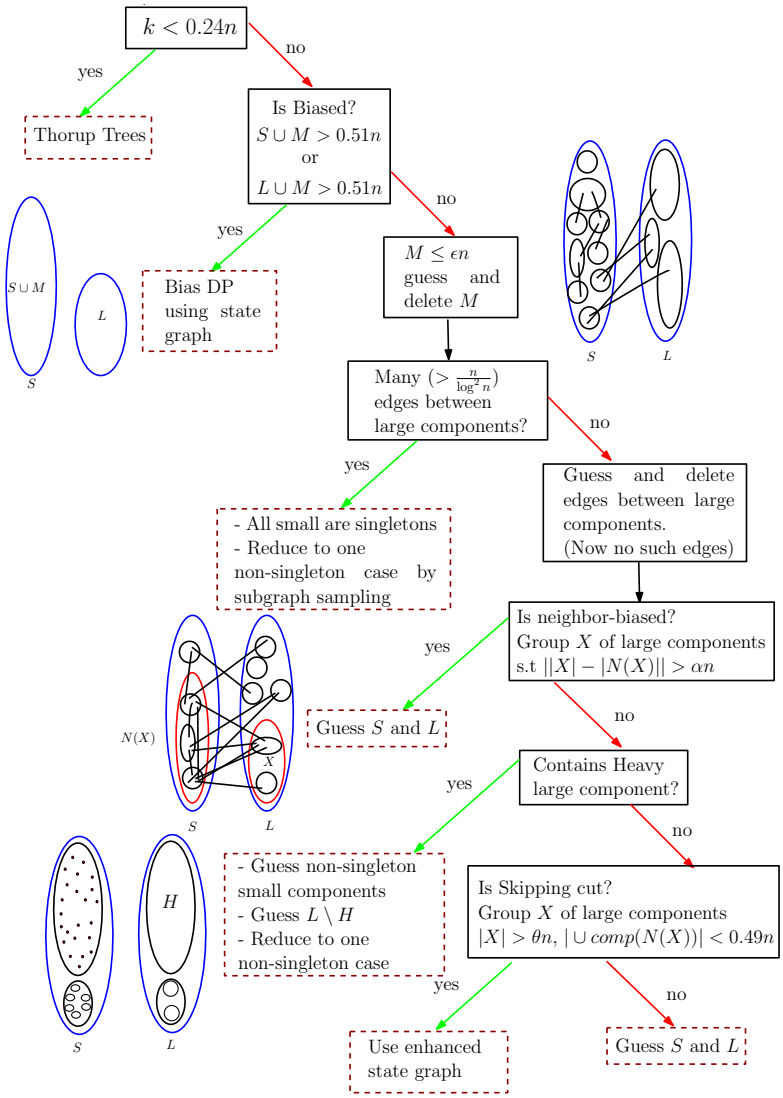


Figure 1 Case Tree depicting the various cases our algorithm branches into. Leaf cases are denoted by dotted brown boxes. Supporting figures are inserted to visualize the cases. The notation $\cup comp(N(X))$ denotes the set of all vertices in components having vertices from $N(X)$.

From now onwards we assume that $k \geq 0.24n$. Let C be a hypothetical solution, that is, an optimum k-cut. We split the parts of the solution into small, medium, and large, as follows. A component is said to be *small* if its size is at most $\log n$, *medium* if its size is greater than $\log n$ and less than $\frac{n}{\log n}$, and *large* if its size is at least $\frac{n}{\log n}$. Given a cut C

of G, we denote the set of all vertices in small, medium, and large components in C by S, M and L, respectively. The next case we deal with is that $|S|$ is at most $n(\frac{1}{2}-\epsilon)$. Here, our main observation is that we can make the $2^{n+o(n)}$ time algorithm already run in time $\mathcal{O}((2-\varepsilon)^n)$; only build the state graph, S_G, for vertex sets of size at most $n(\frac{1}{2}-\epsilon)$. For each choice of S use the $\binom{n}{2k}3^{2k}$ time algorithm to find the best partition of $V(G)\setminus S$ into medium and large size parts. Since, $k \leq \frac{n}{\log n}$, the $\binom{n}{2k}3^{2k}$ time algorithm runs in time $(\frac{en}{n/\log n})^{n/\log n}3^{\mathcal{O}(n/\log n)} = 2^{o(n)}$.

The next case we handle is when $|S \cup M| \geq n(\frac{1}{2}+\epsilon)$. This is the first interesting case. We use the fact that $k > 0.24n$ to infer that there are many small parts. Indeed, observe that at most $0.2n$ parts of C can have size at least 5. Thus, at least $0.04n$ parts of C have size at most 4. To handle this case we extend the idea of Koivisto [36] for speeding up the algorithm for Set Cover when all sets are small (also see the recent generalization by Nederlof [43]). This case is similar to the Set Cover case handled by Nederlof in [43] and our algorithm is based on ideas similar to those used in [36, 43, 45]. Nevertheless we are not able to directly apply these results (it would not be surprising to us if the *methods* used in [43] do apply in this setting).

Specifically, our algorithm proceeds as follows, we pick uniformly at random a set Q, which has the following properties with probability close to 1.

- $|Q| = \frac{n}{2} \pm o(n)$,
- $|Q \cap (S \cup M)| = \frac{|S\cup M|}{2} \pm o(n)$
- At least $\frac{n}{100} - o(n)$ parts P of C of size at most 10 satisfy that $|P \setminus Q| > |P \cap Q|$.
- At least $\frac{n}{100} - o(n)$ parts P of C of size at most 10 satisfy that $|P \setminus Q| < |P \cap Q|$.

Consider now the *ordered* partition **of parts in** C in which: all parts with $|P \cap Q| > |P \setminus Q|$ come first, ordered by size (smaller parts first). Then come all parts P with $|P \cap Q| = |P \setminus Q|$ in any order, followed by all parts with $|P \cap Q| < |P \setminus Q|$ ordered by size (this time smaller parts last). A simple counting argument shows that the path C' in the state graph that corresponds to this ordered partition only visits sets X with the following property:

- $|X| \leq n(\frac{1}{2} - \frac{\epsilon}{10})$ or,
- $|X| \geq n(\frac{1}{2} + \frac{\epsilon}{10})$ or,
- $|X \cap L| \geq \frac{n}{2}(\frac{1}{2} + \frac{\epsilon}{100})$

Thus, to find the path C' we only need to build the subgraph of the state graph with vertices of the three types above. The total number of vertices of this types is upper bounded by $\mathcal{O}((2-\varepsilon)^n)$, leading to an improved algorithm.

The instances not handled by any of the previously discussed cases must satisfy: $k > 0.24n$, $|S|$ and $|L|$ are both between $n(\frac{1}{2}-\epsilon)$ and $n(\frac{1}{2}+\epsilon)$, and $|M|$ is at most $2\epsilon n$. Note that we (the algorithm designers) have control over ϵ and we can choose it as small as we like. The price we pay is that as ϵ becomes smaller the running time of our algorithm comes closer and closer to 2^n. This prevents us from choosing $\epsilon = o(1)$. We design an algorithm with running time $\mathcal{O}((2-\delta)^n)$ for instances where $|S|$ and $|L|$ are both between $n(\frac{1}{2}-\tau)$ and $n(\frac{1}{2}+\tau)$ and $M = \emptyset$. This implies a $\mathcal{O}((2-\epsilon)^n)$ time algorithm for the case where $|M| \leq 2\epsilon n$, whenever ϵ is small enough compared to δ. We can just guess M, run the $n^{\mathcal{O}(k)}$ algorithm on $G[M]$ and run the $\mathcal{O}((2-\delta)^n)$ time algorithm on $G - M$. The total running time is at most $\mathcal{O}(\binom{n}{2\epsilon n}(2^{o(n)} + (2-\delta)^n))$, which beats 2^n whenever ϵ is small enough.

We now turn our attention to the case when $k > 0.24n$, $|S|$ and $|L|$ are both between $n(\frac{1}{2}-\epsilon)$ and $n(\frac{1}{2}+\epsilon)$, and $M = \emptyset$. Our initial intuition was that it should not be too hard to extend the biasing argument from the case that $|S \cup M| \geq n(\frac{1}{2}+\epsilon)$ to also handle these cases, however we were unable to do this. In fact, for a long time we were only able to handle a very special subcase: when all small parts of the optimal solution have size precisely 1, and there

is one large part that covers all of L. We call this special case the *at most one non-singleton case*. Observe that solving this case is equivalent to solving the Densest $(n-k+1)$-subgraph problem. Thus we can solve this case in $\mathcal{O}(2^{\frac{\omega n}{3}} n^{\mathcal{O}(1)})$ time using a known algorithm for Densest ℓ-subgraph by Chang et al. [12]. Their algorithm is an adaptation of the celebrated split and list algorithm of Williams [51] for the Max-Cut problem. At a glance it would appear that this very restricted special case is not useful at all for handling the general case. However, this special case turns out to be pivotal! Indeed, the (rather long and technical) remainder of our argument is to reduce most of the remaining instances of the problem to precisely this special case.

Hard Case. We now provide a high level overview of the tools and techniques used to handle the hard case where $k > 0.24n$, $|S|$ and $|L|$ are both between $n(\frac{1}{2}-\epsilon)$ and $n(\frac{1}{2}+\epsilon)$, and $M = \emptyset$.

First we divide further into two cases based on whether the number of cut edges between the large components in C is greater than $\frac{n}{\log^2 n}$ or not. The idea behind such a split is that there are only $\binom{n^2}{n/\log^2 n} \leq n^{2n/\log^2 n} = 2^{\log n \cdot \frac{2n}{\log^2 n}} = 2^{o(n)}$ subsets of edges of size at most $\frac{n}{\log^2 n}$. So if there are only a few($\leq \frac{n}{\log^2 n}$) cut edges between the large components, we guess and remove them with a $2^{o(n)}$ overhead. This reduces the few cut edges between large components case to one that has no such cut edges. But the no cut edges between large components case turns out to be harder than the many($> \frac{n}{\log^2 n}$) cut edges between large components case. We next show how to handle the latter case followed by the harder one.

1.) Many cut edges between large components. To handle this case, we first observe that all small components are singletons. Indeed, there are at most $\log n$ large parts but there are at least $\frac{n}{\log^2 n}$ cut edges between them. So there exists two large parts P_1 and P_2 having more than $\log n$ cut edges between them. If there exists a non-singleton small part, then combining P_1 and P_2 into a single part and removing a vertex from a non-singleton small part, which by definition has size at most $\log n$, as a singleton gives a strictly better cut.

We use this property to reduce to disjoint instances of the at most one non-singleton case. Here, (a) except for at most $\log n$ many vertices in L, all vertices in L have at least $\frac{n}{\log^4 n}$ neighbors within their large component in C – Suppose a set X of $\log n$ vertices in L have at most $\frac{n}{log^4 n}$ neighbors within their large component. Then, making $L - X$ one big component and all other vertices in X a singleton yields a better cut than C. Also, (b) the number of cut edges between large parts can be upper bounded by $n \log n$ – If not, we can remove $\log n$ vertices in L as singletons and merge all other vertices in L into one big component to obtain a strictly better cut. This is because there are at most $\log n$ large parts.

Next we sample a subgraph G' of G having $V(G') = V(G)$. We keep every edge in G in G' with probability $\frac{1}{\log^3 n}$. We use (a) and (b) to show that G' with high probability satisfies: (A) contains only few ($\leq \frac{n}{\log^2 n}$) cut edges between large components in C and (B) except for $(\log n)^{\mathcal{O}(1)}$ many vertices in L, all vertices in L still have at least $\frac{n}{\log^8 n}$ neighbors in G' within their large component in C.

In G', because of (A), with just $2^{o(n)}$ overhead we guess and remove all remaining cut edges between large components in C. Next, using (B) we show that for each large component X in C we have a set of $(\log n)^{\mathcal{O}(1)}$ vertices $V_X \subset X$ such that $X \subseteq N_{G'}(V_X)$. For every large component X, we guess V_X – There are at most $\log n$ large components and for each, this guessing incurs a $n^{(\log n)^{\mathcal{O}(1)}}$ overhead. Recall that all small components in C are singletons. Next because we removed all cut edges between large components in C in G', V_X has no neighbors in any large component apart from X in G'. Thus for every distinct pair X_1, X_2

of large components in C, every vertex in $N_{G'}(V_{X_1}) \cap N_{G'}(V_{X_2})$ is a singleton in C and can be removed. Finally, in the resulting graph each $V_X \cup N(V_X)$ is a disjoint instance of the at most one non-singleton case.

2.) No cut edges between large components. We handle this case by either reducing to the at most one non-singleton case or by identifying S and L. In the latter case, we use S and L to obtain a cut as good as C. We now show how to efficiently compute a cut as good as C given S and L. To find the part of the cut induced by L, we just use the n^k exact algorithm on $G[L]$ as there are at most $\log n$ large parts. So we shift our focus to finding the part of the cut induced by S. For this, we use the state graph with some guessing. Recall that the vertices of the state graph S_G correspond to subsets of vertices and the edges correspond to components. A path from $\emptyset$ to any subset S in S_G can be mapped to the cut containing all parts corresponding to the edges in P and vice versa. Thus using less than 2^n time, we precompute and store for each state X having $|X| \leq n(\frac{1}{2} - \delta)$ the best ℓ-cut of X containing only small components - $small(X, \ell)$. For this we only build S_G for vertex sets of size at most $n(\frac{1}{2} - \epsilon)$ and edges corresponding to small parts. For bounding our run times, we use that for $\tau \in [0, 1]$, $\binom{n}{\tau n} \leq 2^{h(\tau)n}$, where h is the binary entropy function [32]. So precomputing $small$ takes time at most $\mathcal{O}(2^{h(\frac{1}{2}-\delta)n+o(n)})$. Given S, we guess a union X of small parts in C such that $|S \setminus S'| \leq n(\frac{1}{2} - \delta)$. Then we do a lookup of $small(S', \ell) \cup small(S \setminus S', k - \ell)$ to obtain the part of the cut induced by S. Thus given S and L, we can compute C in time $\binom{\frac{n}{2}+\epsilon n}{(\epsilon+\delta)n} \leq 2^{h(\epsilon+\delta)n}$ since $|S - \frac{n}{2}| \leq \epsilon n$.

Next, to explain the subcases we divide into, we introduce some properties of the cut. Why we need these properties will become clear when we use them to solve the subcases. We say that C is α-*neighbor-biased* if there is a union X of large components whose neighborhood size substantially differs from its size. More precisely if $||X| - |N(X)|| > \alpha n$. If there is no such union of large components, we say C is *neighbor-balanced*. We call a large component θ-*heavy* if it contains all vertices of L except at most θn of them. Look at the order: $\epsilon, \delta, \alpha, \theta$, in which we defined our parameters. We set these parameters such that for each parameter, the ones before it in the order are small enough for our ideas to work. So we start by setting θ which is the highest, followed by α, then δ and finally ϵ. Another nice tool that we use to recover some sets and their neighborhoods is: all p-sized subsets of vertices having a q-sized neighborhood in a graph can be enumerated in time $\mathcal{O}(n\binom{p+q}{q})$ [22, Lemma 3.2].

With that, we divide into three subcases - (i) neighbor-biased (ii) neighbor-balanced with a heavy large component and (iii) neighbor-balanced with no heavy large component.

2.1) α-Neighbor-biased. The core idea in this case is to use the bias in size between X and $N(X)$ to efficiently guess L. We guess X in time $\binom{|X|+|N(X)|}{|X|}$ and recover $N(X)$. Next we guess $L \setminus X$ in time $2^{n-|X|-|N(X)|}$. This lets us recover L and S in time $\binom{|X|+|N(X)|}{|X|} \cdot 2^{n-|X|-|N(X)|}$. Given S and L we do some guessing and lookup of $small$ to compute C in time $2^{h(\epsilon+\delta)n}$ as discussed above. Our final runtime ends up being $\binom{|X|+|N(X)|}{|X|} \cdot 2^{n-|X|-|N(X)|} \cdot 2^{h(\epsilon+\delta)n}$. A simple calculation simplifying $\binom{|X|+|N(X)|}{|X|}$ in terms of h shows that the final runtime is substantially less than 2^n as long as α is sufficiently large as compared to ϵ and δ.

2.2) α-Neighbor-balanced with a θ-heavy large component. Set $\theta = 10^{-5}$. Here we reduce to the at most one non-singleton case. Recall that the case we reduce to can be solved in $\mathcal{O}(2^{\frac{\omega}{3}n})$ time. Thus we can afford an overhead of $2^{(0.999-\frac{\omega}{3})n}$ to get to that case.

First we guess the set $L \setminus H$ of vertices in L not in the heavy component H. This incurs a cost of $\binom{n}{\theta n} \leq 2^{h(\theta)n}$. Then we guess the set X of vertices in non-singleton small components. We show that $|X|$ is actually very small and that the $\binom{n}{|X|}$ overhead to guess X is acceptable.

Observe that we can compute the cut induced by $L \setminus H$ in $2^{o(n)}$ time using the n^k algorithm and the cut induced by X in $\mathcal{O}(1)$ time by a lookup of *small*. Thus after guessing $L \setminus H$ and X we reduce to the one non-singleton case. All that remains to do now is to prove $|X|$ is small enough to be able to guess X. We bound $|X|$ by $0.01n$. Then $\binom{n}{|X|} \leq \binom{n}{0.01n} \leq 2^{h(0.01)n}$, which is an easily affordable overhead.

Suppose $|X| > 0.01n$, we show that C can be improved. We call a non-singleton component *good* if the heavy component is adjacent to 99% of the vertices in it. Since C is neighbor-balanced and $0.01 >> (\theta = 10^{-5})$, we can show that the heavy component is adjacent to almost all vertices in many non-singleton components; that there are at least $n/(\log n)^{\mathcal{O}(1)}$ good components. Because there are $n/(\log n)^{\mathcal{O}(1)}$ good components and good components are all small, we can find an $i(\leq \log n)$ such that there are i good components of size i. Merging $i-1$ good components of size i with the heavy component and breaking one good component of size i into singletons improves C; the extra cost to break is only $0.05\binom{i}{2}$ while the savings from merging is $0.99(i-1)i$.

2.3) Neighbor-balanced with no heavy large component. As a first step, we make our state graph more robust to handle special cases. Recall that the edges of the state graph correspond to components. We build the state graph for vertex sets of size either at most $n(\frac{1}{2} - \epsilon)$ or at least $n(\frac{1}{2} + \epsilon)$ and edges corresponding to small parts. We then add a few additional edges that correspond to groups of components. For each pair X, $X \cup Y$ of vertex sets in the new state graph, we draw an edge from X to $X \cup Y$ if Y is a union of a set of at most $\log n$ connected components in $G[V \setminus X]$. We call the constructed state graph the *enhanced* state graph. Observe that the enhanced state graph has size $\mathcal{O}((2-\varepsilon)^n)$. Now we demonstrate how the new edges help.

Skipping Cut. Suppose C has a union of small components S' and union of large components L' such that (i) $|S'| \leq n(\frac{1}{2} - \theta)$ (ii) $|S' \cup L'| \geq n(\frac{1}{2} + \theta)$ and (iii) $N(L') \subseteq S'$. Also recall that C does not contain edges between large components. In this case, there is a path P in the new state graph from $\emptyset$ to V corresponding to C. We decompose P into 4 portions: (1) a path from $\emptyset$ to S' using the edges corresponding to parts in S', (2) a edge from S' to $S' \cup L'$ corresponding to parts in L', (3) a path from $S' \cup L'$ to $S \cup L'$ using the edges corresponding to parts in $S \setminus S'$ and (4) an edge from $S \cup L'$ to V corresponding to parts in $L \setminus L'$. We say C is a *skipping* cut if it satisfies (i), (ii) and (iii) as we can find C in the modified state graph by skipping between states of size at most $n(\frac{1}{2} - \theta)$ and states of size at least $n(\frac{1}{2} + \theta)$ but by avoiding states of size $\frac{n}{2}$. In conclusion, if C is neighbor balanced with no heavy large component but C is a skipping cut then we can find C using the enhanced state graph in time $\mathcal{O}(2-\epsilon)^n$.

We now turn our attention to the final case: neighbor-balanced with no heavy large component and not skipping. To solve this case we reduce to a special case in which S and L can be identified efficiently using $2^{\frac{n}{2}}$ time. Thus we can afford an overhead of $2^{0.499n}$ to get to that case; but we incur a much smaller overhead. First we highlight the properties of C required for the special case and show how to use them to obtain S and L. Then we show how to guess a few vertices in S to reduce to the case we want.

Let C satisfy the following properties: (i) All large components in C can be grouped into two parts L_1 and L_2 each having size nearly $\frac{n}{4}$ (ii) $N(L_1) \cap N(L_2) = \emptyset$ (iii) All small components in C are of size two and contain one vertex from $N(L_1)$ and the other from $N(L_2)$. To find L and S, we first guess L_1 in time $2^{|L_1|+|N(L_1)|}$ and recover $N(L_1)$. We then recover $N(L_2)$ as $N(N(L_1)) \setminus L_1$. Then the set of remaining vertices is L_2, yielding $L = L_1 \cup L_2$. This in total only takes about $2^{|L_1|+|N(L_1)|} \leq 2^{\frac{n}{2}}$ time since $|L_1| \sim \frac{n}{4}$.

Suppose property (i) is not satisfied, let L' be a group of large components having size at least θn but less than $\frac{n}{4}$; such a group exists because there is no heavy large component. Then C is a skipping cut with witness L' and S' for some $S' \supseteq N(L')$ which exists because C is neighbor balanced. All that remains to be done is to show that we can guess some vertices in S to satisfy properties (ii) and (iii). Let S' be the set of vertices in small components of size two that contain one vertex from $N(L_1) \setminus N(L_2)$ and the other from $N(L_2) \setminus N(L_1)$. Also let $\mathcal{F}'$ be the family containing all such components. We guess $S \setminus S'$. One can easily verify that removing $S \setminus S'$ guarantees properties (ii) and (iii). To conclude that such a guess is feasible, we need to bound the size of $S \setminus S'$. We prove $|S \setminus S'| \leq 30(\epsilon + \theta + \alpha)n$.

Let $\mathcal{F}$ to be the family of all small components X in C that satisfy: (a) X has no vertex from $S \setminus N(L)$, (b) X has a vertex from $N(L_1)$ and a vertex from $N(L_2)$, and (c) X has no vertex from $N(L_1) \cap N(L_2)$.
Observe that the subfamily of all components of size two in $\mathcal{F}$ is $\mathcal{F}'$. We use this to obtain a bound on $|S \setminus S'|$. By definition, every component in $\mathcal{F}$ has at least two vertices, one in $N(L_1) \setminus N(L_2)$ and one in $N(L_2) \setminus N(L_2)$. Thus $|S| - 2|\mathcal{F}|$ will account for all but two vertices in components of size three or more in $\mathcal{F}$ and for all vertices in S not in any component in $\mathcal{F}$. This in turn implies $3 \cdot (|S| - 2|\mathcal{F}|)$ is an upper bound for the number of vertices in S not contained in any component of size two in $\mathcal{F}$, i.e bound for $|S \setminus S'|$. We will show $|S| - 2|\mathcal{F}| \leq 10(\epsilon + \alpha + \theta)n$ from which we can infer $|S \setminus S'| \leq 3(|S| - 2|\mathcal{F}|) \leq 30(\epsilon + \alpha + \theta)n$.

Next, for each of the properties $(a) - (c)$, we bound the number of small components that do not satisfy that property. Since $|S|$ and $|L|$ are both between $n(\frac{1}{2} - \epsilon)$ and $n(\frac{1}{2} + \epsilon)$ and C is α-neighbor-balanced, there are at most $(2\epsilon + \alpha)n$ vertices in S that are not in $N(L)$. Thus $|S \setminus N(L)| \leq (2\epsilon + \alpha)n$. So at most $(2\epsilon + \alpha)n$ small components do not satisfy (a). At most $(\epsilon + \theta/2)n$ small components do not have a neighbor in L_1. If not, since $|S| \leq n(\frac{1}{2} + \epsilon)$, we can show that C is a skipping cut with witness $L' = L_1$ and some $S' \supset N(L_1)$. The same argument holds with respect to L_2. Thus, at most $(2\epsilon + \theta)n$ small components do not satisfy (b). Next, since C is α-neighbor-balanced, $|N(L)| = |N(L_1)| + |N(L_2)| - |N(L_1) \cap N(L_2)| \leq |L_1| + \alpha n + |L_2| + \alpha n - |N(L_1) \cap N(L_2)|$. So $|N(L_1) \cap N(L_2)| \leq |L| - |N(L)| + 2\alpha n \leq 3\alpha n$. This bounds the number of small components not satisfying (c) by $3\alpha n$.

For an easy read, until this point we mentioned $k > 0.24n$. But in our algorithm, we set $k > \frac{n}{4} - \epsilon n$ and need it for this case to work. Since $k > \frac{n}{4} - \epsilon n$, using all the three bounds obtained, $|\mathcal{F}| \geq 0.25n - \epsilon n - (2\epsilon + \alpha)n - (2\epsilon + \theta)n - 3\alpha n \geq 0.25n - (5\epsilon + 4\alpha + \theta)n$. Thus, $|S| - 2|\mathcal{F}| \leq (0.5 + \epsilon)n - 2(0.25n - (5\epsilon + 4\alpha + \theta)n) \leq (10\epsilon + 8\alpha + 2\theta)n \leq 10(\epsilon + \alpha + \theta)n$. Note that the main proof is structured differently for tighter bounds with these being the core ideas.

2 Preliminaries

Given a graph G, we use $V(G)$ and $E(G)$ to denote the vertex sets and edge sets, respectively. For any subset $X \subseteq V(G)$, let $G[X]$ denote the induced subgraph of G on X. A subset $X \subseteq V(G)$ is said to be connected if $G[X]$ is a connected graph. We denote the number of connected components in G by $cc(G)$. Given a path P in G, let $E(P)$ denote the edges in the path. Given a subset $E' \subseteq E(G)$, we denote the subgraph G' having $V(G') = V(G)$ and $E(G') = E(G) \setminus E'$ by $G \setminus E'$. For any subset $X \subseteq V(G)$, let $N_G(X)$ denote the set of vertices in G adjacent to some vertex in X and let $N_G[X] = N_G(X) \cup X$. We will omit the subscript when the graph is clear from context.

A k-*cut* of a graph G is a partition of the vertex set $V(G)$ into k non-empty parts. We will refer to the parts of a cut as *components* of the cut. The *weight* of a k-cut is the total number of edges with endpoints in different parts of the partition. We refer to the edges of G having end points in different components in C as *cut edges* of C. We use $\mathsf{best}(X, l)$ to denote a least weight l-cut of $G[X]$. For any two disjoint subsets A, B of $V(G)$, we denote the number of edges having one end point in A and the other in B by $w(A, B)$. For every $E' \subseteq E(G)$, we let $w(E') = |E'|$. For every $X \subseteq V(G)$, we let $\delta(X)$ denote the set of edges having exactly one end point in X in G.

If G is an edge-weighted graph, then the *weight* of a k-cut is the sum of weights of edges with endpoints in different parts of the partition. For any two disjoint subsets A, B of $V(G)$, we denote the sum of weights of edges having one end point in A and the other in B by $w(A, B)$. For every $E' \subseteq E(G)$, we denote the sum of weights of all edges in E' by $w(E')$. We consider G to be a simple unweighted graph throughout the paper except in Section where we provide the $2^{n+o(n)}(\log W)^{\mathcal{O}(1)}$ time algorithm for edge-weighted graphs.

We also need the following lemma to enumerate connected sets of size $b+1$ with at most f neighbors in our algorithms.

▶ **Lemma 3** ([22, Lemma 3.2]). *All vertex sets of size $b+1$ with f neighbors in a graph G can be enumerated in time $\mathcal{O}(n\binom{b+f}{b})$ by making use of polynomial space.*

There exists an algorithm for Min k-Cut using Thorup trees that runs in time $\mathcal{O}(\binom{n}{2k}3^{2k})$. We express this runtime in this way because it enables us to obtain a $2^{o(n)}$ algorithm for k-cut whenever $k \leq \frac{n}{\log n}$. Given a polynomial in n, m, and $\log n$ sized family of spanning trees that contains a Thorup tree, the algorithm for each tree guesses all possible sets of edges of size at most $2k-2$, removes them and contracts the remaining forests into vertices and solves k-cut on this contracted graph in time 3^{2k}.

▶ **Lemma 4** (Exact Algorithm [50]). *Given a graph G, positive integer k, the min k-cut of G can be found in time $\mathcal{O}(\binom{n}{2k}3^{2k})$.*

The cut induced by a cut C of X on a subset Y of X is the cut obtained by taking the intersection of each component in C with Y. We define the union of two disjoint cuts C_1 of vertex set X_1 and C_2 of vertex set X_2 as the cut C of $X_1 \cup X_2$ that induces cut C_1 on X_1 and cut C_2 on X_2. We define the union of a subfamily $\{Y_0, \cdots, Y_z\}$ of components of a cut to be $\cup_{i \leq z} Y_i$, the set of all vertices in any component in the subfamily. Given a family $\mathcal{F}$ of subsets of $V(G)$, let $U(\mathcal{F}) = \cup_{X \in \mathcal{F}} X$ denote the set of all vertices in some set in $\mathcal{F}$. We will refer to a cut having a single component as just a set of vertices for convenience.

We also need the following classical single source shortest path algorithm to traverse the state graph, S_G (formally defined in later)[2].

▶ **Lemma 5** (Dijkstra's Algorithm). *Given an edge weighted directed graph D on n vertices and m edges, with two special vertices s and t, we can find a minimum weight shortest path from s to t in time $\mathcal{O}((n+m)\log n)$.*

We also need the following upper bounds on binomial coefficients for our purposes.

▶ **Lemma 6** ([32]). *Let n be a positive integer and $\alpha \in [0, 1]$, then $\binom{n}{\alpha n} \leq 2^{h(\alpha)n}$, where $h(\alpha)$ represents the entropy of a Bernoulli random variable with probability of success α, satisfying $h(\alpha) = -\alpha \log \alpha - (1-\alpha)\log(1-\alpha)$. Further, for a positive integer $k \leq n$, $\binom{n}{k} \leq \left(\frac{en}{k}\right)^k$, where e is the base of natural logarithms.*

[2] S_G will be a DAG, so one can also resort to other means to find single source shortest paths.

By truncating the Taylor series for h, given by

$$h(p) = 1 - \frac{1}{2\ln 2}\sum_{n=1}^{\infty}\frac{(1-2p)^{2n}}{n(2n-1)},$$

we get the following useful upper bound for h, when η is close to $\frac{1}{2}$.

▶ **Lemma 7.** *For $\eta \leq 1/10$, we have that $h(\frac{1}{2}-\eta) \leq 1 - \frac{\eta^2}{\ln 2}$.*

We also need the following definition of "contraction and uncontraction of edges".

▶ **Definition 8** ((Un)Contraction of Cuts and Graphs). *Given a graph G and an edge $e = uv$, the graph G/uv obtained by* contracting *the edge uv, is the one where we delete the vertices $\{u, v\}$, and introduce a new vertex v_e and add edges from v_e to every vertex in $N(u)$ and $N(v)$. If $w \in N(u) \cap N(v)$, then the process will make multi-edges.*

Given a cut C and an edge uv such that, u, v belongs to the same part in C, then by contraction of cut*, denoted by C/uv, we mean the cut constructed from C, where we replace the part P in C, that contains u and v with $(P \setminus \{u, v\}) \cup \{v_e\}$. Similarly, given a vertex set X, we define* uncontraction of a cut *C, denoted by $C/^{-1}X$, as a cut where we replace each vertex $x \in X$, with the set of vertices that has been contracted into x.*

3 Proof of Main Theorem and Case Breakdown

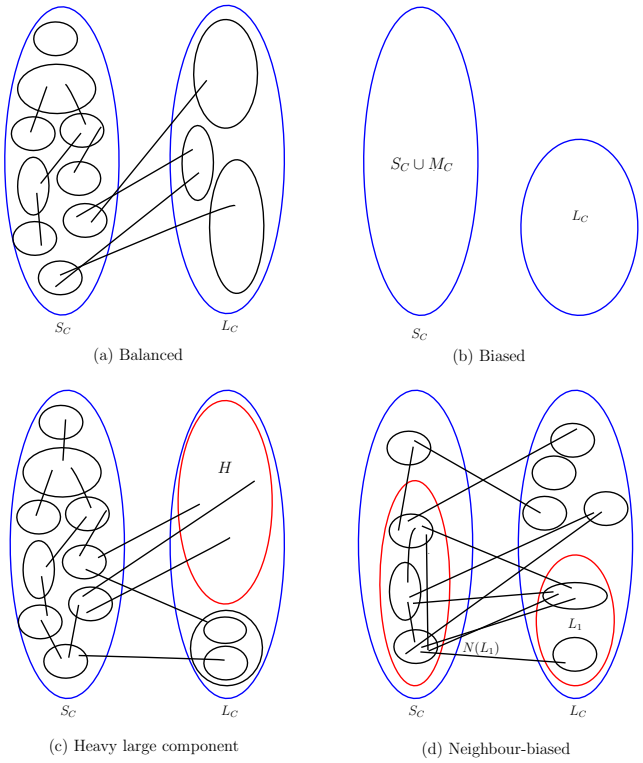
In this section we provide an overview of our algorithm with a case breakdown showing how we organize our cases throughout the paper. We first introduce some basic terminology used in the algorithm. Then we provide a guide to the different cases with pointers to various subroutines called in the algorithm. Finally we define a data structure that is used by most of our subroutines and state a known result for the at most one non-singleton case.

3.1 Properties of Cuts and Special Cases

In this section we define multiple properties of cuts that will be helpful for designing our algorithm. Let G be a graph on n vertices. We will design algorithms to find a cut with different combinations of these properties in G and then combine them to obtain our final algorithm. We divide the components of any cut of G into three types depending on their size.

▶ **Definition 9** (Component Types). *A component is said to be* small *if its size is at most $\log n$. A component is said to be* medium *if its size is greater than $\log n$ and less than $\frac{n}{\log n}$. A component is said to be* large *if its size is at least $\frac{n}{\log n}$. Given a cut C of G, we denote the set of all vertices in small, medium and large components in C by S_C, M_C and L_C respectively.*

This notion is useful because the number of subsets of $V(G)$ that can be small or medium components is at most $n\binom{n}{\frac{n}{\log n}} \leq n(\frac{en}{n/\log n})^{n/\log n} = 2^{o(n)}$. Also, for any subset L of $V(G)$ and $\ell \leq \log n$, we can find a ℓ-cut of $G[L]$ having weight no more than a minimum weight an ℓ-cut of $G[L]$ containing only large components in time $2^{o(n)}$ using Lemma 4. This is because there can be at most $\log n$ large components in any cut. We will use both these facts crucially while designing our algorithms. The above discussion leads us to define the following definitions.



(a) Balanced (b) Biased

(c) Heavy large component (d) Neighbour-biased

Figure 2 Visualizing cut properties (a) Balanced cut has $|S_C|$ nearly equal to $|L_C|$ and no medium components. (b) Biased cut has one of $|S_C \cup M_C|$ or $|L_C \cup M_C|$ substantially higher. (c) Heavy large component H contains nearly all vertices in L_C. (d) Neighbor-biased cut has a union of heavy components L_1 such that $||L_1| - |N(L_1)||$ is high.

▶ **Definition 10.** *For a subset $X \subseteq V(G)$ and $\ell \leq n$, $\mathsf{small}(X, \ell)$ represents an ℓ-cut of $G[X]$ having weight no more than the weight of a least weight ℓ-cut of $G[X]$ containing only small and medium components.*

▶ **Definition 11** (Biased and Balanced Cuts)**.** *For any $0 < \delta < \frac{1}{2}$, a cut C is said to be* δ-biased *if either $|M_C \cup S_C| \geq n(\frac{1}{2} + \delta)$ or $|M_C \cup L_C| \geq n(\frac{1}{2} + \delta)$ and it is said to be* δ-balanced *if it contains no medium components, $|S_C| \leq n(\frac{1}{2} + \delta)$ and $|L_C| \leq n(\frac{1}{2} + \delta)$.*

▶ **Definition 12** (Properties of Large Components)**.** *We say that a cut C of G has* many edges between large components *if the number of cut edges of C having both end points in L_C is greater than $\frac{n}{\log^2 n}$. We say that the cut has* few edges between large components *if the number of such edges is at most $\frac{n}{\log^2 n}$.*

Observe that we can reduce the case of few edges between large components to no edges by guessing the set of edges between the large components with just $2^{o(n)}$ guesses. This again will prove helpful for us to get more structure.

▶ **Definition 13.** *For any $0 \leq \alpha < 1$, we say that a large component X in a cut C of G is* α-heavy *if it has all vertices in L_C except for at most αn vertices, i.e $|L_C \setminus X| \leq \alpha n$.*

▶ **Definition 14.** *For any $0 \leq \alpha < \frac{1}{2}$ and $0 < \rho < \frac{1}{2}$, we say that a α-balanced cut C is ρ-neighbor-biased if there is a union X of large components in C having p vertices such that $N(X) \subseteq S_C$, $|N(X)| = q$ and $|p - q| > \rho n$. Otherwise the cut is said to be ρ-neighbor-balanced.*

We note that we will use the last two properties only on balanced cuts having no edges between large components. Figure 2 captures all the properties defined above.

3.2 Steps of Our Algorithm

Armed with all the technical definitions, we now provide a guide to the different cases our algorithm for Min k-Cut handles. First, we summarize the different parameters that we use in our algorithm and the final value we set them to obtain our main result.

- $\alpha_1 = 10^{-20}$: $k < \frac{n}{4} - \alpha_1 n$ and $k \geq \frac{n}{4} - \alpha_1 n$
- $\alpha_2 = 10^{-20}$: α_2-balanced or $\frac{\alpha_2}{2}$-biased cut
- $\alpha_3 = 10^{-5}$: α_3-heavy large component
- $\alpha_4 = 10^{-2}$: number of non singletons
- $\alpha_5 = 10^{-5}$: α_5-neighbor biased or balanced
- $\delta = 10^{-20}$: δ-small cut data structure

Note that our results will work for a range of these parameters but for convenience we fix these parameters to a particular value and obtain our final result. Given an input graph G on n vertices and integer k, let C be a *hypothetical* min k-cut of G. Our goal is to compute C. We divide into cases based on the value of k and the properties of C. We use Lemmas 6 and 7 to upper bound the running time for each case by $(2-\varepsilon)^n$ for some $\varepsilon > 10^{-50}$. In the summary, for each case we mention which one of the two Lemmas is used to bound the running time in parenthesis.

(Case 1: $k < \frac{n}{4} - \alpha_1 n$.) In this case we apply Lemma 15 and directly solve the problem in time $\frac{2^{n+o(n)}}{(1+4\alpha_1)^{\alpha_1 n}}$ (Lemma 6).

▶ **Lemma 15 (*).** *There exists an algorithm that takes as input a graph G on n vertices, an integer $k < \frac{n}{4} - \alpha_1 n$ and returns the min k-cut of G in time $\frac{2^{n+o(n)}}{(1+4\alpha_1)^{\alpha_1 n}}$.*

(Case 2: $k \geq \frac{n}{4} - \alpha_1 n$) Since k is large, we have the property that the number of components of size at most 4 is at least $0.04n$. We then break this case based on how efficiently we can use this property. We use it in the case when C is $\frac{\alpha_2}{2}$-biased and handle the balanced case separately, leading to the subsequent subcases. This case is handled in Lemma 16 in time $2^{n\frac{1}{2}(1+h(\frac{1}{2}+\frac{\alpha_2}{20}))+o(n)} + 2^{(h(\alpha_2)+h(\frac{1}{2}-\alpha_2))n+o(n)}$ (Lemma 6).

▶ **Lemma 16 (*).** *There exists an algorithm that takes as input a graph G on n vertices, an integer $k \geq \frac{n}{4} - \alpha_1 n$ and returns the min k-cut of G in time $2^{n\frac{1}{2}(1+h(\frac{1}{2}+\frac{\alpha_2}{20}))+o(n)} + 2^{h(\alpha_2)+h(\frac{1}{2}-\alpha_2)n+o(n)}$, when $\alpha_1 = \alpha_2 = 10^{-20}$.*

Case 2a: ($\frac{\alpha_2}{2}$-biased case) We completely handle this case in Lemma*[3] 5.3 in time $2^{\frac{n}{2}(1+h(\frac{1}{2}+\frac{\alpha_2}{20}))+o(n)}$ (Lemma 7).

Case 2b: (α_2-balanced case) We further divide this case based on whether the number of edges between large components in C is large ($> \frac{n}{\log^2 n}$) or not. We handle this in Lemma* 5.4 in time $2^{h(\frac{1}{2}-\alpha_2)n+o(n)}$ (Lemma 7).

Case 2b (i): (Many edges between large components) We solve this case by reducing to the case of finding a cut having at most one non-singleton component. We handle this case in Lemma* 7.1 using Lemma* 3.2 as a subroutine in time $2^{\frac{\omega n}{3}+o(n)}$ (Lemma 6).

[3] We use Lemma* to refer to the Lemmas stated and proved in the full version; Lemma* 5.3 is Lemma 5.3 in the full version.

Case 2b (ii): (Few edges between large components) This is our final case and we deal with this by designing algorithms for the following cases. This case can be found in Lemma* 7.2 and the algorithm runs in time $2^{h(\frac{1}{2}-\alpha_2)n+o(n)}$ (Lemma 7).

- **(a) (α_5-neighbor biased)** This case is handled in Lemma* 7.3. Here we use the fact that C is neighbor-biased to find a family of partitions of $V(G)$ containing the partition $S_C \dot\cup L_C$ of $V(G)$ and use Lemma* 3.1 to find a cut having weight no more than a α_2-balanced cut C' with $S_C = S'_C$ and $L_C = L'_C$. We handle this case in time $2^{h(\frac{1}{2}-\alpha_2)n+o(n)}$ (Lemma 7).
- **(b) (α_5-neighbor balanced and α_3-heavy component)** In this case we argue that there are not too many non-singleton components and solve the case by reducing to the case of finding a cut having at most one non-singleton component. We handle this case in Lemma* 7.4 using Lemma* 3.2 as a subroutine in time $2^{(h(\alpha_3)+h(\alpha_4)+\frac{\omega}{3})n+o(n)}$ (Lemma 6).
- **(c) (α_5-neighbor balanced and no α_3-heavy component)** We handle this case completely in Lemma* 7.5. Here we first run Lemma* 4.2 that involves using the state graph. Then similar to Case A, here we find a family of partitions of $V(G)$ containing the partition $S_C \dot\cup L_C$ of $V(G)$ and use Lemma* 3.1 on it. This case runs in time $2^{h(\frac{1}{2}-\alpha_2)n+o(n)}$ (Lemma 7).

A case tree depicting the various cases can be found in Figure 1. We now provide the proof of our main Theorem assuming Lemma 15($k < \frac{n}{4} - \alpha_1 n$) and Lemma 16($k \geq \frac{n}{4} - \alpha_1 n$).

Proof of Theorem 1. If $k < \frac{n}{4} - \alpha_1 n$ we run Lemma 15 with G and k else we run Lemma 16 with G and k and return the obtained k-cut. The correctness follows from the corresponding Lemmas. To bound the running time of the algorithm, we show that the running time of each subroutine mentioned above in the summary is bounded by $\mathcal{O}(2-\varepsilon)^n$, for some $\varepsilon > 10^{-50}$. To do this, we set the values of the parameters as summarized in the beginning of this subsection and use Lemma 7 and 6 as mentioned in the summary. ◀

4 Conclusion

In this paper we designed the first algorithm for Min k-Cut running in time better than 2^n. In particular, we designed an algorithm with running time $\mathcal{O}((2-\varepsilon)^n)$, for $\varepsilon > 10^{-50}$. We hope that our methods will be useful to design $\mathcal{O}(c^n)$ time algorithms for $c < 2$ for other graph partitioning problems for which progress has stopped at 2^n, including Cutwidth, Edge Multiway Cut, and more generally Edge Multicut on undirected graphs. Finally, it remains an interesting open problem to obtain a $\mathcal{O}(c^n)$ time algorithm for Min k-Cut for a constant $c < 2$ which is bounded away from 2 by more than a rounding error.

References

1 Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 39–51. Springer, 2014. doi:10.1007/978-3-662-43948-7_4.

2 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 522–539. SIAM, 2021.

3 Arturs Backurs and Christos Tzamos. Improving viterbi is hard: Better runtimes imply faster clique algorithms. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 311–321. PMLR, 2017. URL: http://proceedings.mlr.press/v70/backurs17a.html.

4 Andreas Björklund. Determinant sums for undirected hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014. doi:10.1137/110839229.

5 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets möbius: fast subset convolution. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 67–74. ACM, 2007. doi:10.1145/1250790.1250801.

6 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Trimmed moebius inversion and graphs of bounded degree. *Theory Comput. Syst.*, 47(3):637–654, 2010. doi:10.1007/s00224-009-9185-7.

7 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. The traveling salesman problem in bounded degree graphs. *ACM Trans. Algorithms*, 8(2):18:1–18:13, 2012. doi:10.1145/2151171.2151181.

8 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Counting connected subgraphs with maximum-degree-aware sieving. In Wen-Lian Hsu, Der-Tsai Lee, and Chung-Shou Liao, editors, *29th International Symposium on Algorithms and Computation, ISAAC 2018, December 16-19, 2018, Jiaoxi, Yilan, Taiwan*, volume 123 of *LIPIcs*, pages 17:1–17:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ISAAC.2018.17.

9 Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set partitioning via inclusion-exclusion. *SIAM J. Comput.*, 39(2):546–563, 2009. doi:10.1137/070683933.

10 Andreas Björklund, Petteri Kaski, and Ryan Williams. Solving systems of polynomial equations over GF(2) by a parity-counting self-reduction. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 26:1–26:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.26.

11 Ivan Bliznets, Fedor V. Fomin, Michal Pilipczuk, and Yngve Villanger. Largest chordal and interval subgraphs faster than 2^n. *Algorithmica*, 76(2):569–594, 2016. doi:10.1007/s00453-015-0054-2.

12 Maw-Shang Chang, Li-Hsuan Chen, Ling-Ju Hung, Peter Rossmanith, and Guan-Han Wu. Exact algorithms for problems related to the densest k-set problem. *Inf. Process. Lett.*, 114(9):510–513, 2014. doi:10.1016/j.ipl.2014.04.009.

13 Rajesh Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, Marcin Pilipczuk, and Michał Pilipczuk. Designing FPT algorithms for cut problems using randomized contractions. *SIAM J. Comput.*, 45(4):1171–1229, 2016. doi:10.1137/15M1032077.

14 Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as CNF-SAT. *ACM Trans. Algorithms*, 12(3):41:1–41:24, 2016. doi:10.1145/2925416.

15 Marek Cygan, Pawel Komosa, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, Saket Saurabh, and Magnus Wahlström. Randomized contractions meet lean decompositions. *ACM Trans. Algorithms*, 17(1):6:1–6:30, 2021. doi:10.1145/3426738.

16 Marek Cygan and Marcin Pilipczuk. Faster exponential-time algorithms in graphs of bounded average degree. *Inf. Comput.*, 243:75–85, 2015. doi:10.1016/j.ic.2014.12.007.

17 Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. Scheduling partially ordered jobs faster than 2^n. *Algorithmica*, 68(3):692–714, 2014. doi:10.1007/s00453-012-9694-7.

18 Fedor V. Fomin, Serge Gaspers, Daniel Lokshtanov, and Saket Saurabh. Exact algorithms via monotone local search. *J. ACM*, 66(2):8:1–8:23, 2019. doi:10.1145/3284176.

19 Fedor V. Fomin, Serge Gaspers, and Saket Saurabh. Improved exact algorithms for counting 3- and 4-colorings. In Guohui Lin, editor, *Computing and Combinatorics, 13th Annual International Conference, COCOON 2007, Banff, Canada, July 16-19, 2007, Proceedings*, volume 4598 of *Lecture Notes in Computer Science*, pages 65–74. Springer, 2007. doi:10.1007/978-3-540-73545-8_9.

20 Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2010. doi:10.1007/978-3-642-16533-7.

21 Fedor V. Fomin, Dieter Kratsch, Ioan Todinca, and Yngve Villanger. Exact algorithms for treewidth and minimum fill-in. *SIAM J. Comput.*, 38(3):1058–1079, 2008. doi:10.1137/050643350.

22 Fedor V. Fomin and Yngve Villanger. Treewidth computation and extremal combinatorics. *Comb.*, 32(3):289–308, 2012. doi:10.1007/s00493-012-2536-z.

23 Olivier Goldschmidt and Dorit S. Hochbaum. A polynomial algorithm for the k-cut problem for fixed k. *Math. Oper. Res.*, 19(1):24–37, 1994. doi:10.1287/moor.19.1.24.

24 Alexander Golovnev, Alexander S. Kulikov, and Ivan Mihajlin. Families with infants: Speeding up algorithms for np-hard problems using FFT. *ACM Trans. Algorithms*, 12(3):35:1–35:17, 2016. doi:10.1145/2847419.

25 Anupam Gupta, Euiwoong Lee, and Jason Li. Faster exact and approximate algorithms for k-cut. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 113–123. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00020.

26 Anupam Gupta, Euiwoong Lee, and Jason Li. An FPT algorithm beating 2-approximation for k-cut. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2821–2837. SIAM, 2018. doi:10.1137/1.9781611975031.179.

27 Anupam Gupta, Euiwoong Lee, and Jason Li. The number of minimum k-cuts: improving the karger-stein bound. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 229–240. ACM, 2019. doi:10.1145/3313276.3316395.

28 Anupam Gupta, Euiwoong Lee, and Jason Li. The karger-stein algorithm is optimal for k-cut. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 473–484. ACM, 2020. doi:10.1145/3357713.3384285.

29 Anupam Gupta, Euiwoong Lee, and Jason Li. The number of minimum k-cuts: Improving the karger-stein bound. *To appear in STOC 2020*, abs/1906.00417, 2020. arXiv:1906.00417.

30 Zhiyang He and Jason Li. Breaking the n^k barrier for minimum k-cut on simple graphs. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 131–136. ACM, 2022.

31 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

32 Stasys Jukna. *Extremal combinatorics: with applications in computer science*. Springer Science & Business Media, 2011.

33 David R. Karger and Clifford Stein. A new approach to the minimum cut problem. *J. ACM*, 43(4):601–640, 1996. doi:10.1145/234533.234534.

34 Ken-ichi Kawarabayashi and Bingkai Lin. A nearly 5/3-approximation FPT algorithm for min-k-cut. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 990–999. SIAM, 2020. doi:10.1137/1.9781611975994.59.

35 Ken-ichi Kawarabayashi and Mikkel Thorup. The minimum k-way cut of bounded size is fixed-parameter tractable. In *52nd Annual IEEE Symposium on Foundations of Computer Science, FOCS, 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 160–169. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.53.

36 Mikko Koivisto. Partitioning into sets of bounded cardinality. In Jianer Chen and Fedor V. Fomin, editors, *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, volume 5917 of *Lecture Notes in Computer Science*, pages 258–263. Springer, 2009. doi:10.1007/978-3-642-11269-0_21.

37 Jason Li. Faster minimum k-cut of a simple graph. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1056–1077. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00068.

38 Daniel Lokshtanov, Ivan Mikhailin, Ramamohan Paturi, and Pavel Pudlák. Beating brute force for (quantified) satisfiability of circuits of bounded treewidth. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 247–261. SIAM, 2018. doi:10.1137/1.9781611975031.18.

39 Daniel Lokshtanov, Ramamohan Paturi, Suguru Tamaki, R. Ryan Williams, and Huacheng Yu. Beating brute force for systems of polynomial equations over finite fields. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2190–2202. SIAM, 2017. doi:10.1137/1.9781611974782.143.

40 Daniel Lokshtanov, Saket Saurabh, and Vaishali Surianarayanan. A parameterized approximation scheme for min k-cut. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 798–809. IEEE, 2020. doi:10.1109/FOCS46700.2020.00079.

41 Pasin Manurangsi. Inapproximability of maximum biclique problems, minimum k-cut and densest at-least-k-subgraph from the small set expansion hypothesis. *Algorithms*, 11(1):10, 2018. doi:10.3390/a11010010.

42 Joseph Naor and Yuval Rabani. Tree packing and approximating k-cuts. In S. Rao Kosaraju, editor, *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA*, pages 26–27. ACM/SIAM, 2001. URL: http://dl.acm.org/citation.cfm?id=365411.365415.

43 Jesper Nederlof. Finding large set covers faster via the representation method. In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, volume 57 of *LIPIcs*, pages 69:1–69:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ESA.2016.69.

44 Jesper Nederlof. Bipartite TSP in $O(1.9999^n)$ time, assuming quadratic time matrix multiplication. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 40–53. ACM, 2020. doi:10.1145/3357713.3384264.

45 Jesper Nederlof, Jakub Pawlewicz, Céline M. F. Swennenhuis, and Karol Wegrzycki. A faster exponential time algorithm for bin packing with a constant number of bins via additive combinatorics. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1682–1701. SIAM, 2021. doi:10.1137/1.9781611976465.102.

46 Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-sat. *J. ACM*, 52(3):337–364, 2005. doi:10.1145/1066100.1066101.

47 R Ravi and Amitabh Sinha. Approximating k-cuts using network strength as a lagrangean relaxation. *European Journal of Operational Research*, 186(1):77–90, 2008.

48 Huzur Saran and Vijay V. Vazirani. Finding k cuts within twice the optimal. *SIAM J. Comput.*, 24(1):101–108, 1995. doi:10.1137/S0097539792251730.

49 Uwe Schöning. A probabilistic algorithm for k-sat and constraint satisfaction problems. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 410–414. IEEE Computer Society, 1999. doi:10.1109/SFFCS.1999.814612.

50 Mikkel Thorup. Minimum k-way cuts via deterministic greedy tree packing. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 159–166. ACM, 2008. doi:10.1145/1374376.1374402.

51 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. doi:10.1016/j.tcs.2005.09.023.

52 Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In Thore Husfeldt and Iyad A. Kanj, editors, *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, volume 43 of *LIPIcs*, pages 17–29. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.IPEC.2015.17.

53 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018*, pages 3447–3487. World Scientific, 2018.

54 Virginia Vassilevska Williams. Some open problems in fine-grained complexity. *SIGACT News*, 49(4):29–35, 2018. doi:10.1145/3300150.3300158.

55 Gerhard J. Woeginger. Exact algorithms for np-hard problems: A survey. In Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi, editors, *Combinatorial Optimization – Eureka, You Shrink!, Papers Dedicated to Jack Edmonds, 5th International Workshop, Aussois, France, March 5-9, 2001, Revised Papers*, volume 2570 of *Lecture Notes in Computer Science*, pages 185–208. Springer, 2001. doi:10.1007/3-540-36478-1_17.

56 Gerhard J. Woeginger. Open problems around exact algorithms. *Discret. Appl. Math.*, 156(3):397–405, 2008. doi:10.1016/j.dam.2007.03.023.

57 Or Zamir. Breaking the 2^n barrier for 5-coloring and 6-coloring. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 113:1–113:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.113.

A Tight $(1.5 + \epsilon)$-Approximation for Unsplittable Capacitated Vehicle Routing on Trees

Claire Mathieu
CNRS Paris, France

Hang Zhou
École Polytechnique, Institut Polytechnique de Paris, France

Abstract

In the unsplittable capacitated vehicle routing problem (UCVRP) on trees, we are given a rooted tree with edge weights and a subset of vertices of the tree called terminals. Each terminal is associated with a positive demand between 0 and 1. The goal is to find a minimum length collection of tours starting and ending at the root of the tree such that the demand of each terminal is covered by a single tour (i.e., the demand cannot be split), and the total demand of the terminals in each tour does not exceed the capacity of 1.

For the special case when all terminals have equal demands, a long line of research culminated in a quasi-polynomial time approximation scheme [Jayaprakash and Salavatipour, TALG 2023] and a polynomial time approximation scheme [Mathieu and Zhou, TALG 2023].

In this work, we study the general case when the terminals have arbitrary demands. Our main contribution is a polynomial time $(1.5 + \epsilon)$-approximation algorithm for the UCVRP on trees. This is the first improvement upon the 2-approximation algorithm more than 30 years ago. Our approximation ratio is essentially best possible, since it is NP-hard to approximate the UCVRP on trees to better than a 1.5 factor.

2012 ACM Subject Classification Theory of computation → Routing and network design problems

Keywords and phrases approximation algorithms, capacitated vehicle routing, graph algorithms, combinatorial optimization

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.91

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2202.05691`

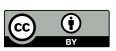
Funding This work was partially funded by the grant ANR-19-CE48-0016 from the French National Research Agency (ANR).

1 Introduction

In the *unsplittable capacitated vehicle routing problem (UCVRP)* on *trees*, we are given a rooted tree with edge weights and a subset of vertices of the tree called *terminals*. Each terminal is associated with a positive *demand* between 0 and 1. The root of the tree is called the *depot*. The goal is to find a minimum length collection of tours starting and ending at the depot such that the demand of each terminal is covered by a *single* tour (i.e., the demand cannot be split), and the total demand of the terminals in each tour does not exceed the *capacity* of 1.

The UCVRP on trees has been well studied in the special setting when all terminals have *equal* demands:[1] Hamaguchi and Katoh [17] gave a polynomial time 1.5-approximation; the approximation ratio was improved to 1.35078 by Asano, Katoh, and Kawashima [3] and

[1] Up to scaling, the equal demand setting is equivalent to the *unit demand* version of the capacitated vehicle routing problem in which each terminal has unit demand, and the capacity of each tour is a positive integer k.



was further reduced to 4/3 by Becker [4]; Becker and Paul [5] gave a bicriteria polynomial time approximation scheme; and very recently, Jayaprakash and Salavatipour [18] gave a quasi-polynomial time approximation scheme, based on which Mathieu and Zhou [21] designed a polynomial time approximation scheme.

In this work, we study the UCVRP on trees in the general setting when the terminals have *arbitrary* demands. Our main contribution is a polynomial time $(1.5+\epsilon)$-approximation algorithm (Theorem 1). This is the first improvement upon the 2-approximation algorithm of Labbé, Laporte, and Mercure [20] more than 30 years ago. Our approximation ratio is essentially best possible, since it is NP-hard to approximate the UCVRP on trees to better than a 1.5 factor [14].

▶ **Theorem 1.** *For any $\epsilon > 0$, there is a polynomial time $(1.5+\epsilon)$-approximation algorithm for the unsplittable capacitated vehicle routing problem on trees.*

The UCVRP on trees generalizes the UCVRP on *paths*. The latter problem has been studied extensively due to its applications in scheduling, see Section 1.1. As an immediate corollary of Theorem 1, we obtain the first polynomial time $(1.5+\epsilon)$-approximation algorithm for the UCVRP on paths. This ratio is essentially best possible, since it is NP-hard to approximate the UCVRP on paths to better than a 1.5 factor.

1.1 Related Work

Originally introduced by Dantzig and Ramser in 1959 [10], the UCVRP generalizes the *traveling salesman problem*, and is one of the most basic problems in Operations Research.

UCVRP on general metrics

The classical tour partitioning algorithm [16] introduced by Haimovich and Rinnooy Kan in 1985 was proved to be a constant-factor approximation on general metrics [2]. Very recently, Blauth, Traub, and Vygen [6] achieved the first improvement upon the tour partitioning algorithm. The best-to-date approximation ratio for general metrics stands at roughly 3.194 due to Friggstad, Mousavi, Rahgoshay, and Salavatipour [13].

UCVRP on paths

The UCVRP on paths is equivalent to the scheduling problem of minimizing the makespan on a single batch processing machine with non-identical job sizes [26]. Many heuristics have been proposed and evaluated empirically, e.g., [26, 12, 22, 9, 19, 24, 7, 1, 23]. Prior to our work, the best approximation ratio for the UCVRP on paths was 1.6 due to Wu and Lu [27].

The UCVRP on paths has also been studied in special cases. For example, when the optimal value is at least $\Omega(1/\epsilon^6)$ times the maximum distance between any terminal and the depot, asymptotic polynomial time approximation schemes are known [11, 25, 8].[2] In contrast, the algorithm in Theorem 1 applies to *any* path instance.

UCVRP in the Euclidean plane

In the two-dimensional Euclidean plane, the UCVRP admits a $(2+\epsilon)$-approximation [15].

[2] The UCVRP on paths is called the *train delivery problem* in [11, 25, 8].

2 Overview of Techniques

To prove Theorem 1, at a high level, our approach is to modify the problem and add enough structural constraints so that the structured problem contains a $(1.5 + O(\epsilon))$-approximate solution and can be solved in polynomial time by dynamic programming.

2.1 Preprocessing

We start by some preprocessing as in [21]. We assume without loss of generality that every vertex in the tree has two children, and the terminals are the leaf vertices of the tree [21]. Furthermore, we assume that the tree has bounded distances (Section 3.2). Next, we decompose the tree into *components* (Figure 1 and Section 3.3).

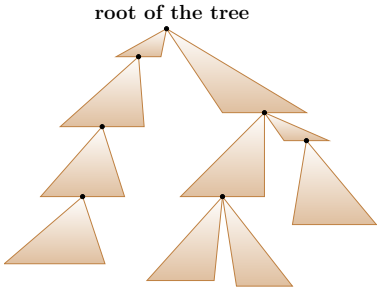


Figure 1 Decomposition of the tree into *components*. Figure extracted from [21]. Each brown triangle represents a *component*. Each component has a *root* vertex and at most one *exit* vertex.

2.2 Solutions Within Each Component

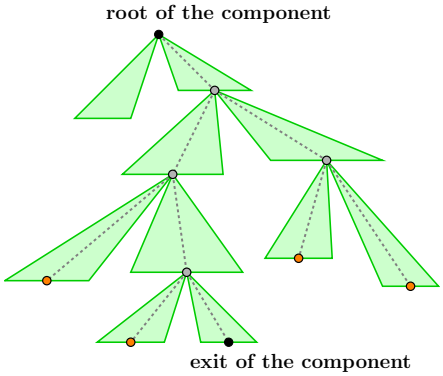
A significant difficulty is to compute solutions within each component. It would be natural to attempt to extend the approach in the setting when all terminals have equal demands [21]. In that setting, the *demands of the subtours*[3] in each component are among a polynomial number of values; since the component is visited by a constant number of tours in a near-optimal solution, that solution inside the component can be computed exactly in polynomial time using a simple dynamic program. However, when the terminals have arbitrary demands, the demands of the subtours in each component might be among an exponential number of values.[4] Indeed, unless $P = NP$, we cannot compute in polynomial time a better-than-1.5 approximate solution inside a component, since that problem generalizes the bin packing problem.

To compute in polynomial time good approximate solutions within each component, at a high level, we simplify the solution structure in each component, so that the demands of the subtours in that component are among a *constant* $O_\epsilon(1)$ number of values,[5] while increasing the cost of the solution by at most a multiplicative factor $1.5 + O(\epsilon)$.

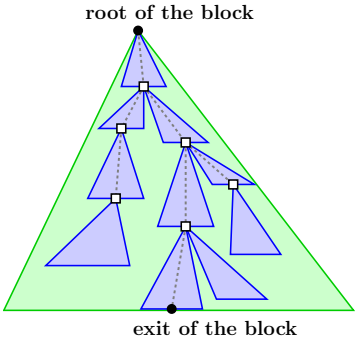
Where does the 1.5 factor come from? Intuitively, our construction creates an additional subtour to cover a selected subset of terminals, charging each edge on that subtour to *two* existing subtours using that edge, thus adding a 0.5 factor to the cost.

In the rest of this section, we explain our approach in more details.

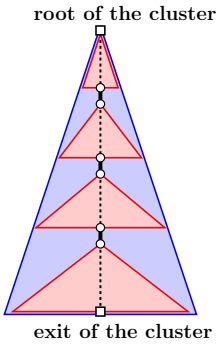
[3] The *demand of a subtour* is the total demand of the terminals visited by that subtour.
[4] For example, consider a component that is a star graph with $\Theta(n)$ leaves, where the i^{th} leaf has demand $1/2^i$.
[5] The notation $O_\epsilon(1)$ stands for $O(f(\epsilon))$ where $f(\epsilon)$ is any function on ϵ.



(a) Decomposition of a component into *blocks*. The orange nodes represent the *big* terminals in the component. The black nodes represent the *root* and the *exit* vertices of the component (defined in Lemma 6). The gray nodes are the branching vertices in the subtree spanning the orange and the black nodes. Splitting the component at the orange, the black, and the gray nodes results in a set of *blocks*, represented by green triangles. Each block has a *root* vertex and at most one *exit* vertex. See Section 4.1.



(b) Decomposition of a block into *clusters*. The green triangle represents a block. Each blue triangle represents a *cluster*. Each cluster has a *root* vertex and at most one *exit* vertex. A cluster is *passing* if it has an exit vertex, and is *ending* otherwise. Each passing cluster has a *spine* (dashed). See Section 4.2.



(c) Decomposition of a passing cluster into *cells*. The blue triangle represents a passing cluster. Removing the thick edges from the cluster results in a set of at most $1/\epsilon$ *cells*. Each red triangle represents a cell. Each of those cells has a *root* vertex, an *exit* vertex, and a *spine* (dashed). See Section 4.3.

Figure 2 Three-level decomposition of a component.

2.2.1 Multi-Level Decomposition (Section 4)

We partition each component into $O_\epsilon(1)$ parts using a *multi-level decomposition*.

In the first level, the component is decomposed into $O_\epsilon(1)$ *blocks* so that all terminals strictly inside a block are *small* (we distinguish *big* and *small* terminals depending on their demands). See Figure 2a and Section 4.1.

In the second level, each block is decomposed into $O_\epsilon(1)$ *clusters* so that the overall demand of each cluster is roughly an ϵ fraction of the demand of a component. See Figure 2b and Section 4.2.

In the third level, each cluster is decomposed into $O_\epsilon(1)$ *cells* so that the *spine* of each cell is roughly an ϵ fraction of the *spine* of the cluster, where the *spine* of a cell (resp. a cluster) is the path traversing that cell (resp. that cluster). See Figure 2c and Section 4.3.

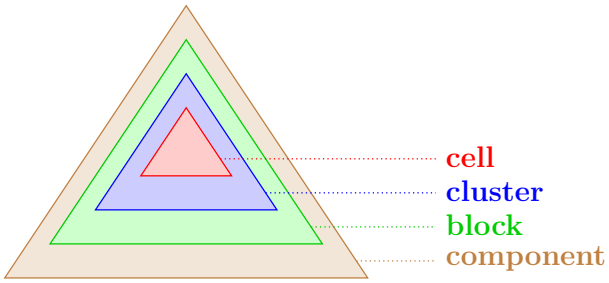


Figure 3 Relation of the multiple levels in the decomposition.

Comparison with the decomposition in [21]

The distinction between big and small terminals plays an important role in UCVRP. This distinction does not exist in the equal demand setting in [21]. In the current paper, the decomposition into blocks is new and enables us to deal with big and small terminals separately; the decomposition into clusters is similar to the decomposition in [21]; the decomposition into cells is a main novelty (see the usage of cells in Section 2.2.2).

2.2.2 Simplifying the Local Solution (Section 5)

The main technical contribution in this paper is the Local Theorem (Theorem 13), which simplifies a local solution inside a component so that, *in each cell, a single subtour visits all small terminals*, while increasing the cost of the local solution by at most a multiplicative factor $1.5 + O(\epsilon)$. The Local Theorem builds upon techniques from [5, 21] together with substantial new ideas.

A first attempt is to reassign all small terminals of each cluster to a single subtour. However, there are two obstacles. First, in order to maintain the connectivity of the resulting subtours, we need to pay for an extra copy of the spines of the clusters, which is expensive. Secondly, using a lemma of Becker and Paul [5], the resulting subtours exceed their capacities slightly. To reduce the demands of the subtours exceeding capacities, an extra cost of only an ϵ fraction of the solution cost is sufficient in the equal demand setting [21], but this is no longer achievable in the arbitrary demand setting.

To overcome those obstacles, we decompose each cluster into *cells* and we reassign all small terminals of each cell to a single subtour. In the analysis, we introduce the technical concept of *threshold cells* (Figure 4a), and we ensure that each cluster contains *at most*

one threshold cell. In order to maintain the connectivity of the resulting subtours, we only need to pay for an extra copy of *the spines of the threshold cells* (Figure 4b), whose cost is negligible.

To reduce the demand of each resulting subtour exceeding capacity, we select some cells from that subtour, and we remove all pieces in that subtour belonging to those cells. We show that each removed piece is connected to the root through at least *two* subtours in the solution (Lemma 20). That property is a main technical novelty in this paper. It enables us to reconnect all removed pieces with an extra cost of at most *half* of the solution cost (Lemma 21), hence an approximation ratio of $1.5 + O(\epsilon)$.

2.3 Postprocessing

We modify the tree of components using the techniques in [21] so that the new tree has only $O_\epsilon(1)$ levels of components. Consider a near-optimal solution in the new tree. We apply the Local Theorem (Theorem 13) to simplify the local solutions in all components. Then we combine the simplified local solutions into a global solution. The combination requires particular care to deal with the additional subtour in each component created in the Local Theorem.

Next, we apply the *adaptive rounding* technique to the resulting global solution. The adaptive rounding technique for capacitated vehicle routing was first used by Jayaprakash and Salavatipour [18] in their design of a QPTAS in the equal demand setting. This technique enables us reduce the number of subtour demands in each subtree to a constant $O_\epsilon(1)$.

Finally, we design a polynomial time dynamic program to compute the best solution that satisfies the structural constraints established previously. The computed solution is a $(1.5 + O(\epsilon))$-approximation.

This completes the proof of Theorem 1. See the full version of the paper for more details.

▶ Remark 2. When the overall cost of all edges in the tree is fixed, letting W denote this cost, it is possible to adapt our analysis to obtain an *asymptotic polynomial time approximation scheme*. To that end, we observe that in the proof of the Local Theorem (Theorem 13), the extra cost to connect all removed pieces in a component is at most twice the overall cost of all edges in that component, so the overall extra cost over all components is at most $2W$. Thus the cost of the computed solution is at most $1 + O(\epsilon)$ times the optimal cost plus $2W$.

3 Preliminaries

3.1 Formal Problem Description and Notations

Let T be a rooted tree (V, E) with edge weights $w(u, v) \geq 0$ for all $(u, v) \in E$. Let n denote the number of vertices in V. The *cost* of a tour (resp. a subtour) t, denoted by $\text{cost}(t)$, is the overall weight of the edges on t. For a set S of tours (resp. subtours), the *cost* of S, denoted by $\text{cost}(S)$, is $\sum_{t \in S} \text{cost}(t)$.

▶ **Definition 3** (UCVRP on trees). *An instance of the* unsplittable capacitated vehicle routing problem (UCVRP) *on* trees *consists of*

- *an edge weighted* tree $T = (V, E)$ *with* root $r \in V$ *representing the* depot,
- *a set* $V' \subseteq V$ *of* terminals,
- *for each terminal* $v \in V'$, *a* demand *of* v, *denoted by* $\text{demand}(v)$, *which belongs to* $(0, 1]$.

A feasible solution is a set of tours *such that*

- *each tour starts and ends at* r,

- *the demand of each terminal is covered by a* single *tour, i.e., the demand cannot be split,*
- *the total demand of the terminals covered by each tour does not exceed the* capacity *of 1.*

The goal is to find a feasible solution of minimum cost.

For any two vertices $u, v \in V$, let $\text{dist}(u, v)$ denote the distance between u and v in the tree T.

We say that a tour (resp. a subtour) *visits* a terminal if it covers the demand of that terminal. For technical reasons, we allow *dummy* terminals of appropriate demands to be included. The *demand* of a tour (resp. a subtour) t, denoted by $\text{demand}(t)$, is defined to be the total demand of all terminals (including dummy terminals) visited by t.

3.2 Reduction to Instances of Bounded Distances

▶ **Definition 4** (bounded distances, Definition 2.1 in [21]). *Let $D_{\min}$ (resp. $D_{\max}$) denote the minimum (resp. maximum) distance between the depot and any terminal in the tree T. We say that T has* bounded distances *if $D_{\max} < (1/\epsilon)^{(1/\epsilon)-1} \cdot D_{\min}$.*

The next theorem (Theorem 5) enables us to assume without loss of generality that the tree T has bounded distances.

▶ **Theorem 5** (Theorem 2.3 and Section 9 in [21]). *For any $\rho \geq 1$, if there is a polynomial time ρ-approximation algorithm for the UCVRP on trees with* bounded distances*, then there is a polynomial time $(1 + 5\epsilon)\rho$-approximation algorithm for the UCVRP on trees with* general distances*.*

3.3 Decomposition Into Components

The next lemma decomposes the tree T into *components*.

▶ **Lemma 6** (Lemma 4.2 in [21]). *Let $\Gamma = 12/\epsilon$. There is a polynomial time algorithm to compute a partition of the edges of the tree T into a set $\mathcal{C}$ of* components *(see Figure 1), such that all of the following properties are satisfied:*

1. *Every component $c \in \mathcal{C}$ is a connected subgraph of T; the* root *vertex of the component c, denoted by r_c, is the vertex in c that is closest to the depot.*
2. *A component c shares vertices with other components at vertex r_c and possibly at one other vertex, called the* exit *vertex of the component c and denoted by e_c. We say that c is an* internal *component if c has an exit vertex, and is a* leaf *component otherwise.*
3. *The total demand of the terminals in each component $c \in \mathcal{C}$ is at most 2Γ.*
4. *The number of components in $\mathcal{C}$ is at most $\max\{1, 3 \cdot \text{demand}(T)/\Gamma\}$, where $\text{demand}(T)$ denotes the total demand of the terminals in the tree T.*

▶ **Definition 7** (Definition 4.4 in [21]). *Let $c \in \mathcal{C}$ be any component. A* subtour in component c *is a path t that starts and ends at the root r_c of component c, and such that every vertex on t is in component c. We say that a subtour t is a* passing subtour *if c has an exit vertex and that vertex belongs to t, and is an* ending subtour *otherwise.*

4 Multi-Level Decomposition in a Component

Let $c \in \mathcal{C}$ be any component. We partition c using a *multi-level decomposition*: first, c is decomposed into *blocks* (Section 4.1); next, each block is decomposed into *clusters* (Section 4.2); and finally, each cluster is decomposed into *cells* (Section 4.3).

We introduce some notations. Let z denote any block (resp. any cluster or any cell). Then z has a *root* vertex and at most one *exit* vertex. We say that a terminal v is *strictly* inside z if v belongs to z and v is different from the root vertex and the exit vertex of z. The *demand* of z is defined as the total demand of all terminals *strictly* inside z. If z has no exit vertex, then z is called *ending*; otherwise z is called *passing*, and the path between the root vertex and the exit vertex of z is called the *spine* of z.

We distinguish *big* and *small* terminals depending on their demands.

▶ **Definition 8** (big and small terminals). *Let* $\alpha = \epsilon^{(1/\epsilon)+1}$. *Let* $\Gamma' = \epsilon \cdot \alpha/\Gamma$, *where* Γ *is defined in Lemma 6. We say that a terminal* v *is* big *if* $\text{demand}(v) > \Gamma'$ *and* small *otherwise.*

4.1 Decomposition of a Component Into Blocks (Figure 2a)

Let c be a component. Let $U \subseteq V$ denote the set of vertices consisting of the big terminals in c, the *root* vertex of c, and possibly the *exit* vertex of c if c is an *internal* component (see Lemma 6 for definitions). Let T_U denote the subtree of c spanning the vertices in U. We say that a vertex in T_U is a *key* vertex if either it belongs to U or it has two children in T_U. We define a *block* to be a maximally connected subgraph of component c in which any key vertex has degree 1; in other words, blocks are obtained by splitting the component at the key vertices. Note that any terminal strictly inside a block is small. The blocks form a partition of the edges of component c.

4.2 Decomposition of a Block Into Clusters (Figure 2b)

As an adaptation from Lemma 6, we decompose a block into clusters in Lemma 9.

▶ **Lemma 9.** *Let* b *be any block. There is a polynomial time algorithm to compute a partition of the edges of the block* b *into a set of* clusters, *such that all of the following properties are satisfied:*

1. *Every cluster* x *is a connected subgraph of* b*; the* root *vertex of the cluster* x*, denoted by* r_x*, is the vertex in* x *that is closest to the depot.*
2. *A cluster* x *shares vertices with other clusters at vertex* r_x *and possibly at one other vertex, called the* exit *vertex of the cluster* x *and denoted by* e_x*. If block* b *has an exit vertex* e_b*, then there is a cluster* x *in* b *such that* $e_x = e_b$*.*
3. *The demand of each cluster in* b *is at most* $2\Gamma'$*.*
4. *The number of clusters in* b *is at most* $3 \cdot (\text{demand}(b)/\Gamma' + 1)$*.*

4.3 Decomposition of a Cluster Into Cells (Figure 2c)

Let x be any cluster.

Case 1: x is an ending cluster. The decomposition of x consists of a single *cell*, which is the entire cluster x.

Case 2: x is a passing cluster. Let ℓ denote the cost of the spine of cluster x. If $\ell = 0$, the decomposition of x consists of a single *cell*, which is the entire cluster x. Next, we assume that $\ell > 0$. For each integer $i \in [1, (1/\epsilon) - 1]$, there exists a unique edge (u, v) on the spine of cluster x satisfying $\min(\text{dist}(r_x, u), \text{dist}(r_x, v)) \leq i \cdot \epsilon \cdot \ell < \max(\text{dist}(r_x, u), \text{dist}(r_x, v))$; let e_i denote that edge. Removing the edges $e_1, e_2, \ldots, e_{(1/\epsilon)-1}$ from cluster x results in at most $1/\epsilon$ connected subgraphs; each subgraph is called a *cell*. Observe that those cells form a partition of the vertices of cluster x.

The (unique) cell inside an ending cluster is an ending cell, and any cell inside a passing cluster is a passing cell. Fact 10 follows directly from the construction.

▶ **Fact 10.** *Let x be a passing cluster. The cost of the spine of any cell in x is at most an ϵ fraction of the cost of the spine of x.*

▶ **Fact 11.** *In any component c, the number of cells and the number of big terminals are both $O_\epsilon(1)$.*

Proof. By Lemma 6, the total demand of the terminals in component c is at most 2Γ. Since the demand of a big terminal is at least Γ', there are at most $2\Gamma/\Gamma' = O_\epsilon(1)$ big terminals in c.

From the construction in Section 4.1, the set U consists of at most $2 + 2\Gamma/\Gamma'$ vertices. Since each vertex in c has at most two children, the number of blocks in c is at most $2|U| \leq 4 + 4\Gamma/\Gamma'$. From the construction in Section 4.2, each block b is partitioned into at most $3 \cdot (\text{demand}(b)/\Gamma' + 1)$ clusters, where demand(b) is at most the total demand of the terminals in component c, which is at most 2Γ. From the construction in Section 4.3, each cluster is partitioned into at most $1/\epsilon$ cells. So the number of cells in c is at most $(4 + 4\Gamma/\Gamma') \cdot (3 \cdot (2\Gamma/\Gamma' + 1)) \cdot (1/\epsilon) = O_\epsilon(1)$. ◀

▶ **Definition 12** (Adaptation from Definition 7)**.** *A* subtour in a cluster (resp. cell) *is a path t that starts and ends at the root of that cluster (resp. cell), and such that every vertex on t is in that cluster (resp. cell). We say that a subtour t is a* passing subtour *if that cluster (resp. cell) has an exit vertex and that vertex belongs to t, and is an* ending subtour *otherwise. The* spine subtour *in a passing cluster (resp. passing cell) consists of the spine of that cluster (resp. cell) in both directions.*

5 Simplifying the Local Solution

In this section, we prove the Local Theorem (Theorem 13).

▶ **Theorem 13** (Local Theorem)**.** *Let c be any component. Let S_c denote a set of at most $(2\Gamma/\alpha) + 1$ subtours in component c visiting all terminals in c. Then there exists a set S_c^* of subtours in component c visiting all terminals in c, such that all of the following properties hold:*

1. *For each cell in c, a single subtour in S_c^* visits all small terminals in that cell;*
2. *S_c^* contains one particular subtour $\bar{t}$ of demand at most 1, and the subtours in $S_c^* \setminus \{\bar{t}\}$ are in one-to-one correspondence with the subtours in S_c, such that for every subtour t in S_c and its corresponding subtour t^* in $S_c^* \setminus \{\bar{t}\}$, the demand of t^* is at most the demand of t, and in addition, if t is a passing subtour in c, then t^* is also a passing subtour in c;*
3. *The cost of S_c^* is at most $1.5 + 2\epsilon$ times the cost of S_c.*

▶ Remark 14. Note that the cost to connect the newly generated subtour $\bar{t}$ to the depot is negligible thanks to the properties of the components.

5.1 Construction of S_c^*

The construction of S_c^* starts from S_c and proceeds in 5 steps. In particular, Step 2 uses a new concept of *threshold cells* and is the main novelty in the construction.

The following lemma due to Becker and Paul [5] will be used in Step 1 and Step 3.

▶ **Lemma 15** (Assignment Lemma, Lemma 1 in [5]). *Let $G = (V[G], E[G])$ be an edge-weighted bipartite graph with vertex set $V[G] = A \uplus B$ and edge set $E[G] \subseteq A \times B$, such that each edge $(a, b) \in E[G]$ has a weight $w(a, b) \geq 0$. For each vertex $b \in B$, let $N(b)$ denote the set of vertices $a \in A$ such that $(a, b) \in E[G]$. We assume that $N(b) \neq \emptyset$ and the weight $w(b)$ of the vertex b satisfies $0 \leq w(b) \leq \sum_{a \in N(b)} w(a, b)$. Then there exists a function $f : B \to A$ such that each vertex $b \in B$ is assigned to a vertex $a \in N(b)$ and, for each vertex $a \in A$, we have*

$$\sum_{b \in B | f(b) = a} w(b) - \sum_{b \in B | (a,b) \in E[G]} w(a, b) \leq \max_{b \in B} \{w(b)\}.$$

Step 1: Combining ending subtours within each cluster

Let A_0 denote S_c. We define a weighted bipartite graph G in which the vertices in one part represent the subtours in A_0 and the vertices in the other part represent the clusters in c.[6] There is an edge in G between a subtour $a \in A_0$ and a cluster x in c if and only if a contains an ending subtour t in x; the weight of the edge is defined to be demand(t). For each cluster x in c, we define the weight of x in G to be the sum of the weights of its incident edges in G. We apply the Assignment Lemma (Lemma 15) to the graph G (deprived of the vertices of degree 0) and obtain a function f that maps each cluster x in c to some subtour $a \in A_0$ such that (a, x) is an edge in G.

We construct a set of subtours A_1 as follows: for every cluster x in c and for every subtour $a \in A_0$ containing an ending subtour t in x, the subtour t is removed from a and added to the subtour $f(x)$. Observe that each resulting subtour in A_1 is connected. From the construction, *for each cluster x, at most one subtour in A_1 has an ending subtour in x*. In particular, for any *ending cell*, which is equivalent to an ending cluster, a single subtour in A_1 visits all small terminals in that cell.

Step 2: Extending ending subtours within threshold cells

Let x be any passing cluster in c such that there is a subtour in A_1 containing an ending subtour in x. From Step 1 of the construction, such a subtour in A_1 is unique; let t_e denote the corresponding ending subtour in x.

We define the **threshold cell** of cluster x to be the deepest cell in x containing vertices of t_e. See Figure 4a.

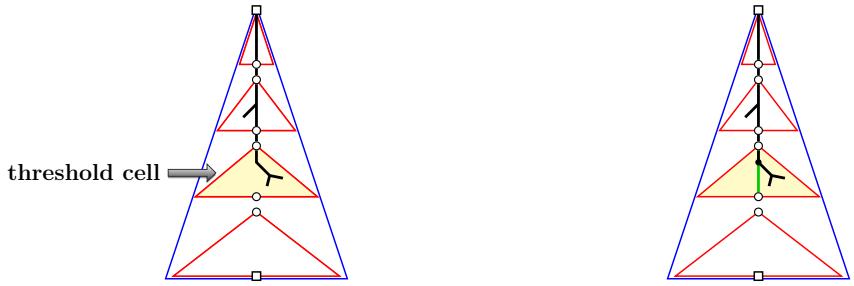
Then we add to t_e the part of the *spine subtour in the threshold cell of x* that does not belong to t_e, resulting in a subtour $\tilde{t}_e$; see Figure 4b.

Let A_2 denote the resulting set of subtours in c after the extension within all threshold cells. From the construction, *for each passing cell s, all subtours in s that are contained in A_2 are passing subtours in s.*

Step 3: Combining passing subtours within each passing cell

We define a weighted bipartite graph G' in which the vertices in one part represent the subtours in A_2 and the vertices in the other part represent the passing cells in c.[7] There is an edge in G' between a subtour $a \in A_2$ and a passing cell s in c if and only if a contains a non-spine passing subtour t in s; the weight of the edge is defined to be the total demand of the small terminals on t. For each passing cell s in c, we define the weight of s in G' to

[6] With a slight abuse, we identify a vertex in G with either a subtour in A_0 or a cluster in c.
[7] With a slight abuse, we identify a vertex in G' with either a subtour in A_2 or a passing cell in c.



(a) Subtour before extension. **(b)** Subtour after extension.

Figure 4 The threshold cell and the extension of an ending subtour. The outermost triangle in blue represents a cluster x. In Figure 4a, the black segments represent the ending subtour t_e in x. The *threshold cell* of cluster x is the *deepest* cell visited by t_e and is represented by the yellow triangle. In Figure 4b, subtour t_e is extended within the threshold cell: the green segment represents the part of the *spine subtour of the threshold cell* that is added to t_e, resulting in a subtour $\tilde{t}_e$.

be the sum of the weights of its incident edges in G'. We apply the Assignment Lemma (Lemma 15) to the graph G' (deprived of the vertices of degree 0) and obtain a function f' that maps each passing cell s in c to some subtour $a \in A_2$ such that (a, s) is an edge in G'.

We construct a set of subtours A_3 as follows: for every passing cell s in c and for every subtour $a \in A_2$ containing a non-spine passing subtour t in s, the subtour t is removed from a except for the spine subtour of s; the removed part is added to the subtour $f'(s)$. Observe that each resulting subtour in A_3 is connected. From the construction, *for each passing cell s, a single subtour in A_3 visits all small terminals in s.*

Step 4: Correcting subtour capacities

For each subtour t_3 in A_3, let t_0 denote the corresponding subtour in A_0. As soon as the demand of t_3 is greater than the demand of t_0, we repeatedly modify t_3 as follows: find a terminal v that is *visited by t_3 but not visited by t_0*; let s denote the cell containing v and let t_s denote the subtour of t_3 in cell s; if s is an ending cell, then remove t_s from t_3; and if s is a passing cell, then remove t_s from t_3 except for the spine subtour of s.

Let A_4 denote the resulting set of modified subtours. Observe that each subtour in A_4 is connected. From the construction, *the demand of each subtour in A_4 is at most the demand of the corresponding subtour in A_0.* Note that the big terminals in each subtour in A_4 are the same as the big terminals in the corresponding subtour in A_0.[8]

Let $\mathcal{R}$ denote the set of the removed pieces. We claim that the total demand of the pieces in $\mathcal{R}$ is at most 1 (Lemma 22).

Step 5: Creating an additional subtour

We connect all pieces in $\mathcal{R}$ by a single subtour $\bar{t}$, which is the minimal subtour in component c that connects all pieces in $\mathcal{R}$ to the root of component c.

Finally, let S_c^* denote $A_4 \cup \{\bar{t}\}$.

[8] Any big terminal cannot be removed, since it is the exit vertex of some cell, thus belongs to the spine of that cell.

5.2 Analysis on the Cost of S_c^*

From the construction of S_c^*, we observe that the cost of S_c^* equals the cost of S_c plus the extra costs in Step 2 and in Step 5 of the construction, denoted by W_2 and W_5, respectively.

To analyze the extra costs, first, in a preliminary lemma (Lemma 16), we bound the overall cost of the spines of the threshold cells. Lemma 16 will be used to analyze both W_2 (Corollary 17) and W_5 (Lemma 21).

▶ **Lemma 16.** *The overall cost of the spines of all threshold cells in the component c is at most $(\epsilon/2) \cdot \text{cost}(S_c)$.*

Proof. Consider any threshold cell s. Let x be the passing cluster that contains s. By Fact 10, the cost of the spine of cell s is at most an ϵ fraction of the cost of the spine of x. Since x is a passing cluster, at least one subtour in S_c contains a passing subtour in x; let t_x denote that passing subtour in x. Observe that t_x contains each edge of the spine of cluster x in both directions (Definition 12), so the cost of the spine of x is at most $\text{cost}(t_x)/2$. Thus the cost of the spine of s is at most $(\epsilon/2) \cdot \text{cost}(t_x)$. We *charge* the cost of the spine of s to t_x.

From the construction, each cluster contains at most one threshold cell. Thus the costs of the spines of all threshold cells are charged to disjoint parts of S_c. The claim follows. ◀

Observe that the extra cost in Step 2 of the construction is at most the overall cost of the spine subtours in all threshold cells in the component c, which equals twice the overall cost of the spines of those cells by Definition 12.

▶ **Corollary 17.** *The extra cost W_2 in Step 2 of the construction is at most $\epsilon \cdot \text{cost}(S_c)$.*

Next, we bound the extra cost in Step 5 of the construction.

▶ **Fact 18.** *Let t denote any subtour in S_c. Let x denote any cluster in c. Let r_c and r_x denote the root vertices of component c and of cluster x, respectively; let e_x denote the exit vertex of cluster x. If the r_c-to-r_x path (resp. the r_c-to-e_x path) belongs to t, then that path belongs to the corresponding subtour of t throughout the construction in Section 5.1.*

▶ **Definition 19** (nice edges)**.** *We say that an edge e in component c is* nice *if e belongs to* at least two *subtours in A_2.*

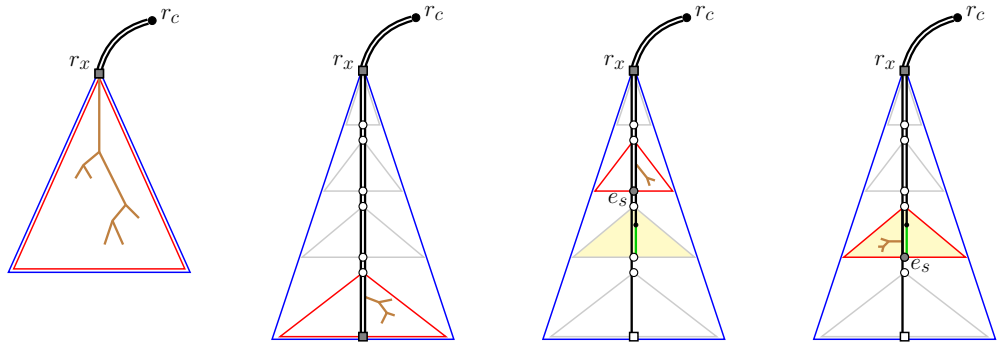
The next Lemma (Lemma 20) is the main novelty in the analysis.

▶ **Lemma 20.** *Any piece in $\mathcal{R}$ is connected to the root r_c of component c through nice edges in c.*

Proof. Consider any piece $q \in \mathcal{R}$. Let s be the cell containing q. Let x be the cluster containing q. See Figure 5. Let r_s and r_x denote the root vertices of cell s and of cluster x, respectively. Observe that the terminals in x are visited by at least two subtours in S_c. This is because, if all terminals in cluster x are visited by a single subtour in S_c, then those terminals belong to the corresponding subtour throughout the construction, thus none of those terminals belongs to a piece in $\mathcal{R}$, contradiction. Thus the r_c-to-r_x path belongs to at least two subtours in S_c. By Fact 18, the r_c-to-r_x path belongs to at least two subtours in A_2, thus every edge on the r_c-to-r_x path is nice. It suffices to show the following Claim:

Piece q is connected to vertex r_x through nice edges in c. (*)

There are two cases:



(a) Case 1. **(b)** Subcase 2(i). **(c)** Subsubcase 2(ii)(α). **(d)** Subsubcase 2(ii)(β).

Figure 5 Illustrations for the different cases in the proof of Lemma 20. A piece $q \in \mathcal{R}$ is in brown. The cell s containing that piece is represented by the triangle in red; the cluster x containing that piece is represented by the outermost triangle in blue. The black node r_c is the root of component c. In Figure 5a, x is an ending cluster. In Figure 5b, x is a passing cluster, and the solution S_c contains two passing subtours in x. In Figures 5c and 5d, x is a passing cluster, and the solution S_c contains a unique passing subtour in x; the yellow triangle represents the threshold cell of x. In the case when q belongs to the threshold cell (Figure 5d), q is connected to r_c through at least two subtours, thanks to the extension of the ending subtour within the threshold cell.

Case 1: x is an ending cluster. See Figure 5a. From the decomposition in Section 4.3, s is an ending cell and s equals x. Piece q is an ending subtour in x and in particular contains r_x. Claim (*) follows trivially.

Case 2: x is a passing cluster. Let e_s and e_x denote the exit vertices of cell s and of cluster x, respectively. Observe that at least one subtour in S_c contains a passing subtour in x. There are two subcases.

Subcase 2(i): At least two subtours in S_c contain passing subtours in x. See Figure 5b. Then the r_c-to-e_x path belongs to at least two subtours in S_c. By Fact 18, the r_c-to-e_x path belongs to at least two subtours in A_2, thus each edge on the spine of x is nice. Since piece q contains a vertex on the spine of x, Claim (*) follows.

Subcase 2(ii): Exactly one subtour in S_c contains a passing subtour in x. See Figures 5c and 5d. Let t_p denote that passing subtour in x. As observed previously, at least two subtours in S_c visit terminals in x, so there must be at least one subtour in S_c that contains an ending subtour in x. Let $t_e^1, \ldots, t_e^m$ (for some $m \geq 1$) denote the ending subtours in x contained in the subtours in S_c. In Step 1 of the construction, the m ending subtours are combined into a single ending subtour, denoted by t_e (recall that the threshold cell of x is defined with respect to t_e); and in Step 2 of the construction, subtour t_e is extended to a subtour $\tilde{t}_e$ (Figure 4). Note that the passing subtour t_p remains unchanged in Steps 1 and 2 of the construction. We observe that cell s is either above or equal to the threshold cell of x. This is because, if cell s is below the threshold cell of x, then all terminals in s are visited by a single subtour in S_c, i.e., the subtour t_p, so those terminals belong to the corresponding subtour of t_p throughout the construction, thus none of those terminals belongs to a piece in $\mathcal{R}$, contradiction. Hence the following two subsubcases.

Subsubcase 2(ii)(α): s is above the threshold cell of x. See Figure 5c. Each edge on the r_x-to-e_s path belongs to both subtours t_p and t_e, hence is nice. Since q contains some vertex on the spine of s, Claim (*) follows.

Subsubcase 2(ii)(β): s equals the threshold cell of x. See Figure 5d. Observe that each edge on the r_x-to-e_s path belongs to $\tilde{t}_e$ due to the extension of the ending subtour t_e within the threshold cell (Step 2 of the construction). Thus each edge on the r_x-to-e_s path belongs to both subtours t_p and $\tilde{t}_e$, hence is nice. Since q contains some vertex on the spine of s, Claim (*) follows. ◀

▶ **Lemma 21.** *The extra cost W_5 in Step 5 of the construction is at most $(0.5+\epsilon)\cdot \text{cost}(S_c)$.*

Proof. Let W_{nice} denote the overall cost of the nice edges in c. We show that $W_5 \leq 2 \cdot W_{\text{nice}}$. Let H be the multi-subgraph in c that consists of the pieces in $\mathcal{R}$ and two copies of each nice edge in c (one copy for each direction). Since any piece in $\mathcal{R}$ is connected to the root r_c of component c through nice edges (Lemma 20), H induces a connected subtour in c. So $W_5 \leq 2 \cdot W_{\text{nice}}$.

Next, we analyze W_{nice}. From the construction, any nice edge e in c is of at least one of the two cases:

Case 1: e belongs to at least two subtours in S_c. Then e has at least 4 copies in S_c, since each subtour to which e belongs contains 2 copies of e (one for each direction). Thus the overall cost of the edges e in this case is at most $0.25 \cdot \text{cost}(S_c)$.

Case 2: e belongs to the spine of a threshold cell in component c. By Lemma 16, the overall cost of the edges e in this case is at most $(\epsilon/2) \cdot \text{cost}(S_c)$.

Hence the overall cost W_{nice} of the nice edges is at most $(0.25+\epsilon/2)\cdot \text{cost}(S_c)$. Therefore, $W_5 \leq 2 \cdot W_{\text{nice}} \leq (0.5+\epsilon)\cdot \text{cost}(S_c)$. ◀

From Corollary 17 and Lemma 21, we conclude that

$$\text{cost}(S_c^*) = \text{cost}(S_c) + W_2 + W_5 \leq (1.5+2\epsilon)\cdot \text{cost}(S_c).$$

Hence the third property of the claim in the Local Theorem (Theorem 13).

5.3 Feasibility

From the construction, S_c^* is a set of subtours in c visiting all terminals in c. The first property of the claim in the Local Theorem (Theorem 13) follows from the construction. The second property of the claim follows from the construction, Fact 18, and the following Lemma 22.

▶ **Lemma 22.** *The total demand of the pieces in $\mathcal{R}$ is at most 1.*

Proof. Observe that the pieces in $\mathcal{R}$ are removed from subtours in A_3. Let t_3 denote any subtour in A_3. Let t_0, t_1, t_2, and t_4 denote the corresponding subtours of t_3 in A_0, A_1, A_2, and A_4, respectively. Let Δ denote the overall demand of the pieces that are removed from t_3 in Step 4 of the construction. Observe that $\Delta = \text{demand}(t_3) - \text{demand}(t_4)$. To bound Δ, first, by Step 1 of the construction and the Assignment Lemma (Lemma 15), the demand of each subtour in A_0 is increased by at most the maximum demand of a cluster. Thus $\text{demand}(t_1) - \text{demand}(t_0)$ is at most the maximum demand of a cluster, which is at most $2\Gamma'$ by the definition of clusters (Section 4.2). By Step 2 of the construction, $\text{demand}(t_2) = \text{demand}(t_1)$. By Step 3 of the construction and the Assignment Lemma (Lemma 15), the demand of each subtour in A_2 is increased by at most the maximum demand of a cell. Thus $\text{demand}(t_3) - \text{demand}(t_2)$ is at most the maximum demand of a cell, which is at most $2\Gamma'$ by the definition of cells (Section 4.3). By Step 4 of the construction, $\text{demand}(t_0) - \text{demand}(t_4)$ is at most the maximum demand of a cell, which is at most $2\Gamma'$. Combining, we have $\Delta = \text{demand}(t_3) - \text{demand}(t_4) \leq 6\Gamma'$.

The number of subtours in A_3 equals the number of subtours in S_c, which is at most $(2\Gamma/\alpha)+1$ by assumption. Thus total demand of the pieces in $\mathcal{R}$ is at most $6\Gamma' \cdot ((2\Gamma/\alpha)+1) < 13\epsilon < 1$, assuming $\epsilon < 1/13$. ◀

This completes the proof of the Local Theorem (Theorem 13).

References

1 Muhammad Al-Salamah. Constrained binary artificial bee colony to minimize the makespan for single machine batch processing with non-identical job sizes. *Applied Soft Computing*, 29:379–385, 2015.

2 Kemal Altinkemer and Bezalel Gavish. Heuristics for unequal weight delivery problems with a fixed error guarantee. *Operations Research Letters*, 6(4):149–158, 1987.

3 Tetsuo Asano, Naoki Katoh, and Kazuhiro Kawashima. A new approximation algorithm for the capacitated vehicle routing problem on a tree. *Journal of Combinatorial Optimization*, 5(2):213–231, 2001.

4 Amariah Becker. A tight 4/3 approximation for capacitated vehicle routing in trees. In *International Conference on Approximation Algorithms for Combinatorial Optimization Problems*, volume 116, pages 3:1–3:15, 2018.

5 Amariah Becker and Alice Paul. A framework for vehicle routing approximation schemes in trees. In *Workshop on Algorithms and Data Structures*, pages 112–125. Springer, 2019.

6 Jannis Blauth, Vera Traub, and Jens Vygen. Improving the approximation ratio for capacitated vehicle routing. *Mathematical Programming*, pages 1–47, 2022.

7 Huaping Chen, Bing Du, and George Q. Huang. Scheduling a batch processing machine with non-identical job sizes: a clustering perspective. *International Journal of Production Research*, 49(19):5755–5778, 2011.

8 Jing Chen, He Guo, Xin Han, and Kazuo Iwama. The train delivery problem revisited. In *International Symposium on Algorithms and Computation*, pages 601–611. Springer, 2013.

9 Purushothaman Damodaran, Praveen Kumar Manjeshwar, and Krishnaswami Srihari. Minimizing makespan on a batch-processing machine with non-identical job sizes using genetic algorithms. *International Journal of Production Economics*, 103(2):882–891, 2006.

10 George B. Dantzig and John H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

11 Aparna Das, Claire Mathieu, and Shay Mozes. The train delivery problem-vehicle routing meets bin packing. In *International Workshop on Approximation and Online Algorithms*, pages 94–105. Springer, 2010.

12 Lionel Dupont and Clarisse Dhaenens-Flipo. Minimizing the makespan on a batch machine with non-identical job sizes: an exact procedure. *Computers & Operations Research*, 29(7):807–819, 2002.

13 Zachary Friggstad, Ramin Mousavi, Mirmahdi Rahgoshay, and Mohammad R. Salavatipour. Improved approximations for capacitated vehicle routing with unsplittable client demands. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 251–261. Springer, 2022.

14 Bruce L. Golden and Richard T. Wong. Capacitated arc routing problems. *Networks*, 11(3):305–315, 1981.

15 Fabrizio Grandoni, Claire Mathieu, and Hang Zhou. Unsplittable Euclidean Capacitated Vehicle Routing: A $(2+\epsilon)$-Approximation Algorithm. In *Innovations in Theoretical Computer Science (ITCS)*, volume 251 of *LIPIcs*, pages 63:1–63:13, 2023.

16 Mordecai Haimovich and Alexander H. G. Rinnooy Kan. Bounds and heuristics for capacitated routing problems. *Mathematics of Operations Research*, 10(4):527–542, 1985.

17 Shin-ya Hamaguchi and Naoki Katoh. A capacitated vehicle routing problem on a tree. In *International Symposium on Algorithms and Computation*, pages 399–407. Springer, 1998.

18 Aditya Jayaprakash and Mohammad R. Salavatipour. Approximation schemes for capacitated vehicle routing on graphs of bounded treewidth, bounded doubling, or highway dimension. *ACM Transactions on Algorithms (TALG)*, 19(2), 2023.

19 Ali Husseinzadeh Kashan, Behrooz Karimi, and Fariborz Jolai. Effective hybrid genetic algorithm for minimizing makespan on a single-batch-processing machine with non-identical job sizes. *International Journal of Production Research*, 44(12):2337–2360, 2006.

20 Martine Labbé, Gilbert Laporte, and Hélene Mercure. Capacitated vehicle routing on trees. *Operations Research*, 39(4):616–622, 1991.

21 Claire Mathieu and Hang Zhou. A PTAS for capacitated vehicle routing on trees. *ACM Transactions on Algorithms (TALG)*, 19(2), 2023.

22 Sharif Melouk, Purushothaman Damodaran, and Ping-Yu Chang. Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing. *International Journal of Production Economics*, 87(2):141–147, 2004.

23 İbrahim Muter. Exact algorithms to minimize makespan on single and parallel batch processing machines. *European Journal of Operational Research*, 285(2):470–483, 2020.

24 N. Rafiee Parsa, Behrooz Karimi, and Ali Husseinzadeh Kashan. A branch and price algorithm to minimize makespan on a single batch processing machine with non-identical job sizes. *Computers & Operations Research*, 37(10):1720–1730, 2010.

25 Thomas Rothvoß. The entropy rounding method in approximation algorithms. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 356–372. SIAM, 2012.

26 Reha Uzsoy. Scheduling a single batch processing machine with non-identical job sizes. *The International Journal of Production Research*, 32(7):1615–1635, 1994.

27 Yuanxiao Wu and Xiwen Lu. Capacitated vehicle routing problem on line with unsplittable demands. *Journal of Combinatorial Optimization*, pages 1–11, 2020.

Online Demand Scheduling with Failovers

Konstantina Mellou
Microsoft Research, Redmond, WA, USA

Marco Molinaro
Microsoft Research, Redmond, WA, USA
PUC-Rio de Janeiro, Brazil

Rudy Zhou[1]
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

Motivated by cloud computing applications, we study the problem of how to optimally deploy new hardware subject to both power and robustness constraints. To model the situation observed in large-scale data centers, we introduce the *Online Demand Scheduling with Failover* problem. There are m identical devices with capacity constraints. Demands come one-by-one and, to be robust against a device failure, need to be assigned to a pair of devices. When a device fails (in a failover scenario), each demand assigned to it is rerouted to its paired device (which may now run at increased capacity). The goal is to assign demands to the devices to maximize the total utilization subject to both the normal capacity constraints as well as these novel failover constraints. These latter constraints introduce new decision tradeoffs not present in classic assignment problems such as the Multiple Knapsack problem and AdWords.

In the worst-case model, we design a deterministic $\approx \frac{1}{2}$-competitive algorithm, and show this is essentially tight. To circumvent this constant-factor loss, which represents substantial capital losses for big cloud providers, we consider the stochastic arrival model, where all demands come i.i.d. from an unknown distribution. In this model we design an algorithm that achieves sub-linear additive regret (i.e. as OPT or m increases, the multiplicative competitive ratio goes to 1). This requires a combination of different techniques, including a configuration LP with a non-trivial post-processing step and an online monotone matching procedure introduced by Rhee and Talagrand.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases online algorithms, approximation algorithms, resource allocation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.92

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2209.00710`

Funding *Marco Molinaro*: Supported in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES, Brasil) - Finance Code 001, by Bolsa de Produtividade em Pesquisa #312751/2021-4 from CNPq, FAPERJ grant "Jovem Cientista do Nosso Estado".

Acknowledgements We thank the anonymous reviewers for their valuable suggestions. We also thank Alok Gautam Kumbhare and Ishai Menache for useful discussions.

1 Introduction

A critical challenge faced by cloud providers is how to deploy new hardware to satisfy the increasing demand for cloud resources, and the main bottleneck in this process is power. Data centers consist of power devices with limited capacity and each demand for hardware (e.g.

[1] Work performed as intern at Microsoft Research, Redmond.

rack of servers) has a power requirement. The goal is to assign demands to power devices to fulfill their requirements while using the available power in the data centers efficiently. This allows cloud providers to maximize return on investment on existing data centers before incurring large capital expenses for new data centers to accommodate additional demand.

An important consideration that sets this demand assignment process apart from other applications is *reliability*. Cloud users are promised a high availability of service which mandates that cloud capacity can only be unavailable for very short durations (between a few minutes and a few hours per year). As a result, assigning each demand to a single power device leads to an unacceptable level of risk; if that device fails, the capacity for the demand becomes unavailable, leading to potentially millions of dollars in costs for the provider and jeopardizing the cloud business model that is highly dependent on users' trust. To this end, power *redundancy* is built into the assignment process.

We consider a specific model of redundancy used by large cloud providers [19]. In this model, each demand gets assigned to two power devices. In normal operations (no device failure), the demand obtains half of its required power from each device. If one of the devices fails, the remaining device must provide the full power amount to the demand (see Figure 1 for an example). In these failover scenarios, the remaining devices may run at an increased capacity temporarily to accommodate their increased load. The provider uses this time to take ad-hoc corrective actions, for instance, shut down certain workloads and reduce the power of others in order to bring the power utilization of each device back within its normal limits. As in [19] we consider a single device failure at a time.

This architecture is favored in practice because it provides strong reliability guarantees with a small increase in overhead and complexity. One could consider more complex architectures, where demands could be assigned with a power split other than half-half to each device or to more than two devices, but this comes at an increased cost in hardware and operational complexity. Further, a common goal of large cloud providers is to provide statistical guarantees for high service availability, e.g., 99.99% availability for certain cloud resources or services; cloud operators have determined that accounting for a single device failure with the described architecture provides such target guarantees.

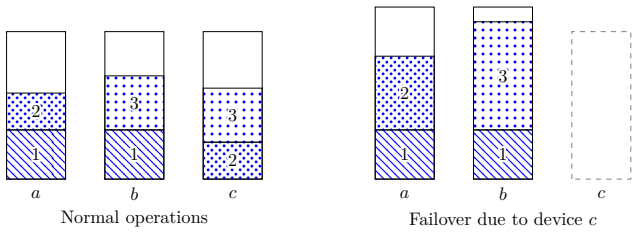
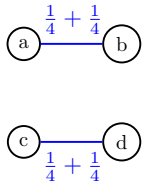
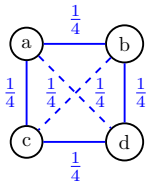


Figure 1 In normal operations (left), each demand (denoted with a different pattern) is assigned to two devices and gets half of its required power from each device. In the failover scenario where device c has failed (right), the demands that were assigned to c now get their full power from the remaining devices that may run at increased capacity.

We introduce the *Online Demand Scheduling with Failover* problem (FAILOVER) to model this issue of assigning demands to power devices with redundancy. Formally, in this problem there are m identical devices (or machines) and n demands. Each device has two capacities: a nominal capacity that is normalized to 1 and a failover capacity $B \geq 1$. Each demand j has some size $s_j \geq 0$, which for convenience is defined as its per-device power requirement (so the total power requirement of the demand is $2s_j$). The demands arrive online one-by-one and there is no knowledge about future demands. The goal is to irrevocably assign the arriving

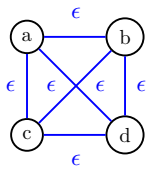


(a) Bad assignment.

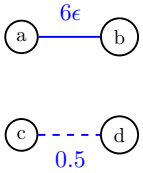


(b) Good assignment.

Figure 2 Illustration of Example 1.



(a) Bad assignment.



(b) Good assignment.

Figure 3 Illustration of Example 2.

demands to pairs of devices (or edges, where we consider each device as a node) satisfying:

1. (Nominal Constraints) For every device u, its total load has to be at most 1, namely $L_u := \sum_{v \neq u} L_{uv} \leq 1$, where we define $L_{uv} = \sum_{j \to uv} s_j$ to be the total load on edge uv (i.e., all demands assigned to the pair of devices uv).
2. (Failover Constraints) For every device u, we have $L_u + \max_{v \neq u} L_{uv} \leq B$ (i.e., if a device $v \neq u$ fails, all demands assigned to uv have to be supplied solely by u, which sees its load increased by the amount L_{uv} that was formerly supplied to them by device v; the increased load has to fit the failover capacity B).

We assume that each demand size s_j fits on a pair of devices by itself, so $s_j \in [0, \min(1, B/2)]$. We are not allowed to reject demands, so the algorithm assigns arriving demands to the available devices until a demand cannot be scheduled, in which case the algorithm terminates. Our objective is to maximize the total size of all assigned demands (i.e., the utilization). We compare the algorithm against the optimal offline strategy that knows the demand sequence in advance (but still subject to the same no-rejection requirement). We use OPT to denote the total utilization of this optimal offline strategy.

This problem has similarities with several classical packing problems. For example, in the Multiple Knapsack problem (and related problems such as Generalized Assignment [18], AdWords [14], etc.) we are given a set of items each with a weight and size, and the goal is to select a subset of the items to pack in capacitated bins in order to maximize the total weight. However, one fundamental difference in our setting, besides the need to assign each demand to a pair of devices instead of a single device, is the failover constraint. Unlike in previously studied resource allocation problems, here the capacity constraints are not just determined by the total demand incident to a node, but rather they depend also on how the demands are arranged across its edges. See the next example.

▶ **Example 1.** Consider an instance with 4 power devices a, b, c, d with failover capacity $B = 1$. There are 6 demands of size $\frac{1}{4}$ that arrive sequentially; suppose 4 demands have arrived so far. One possible assignment has placed 2 demands on each of the pairs ab and cd (see Figure 2a), while a different one may place each of the 4 demands on a different pair (see Figure 2b, in solid lines). While in the second option all remaining demands can be placed (dashed lines in Figure 2b), the first option cannot accommodate more demands due to the Failover capacity. To see this, suppose we assign another demand to device a, say. If device b fails, then the total load on a will become at least $\frac{5}{4}$ violating its Failover capacity.

The above example suggests that due to the Failover constraints we should "spread out" the demands by not putting too many demands on one edge, because if one of its endpoints fails then this edge can have a large contribution to the Failover constraint of the other endpoint. However, there is a danger in spreading out the demands too much and not leaving enough devices free.

▶ **Example 2.** Consider again the same 4 power devices a, b, c, d with failover capacity $B = 1$. Now, there are 7 demands; the first 6 have a small size $\varepsilon > 0$ and the last demand has size 0.5. Assume the first 6 demands have arrived. A first option is to assign one demand of size ϵ per device pair (see Figure 3a). In this case, the remaining demand of size 0.5 cannot be placed, as the Failover capacities would be exceeded. The second option groups the first 6 demands on a single edge (see Figure 3b); in this case, all demands can be fulfilled by assigning the last demand on a disjoint edge (dashed edge of Figure 3b).

Taking these two examples together, there is a delicate balance between spreading demands out across edges to minimize their impact in failover scenarios and leaving enough devices open for future demands, as to not prematurely end up with an unassignable demand.

1.1 Our results

We first consider the FAILOVER problem in the worst-case and design a deterministic algorithm with competitive ratio $\approx \frac{1}{2}$. Since no deterministic algorithm can be better than $\frac{1}{2}$-competitive (see the full version of the paper for upper bound), this result is almost best possible.

▶ **Theorem 3.** *There is a deterministic poly-time online algorithm for* FAILOVER *in the worst-case model with competitive ratio* $\frac{1}{2} - O(\frac{1}{m^{1/3}})$,[2] *where* m *is the number of devices.*

A $\frac{1}{2}$-competitive solution may, roughly speaking, underutilize by a factor of $\frac{1}{2}$ the available power; in the context of big cloud providers, this inefficiency translates to substantial capital expenses due to the extra data centers required to accommodate the demands. Since such losses are unavoidable in the worst-case model, we consider the FAILOVER problem in the *stochastic arrival model*. Here the demand sizes are drawn i.i.d. from an unknown distribution μ supported on $[0, \min(1, B/2)]$.

We show that in this stochastic model it is possible to obtain *sublinear additive regret*.

▶ **Theorem 4.** *For the* FAILOVER *problem in the stochastic arrival model, there is a poly-time algorithm that achieves utilization* $\mathsf{OPT} - O(\mathsf{OPT}^{5/6} \log \mathsf{OPT})$ *with probability* $1 - O(\frac{1}{m})$.

We remark that since OPT grows like $\Theta(\xi)$, where $\xi := \min\{n, m\}$ (see Lemma 9), this guarantee implies the multiplicative approximation $(1 - O(\log \xi \,/\, \xi^{1/6})) \cdot \mathsf{OPT}$. So as the number of demands and devices grow, the competitive ratio of this algorithm goes to 1.

As a subroutine of this algorithm, we need to solve the natural *offline minimization* variant of demand scheduling with failover: Given a collection of demands, minimize the number of devices needed to assign all demands satisfying the Nominal and Failover constraints. We also design an (offline) algorithm with sublinear additive regret for this problem (Section 4).

1.2 Technical Overview

We illustrate the main technical challenges in the FAILOVER problem in both the worst-case and stochastic models, as well as in the offline minimization subproblem needed for the latter.

Online Worst-Case (Section 2). The examples from Figure 2 and 3 show that the main difficulty is dealing with the trade-off between spreading out the demands, which allows for a better use of the failover budgets, and co-locating demands on fewer edges, keeping some edges free for future big demands.

[2] Throughout the paper we use $O(x)$ to mean "$\leq cst \cdot x$" for some constant cst independent of x.

To effectively strike this balance and get near optimal guarantees, the main idea is to group demands based on their sizes using intervals I_k and schedule each group separately on cliques of size k. That is, we will "open" a set of k unused devices and assign the demands in I_k only to the edges between these devices (opening new k-cliques as needed). Interestingly, we assign at most one demand per edge of the clique (other than for tiny demands, which are handled separately). This means the algorithm tries to co-locate demands in controlled *regions*, which allows for the right use of the failover budgets.

Online Stochastic Arrivals (Section 3). First, note that because demands are i.i.d. from a distribution with bounded support, the total utilization of the first ℓ demands grows as $\ell \cdot \mathbb{E}_{S\sim\mu}S$. Thus, it suffices to show that our algorithm "survives" for as many demand arrivals as possible without needing to reject one due to lack of space. Our approach is to try and assign prefixes of arrivals to the (approximately) minimum number of devices possible. This ensures that if our algorithm fails due to needing more than m devices to feasibly assign another demand, then OPT will fail shorty after.

Our algorithm is based on a learn-and-pack framework, where we use knowledge of the first ℓ arrivals to compute a good template assignment for the next ℓ arrivals. To compute this template, we need a subroutine that (approximately) solves the offline minimization subproblem mentioned above. Concretely, we run the subroutine on the realized sizes of the first ℓ arrivals, which gives a possible assignment of these demands into, say m' unused devices. We use the "slots" of this possible assignment as a template to assign the future ℓ demands by employing the *online monotone matching* process of Rhee-Talgrand [16]: For each future arrival, we assign it to a (carefully-chosen) open slot in the template that has a larger size – if we cannot find such an open slot, then we assign this demand to its own disjoint edge (using 2 more devices).

It is known that this matching process leaves $o(\ell)$ unmatched demands with high probability. Further, our offline minimization subroutine has sublinear additive regret, that is, it uses only $o(\ell)$ more devices than the optimal offline assignment. Since these losses are sublinear in the prefix size, it seems that by repeating this process together with doubling the prefix size we should obtain a final sublinear regret guarantee.

But there is still a major issue: This strategy uses *disjoint* sets of devices to fulfill the first ℓ demands and the next ℓ demands (for each doubling ℓ). But this is possibly very wasteful: even using the optimal assignment for each of these ℓ demands separately may require many more devices (up to double) compared to reusing the leftover space from the first batch of ℓ demands for the next batch (i.e. assigning the batches to a common set of devices). Wasting a constant fraction of devices would lead to the unwanted constant-competitive loss. To overcome this, we show that M_ℓ, the minimum number of devices to assign ℓ i.i.d. demands, is approximately linear in ℓ (Theorem 8), e.g. $M_\ell + M_\ell$ (assigning batches separately) is approximately $M_{2\ell}$ (assigning them together). This is a non-trivial task (another Rhee-Talagrand paper [15] is entirely devoted to doing this for the simpler Bin Packing problem). Perhaps surprisingly, our proof relies on our algorithm for the offline device minimization problem, which is LP-based. The crucial property is that the optimal LP value doubles if we duplicate the items on its input, which (with additional probabilistic arguments) translates into the additivity of M_ℓ.

Offline Minimization (Section 4). Our algorithm for offline minimization of the number of devices needed to fulfill a set of demands is based on a configuration LP inspired by the classic Gomory-Gilmore LP for the Bin Packing problem. Consider a fixed assignment of

demands to some number of devices. We want to interpret each device as a configuration, which captures the arrangement of demands on this device's edges. Our LP will minimize the number of configurations needed in order to assign all demands.

There is a tension between two issues in this approach. First, the Failover constraint depends not only on the subset of demands on this device's edges, but also how they are arranged within these edges (because the most-loaded edge contributes to the Failover constraint). This suggests that a configuration should not only specify a subset of demands, but also have enough information about the edge assignment to control the most-loaded edge. Second, each demand must be assigned to a pair of devices rather than a single device, so our configurations are not "independent" of each other. Thus, we need to "match" configurations to ensure that a collection of configurations can be realized in an edge assignment. In summary, our configurations should be expressive enough to capture the Failover constraints, but also simple enough so that we can actually realize them in an actual assignment.

Our solution to this is to define a configuration to be a subcollection, say C, of demands satisfying $\sum_{s\in C} s \leq 1$ (the Nominal constraint) and $\sum_{s\in C} s + \max_{s\in C} s \leq B$ (a relaxed Failover constraint). Note that this notion of configuration does not capture the arrangement of the demands C across a device's edges – we assume the best case that every demand is on its own edge to minimize their impact in failover scenarios. It is not clear that there even exists a near-optimal assignment that assigns at most one demand per edge, let alone that we can obtain one from the LP solution. However, our LP post-processing procedure will show that – by opening slightly more devices – we can match configurations of this form to realize them in a near-optimal assignment.

1.3 Related work

Despite a vast literature on assignment-type problems, none of the ones considered addresses the main issue of redundancy, modeled in the Failover problem. Arguably the Coupled Placement [11] problem is the closest to Failover. Given a bipartite graph with capacities at the nodes and a set of jobs, the goal is to assign a subset of the jobs to the edges of the graph to maximize the total value (each assigned job gives a value that also depends on its assigned edge), while respecting the capacity of the nodes (each assigned job consumes capacity from its edge's endpoints). [11] gives a $\frac{1}{15}$-approximation to the offline version of this problem (see also [1]). While this problem involves the allocation of jobs to a pair of nodes (albeit on a bipartite graph) and has the additional difficulty that the value and consumption of a job depends on which pair of nodes it is assigned, it does not have any Failover type constraints, a crucial component of our problem.

As already mentioned, several classic assignment problems are related to ours, such as the Multiple Knapsack [4], Generalized Assignment (GAP) [18], and AdWords problem [14, 7]. The latter is the closest to our problem: there are m bins (i.e. advertisers) of different capacities, and jobs (i.e. keyword searches) that come one-by-one and need to be assigned to the bins; each assignment consumes some of the bin's capacity and incurs an equal amount of value (i.e. bid). The goal is maximize total value subject to bin capacities. Despite the similarities, this problem does not consider critical aspects of our problem, namely the need to assign a job/demand to a pair of bins/devices and the Failover constraints.

There is also a large literature on survivable network design problems, where failures in the network are explicitly considered [6], but the nature of the problems is quite different from our assignment problem, as the focus there is typically on routing flows.

Finally, a problem related to our device minimization problem, and from which we borrow some tools and techniques, is Bin Packing. Here jobs of different sizes need to be assigned to a minimum number of bins of size 1. Results are known in both offline [9] and online settings [2, 16]. In the online stochastic setting, [16] obtains an additive $+O(\sqrt{\mathsf{OPT}}\cdot\log^{3/4}\mathsf{OPT})$ sublinear approximation (see [5, 8, 13] for improvements under different assumptions).

2 FAILOVER Problem in the Online Worst-Case Model

In this section we consider the FAILOVER in the online worst-case model. We design an algorithm that achieves competitive ratio $\approx \frac{1}{2}$ in this setting (restated from the introduction).

▶ **Theorem 3.** *There is a deterministic poly-time online algorithm for* FAILOVER *in the worst-case model with competitive ratio* $\frac{1}{2} - O(\frac{1}{m^{1/3}})$,[3] *where* m *is the number of devices.*

Recall that in the full version of the paper, we also show the almost matching upper bound of $\frac{1}{2}$ on such competitive ratio, and design another algorithm whose competitive ratio approaches 1 as the size of the largest demand goes to 0. To convey the main ideas more clearly, here we focus only on Theorem 3.

2.1 Algorithm

As suggested in the technical overview, our algorithm will group demands by size, and assign each group of demands to sub-cliques of an appropriate size. To make this precise, set in hindsight $L := m^{1/3}$ and for $k = 2, \ldots, L-1$ define the interval

$$I_k := \left(\min\left\{\frac{1}{k}, \frac{B}{k+1}\right\}, \ \min\left\{\frac{1}{k-1}, \frac{B}{k}\right\}\right].$$

(Notice there is no $k = 1$, because the upper limit of I_2 is the max size of a demand.) This definition ensures that it is feasible to assign one demand of such size to each edge of a k-clique, as we argue in the next subsection. Also define the interval of small sizes

$$I_{\geq L} := \left[0, \min\left\{\frac{1}{L-1}, \frac{B}{L}\right\}\right].$$

The algorithm is then the following:

Algorithm 1 **FailoverWostCase.**

1: When a demand arrives, determine the interval I_k (or $I_{\geq L}$) that it belongs to based on its size.

2: If it belongs to an interval I_k with $k \in \{2, \ldots, L-1\}$, assign the demand to any "empty" edge (i.e. that has not received any demands) of a k-clique opened for I_k. If no such edge exists, then open a new k-clique for I_k.

3: Otherwise it belongs to $I_{\geq L}$, so assign it to an edge of one of its L-cliques using *first-fit* (so here we **can assign multiple demands to the same edge**) making sure that the total load on each edge is at most $\min\{\frac{1}{L-1}, \frac{B}{L}\}$. By first-fit we mean that the edges of the $I_{\geq L}$ cliques are arbitrarily ordered and the demand is assigned to the first possible edge. Open a new L-clique for $I_{\geq L}$ if need be.

4: If the demand cannot fit in the appropriate clique and it is not possible to open a new clique (i.e. there are not enough unused machines to form a clique of the desired size), then stop.

[3] Throughout the paper we use $O(x)$ to mean "$\leq cst \cdot x$" for some constant cst independent of x.

2.2 Analysis

We first quickly verify that the assignment done by the algorithm is feasible, i.e. satisfies the Nominal and Failover constraints. Consider a node/machine u on an I_k clique opened by the algorithm (for machines in an $I_{\geq L}$ clique the argument is analogous). For the Nominal capacity constraint: Every demand assigned to u is actually assigned to one of the $k-1$ edges in this clique incident on u; each such edge receives at most 1 demand from I_k (and no other demands), so using the upper limit of this interval we see that u receives total size at most $(k-1)\cdot\min\{\frac{1}{k-1}, \frac{B}{k}\} \leq 1$, so within its Nominal capacity. For the Failover capacity: in a failover scenario one of these $(k-1)$ demands has "both ends" assigned to u, so the total size it receives is now $k\cdot\min\{\frac{1}{k-1}, \frac{B}{k}\} \leq B$, so within the Failover capacity. Hence the algorithm produces a feasible assignment.

Now we show that the value obtained by the algorithm is at least $\left(\frac{1}{2} - O(\frac{1}{m^{1/3}})\right)\mathsf{OPT}$. The idea is to show that for (essentially) each clique opened by the algorithm, we get on average value at least $\approx \frac{1}{4}$ per vertex. Since each node has Nominal capacity 1 and each demand must be scheduled on two nodes, OPT can only get at most $\frac{1}{2}$ value from each node on average, showing that our algorithm is a $\approx \frac{1}{2}$-approximation. However, there are two exceptions where we may get less than $\approx \frac{1}{4}$ per vertex on average. The first is the last clique for each I_k, which may not be "fully used" (but by setting L appropriately there are not too many nodes involved in this loss). More importantly, the second exception is the "big items" I_2, which may not allow us to get average value $\frac{1}{4}$ per node (e.g. when the failover is $B=1$, a demand of size $\frac{1}{3}+\varepsilon$ falls in the group I_2 and is put by itself on an edge, giving value $\frac{1}{6}+\frac{\varepsilon}{2} \ll \frac{1}{4}$ per node used). However, in this case we show that we can obtain a stronger upper bound for these demands for OPT.

We now make this precise. Assume throughout that the algorithm has stopped before the end of the input (else it scheduled everything, so it is OPT). We account for the value obtained on each type of clique separately.

Cliques for $I_{\geq L}$. We will use two observations: (i) when the algorithm opens a new $I_{\geq L}$ clique, every edge of the previous $I_{\geq L}$ cliques has some demand assigned to it, and (ii) across all $I_{\geq L}$ cliques, out of all edges with some demand assigned to them, at most one can have total size assigned to it less than $\alpha := \frac{1}{2}\min\left\{\frac{1}{L-1}, \frac{B}{L}\right\}$ (i.e. half of its "capacity").

Both observations stem from the first-fit strategy used to assign these demands. In particular, the algorithm will only open a new clique when a demand in $I_{\geq L}$ does not fit in the edges of the existing cliques, implying that all of these edges already have some demand assigned; this shows the first statement. For the second statement, by contradiction assume that at some point there are at least two edges on $I_{\geq L}$ cliques with total load less than α. Then the first demand that was assigned to the last such edge has size less than α. But this means that it could have been assigned to an earlier edge with load less than α, contradicting the first-fit procedure.

Let $c_{\geq L}$ be the total number of $I_{\geq L}$ cliques that the algorithm opened, and $m_{\geq L} := c_{\geq L}\cdot L$ the number of nodes associated with those cliques. Combining the above two observations, at the end of the execution either: (i) every edge of the first $c_{\geq L}-1$ of these cliques has load at least α or; (ii) all but one edge in the first $c_{\geq L}-1$ cliques has load at least α and some edge of the last $c_{\geq L}$-th (e.g., the one that "opened" it) has load at least α. In both cases, the total size of demands assigned by the algorithm to the edges of these cliques is at least

$$(c_{\geq L}-1)\cdot\binom{L}{2}\cdot\alpha = (c_{\geq L}-1)\cdot\frac{L}{4}\cdot\min\left\{1, \frac{(L-1)B}{L}\right\} \geq m_{\geq L}\cdot\frac{1}{4}\left(1-\frac{1}{L}\right) - O(L), \qquad (1)$$

yielding roughly average value $\frac{1}{4}$ from each node of these cliques, as claimed.

Cliques for I_k, for $k \geq 3$. Consider any clique for I_k except the last one to be opened. All edges of this clique have some demand from I_k assigned to it; given the lower limit for this interval, this means that the algorithm has assigned to each such clique total size at least

$$\binom{k}{2} \cdot \min\left\{\frac{1}{k}, \frac{B}{k+1}\right\} = \frac{k}{2} \cdot \min\left\{\frac{k-1}{k}, \frac{B(k-1)}{k+1}\right\}.$$

Since $k \geq 3$ and $B \geq 1$, the right-hand side is at least $\frac{k}{4}$. Letting again c_k denote the number of cliques for I_k that the algorithm opens and m_k the corresponding number of nodes/machines, we can count the total value of all but the last I_k clique and we see that the algorithm has assigned to them total size at least $(c_k - 1) \cdot \frac{k}{4} = m_k \cdot \frac{1}{4} - O(k)$.

Cliques for I_2. (Recall that there is no $k = 1$, so this is the last case to consider.) Given the lower limit of the interval I_2, each I_2 clique (which being a 2-clique is just an edge) has a demand of size at least $\min\{\frac{1}{2}, \frac{B}{3}\}$ assigned to it. So the algorithm assigns total size at least $m_2 \cdot \min\{\frac{1}{4}, \frac{B}{6}\}$ to these I_2 cliques, where m_2 is the number of nodes in these cliques.

Total value of Alg. Since we assumed that the algorithm stops at some point, it means that it could not open more cliques. This means that all but at most $L-1$ nodes belong to one such clique (the worst case is that it tried to open an L-clique but could not), so $m_{\geq L} + \sum_{k=3}^{L-1} m_k + m_2 \geq m - L$. Then adding the above estimates for the values obtained on each type of clique, we see that the algorithm gets total value at least

$$\begin{aligned}\mathsf{Alg} &\geq \frac{1}{4}\left(1 - \frac{1}{L}\right) \cdot \left(m - m_2 - L\right) - O(L^2) + m_2 \cdot \min\left\{\frac{1}{4}, \frac{B}{6}\right\} \\ &= \frac{1}{4} \cdot (m - m_2) + m_2 \cdot \min\left\{\frac{1}{4}, \frac{B}{6}\right\} - O(m^{2/3})\end{aligned}$$

where the last inequality uses the fact that $L = m^{1/3}$.

If the minimum in the last line is $\frac{1}{4}$, then we obtain $\mathsf{Alg} \geq \left(\frac{1}{2} - O(\frac{1}{m^{1/3}})\right)\mathsf{OPT}$ as desired (recall $\mathsf{OPT} \leq \frac{m}{2}$ since each machine has Nominal capacity 1 and each demand is assigned to two machines). **So assume this is not the case, namely $B < \frac{3}{2}$.** Under this assumption

$$\mathsf{Alg} \overset{with\ ass.}{\geq} \frac{1}{4} \cdot (m - m_2) + \frac{B}{6} \cdot m_2 - O(m^{2/3}) \tag{2}$$

Value of OPT. We analyze OPT again under the assumption $B < \frac{3}{2}$. The Failover constraints also ensure that in order to accommodate the demand from I_2 in case of failure, any node that receives a demand from I_2 can have total size assigned to it a most $B - \min\{\frac{1}{2}, \frac{B}{3}\} \overset{with\ ass.}{=} \frac{2B}{3}$, due to the assumption $B < \frac{3}{2}$. For all other nodes, OPT can assign at most size 1 per node due to the Nominal capacity constraint. Let m_2^{OPT} be the number of nodes where OPT schedules a demand from I_2. Again, since the size of each demand is counted towards the Nominal capacity of two nodes, the total size scheduled by OPT is

$$\mathsf{OPT} \leq \frac{1}{2}\left(m_2^{\mathsf{OPT}} \cdot \frac{2B}{3} + (m - m_2^{\mathsf{OPT}}) \cdot 1\right) = \frac{1}{2} \cdot (m - m_2^{\mathsf{OPT}}) + \frac{B}{3} \cdot m_2^{\mathsf{OPT}} \tag{3}$$

Notice that since every demand in I_2 has size $> \min\{\frac{1}{2}, \frac{B}{3}\} \geq \frac{1}{3}$, the Failover constraints ensure that in OPT (as well as in our algorithm) the demands from I_2 that are scheduled form a matching, i.e. no 2 such demands can share a node/machine. So m_2^{OPT} (resp. m_2) is

just twice the number of I_2 demands scheduled by OPT (resp. our algorithm). Moreover, both Alg and OPT schedule a prefix of the instance. Since OPT gets at least as much value as Alg, it means that it scheduled a prefix that is at least as long; in particular it schedules at least as many I_2 demands as our algorithm. Together these observations imply that that $m_2^{\mathsf{OPT}} \geq m_2$. Then given inequalities (2) and (3), under the assumption $B < \frac{3}{2}$ we obtain that $\mathsf{Alg} \geq \left(\frac{1}{2} - O(\frac{1}{m^{1/3}})\right)\mathsf{OPT}$ as desired. This concludes the proof of Theorem 3.

3 Sublinear Additive Regret in the Stochastic Model

We now consider FAILOVER in the online stochastic model, where, instead of being adversarial, the size S_t of each demand now comes independently from an unknown distribution μ over $[0, \min\{1, \frac{B}{2}\}]$. Again, at time t the algorithm observes the size S_t of the current demand and irrevocably assigns it to two of the m machines. We still use $\mathsf{OPT} = \mathsf{OPT}(S_1, \ldots, S_n)$ to denote the value of (sum of the sizes scheduled by) the optimal strategy, which is now a random quantity. Our main result is algorithm FailoverStochastic that achieves a sublinear additive loss compared to OPT in this model (restated from the introduction for convenience).

▶ **Theorem 4.** *For the* FAILOVER *problem in the stochastic arrival model, there is a poly-time algorithm that achieves utilization* $\mathsf{OPT} - O(\mathsf{OPT}^{5/6} \log \mathsf{OPT})$ *with probability* $1 - O(\frac{1}{m})$.

The algorithm relies on a learn-and-pack approach that uses previously seen items to compute a *template* for packing the next items. This process is performed in rounds. Each round starts by assigning the first demand of the round on a pair of (empty) machines. Then, we iteratively create a template based on the first $n_k := 2^k$ items of the round, which we use to schedule the next n_k items. When the number of machines needed for the template (along with some slack) exceeds the number of available machines, the current round terminates and the next round begins. The next round maintains no knowledge of the previous demands; it only takes as input the number of empty machines $\tilde{m}$ which it is allowed to use.

Before describing the algorithm in more detail, an important question that arises is how to use the templates to schedule the future demands. A crucial component in this process are *monotone matchings*, which only match two values if the second is at least as big as the first.

▶ **Definition 5** (Monotone matching). *Given two sequences* $x_1, \ldots, x_n \in \mathbb{R}$ *and* $y_1, \ldots, y_n \in \mathbb{R}$, *a* monotone matching π *from the* x_t*'s to the* y_t*'s is an injective function from a subset* $I \in \{1, \ldots, n\}$ *to* $\{1, \ldots, n\}$ *such that* $x_i \leq y_{\pi(i)}$ *for all* $i \in I$. *We say that* x_i *is* matched *to* $y_{\pi(i)}$ *if* $i \in I$, *and* x_i *is* unmatched *otherwise.*

Monotone matchings will allow us to match future demands (x_i's) to the demands that are part of a template ($y_{\pi(i)}$'s) and put the former in the place of the latter (since $x_t \leq y_{\pi(i)}$). A surprising result of Rhee and Talagrand [16] is that if the two sequences are sampled i.i.d. from the same distribution, then almost all items can be matched, and moreover such a matching can be found online (see the paper for a more general result where the sequences may come form different distributions).

▶ **Theorem 6** (Monotone Matching Theorem [16]). *Suppose the random variables* $A_1, \ldots, A_n$ *and* $B_1, \ldots, B_n$ *are all sampled independently from a distribution* μ. *Then there is a constant cst such that with probability at least* $1 - e^{-cst \cdot \log^{3/2} n}$ *there is a monotone matching* π *of the* A_i*'s to the* B_i*'s where at most* $cst \cdot \sqrt{n} \log^{3/4} n$ *of the* A_i*'s are unmatched. Moreover, this matching can be computed even if the sequence* $A_1, \ldots, A_n$ *is revealed online.*

3.1 Algorithm

We are now ready to present the details of the FailoverStochastic algorithm.

FailoverStochastic. The algorithm just repeatedly calls the procedure OneRound below, passing to it the number of machines that are still available/unopened (e.g. initially it calls OneRound(m)); it does this for $\frac{\log m}{\log 4/3}$ rounds.

OneRound($\tilde{m}$). This procedure receives as input the number $\tilde{m}$ of machines that it is allowed to open. It is convenient to rename the demands and use Y_t to denote the t-th demand seen by OneRound (which are still sampled i.i.d. from μ). Similar to the work of Rhee and Talagrand [16], this algorithm works in phases: As mentioned earlier, each phase k sees the previous $n_k = 2^k$ items and creates a template based on them, which will then be used to schedule the next n_k items. To create this template, we define the offline problem OffMinFailover of minimizing the number of machines that are required to schedule these n_k items. To solve this problem, we design an approximation algorithm OffMinFailoverAlg achieving a sublinear approximation guarantee (more on this soon). Specifically, let $\overline{\mathsf{OPT}}_{mach}(x_1, \ldots, x_n)$ be the number of machines that OffMinFailoverAlg (with $\varepsilon = 1/n_k^{1/6}$) uses to schedule the demands $x_1, \ldots, x_n$. OneRound is then as follows:

Algorithm 2 OneRound**:** Given a number of available machines $\tilde{m}$.

1: Assign the first demand Y_1 to an empty edge by itself, opening 2 machines.

2: For phases $k = 0, 1, 2, \ldots$

(a) See the first n_k items $Y_1, \ldots, Y_{n_k}$. Run the algorithm OffMinFailoverAlg from Section 4 (with $\varepsilon = 1/n_k^{1/6}$) to find a solution for them that uses $\overline{\mathsf{OPT}}_{mach}(Y_1, \ldots, Y_{n_k})$ machines; let $templ(t)$ denote the pair of machines that Y_t is assigned to. This solution is our template.

(b) STOP if

$$\#\{\text{already open machines}\} + \underbrace{\overline{\mathsf{OPT}}_{mach}(Y_1, \ldots, Y_{n_k})}_{\text{machines from template}} + \underbrace{cst_1 \cdot \sqrt{n_k} \log^{3/4} n_k}_{\text{predicted unmatched demand}} + 2m^{5/6} > \tilde{m}.$$

(c) Else, open a clique of $\overline{\mathsf{OPT}}_{mach}(Y_1, \ldots, Y_{n_k})$ machines. Upon the arrival of each of the next n_k demands $Y_{n_k+1}, \ldots, Y_{2n_k}$, assign them to machines based on the template. More precisely, find the Rhee-Talagrand monotone matching π guaranteed by Theorem 6 from the new to the old demands (as the new ones arrive online). Schedule each matched new demand Y_t to the pair of machines that $Y_{\pi(t)}$ occupied in the template, namely the machine pair $templ(\pi(t))$. For each unmatched new demand, schedule it on an edge by itself (opening two more machines for each). If at any point the execution tries to open more than $\tilde{m}$ machines, declare FAIL.

3.2 Analysis

We next discuss the main ideas for the analysis of the algorithm FailoverStochastic, leading to the proof of Theorem 4. We assume throughout that m is at least a sufficiently large constant, else the success probability $1 - O(\frac{1}{m})$ trivially holds. Due to space constraints, we mostly state and discuss at a high-level the main components of the proof and show how they imply Theorem 4, deferring details to the full version of the paper.

First, we need to develop two important and complex components. Let $\mathsf{OPT}_{mach}(J)$ denote the minimum number of devices needed to assign all demands from a set of demands J, satisfying the Nominal and Failover constraints.

First component. The first component is the aforementioned algorithm OffMinFailoverAlg that is called within OneRound. It relies on a novel configuration LP, (LP_{mach}), and a post-processing algorithm to realize a rounded LP solution as a feasible assignment. It has the following guarantee:

▶ **Theorem 7.** *There exists a poly-time algorithm,* OffMinFailoverAlg*, that given $\varepsilon \in (0,1)$, finds a solution for* OffMinFailover *with at most $\big(1+O(\varepsilon)\big)\mathsf{LP}_{mach} + O(\frac{1}{\varepsilon^5}) \le \big(1+O(\varepsilon)\big)\mathsf{OPT}_{mach} + O(\frac{1}{\varepsilon^5})$ machines.*

Choosing ε appropriately, we can create a template using at most $\mathbb{E}\,\mathsf{OPT}_{mach}(Y_1,\ldots,Y_{n_k}) + o(n_k)$ devices in expectation for the next n_k arrivals. This result is proved in Section 4.

Second component. Recall from the technical overview that a worrisome aspect of FailoverStochastic is that each call to OneRound does not re-use machines from previous rounds. To show that this is not too wasteful, we prove that $\mathbb{E}\,\mathsf{OPT}_{mach}(X_1,\ldots,X_T)$ is approximately linear in T. We do so by giving a quantitative convergence theorem of $\mathbb{E}\,\mathsf{OPT}_{mach}(X_1,\ldots,X_T)$ to $T\cdot c(\mu)$, where $c(\mu)$ is a constant that characterizes the "average number of devices needed per demand." Furthermore, we use the bounded-differences inequality [3] to show that the number of machines $\mathsf{OPT}_{mach}(X_1,\ldots,X_T)$ is concentrated around this mean. That is, in the full version of the paper we show the following:

▶ **Theorem 8.** *Let μ be a distribution supported on $[0,\min\{1,\frac{B}{2}\}]$. Then there exists a scalar $c(\mu)$ such that for every $T\in\mathbb{N}$ and $\lambda>0$, we have*

$$\mathsf{OPT}_{mach}(X_1,\ldots,X_T) \;\in\; T\cdot c(\mu) \pm O(T^{5/6}) \pm \lambda\sqrt{T}$$

with probability at least $1-2e^{-\frac{\lambda^2}{2}}$, where $X_1,\ldots,X_T$ are i.i.d. samples from μ.

Thus splitting the first $2n_k$ demands into two rounds of n_k demands each costs us only an extra $o(n_k)$ devices.

With those two results in hand, the core of the analysis is that OneRound gets good value density, i.e., the ratio of value over number of machines m. We use $\mathbb{E}S_0$ to denote the expected value of the size of a demand (which is the same as $\mathbb{E}S_t$ for any t).

Specifically, according to Theorem 8, there is a scalar $c(\mu)$ such that OPT is able to fit roughly $\frac{1}{c(\mu)}$ demands per machine. Each such demand gives value roughly $\mathbb{E}S_0$; so the intuition is that the best possible density value/machine should be around $\frac{\mathbb{E}S_0}{c(\mu)}$. We first make this formal in the next lemma.

▶ **Lemma 9.** *With probability at least $1-\frac{2}{m^2}$ we have*

$$\mathsf{OPT} \le m\cdot\frac{\mathbb{E}S_0}{c(\mu)} + O(m^{5/6}) \quad \textit{and} \quad \mathsf{OPT} \ge \min\left\{ n\cdot\mathbb{E}S_0 - \sqrt{n\log m}\;,\; m\cdot\frac{\mathbb{E}S_0}{c(\mu)} - O(m^{5/6}) \right\}.$$

Crucially, the next lemma says that OneRound almost achieves this density.

▶ **Lemma 10.** *Let Open be the number of machines opened by* OneRound$(\tilde{m})$ *(which is a random variable). Then with probability at least $1-\frac{1}{m^2}$, the total value of the demands scheduled by* OneRound$(\tilde{m})$ *is at least*

$$\textit{value of } \mathsf{OneRound}(\tilde{m}) \ge \frac{\mathbb{E}S_0}{c(\mu)}\cdot \mathit{Open} - O(m^{5/6}).$$

Given this lemma, we see that the total value of the FailoverStochastic algorithm (which repeatedly calls OneRound) is approximately $\frac{\mathbb{E}S_0}{c(\mu)}$ times the total machines opened during the execution. By showing that the number of machines FailoverStochastic opens is $\approx m$, we then almost match the upper bound on OPT from Lemma 9.

▶ **Lemma 11.** *There is a constant cst_5 such that with probability $1-O(\frac{1}{m})$, FailoverStochastic opens at least $m - 5cst_5 \cdot m^{5/6}$ machines.*

These lemmas quickly lead to the proof of Theorem 4.

Proof of Theorem 4. Let $L := \frac{\log m}{\log 4/3}$ denote the number of calls to OneRound that FailoverStochastic makes, and let val_i and $Open_i$ be the value obtained and number of machines opened by the i-th call. Employing Lemma 10 on these L calls, we have that with probability at least $1 - \frac{L}{m^2}$ the total value of FailoverStochastic is

$$\text{algo value} \;=\; val_1 + \ldots + val_L \;\geq\; \frac{\mathbb{E}S_0}{c(\mu)} \cdot \sum_{i \leq L} Open_i \;-\; O(m^{5/6} \log m).$$

Moreover, from Lemma 11, with probability at least $1 - O(\frac{1}{m})$ the total number of machines open $\sum_{i \leq L} Open_i$ is at least $m - 5cst_5 \cdot m^{5/6}$, in which case we get

$$\text{algo value} \;\geq\; m \cdot \frac{\mathbb{E}S_0}{c(\mu)} \;-\; O(m^{5/6} \log m). \tag{4}$$

Furthermore, from Lemma 9 we have that $\mathsf{OPT} \leq m \cdot \frac{\mathbb{E}S_0}{c(\mu)} + O(m^{5/6})$ with probability at least $1 - \frac{2}{m^2}$. So by taking a union bound and combining this with the above lower bound on the algorithm's value, we get that with probability $1 - O(\frac{1}{m})$

$$\text{algo value} \;\geq\; \mathsf{OPT} \;-\; O(m^{5/6} \log m).$$

Since (4) also implies that $\mathsf{OPT} \geq \Omega(m)$, the previous bound is at least $\mathsf{OPT} - O(\mathsf{OPT}^{5/6} \log \mathsf{OPT})$. This concludes the proof of Theorem 4. ◀

We conclude this section by proving the lower bound on the value density of OneRound from Lemma 10. We defer the proofs of Lemma 9 and 11 to the full version of the paper.

3.2.1 Proof of Lemma 10

First, we control in high-probability the number of phases that OneRound($\tilde{m}$) executes before stopping or failing; this is important to avoid dependencies on the total number of demands n in the instance, which can be arbitrarily bigger than the scale of the effective instance.

▷ Claim 12. With probability $1 - \frac{1}{m^3}$, the number of phases within OneRound is at most

$$\bar{k} := \log\left(\frac{\tilde{m}}{c(\mu)} + O(\tilde{m}^{5/6}) + 3\log^{\frac{3}{2}} m\right). \tag{5}$$

Proof. Recall that the demand sizes $Y_1, Y_2, \ldots$ that OneRound sees are still i.i.d. samples from the original distribution μ. Using Theorem 8, it is not hard to show that with probability at least $1 - \frac{1}{m^3}$ OneRound can schedule at most $\frac{\tilde{m}}{c(\mu)} + O(\tilde{m}^{5/6}) + 3\log^{\frac{3}{2}} m$ many of these demands (for intuition, Theorem 8 indicates that even OPT requires more than $\tilde{m}$ machines to schedule these many demands). Since this quantity is exactly $n_{\bar{k}}$, OneRound cannot complete phase $\bar{k}$ (there are $2n_{\bar{k}}$ demands by the end of it) and the claim holds. ◁

Next, we need to bound how many machines are opened by OneRound, which in particular affects the probability of it failing. For a phase k, let $M_k := \overline{\mathsf{OPT}}_{mach}(Y_1, \ldots, Y_{n_k})$ denote the number of machines in the template solution, and let U_k be the number of additional machines that had to be open to accommodate the unmatched demands among $Y_{n_k+1}, \ldots, Y_{2n_k}$, namely twice the number of unmatched items. Notice that these quantities are well defined even for phases that the algorithm did not execute. The quantity $M_k + U_k$ is then the number machines that the algorithm OneRound opens in phase k (if it executes it). We have the following bounds for the number of machines open, at least for a phase k where the number of items n_k is sufficiently large (but still sublinear in m).

▷ Claim 13. Let $k_0 := (\frac{2}{cst_2} \log m)^{2/3}$ for a sufficiently small constant cst_2. Then there is a constant cst_1 such that:

1. For $k \geq k_0$, we have $M_k \in n_k \cdot c(\mu) \pm cst_1 \cdot n_k^{5/6}$ with probability $\geq 1 - \frac{1}{m^3}$
2. For $k \geq k_0$, we have $U_k \leq cst_1 \cdot \sqrt{n_k} \log^{3/4} n_k$ with probability $\geq 1 - \frac{1}{m^3}$
3. $n_{k_0} \leq m^{5/6}$.

Proof. Consider a phase $k \geq k_0$. Since the demand sizes $Y_1, \ldots, Y_{2n_k}$ seen in this phase are i.i.d. samples from the original distribution μ, we can bound the minimum number of machines $\mathsf{OPT}_{mach}(Y_1, \ldots, Y_{n_k})$ (using Theorem 8 with $\lambda = n_k^{1/3}$) as

$$\mathsf{OPT}_{mach}(Y_1, \ldots, Y_{n_k}) \in n_k \cdot c(\mu) \pm O(n_k^{5/6})$$

with probability at least $1 - 2e^{-\frac{n_k^{2/3}}{2}}$. Moreover, employing the guarantee of the algorithm OffMinFailoverAlg used to build the template (Theorem 7 with $\varepsilon = 1/n_k^{1/6}$), we get that M_k is in the range $n_k \cdot c(\mu) \pm cst_1 \cdot n_k^{5/6}$ with probability at least $1 - 2\exp(-\frac{n_k^{2/3}}{2})$ for some constant cst_1. But since $n_k = 2^k \geq 2^{k_0}$, a quick calculation shows that this probability is at least $1 - \frac{1}{m^3}$, proving the first item of the claim.

To control U_k, we can use the Monotone Matching Theorem (Theorem 6) with the first sequence of sizes being the demands from the template, i.e., $(B_1, \ldots, B_{n_k}) = (Y_1, \ldots, Y_{n_k})$, and the second one being the demands that we attempted to match to them, namely $(A_1, \ldots, A_{n_k}) = (Y_{n_k+1}, \ldots, Y_{2n_k})$ to obtain that the number of unmatched demands is at most $cst \cdot \sqrt{n_k} \log^{3/4} n_k$ with probability at least $1 - e^{-cst \cdot \log^{3/2} n_k}$, and hence with this probability $U_k \leq 2cst \cdot \sqrt{n_k} \log^{3/4} n_k$. Again because $k \geq k_0$, we get that this probability is at least $1 - \frac{1}{m^3}$, proving Item 2 of the claim (by taking $cst_1 \geq 2cst$ we can just replace the latter by the former).

The last item $n_{k_0} \leq m^{5/6}$ of the claim can be directly verified using the fact that we assumed m is at least a sufficiently large constant. ◁

Recall that OneRound only fails when the number of machines $M_k + U_k$ actually opened in a phase is bigger than it "predicted" in Line 2.b, and this prediction is exactly M_k plus the upper bound U_k from Claim 13 plus a slack of $2m^{5/6}$. By considering all phases, it is now easy to upper bound the probability that OneRound fails ($\bar{k}$ is defined in (5)).

▷ Claim 14. The probability that OneRound fails is at most $\frac{\bar{k}+1}{m^3}$.

Proof. Fix any phase k, and we claim that the probability that OneRound fails on this phase is at most $\frac{1}{m^3}$. If OneRound fails on phase k, then it did not STOP in Line 2.b, so

$$\#\,[\text{machines open before phase } k] + \overline{\mathsf{OPT}}_{mach}(Y_1, \ldots, Y_{n_k}) + cst_1 \cdot \sqrt{n_k} \log^{3/4} n_k + 2m^{5/6} \leq \tilde{m},$$

but it ran out of machines during phase k, namely

$$\#\,[\text{machines open before phase } k] + (M_k + U_k) > \tilde{m}.$$

Since $M_k = \overline{\mathsf{OPT}}_{mach}(Y_1, \ldots, Y_{n_k})$, these observations imply that $U_k > cst_1 \cdot \sqrt{n_k}\, \log^{3/4} n_k + 2m^{5/6}$. This is impossible if $n_k \leq m^{5/6}$, because the number of machines U_k opened for the unmatched demands is at most twice the number n_k of demands considered for the matching. So we must have $n_k > m^{5/6}$ (and so from Claim 13 $k \geq k_0$) and at least $U_k > cst_1 \cdot \sqrt{n_k}\, \log^{3/4} n_k$; but again by Claim 13 the latter happens with probability at most $\frac{1}{m^3}$. Thus, the probability that OneRound fails on phase k is at most $\frac{1}{m^3}$.

Moreover, by Claim 12, with probability at least $1 - \frac{1}{m^3}$ OneRound has at most $\bar{k}$ phases. Then taking a union bound, we see that the event that OneRound has at most $\bar{k}$ phases and in all of them it does not fail holds with probability at least $1 - \frac{\bar{k}+1}{m^3}$; in particular, with at least this much probability the algorithm does not fail in its execution, proving the claim. ◁

We now finally lower bound the value that OneRound gets. Let τ be the (random) index of the last phase attempted by OneRound, namely where Line 2.c is executed. As long as it does not fail on the last phase τ (which by the previous claim happens with probability at least $1 - \frac{\bar{k}+1}{m^3}$) OneRound gets the value of all items up until this phase, that is

$$\text{value of OneRound} \geq Y_1 + \ldots + Y_{2n_\tau} \geq Y_1 + \ldots + Y_{2n_{\min\{\tau,\bar{k}\}}}. \tag{6}$$

Recall that the Y_i's are independent and each has mean $\mathbb{E}S_0$. Then employing the Chernoff bound (Theorem 2.8 of [3]), for any fixed $t \leq n_{\bar{k}}$ we have that

$$Y_1 + \ldots + Y_t \geq t \cdot \mathbb{E}S_0 - \sqrt{n_{\bar{k}} \log(m^3 \cdot n_{\bar{k}})} \qquad \text{with probability at least } 1 - \frac{1}{m^3 \cdot n_{\bar{k}}}.$$

Then taking a union bound over (6), the previous displayed inequality for all $t \leq n_{\bar{k}}$, and over the event that OneRound has at most $\bar{k}$ phases (which holds with probability at least $1 - \frac{1}{m^3}$) we get that

$$\begin{aligned} \text{value of OneRound} &\geq 2n_{\min\{\tau,\bar{k}\}} \cdot \mathbb{E}S_0 - \sqrt{n_{\bar{k}} \log(m^3 \cdot n_{\bar{k}})} \\ &= 2n_\tau \cdot \mathbb{E}S_0 - \sqrt{n_{\bar{k}} \log(m^3 \cdot n_{\bar{k}})} \\ &\geq 2n_\tau \cdot \mathbb{E}S_0 - O(m^{5/6}) \qquad \text{with probability} \geq 1 - \frac{\bar{k}+3}{m^3}. \end{aligned} \tag{7}$$

To conclude the proof of Lemma 10 we just need to relate this quantity to the number of machines opened by OneRound. Let $Open_\ell$ be the number of machines opened until (including) phase ℓ, and recall that $Open$ is the number of machines opened over all phases. Since the number of machines opened on phase k is $M_k + U_k$ (plus two machines for the first demand Y_1), we have

$$Open_\ell = 2 + (M_1 + U_1) + \ldots + (M_\ell + U_\ell) \tag{8}$$

To upper bound the right-hand side, for the phases $k < k_0$ we just use the fact that $M_k + U_k \leq 2n_k + 2n_k = 4n_k$, since both in the template and for the unmatched demands we never open more than 2 machines per demand considered (and n_k demands are considered in each part). For each phase $k = k_0, \ldots, \bar{k}$ we can use Claim 13 to upper bound $M_k + U_k$ with probability at least $1 - \frac{2}{m^3}$ by

$$M_k + U_k \leq n_k \cdot c(\mu) + cst_1 \cdot n_k^{5/6} + cst_1 \cdot \sqrt{n_k}\, \log^{3/4} n_k \leq n_k \cdot c(\mu) + cst_3 \cdot n_k^{5/6}$$

for some constant cst_3. Together these bounds give that with probability at least $1 - \frac{2\ell}{m^3}$

$$Open_\ell \leq 2 + \sum_{k<k_0} 4n_k + \sum_{k=k_0}^{\ell} \left(n_k \cdot c(\mu) + cst_3 \cdot n_k^{5/6} \right).$$

To further upper bound the first summation on the right-hand side, because of the exponential relationship $n_k = 2^k$, we have $\sum_{k<k_0} 4n_k \leq 8n_{k_0-1} \leq O(m^{5/6})$, the last inequality coming from Claim 13; for the second summation, we analogously have $\sum_{k=k_0}^{\ell} n_k \leq 2n_\ell$ and $\sum_{k=k_0}^{\ell} n_k^{5/6} \leq O(n_\ell^{5/6})$. Therefore,

$$Open_\ell \leq 2n_\ell \cdot c(\mu) + O(n_\ell^{5/6}) + O(m^{5/6}) \qquad \text{with probability at least } 1 - \frac{2\ell}{m^3}. \tag{9}$$

Finally, since by Claim 12 the number of phases τ performed by OneRound is at most $\bar{k}$ with probability at least $1 - \frac{1}{m^3}$, the total number of machines open can be upper bounded

$$Open \leq Open_{\min\{\tau,\bar{k}\}} \leq 2n_\tau \cdot c(\mu) + O(n_{\bar{k}}^{5/6}) + O(m^{5/6}) \leq 2n_\tau \cdot c(\mu) + O(m^{5/6})$$

with probability at least $1 - \frac{2\bar{k}+1}{m^3}$.

Finally, taking a union bound to combine this inequality with (7), we get that

$$\text{value of OneRound} \geq \frac{\mathbb{E}S_0}{c(\mu)} \cdot Open - O(m^{5/6})$$

with probability at least $1 - \frac{3\bar{k}+4}{m^3}$. Since m is at least a sufficiently large constant, we have $m \geq 3\bar{k} + 4$, and the bound from the displayed inequality holds with probability at least $1 - \frac{1}{m^2}$. This finally concludes the proof of Lemma 10.

4 Offline Machine Minimization

In this section we consider the aforementioned (offline) minimization version of FAILOVER, which we call OFFMINFAILOVER: Given a failover capacity $B \geq 1$ and a collection of demands such that demand j has size $s_j \in [0, \min\{1, \frac{B}{2}\}]$, we need to assign *all demands* to pairs of machines while satisfying the Nominal and Failover constraints, and the goal is to minimize the number of machines used. As before, we use $\mathsf{OPT}_{mach} = \mathsf{OPT}_{mach}(s_1, \ldots, s_n)$ to denote the cost of (i.e. number of machines in) the optimal solution.

The main result of this section (Theorem 7, restated) is an efficient algorithm with a sublinear additive regret for this problem (when ε is set appropriately). We remark that a sublinear regret (compared to, say, a constant approximation) is necessary due to its use in Section 3. In fact, the algorithm compares against the stronger optimum of an LP relaxation for the problem (denoted by (LP_{mach}), and defined below). We let LP_{mach} denote the optimal value of this LP.

▶ **Theorem 7.** *There exists a poly-time algorithm,* OffMinFailoverAlg*, that given $\varepsilon \in (0,1)$, finds a solution for* OFFMINFAILOVER *with at most $\big(1 + O(\varepsilon)\big)\mathsf{LP}_{mach} + O(\frac{1}{\varepsilon^5}) \leq \big(1 + O(\varepsilon)\big)\mathsf{OPT}_{mach} + O(\frac{1}{\varepsilon^5})$ machines.*

As hinted above, our algorithm is based on converting a solution of a configuration LP into a good assignment of demands to pairs of machines. But crucially, while the configuration of each machine controls the total size of demands serviced by it, it has no information how these demands are distributed over the "edges" incident to the machine, which is important for adequately handling the Failover constraints. The post-processing of the LP solution is the one in charge of creating a feasible (and low-cost) assignment from this limited control offered by the LP.

4.1 Configuration LP

Consider an assignment of the demands into some number of machines. We can view the collection of demands assigned to (the edges incident to) a given machine as a configuration. Precisely, we define a *configuration* C to be a subset of the demands such that $\sum_{s\in C} s \leq 1$ and $\sum_{s\in C} s + \max_{s\in C} s \leq B$. Note that the first constraint is exactly the Nominal constraint, while the second is a relaxation of the Failover constraint, because the most-loaded edge incident on some machine can be larger than the single largest demand assigned to that machine. Thus, our notion of configuration does not take in to account how the demands are assigned to the respective edges incident on each machine.

To define our configuration LP, we suppose the input collection of demands is partitioned into T *demand types* such that type t consists of n_t-many demands each with size s_t. Thus each configuration C can be represented by a number $n_t(C) \in \mathbb{N}$ of demands for each type t such that $\sum_t n_t(C)\cdot s_t \leq 1$ and $\sum_t n_t(C)\cdot s_t + \max_{t|\, n_t(C)>0} s_t \leq B$. We are ready to define our configuration LP:

$$\begin{array}{lrll} \min & \sum_C x_C & & \\ s.t. & \sum_C n_t(C)\cdot x_C & \geq 2n_t & \forall t \\ & x & \geq 0 & \end{array} \qquad (\mathsf{LP}_{mach})$$

Note that the definition of (LP_{mach}) depends on how the demands are partitioned into types. We show in the full version of the paper that the optimal value of (LP_{mach}) does not depend on the particular type partition. Thus, throughout the analysis, we will use whichever type partition is convenient (unless a particular one is specified).

It is immediate that (LP_{mach}) is a relaxation of OffMinFailover by taking the natural setting of the x-variables defined by a feasible assignment to machines: just let x_C be the number of machines whose collection of demand sizes assigned to its edges are exactly those in C. In particular, we have that $\mathsf{LP}_{mach} \leq \mathsf{OPT}_{mach}$.

Although (LP_{mach}) has exponentially many variables in general, we can approximately solve it via column generation similar to the standard bin packing configuration LP [10, 17] (proof in the full version of the paper).

▶ **Lemma 15.** *We can find in poly-time an extreme point solution of* (LP_{mach}) *with objective value at most* $\mathsf{LP}_{mach} + 1$.

Further, observe that (LP_{mach}) only has T non-trivial constraints, so by the standard rank argument (see for example Lemma 2.1.3 of [12]) any extreme point solution of (LP_{mach}) has at most T non-zero variables. Thus, the next lemma follows immediately by rounding up all the fractional variable of an extreme point solution.

▶ **Lemma 16.** *Given an extreme point of* (LP_{mach}) *with objective value Val, rounding up all fractional variables to the next largest integer gives an integral solution to* (LP_{mach}) *with objective value at most* $Val + T$.

To summarize this section, we can efficiently obtain a collection of configurations, each corresponding to a machine, that "covers" all the demands. However, these configurations do not specify how to actually assign the demands to the edges incident on the corresponding machine. This is the goal of the next section.

4.2 Matching configurations

We say that a collection $\mathcal{C}$ of configurations is feasible if it comes from an integer solution for (LP_{mach}), i.e. setting x_C to be the number of times C appears in $\mathcal{C}$ gives a feasible solution for (LP_{mach}). Our goal in this section is to realize such collection by actually assigning demands to edges. The main challenge is satisfying the actual Failover constraints.

For simplicity assume $\sum_{C \in \mathcal{C}} n_t(C) = 2n_t$ for all types t, i.e. each demand appears on exactly 2 configurations (drop from the configurations what is extra). We can think of $\mathcal{C}$ (with, say, N configurations) as a graph on N nodes/machines, where node/machine $C \in \mathcal{C}$ has $n_t(C)$ "slots" for demands of type t. While this gives the right number of slots $2n_t$ to accommodate the demands of each type t, we still need to specify to which *edge* (pair of machines) each of the n_t demands of type t is assigned in a way that satisfies the Nominal and Failover constraints. (We can alternatively see this as a graph realization problem: each node C as having a requirement $n_t(C)$ of "edges of type t" (which we call its t-*degree*) and we want to create edges of different types (i.e., assignment of demands to pairs of nodes) to satisfy these requirements while also satisfying the Nominal and Failover constraints.)

To see the challenge, consider a fixed node/configuration C. Regardless of how we assign demands to edges (as long as it is consistent with the slots of the configurations), the Nominal constraint of C is satisfied: it will receive total size $\sum_t n_t(C) \cdot s_t = \sum_{s \in C} s$, which is at most 1 by definition of a configuration. This is not the case for the Failover constraint. This is again because the definition of configuration only gives us the relaxed version of the Failover constraint $\sum_{s \in C} s + \max_{s \in C} s \leq B$, In particular, the blue term only considers the largest demand assigned to machine C instead of the most-loaded edge incident to C. However, these two quantities are the same *if we are able to assign at most one demand per edge*. (In the graph realization perspective, it means that it suffices to construct a *simple* graph with the desired t-degrees.) But it is not clear that such an assignment should even exist, let alone be found efficiently.

The main result of this section is that – by opening slightly more machines – we can find such an assignment that realizes any given collection of configurations satisfying both the Nominal and Failover constraints.

▶ **Theorem 17.** *Consider an instance of* OffMinFailover *with T demand types. Given a collection $\mathcal{C}$ of N configurations that is feasible for (LP_{mach}), we can find in poly-time a feasible solution for* OffMinFailover *that uses at most $N + O(DT)$ machines, where D is the maximum number of demands in any configuration in $\mathcal{C}$.*

For that, we will need the following subroutine to assign some demands outside of their respective configurations. This result easily follows by opening disjoint edges as needed, and assigning demands arbitrarily to an already-opened edge is possible.

▶ **Lemma 18.** *There is a poly-time algorithm for* OffMinFailover *that uses at most $8 \cdot S + 2$ machines, where S is the sum of the size of the demands in the instance.*

The algorithm guaranteed by Theorem 17 is the following. In order to simplify the notation, as before we assume without loss of generality that $\mathcal{C}$ has $\sum_{C \in \mathcal{C}} n_t(C) = 2n_t$ for all types t.

Proof of Theorem 17. It is clear that MatchConfigs runs in polynomial time, and assigns all demands to edges. Further, this assignment satisfies both the Nominal and Failover constraints, because we assign at most one demand per edge in Step 4 (see discussion in the beginning of this section), and Step 5 guarantees a feasible assignment for the remaining demands.

It remains to show that it opens $N + O(DT)$ machines. In particular, by Lemma 18 it suffices to show that the total size of all unassigned demands that reach Step 5 is $O(DT)$. When considering demand type t, there are two possibilities:

Algorithm 3 MatchConfigs: Given a collection $\mathcal{C}$ of N configurations.

1: Open N machines – one corresponding to each configuration in $\mathcal{C}$.

2: Consider demand types in arbitrary order $t = 1, \dots, T$.

3: When considering demand type t, partition the collection $\mathcal{C}$ into two collections $\mathcal{L}_t$ and $\mathcal{R}_t$ such that their total t-degrees $\sum_{C\in\mathcal{L}_t} n_t(C)$ and $\sum_{C\in\mathcal{R}_t} n_t(C)$ differ by at most D_t, where D_t is the maximum number of type t demands in any configuration. (This can be achieved, e.g., by initializing $\mathcal{L}_t, \mathcal{R}_t = \emptyset$, and adding configurations one-by-one to the set with minimum total t-degree.)

4: Given this partition, as long as there exists a configuration $C \in \mathcal{L}_t$ that is currently assigned less than $n_t(C)$ demands of type t, we pick such a configuration and assign a demand of type t to an arbitrary edge (C, C') (for $C' \in \mathcal{R}_t$) that has not yet been assigned a demand of any type and such that C' is currently assigned less than $n_t(C')$ demands of type t. If no such edge exists, then we stop and move on to the next demand type.

5: Once we are done considering all demand types, assign all the currently unassigned demands to new machines using Lemma 18.

Case 1: Step 4 assigns $n_t(C)$ type t demands to each $C \in \mathcal{L}_t$. In this case it assigns $\sum_{C\in\mathcal{L}_t} n_t(C)$ type t demands to edges between $\mathcal{L}_t$ and $\mathcal{R}_t$, while the total number of type t demands is

$$n_t = \frac{1}{2}\left(\sum_{C\in\mathcal{L}_t} n_t(C) + \sum_{C\in\mathcal{R}_t} n_t(C)\right) \leq \sum_{C\in\mathcal{L}_t} n_t(C) + \frac{D_t}{2}, \tag{10}$$

where the inequality uses the fact that the t-degree of $\mathcal{R}_t$ is at most that of $\mathcal{L}_t$ plus D_t. Thus, at most $\frac{D_t}{2}$ demands of type t remain unassigned and reach Step 5. The total size of these demands it at most $\frac{1}{2}$, since D_t demands of type t are in a valid configuration. Hence the total size of the unassigned demands of all types is at most $\frac{T}{2} \leq O(DT)$.

Case 2: Step 4 fails to assign $n_t(\bar{C})$ to a configuration $\bar{C} \in \mathcal{L}_t$. In this case, for each $C' \in \mathcal{R}_t$, either the edge $(\bar{C}, C')$ is already assigned some demand (call such C' *blocked*) or C' has already been assigned $n_t(C')$ demands of type t. But there are at most D blocked C''s, since the configuration $\bar{C}$ has at most D slots to receive demands. Thus the total number of type-t demands assigned is at least

$$\sum_{C'\in\mathcal{R}_t\setminus\text{blocked}} n_t(C') \geq \sum_{C'\in\mathcal{R}_t} n_t(C') - D\cdot\max_{C'\in\text{blocked}} n_t(C') \geq \sum_{C'\in\mathcal{R}_t} n_t(C') - D\cdot D_t.$$

Moreover, exchanging the roles of $\mathcal{L}_t$ and $\mathcal{R}_t$ in the argument from (10) we get that $\sum_{C'\in\mathcal{R}_t} n_t(C') \geq n_t - \frac{D_t}{2}$, and thus at least $n_t - D\cdot D_t - \frac{D_t}{2}$ demands of type t are assigned by Step 4. Thus at most $O(D\cdot D_t)$ demands (hence total size $O(D)$) of this type remain unassigned and reach Step 5. This a total size of $O(DT)$, over all demand types, that reach the latter step, as desired. ◀

We summarize the main results of this section and the previous with the next theorem: By approximately solving (LP_{mach}) (Lemma 15), rounding the solution (Lemma 16), and using the above algorithm to obtain an assignment of demands to edges (Theorem 17), we obtain the following.

▶ **Theorem 19.** *Consider an instance of* OffMinFailover *that has most T demands types and where each configuration has at most D demands. Then there is a poly-time algorithm that finds a feasible solution that uses at most $\mathsf{LP}_{mach} + O(DT)$ machines.*

To obtain our main result, Theorem 7, we need to modify the input instance to make D and T small enough. In the full version of this paper – by losing a multiplicative $(1 + O(\epsilon))$-factor – we show how to ensure that $D, T = poly(\frac{1}{\epsilon})$ by rounding demand sizes and handling the small demands separately. This concludes the proof of Theorem 7.

References

1. Sara Ahmadian and Zachary Friggstad. Further approximations for demand matching: Matroid constraints and minor-closed graphs. In *44th International Colloquium on Automata, Languages, and Programming, ICALP*, 2017. doi:10.4230/LIPIcs.ICALP.2017.55.
2. János Balogh, József Békési, György Dósa, Leah Epstein, and Asaf Levin. A New and Improved Algorithm for Online Bin Packing. In *26th Annual European Symposium on Algorithms (ESA)*, 2018. doi:10.4230/LIPIcs.ESA.2018.5.
3. Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, 2013.
4. Chandra Chekuri and Sanjeev Khanna. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM Journal on Computing*, 35(3):713–728, 2005. doi:10.1137/S0097539700382820.
5. Janos Csirik, David S Johnson, Claire Kenyon, James B Orlin, Peter W Shor, and Richard R Weber. On the sum-of-squares algorithm for bin packing. *Journal of the ACM (JACM)*, 53(1):1–65, 2006.
6. Anupam Gupta and Jochen Könemann. Approximation algorithms for network design: A survey. *Surveys in Operations Research and Management Science*, 16(1):3–20, 2011. doi:10.1016/j.sorms.2010.06.001.
7. Anupam Gupta and Marco Molinaro. How the experts algorithm can help solve lps online. *Mathematics of Operations Research*, 41(4):1404–1431, 2016. doi:10.1287/moor.2016.0782.
8. Varun Gupta and Ana Radovanović. Interior-point-based online stochastic bin packing. *Operations Research*, 68(5):1474–1492, 2020.
9. Rebecca Hoberg and Thomas Rothvoss. A logarithmic additive integrality gap for bin packing. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, 2017.
10. Narendra Karmarkar and Richard M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*, pages 312–320, 1982. doi:10.1109/SFCS.1982.61.
11. Madhukar Korupolu, Adam Meyerson, Rajmohan Rajaraman, and Brian Tagiku. Coupled and k-sided placements: Generalizing generalized assignment. *Math. Program.*, 154(1–2):493–514, December 2015. doi:10.1007/s10107-015-0930-1.
12. Lap-Chi Lau, R. Ravi, and Mohit Singh. *Iterative Methods in Combinatorial Optimization*. Cambridge University Press, USA, 1st edition, 2011.
13. Shang Liu and Xiaocheng Li. Online bin packing with known T. *arXiv preprint arXiv:2112.03200*, 2021.
14. Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5):22–es, October 2007. doi:10.1145/1284320.1284321.
15. Wansoo T. Rhee and Michel Talagrand. Optimal bin packing with items of random sizes II. *SIAM Journal on Computing*, 18(1):139–151, 1989. doi:10.1137/0218009.
16. Wansoo T Rhee and Michel Talagrand. On-line bin packing of items of random sizes, II. *SIAM Journal on Computing*, 22(6):1251–1256, 1993.
17. Thomas Rothvoß. The entropy rounding method in approximation algorithms. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, 2012.
18. David B Shmoys and Éva Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical programming*, 62(1):461–474, 1993.
19. Chaojie Zhang, Alok Gautam Kumbhare, Ioannis Manousakis, Deli Zhang, Pulkit A Misra, Rod Assis, Kyle Woolcock, Nithish Mahalingam, Brijesh Warrier, David Gauthier, et al. Flex: High-availability datacenters with zero reserved power. In *ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021.

Faster Parameterized Algorithms for Modification Problems to Minor-Closed Classes

Laure Morelle
LIRMM, Université de Montpellier, CNRS, France

Ignasi Sau
LIRMM, Université de Montpellier, CNRS, France

Giannos Stamoulis
LIRMM, Université de Montpellier, CNRS, France

Dimitrios M. Thilikos
LIRMM, Université de Montpellier, CNRS, France

Abstract

Let $\mathcal{G}$ be a minor-closed graph class and let G be an n-vertex graph. We say that G is a *k-apex* of $\mathcal{G}$ if G contains a set S of at most k vertices such that $G \setminus S$ belongs to $\mathcal{G}$. Our first result is an algorithm that decides whether G is a k-apex of $\mathcal{G}$ in time $2^{\mathsf{poly}(k)} \cdot n^2$. This algorithm improves the previous one, given by Sau, Stamoulis, and Thilikos [ICALP 2020, TALG 2022], whose running time was $2^{\mathsf{poly}(k)} \cdot n^3$. The *elimination distance* of G to $\mathcal{G}$, denoted by $\mathsf{ed}_{\mathcal{G}}(G)$, is the minimum number of rounds required to reduce each connected component of G to a graph in $\mathcal{G}$ by removing one vertex from each connected component in each round. Bulian and Dawar [Algorithmica 2017] proved the existence of an FPT-algorithm, with parameter k, to decide whether $\mathsf{ed}_{\mathcal{G}}(G) \leq k$. This algorithm is based on the computability of the minor-obstructions and its dependence on k is not explicit. We extend the techniques used in the first algorithm to decide whether $\mathsf{ed}_{\mathcal{G}}(G) \leq k$ in time $2^{2^{2^{\mathsf{poly}(k)}}} \cdot n^2$. This is the first algorithm for this problem with an explicit parametric dependence in k. In the special case where $\mathcal{G}$ excludes some apex-graph as a minor, we give two alternative algorithms, one running in time $2^{2^{\mathcal{O}(k^2 \log k)}} \cdot n^2$ and one running in time $2^{\mathsf{poly}(k)} \cdot n^3$. As a stepping stone for these algorithms, we provide an algorithm that decides whether $\mathsf{ed}_{\mathcal{G}}(G) \leq k$ in time $2^{\mathcal{O}(\mathsf{tw} \cdot k + \mathsf{tw} \log \mathsf{tw})} \cdot n$, where tw is the treewidth of G. This algorithm combines the dynamic programming framework of Reidl, Rossmanith, Villaamil, and Sikdar [ICALP 2014] for the particular case where $\mathcal{G}$ contains only the empty graph (i.e., for treedepth) with the representative-based techniques introduced by Baste, Sau, and Thilikos [SODA 2020]. In all the algorithmic complexities above, poly is a polynomial function whose degree depends on $\mathcal{G}$, while the hidden constants also depend on $\mathcal{G}$. Finally, we provide explicit upper bounds on the size of the graphs in the minor-obstruction set of the class of graphs $\mathcal{E}_k(\mathcal{G}) = \{G \mid \mathsf{ed}_{\mathcal{G}}(G) \leq k\}$.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Graph minors, Parameterized algorithms, Graph modification problems, Vertex deletion, Elimination distance, Irrelevant vertex technique, Flat Wall Theorem, Obstructions

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.93

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://doi.org/10.48550/arXiv.2210.02167

Funding Supported by the ANR projects ELIT (ANR-20-CE48-0008-01), ESIGMA (ANR-17-CE23-0010), and the French-German Collaboration ANR/DFG Project UTMA (ANR-20-CE92-0027).

Acknowledgements We would like to thank the reviewers for helpful remarks that improved the presentation of the article.

1 Introduction

The *distance from triviality* is a concept formalized by Guo, Hüffner, and Niedermeier [24] to express the closeness of a graph to a supposedly "simple" target graph class. One such a measure of closeness is, for instance, the number of vertices or edges that one must delete/add from/to a graph G to obtain a graph in the target graph class. This concept of distance to a graph class has recently gained the interest of the parameterized complexity community. The motivation is that, if a problem is tractable on a graph class $\mathcal{G}$, it is natural to study other classes of graphs according to their "distance to $\mathcal{G}$". In this paper, we focus on two such measures of distance from triviality: Given a target graph class $\mathcal{G}$, we consider the *vertex deletion distance* to $\mathcal{G}$ and the *elimination distance* to $\mathcal{G}$, which we formalize next.

Given a target graph class $\mathcal{G}$ and a non-negative integer k, we define $\mathcal{A}_k(\mathcal{G})$ as the set of all graphs containing a set S of at most k vertices whose removal results in a graph in $\mathcal{G}$. If $G \in \mathcal{A}_k(\mathcal{G})$, then we say that G is a *k-apex* of $\mathcal{G}$. We refer to S as a *k-apex set of G for the class $\mathcal{G}$*. In other words, we consider the following meta-problem for a fixed class $\mathcal{G}$.

Vertex Deletion to $\mathcal{G}$
Input: A graph G and a non-negative integer k.
Objective: Find, if it exists, a k-apex set of G for the class $\mathcal{G}$.

Throughout the paper, we denote by n (resp. m) the number of vertices (resp. edges) of the input graph of the problem under consideration. The importance of Vertex Deletion to $\mathcal{G}$ can be illustrated by the variety of graph modification problems that it encompasses. For instance, if $\mathcal{G}$ is the class of edgeless (resp. acyclic, planar, bipartite, (proper) interval, chordal) graphs, then we obtain the Vertex Cover (resp. Feedback Vertex Set, Vertex Planarization, Odd Cycle Transversal, (proper) Interval Vertex Deletion, Chordal Vertex Deletion) problem.

The second measure of distance from triviality that we study was recently introduced by Bulian and Dawar [10, 11]. Given a graph class $\mathcal{G}$, we define the *elimination distance* of a graph G to $\mathcal{G}$, denoted by $\mathsf{ed}_{\mathcal{G}}(G)$, as follows:

$$\mathsf{ed}_{\mathcal{G}}(G) = \begin{cases} 0 & \text{if } G \in \mathcal{G}, \\ 1 + \min\{\mathsf{ed}_{\mathcal{G}}(G \setminus \{v\}) \mid v \in V(G)\} & \text{if } G \text{ is connected}, \\ \max\{\mathsf{ed}_{\mathcal{G}}(H) \mid H \text{ is a connected component of } G\} & \text{otherwise.} \end{cases}$$

Given that $\mathsf{ed}_{\mathcal{G}}(G) \leq k$, a set $S \subseteq V(G)$ of vertices recursively deleted from G to achieve $\mathsf{ed}_{\mathcal{G}}(G)$ is called a *k-elimination set of G for $\mathcal{G}$*. We define the (parameterized) class of graphs $\mathcal{E}_k(\mathcal{G}) = \{G \mid \mathsf{ed}_{\mathcal{G}}(G) \leq k\}$. The above notion can be seen as a natural generalization of *treedepth* (denoted by td), which corresponds to the case where $\mathcal{G}$ contains only the empty graph. Treedepth, along with treewidth, are two of the most studied and widely used parameters to measure the structural complexity of a graph [12, 36, 43]. The second meta-problem that we consider is the following, again for a fixed graph class $\mathcal{G}$.

Elimination Distance to $\mathcal{G}$
Input: A graph G and a non-negative integer k.
Objective: Find, if it exists, a k-elimination set of G for the class $\mathcal{G}$.

Unsurprisingly, Vertex Deletion to $\mathcal{G}$ is NP-hard for every non-trivial graph class $\mathcal{G}$ [41], while Elimination Distance to $\mathcal{G}$ is NP-hard even when $\mathcal{G}$ contains only the empty graph [45]. To circumvent this intractability, we study both problems from the parameterized

complexity point of view and consider their parameterizations by k. In this setting, the most desirable behavior is the existence of an algorithm running in time $f(k) \cdot n^{\mathcal{O}(1)}$, where f is a computable function depending only on k. Such an algorithm is called *fixed-parameter tractable*, or FPT-*algorithm* for short, and a parameterized problem admitting an FPT-algorithm is said to belong to the parameterized complexity class FPT. Also, the function f is called *parametric dependence* of the corresponding FPT-algorithm, and the challenge is to design FPT-algorithms with small parametric dependencies and with a polynomial factor of small degree [12, 14, 17, 44]. We may also consider XP-*algorithms*, i.e., algorithms running in time $f(k) \cdot n^{g(k)}$ for some computable functions f and g depending only on k.

In general, for any of the two considered problems, we cannot expect FPT-algorithms for every graph class $\mathcal{G}$. For instance, the two problems are NP-hard, even for $k = 0$, for every graph class $\mathcal{G}$ whose recognition problem is NP-hard. This is the case of 3-colorable graphs, which is a class closed under taking (induced) subgraphs. In this paper, we focus on a family of graph classes that exhibits a nice behavior with respect to the considered problems (and many others): we consider $\mathcal{G}$ to be a *minor-closed* graph class, i.e., such that every minor of a graph in $\mathcal{G}$ (that is, obtained from a subgraph of a graph in $\mathcal{G}$ by contracting edges; see Section 2 for the formal definition) is also in $\mathcal{G}$. Indeed, it turns out that, for every such a family $\mathcal{G}$, the problems become fixed-parameter tractable, as we proceed to discuss.

The *minor-obstruction set* (in short *obstruction set*) of $\mathcal{G}$ is the set of minor-minimal graphs that do not belong to $\mathcal{G}$, and is denoted by $\mathsf{obs}(\mathcal{G})$. Notice that $\mathsf{obs}(\mathcal{G})$ gives a complete characterization of $\mathcal{G}$ as, for every graph G, it holds that $G \in \mathcal{G}$ if and only if, for every $H \in \mathsf{obs}(\mathcal{G})$, H is not a minor of G. Because of Robertson and Seymour's theorem [49], $\mathsf{obs}(\mathcal{G})$ is *finite* for every minor-closed graph class. As checking whether an h-vertex graph H is a minor of G can be done in time $f(h) \cdot n^2$ [31, 47], the finiteness of $\mathsf{obs}(\mathcal{G})$ along with the above characterization imply that, for every minor-closed graph class $\mathcal{G}$, checking whether $G \in \mathcal{G}$ can be done in time $c \cdot n^2$, where c is a constant depending on the graph class $\mathcal{G}$. This meta-theorem implies the *existence* of FPT-algorithms for a wide family of problems, including Vertex Deletion to $\mathcal{G}$ and Elimination Distance to $\mathcal{G}$. Indeed, this follows by observing that if $\mathcal{G}$ is minor-closed, then for every non-negative integer k, the classes $\mathcal{A}_k(\mathcal{G})$ and $\mathcal{E}_k(\mathcal{G})$ are also minor-closed.

As Robertson and Seymour's theorem [49] does *not* give any way to construct the corresponding obstruction sets, the aforementioned argument is not constructive, i.e., it cannot construct the obstruction sets required for the corresponding FPT-algorithms. Moreover, these algorithms are *non-uniform* in k, meaning that we have a *distinct* algorithm for every value of k. Important steps towards the constructibility of such FPT-algorithms were done by Adler, Grohe, and Kreutzer [1] and Bulian and Dawar [11], who respectively proved that $\mathsf{obs}(\mathcal{A}_k(\mathcal{G}))$ and $\mathsf{obs}(\mathcal{E}_k(\mathcal{G}))$ are effectively computable. Hence, for both problems, it is possible to construct uniform (in k) algorithms running in time $f(k) \cdot n^2$ for some computable function f. However, this does not imply any reasonable, or even explicit, parametric dependence of the obtained algorithms.

The main focus of this paper is on the parametric and polynomial dependence of FPT-algorithms to solve Vertex Deletion to $\mathcal{G}$ and Elimination Distance to $\mathcal{G}$, i.e., for recognizing the classes $\mathcal{A}_k(\mathcal{G})$ and $\mathcal{E}_k(\mathcal{G})$, when $\mathcal{G}$ is a minor-closed graph class.

Concerning Vertex Deletion to $\mathcal{G}$, after a number of articles for particular cases of minor-closed classes $\mathcal{G}$, such as graphs of bounded treewidth [19, 34], planar graphs [27, 42], or graphs of bounded genus [37], an explicit FPT-algorithm for *any* minor-closed graph $\mathcal{G}$ was recently proposed by Sau, Stamoulis, and Thilikos [51], running in time $2^{\mathcal{O}(k^c)} \cdot n^3$, where c is a constant that depends on the maximum size of a graph in the obstruction set of $\mathcal{G}$.

Moreover, in the case where $\mathsf{obs}(\mathcal{G})$ contains some apex-graph (that is, a 1-apex for the class of planar graphs), Sau, Stamoulis, and Thilikos [51] gave an improved running time of $2^{\mathcal{O}(k^c)} \cdot n^2$. Note also that the more general variant where $\mathcal{G}$ is a topological-minor-closed graph class is in FPT as well [20].

As for Elimination Distance to $\mathcal{G}$ when $\mathcal{G}$ is minor-closed, no explicit parametric dependence was known, with the notable exception of treedepth, for which Reidl, Rossmanith, Villaamil, and Sikdar [46] gave an algorithm deciding whether $\mathsf{td}(G) \leq k$ in time $2^{\mathcal{O}(k \cdot \mathsf{tw})} \cdot n$, where $\mathsf{tw} := \mathsf{tw}(G)$ (see also [9]). Using our terminology, and given that $\mathsf{tw}(G) \leq \mathsf{td}(G)$ for every graph G, this yields an FPT-algorithm for Elimination Distance to $\mathcal{G}_\emptyset$, where $\mathcal{G}_\emptyset$ is the class consisting of the empty graph, running in time $2^{\mathcal{O}(k^2)} \cdot n$. Note that this algorithm [46], combined with the fact that $\mathsf{td}(G) \leq \log(n) \cdot \mathsf{tw}(G)$ (see [9]), imply an XP-algorithm for the problem of computing td when parameterized by tw, namely an algorithm that computes the value of $\mathsf{td}(G)$ in time $n^{\mathcal{O}(\mathsf{tw}(G)^2)}$. To the best of our knowledge, it is open whether computing td parameterized by tw is in FPT.

Before describing our results, let us mention some recent relevant results dealing with Elimination Distance to $\mathcal{G}$ for classes $\mathcal{G}$ that are not necessarily minor-closed. Agrawal and Ramanujan [4] (resp. Agrawal, Kanesh, Panolan, Ramanujan, and Saurabh [3]) provided FPT-algorithms, with parameter k, when $\mathcal{G}$ is the class of cliques (resp. graphs of bounded degree). Fomin, Golovach, and Thilikos [18] identified sufficient and necessary conditions for the existence of FPT-algorithms when $\mathcal{G}$ is definable in first-order logic (such as having bounded degree). Jansen, de Kroon, and Włodarczyk [26] proved, among other results, that if $\mathcal{G}$ is a hereditary union-closed graph class and Vertex Deletion to $\mathcal{G}$ can be solved in time $2^{k^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$ (as it is the case for every minor-closed class $\mathcal{G}$ by the results of [51]), then there is an algorithm that, given an n-vertex graph G, computes an $\mathcal{O}(\mathsf{ed}_{\mathcal{G}}(G)^3)$-elimination set of G for $\mathcal{G}$ in time $2^{\mathsf{ed}_{\mathcal{G}}(G)^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$. Therefore, for union-closed minor-closed graph classes $\mathcal{G}$, the result of [26] yields an FPT-approximation algorithm for Elimination Distance to $\mathcal{G}$.

Agrawal, Kanesh, Lokshtanov, Panolan, Ramanujan, Saurabh, and Zehavi [2] proved that if $\mathcal{G}$ is hereditary, union-closed, and definable in monadic second-order logic, then Vertex Deletion to $\mathcal{G}$ is (non-uniformly) in FPT if, and only if, Elimination Distance to $\mathcal{G}$ is (non-uniformly) in FPT. Incidentally, they also showed that if $\mathcal{G}$ is defined by excluding a finite number of connected topological minors, then Elimination Distance to $\mathcal{G}$ is (uniformly) in FPT. We note that the results of [2] do not provide explicit parametric dependencies for these FPT-algorithms. Also, let us mention that it was conjectured in [2] that Elimination Distance to $\mathcal{G}$ is in FPT parameterized by a generalization of treewidth called $\mathcal{G}$-*treewidth* (see [2, 16, 26]). Note that, if true, this conjecture would answer the open problem mentioned above of whether computing td parameterized by tw is in FPT.

Our results. In this paper, we provide explicit FPT-algorithms for Vertex Deletion to $\mathcal{G}$ and Elimination Distance to $\mathcal{G}$ for *every* fixed minor-closed graph class $\mathcal{G}$. Our first result is the following.

▶ **Theorem 1.** *For every minor-closed graph class $\mathcal{G}$, there exists an algorithm that solves* Vertex Deletion to $\mathcal{G}$ *in time* $2^{\mathsf{poly}(k)} \cdot n^2$.

The degree of the polynomial function poly in the running time of Theorem 1 and of the other results below, as well as the constants hidden in the $\mathcal{O}$-notation in the running time of the algorithms, depend on the maximum size of a graph in $\mathsf{obs}(\mathcal{G})$. Thus, the algorithm of Theorem 1, while being uniformly FPT in k, is *not* uniform in the target class $\mathcal{G}$, as one needs to know an upper bound on the size of the minor-obstructions. This "meta-non-uniformity"

applies to all the algorithms presented in this paper, and it is also the case, among many others, of the FPT-algorithms in [51]. The algorithm of Theorem 1 improves the algorithm of [51] from cubic to quadratic complexity in n while keeping the same parametric dependence on k. This answers positively one of the open problems posed in [51].

Our next algorithmic results concern Elimination Distance to $\mathcal{G}$ and provide, to the authors' knowledge, the first FPT-algorithms for this problem, when $\mathcal{G}$ is minor-closed, with an *explicit parametric dependence*.

▶ **Theorem 2.** *For every minor-closed graph class $\mathcal{G}$, there exists an algorithm that solves* Elimination Distance to $\mathcal{G}$ *in time* $2^{2^{2^{\mathsf{poly}(k)}}} \cdot n^2$. *In the particular case where* $\mathsf{obs}(\mathcal{G})$ *contains an apex-graph, this algorithm runs in time* $2^{2^{\mathcal{O}(k^2 \log k)}} \cdot n^2$.

Examples of classes $\mathcal{G}$ where $\mathsf{obs}(\mathcal{G})$ contains an apex-graph are graphs of bounded Euler genus, such as planar graphs. Our next result improves the parametric dependence of the algorithm of Theorem 2 when $\mathsf{obs}(\mathcal{G})$ contains an apex-graph, with a worse polynomial factor.

▶ **Theorem 3.** *For every minor-closed graph class $\mathcal{G}$ such that* $\mathsf{obs}(\mathcal{G})$ *contains an apex-graph, there exists an algorithm that solves* Elimination Distance to $\mathcal{G}$ *in time* $2^{\mathsf{poly}(k)} \cdot n^3$.

As discussed later, a crucial ingredient in the algorithms of Theorem 2 and Theorem 3 is to solve Elimination Distance to $\mathcal{G}$ parameterized by the treewidth of the input graph. The following result, which may be of independent interest, deals with this case.

▶ **Theorem 4.** *For every minor-closed graph class $\mathcal{G}$, there exists an algorithm that solves* Elimination Distance to $\mathcal{G}$ *in time* $2^{\mathcal{O}(k \cdot \mathsf{tw} + \mathsf{tw} \log \mathsf{tw})} \cdot n$, *where* tw *denotes the treewidth of the input graph.*

The algorithm of Theorem 4 can be seen as a generalization of the algorithm of Reidl, Rossmanith, Villaamil, and Sikdar [46] deciding whether $\mathsf{td}(G) \leq k$ in time $2^{\mathcal{O}(k \cdot \mathsf{tw})} \cdot n$. Since, for any graph G and any graph class $\mathcal{G}$, $\mathsf{ed}_{\mathcal{G}}(G) \leq \mathsf{td}(G) \leq \mathsf{tw}(G) \cdot \log n$, Theorem 4 implies the existence of an XP-algorithm for Elimination Distance to $\mathcal{G}$ parameterized by treewidth, when $\mathcal{G}$ is minor-closed, running in time $n^{\mathcal{O}(\mathsf{tw}^2)}$. Given that the conjecture of [2] is still open, this is the best type of algorithm that one can expect for Elimination Distance to $\mathcal{G}$ parameterized by treewidth.

Finally, for any minor-closed graph class $\mathcal{G}$, we provide an upper bound on the size of the graphs in the obstruction set of $\mathcal{E}_k(\mathcal{G})$.

▶ **Theorem 5.** *For every minor-closed graph class $\mathcal{G}$ and for every positive integer k, each graph in* $\mathsf{obs}(\mathcal{E}_k(\mathcal{G}))$ *has at most* $2^{2^{2^{\mathsf{poly}(k)}}}$ *vertices. Moreover, if* $\mathsf{obs}(\mathcal{G})$ *contains an apex-graph, this bound drops to* $2^{2^{\mathsf{poly}(k)}}$.

The only previously known bound for the graphs in $\mathsf{obs}(\mathcal{E}_k(\mathcal{G}))$ is the one for treedepth by Dvořák, Giannopoulou, and Thilikos [15], who proved that every graph in $\mathsf{obs}(\mathcal{E}_k(\mathcal{G}_\emptyset))$ has size at most $2^{2^{k-1}}$. Theorem 5 can be seen as a generalization of the results of Sau, Stamoulis, and Thilikos [52], who provided similar upper bounds for the graphs in $\mathsf{obs}(\mathcal{A}_k(\mathcal{G}))$.

These two results are, to the authors' knowledge, the first upper bounds on the size of the graphs in the obstruction set for the elimination distance parameter, and give, as an immediate consequence, the first known upper bound for the size of these obstruction sets.

Our techniques. This paper builds heavily on the techniques recently introduced in [51] in order to deal with Vertex Deletion to $\mathcal{G}$, which are based on exploiting the Flat Wall Theorem of Robertson and Seymour [47], namely the version proved by Kawarabayashi, Thomas, and Wollan [32] and its recent restatement by Sau, Stamoulis, and Thilikos [50]. In a nutshell, the idea of Theorem 1, Theorem 2, and Theorem 3 is that, as far as the treewidth of the input graph is sufficiently large as an appropriate function of k, it is possible to either "branch" into a number of subproblems that depends only on k and where the value of the parameter is strictly smaller, or to find an irrelevant vertex (i.e., a vertex that does not change the answer to the considered problem) and remove it from the graph. The irrelevant vertex technique originates from Robertson and Seymour [47] and is further developped in [50–52]. Once the treewidth is bounded, what remains is to apply the most efficient possible algorithm to solve the problem via dynamic programming on tree decompositions.

Let us focus more particularly on the techniques we use to prove Theorem 1. Contrary to the algorithm of [51] that solves Vertex Deletion to $\mathcal{G}$ for any minor-closed class $\mathcal{G}$, we *avoid using iterative compression*. This explains the improvement from cubic to quadratic complexity in n. The algorithm of Theorem 1 can be seen as an extension of the algorithm of [51] that solves Vertex Deletion to $\mathcal{G}$ in the particular case where $\mathsf{obs}(\mathcal{G})$ contains some apex-graph, and uses ideas that date back to the work of Marx and Schlotter [42] for the Planarization problem, that is, when $\mathcal{G}$ is the class of planar graphs. In Section 3 we provide a sketch of the algorithms claimed in Theorem 1, Theorem 2, and Theorem 3, and in Section 4 we present the algorithm of Theorem 1 in full detail, along with a proof of its correctness.

The proof of Theorem 4 consists of a dynamic programming algorithm that combines the framework of [46] for the particular case where $\mathcal{G}$ contains only the empty graph (i.e., for treedepth) with the representative-based techniques introduced in [5]. A bit more precisely, the idea is to encode the partial solutions (called *characteristic*) via sets of annotated trees with some additional properties. Here, the trees correspond to partial elimination trees and the annotations indicate the representatives, in the leaves of the elimination trees, with respect to the canonical equivalence relation defined for the target class $\mathcal{G}$. The size of the characteristic dominates the running time of the whole algorithm. As usual when dealing with dynamic programming, the formal description of the algorithm is quite technical and lengthy, and has been deferred to the full version of the paper.

Finally, to obtain the upper bound on the size of a graph $G \in \mathsf{obs}(\mathcal{E}_k(\mathcal{G}))$ claimed in Theorem 5, we proceed in two steps. First, we bound the treewidth of G by a function of k. To do so, we observe that if the treewidth of G is big enough, then there is a big enough wall in G, and we find an irrelevant vertex v for Elimination Distance to $\mathcal{G}$ in G. However, $G \setminus \{v\} \in \mathcal{E}_k(\mathcal{G})$ and $G \notin \mathcal{E}_k(\mathcal{G})$, hence we reach a contradiction. The second step is to bound the size of a minor-minimal obstruction of small treewidth. This uses the classic technique of Lagergren [39] (see also [21–23, 28, 29, 38, 40, 52]) combined with the encoding of the tables of the dynamic programming algorithm that we use to prove Theorem 4; see the full paper.

2 Preliminaries

In this section we give some basic definitions needed to understand the main body of the paper. Due to space limitations and the length of all the formal definitions, the complete preliminaries are provided in the full version of the paper (definitions and preliminary results regarding treedepth, treewidth and boundaried graphs, and framework of flat walls).

Minors and obstructions. A graph G' is a *minor* of a graph G, denoted by $G' \preceq G$, if G' can be obtained from G by a sequence of vertex removals, edge removals, and edge contractions. Let $\mathcal{G}$ be a graph class that is closed under taking minors. Recall that the *minor obstruction set* of $\mathcal{G}$ is defined as the set of all minor-minimal graphs that are not in $\mathcal{G}$, and is denoted by $\mathsf{obs}(\mathcal{G})$. Given a finite non-empty collection of non-empty graphs $\mathcal{F}$, we denote by $\mathsf{exc}(\mathcal{F})$ the set containing every graph G that excludes all graphs in $\mathcal{F}$ as minors. We call each graph in $\mathsf{exc}(\mathcal{F})$ *$\mathcal{F}$-minor-free*.

Restating the problems. Let $\mathcal{G}$ be a minor-closed graph class and $\mathcal{F}$ be its obstruction set. Clearly, Vertex Deletion to $\mathcal{G}$ is the same problem as asking, given a graph G and some $k \in \mathbb{N}$, for a vertex set $S \subseteq V(G)$ of at most k vertices such that $G \setminus S \in \mathsf{exc}(\mathcal{F})$. Following the terminology of [5–8, 19, 20, 34, 35, 51], we call this problem $\mathcal{F}$-M-Deletion. Likewise, Elimination Distance to $\mathcal{G}$ is the same problem as asking whether $\mathsf{ed}_{\mathsf{exc}(\mathcal{F})}(G) \leq k$. We thus follow a similar notation and call this problem $\mathcal{F}$-M-Elimination Distance. Using the notation, $\{K_1\}$-M-Elimination Distance is the problem of asking whether $\mathsf{td}(G) \leq k$.

Some conventions. In the rest of the paper, we fix $\mathcal{G}$ to be a minor-closed graph class and $\mathcal{F}$ to be the set $\mathsf{obs}(\mathcal{G})$. From Robertson and Seymour's theorem [49], we know that $\mathcal{F}$ is a finite collection of graphs. Given a graph G, we define its *apex number* to be the smallest integer a for which there is a set $A \subseteq V(G)$ of size at most a such that $G \setminus A$ is planar. An *apex-graph* is a graph with apex number one. Also, we define the *detail* of G, denoted by $\mathsf{detail}(G)$, to be the maximum among $|E(G)|$ and $|V(G)|$. We define three constants depending on $\mathcal{F}$ that will be used throughout the paper whenever we consider such a graph family $\mathcal{F}$. We define $a_{\mathcal{F}}$ as the minimum apex number of a graph in $\mathcal{F}$, we set $s_{\mathcal{F}} := \max\{|V(H)| \mid H \in \mathcal{F}\}$, and we set $\ell_{\mathcal{F}} := \max\{\mathsf{detail}(H)| \mid H \in \mathcal{F}\}$. Given a tuple $\mathbf{t} = (x_1, \ldots, x_\ell) \in \mathbb{N}^\ell$ and two functions $\chi, \psi : \mathbb{N} \to \mathbb{N}$, we write $\chi(n) = \mathcal{O}_{\mathbf{t}}(\psi(n))$ in order to denote that there exists a computable function $\phi : \mathbb{N}^\ell \to \mathbb{N}$ such that $\chi(n) = \mathcal{O}(\phi(\mathbf{t}) \cdot \psi(n))$. Notice that $s_{\mathcal{F}} \leq \ell_{\mathcal{F}} \leq s_{\mathcal{F}}(s_{\mathcal{F}} - 1)/2$, and thus $\mathcal{O}_{\ell_{\mathcal{F}}}(\cdot) = \mathcal{O}_{s_{\mathcal{F}}}(\cdot)$. Observe also that $\mathcal{A}_k(\mathcal{G})$ and $\mathcal{E}_k(\mathcal{G})$ are $K_{s_{\mathcal{F}}+k}$-minor-free graph classes, and thus, due to [53], we can always assume that G has $\mathcal{O}_{s_{\mathcal{F}}}(k\sqrt{\log k} \cdot n)$ edges, otherwise we can directly conclude that (G, k) is a no-instance for both problems.

Walls and flat walls. In this paper we extensively deal with walls and flat walls, following the framework of [50]. Unfortunately, more than ten pages are required to provide all the technical notions to correctly present all this framework, that is necessary to use the tools developed in [50–52]. Thus, formal definitions are provided in the full version of the paper. More precisely, we introduce walls and several notions concerning them (just look at Figure 1 to understand what a wall is). We then provide the definitions of a rendition and a painting in order to define flat walls. There are a number of technical terms (such as tilts, influence, regular flatness pairs, ...) that are not the main focus of this work. Let us just mention that the perimeter of a flat wall of a graph G separates $V(G)$ into two sets X and Y with Y containing the wall. The *compass* of a flat wall is $G[Y]$.

We define canonical partitions and the notion of bidimensionality. Informally speaking, a *canonical partition* of a graph with respect to some wall W refers to a partition of the vertex set of a graph in bags that follow the structure of a wall subgraph of the given graph; see Figure 1 for an illustration. The *bidimensionality* of a vertex set X with respect to a wall W of a graph G intuitively expresses the "spread" of a set X in a W-canonical partition of G. The crucial idea is that a set X of small bidimensionality cannot "destroy" a large (flat) wall too much.

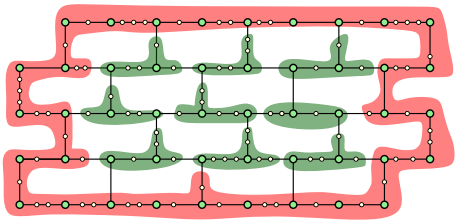


Figure 1 A 5-wall and its canonical partition $\mathcal{Q}$. The red bag is the external bag Q_{ext}.

Finally, we present homogeneous walls. Intuitively, *homogeneous flat walls* are flat walls that allow the routing of the same set of (topological) minors in the augmented flaps (i.e., the flaps together with the apex set) "cropped" by each one of their bricks. Such a homogeneous wall can be detected in a big enough flat wall (Proposition 10) and this "homogeneity" property implies that some central part of a big enough homogeneous wall can be declared irrelevant (Proposition 11).

3 Sketch of the algorithms

In this section we provide a sketch of the algorithms claimed in Theorem 1, Theorem 2 and Theorem 3. As mentioned in the introduction, Theorem 1 can be seen as a generalization of the algorithm of [51] that solves $\mathcal{F}$-M-Vertex Deletion in the particular case where $\mathcal{F}$ contains some apex-graph. While many techniques taken from [51] remain the same, some new ingredients are needed so as to deal with the possible existence of many apices in all graphs in $\mathcal{F}$. On the other hand, Theorem 2 and Theorem 3 can be seen as an adaptation of Theorem 1 to $\mathcal{F}$-M-Elimination Distance. Since these three algorithms follow a common streamline, we sketch all of them simultaneously while pointing out the steps where they differ. Moreover, the full proof of Theorem 1 is given in Section 4, while the proofs of Theorem 2 and Theorem 3 can be found in the full version of the paper.

The first common step is to run the following algorithm that states that a graph G in $\mathcal{A}_k(\mathsf{exc}(\mathcal{F}))$ or $\mathcal{E}_k(\mathsf{exc}(\mathcal{F}))$ either has bounded treewidth or contains a large wall. This result was proved in [51] in the case of $\mathcal{F}$-M-Deletion. The proof in the case of $\mathcal{F}$-M-Elimination Distance, necessary for Theorem 2 and Theorem 3, can be found in the full version of the paper.

▶ **Proposition 6** ([51], full paper)**.** *Let $\mathcal{F}$ be a finite collection of graphs. There exist a function $f_1 : \mathbb{N} \to \mathbb{N}$ and an algorithm with the following specifications:*

`Find-Wall`(G, r, k)
***Input**: A graph G, an odd $r \in \mathbb{N}_{\geq 3}$, and $k \in \mathbb{N}$.*
***Output**: One of the following:*

- **Case 1:** *Either a report that (G, k) is a* no*-instance of $\mathcal{F}$-M-Deletion (resp. $\mathcal{F}$-M-Elimination Distance), or*
- **Case 2:** *a report that G has treewidth at most $f_1(s_{\mathcal{F}}) \cdot r + k$, or*
- **Case 3:** *an r-wall W of G.*

Moreover, $f_1(s_{\mathcal{F}}) = 2^{\mathcal{O}(s_{\mathcal{F}}^2 \cdot \log s_{\mathcal{F}})}$, and the algorithm runs in time $2^{\mathcal{O}_{\ell_{\mathcal{F}}}(r^2+(k+r)\cdot\log(k+r))} \cdot n$ (resp. $2^{\mathcal{O}_{\ell_{\mathcal{F}}}(r^2+k^2)} \cdot n$).

In **Case 1**, we can immediately conclude. In **Case 2**, since the treewidth of G is bounded, we use a dynamic programming algorithm to solve the corresponding problem. Namely, we solve $\mathcal{F}$-M-Deletion on instances of bounded treewidth using the main result from [5].

▶ **Proposition 7** ([5]). *For every finite collection of graphs $\mathcal{F}$, there exists an algorithm that, given a triple (G, tw, k) where G is a graph of treewidth at most tw and k is a non-negative integer, solves $\mathcal{F}$-M-Deletion in time $2^{\mathcal{O}_{\ell_{\mathcal{F}}}(\mathsf{tw}\cdot\log \mathsf{tw})} \cdot n$.*

For $\mathcal{F}$-M-Elimination Distance, we use Theorem 4 to conclude. The proof of this (quite technically involved) dynamic programming algorithm is given in the full version of the paper.

Therefore, it only remains to deal with **Case 3**. Given an r-wall W of G, we want to reduce the size of G. To do so, we observe that we can either:

- **Case 3a:** find a subwall W_a of W and an apex set A_a such that W_a is flat in $G \setminus A_a$ and has a compass of bounded treewidth, or
- **Case 3b:** find a subwall W_b of W that is very "well connected" to an apex set A_b of small size.

The above distinction is done using two algorithmic versions of the Flat Wall Theorem consecutively. The first one comes from [32, Theorem 7.7] and is translated here in the new framework with tilts of [50]. Informally, we say that a graph H is *grasped* by a wall W in a graph G if there is a model of H in G such that the model of every node of H intersects W.

▶ **Proposition 8** ([32]). *There are two functions $f_2, f_3 : \mathbb{N} \to \mathbb{N}$, such that the images of f_2 are odd integers, and an algorithm with the following specifications:*

`Grasped-or-Flat`(G, r, t, W)
***Input**: A graph G, an odd $r \in \mathbb{N}_{\geq 3}$, $t \in \mathbb{N}_{\geq 1}$, and an $f_2(t) \cdot r$-wall W of G.*
***Output**: One of the following:*

- *Either a model of a K_t-minor in G grasped by W, or*
- *a set $A \subseteq V(G)$ of size at most $f_3(t)$ and a flatness pair $(W', \mathfrak{R}')$ of $G \setminus A$ of heigth r such that W' is a $\tilde{W}'$-tilt of some subwall $\tilde{W}'$ of W.*

Moreover, $f_2(t) = \mathcal{O}(t^{26})$, $f_3(t) = \mathcal{O}(t^{24})$, and the algorithm runs in time $\mathcal{O}(t^{24}m + n)$.

We would like to mention that the notion of being grasped by a wall is one of the new main arguments yielding the improvement of the complexity for $\mathcal{F}$-M-Deletion compared to [51].

The second one comes from [51] and adds the condition that W' has a compass of bounded treewidth, at the price of dropping the condition that the model of K_t is grasped by W.

▶ **Proposition 9** ([51]). *There exist a function $f_4 : \mathbb{N} \to \mathbb{N}$ and an algorithm with the following specifications:*

`Clique-Or-twFlat`(G, r, t)
***Input**: A graph G, an odd $r \in \mathbb{N}_{\geq 3}$, and $t \in \mathbb{N}_{\geq 1}$.*
***Output**: One of the following:*

- *Either a report that K_t is a minor of G, or*
- *a tree decomposition of G of width at most $f_4(t) \cdot r$, or*
- *a set $A \subseteq V(G)$ of size at most $f_3(t)$ and a regular flatness pair $(W', \mathfrak{R}')$ of $G \setminus A$ of height r whose $\mathfrak{R}'$-compass has treewidth at most $f_4(t) \cdot r$.*

Moreover, $f_4(t) = 2^{\mathcal{O}(t^2 \log t)}$ and this algorithm runs in time $2^{\mathcal{O}_t(r^2)} \cdot n$. The algorithm can be modified to obtain an explicit dependence on t in the running time, namely $2^{2^{\mathcal{O}(t^2 \log t)} \cdot r^3 \log r} \cdot n$.

`Grasped-or-Flat` is used to find a big enough complete graph "controlled" by the input wall, while we need `Clique-or-twFlat` to find a flat wall whose compass has bounded treewidth. Unfortunately, we cannot obtain both conditions simultaneously, and this is why we need both results. If, after using both algorithms, we obtain a flatness pair $(\tilde{W}', \mathfrak{R}')$ of

$G \setminus A_a$ of heigth r_a whose compass has bounded treewidth, then we are in **Case 3a**. In that case, the following result from [51] provides an algorithm that, given a flatness pair of big enough height, outputs a homogeneous flatness pair.

▶ **Proposition 10** ([51]). *There is a function $f_5 : \mathbb{N}^4 \to \mathbb{N}$, whose images are odd integers, and an algorithm with the following specifications:*

`Homogeneous`$(r, \tilde{a}, a, \ell, t, G, A, W, \mathcal{R})$
***Input**: Five integers $r \in \mathbb{N}_{\geq 3}$, $\tilde{a}, a, \ell, t \in \mathbb{N}$, where $\tilde{a} \leq a$, a graph G, a set $A \subseteq V(G)$ of size at most a, and a flatness pair $(W, \mathfrak{R})$ of $G \setminus A$ of height $f_5(r, a, \tilde{a}, \ell)$ whose $\mathfrak{R}$-compass has treewidth at most t.*
***Output**: A flatness pair $(\breve{W}, \breve{\mathfrak{R}})$ of $G \setminus A$ of height r that is ℓ-homogeneous with respect to $\binom{A}{\leq \tilde{a}}$ and is a W'-tilt of $(W, \mathfrak{R})$ for some subwall W' of W.*
Moreover, $f_5(r, \tilde{a}, a, \ell) = \mathcal{O}(r^{f_6(\tilde{a}, a, \ell)})$ where $f_6(\tilde{a}, a, \ell) = 2^{a^{\tilde{a}} \cdot 2^{\mathcal{O}((\tilde{a}+\ell)\cdot \log(\tilde{a}+\ell))}}$ and the algorithm runs in time $2^{\mathcal{O}(f_6(\tilde{a}, a, \ell) \cdot r \log r + t \log t)} \cdot (n + m)$.

Then we use the next result, that essentially says that the central vertex v of a big enough homogeneous wall is irrelevant, i.e., (G, k) and $(G \setminus v, k)$ are equivalent instances of the corresponding problem. Here, $\mathsf{bid}_{G \setminus A, W}(X)$ denotes the bidimensionality of a set X in the wall W with apex set A.

▶ **Proposition 11** ([52]). *Let $\mathcal{F}$ be a finite collection of graphs. There exist two functions $f_7 : \mathbb{N}^4 \to \mathbb{N}$ and $f_8 : \mathbb{N}^2 \to \mathbb{N}$, and an algorithm with the following specifications:*

`Find-Irrelevant-Vertex`$(k, a, G, A, W, \mathcal{R})$
***Input**: Two integers $k, a \in \mathbb{N}$, a graph G, a set $A \subseteq V(G)$, and a regular flatness pair $(W, \mathcal{R})$ of $G \setminus A$ of height at least $f_7(a, \ell_{\mathcal{F}}, 3, k)$ that is $f_8(a, \ell_{\mathcal{F}})$-homogeneous with respect to $\binom{A}{\leq a}$.*
***Output**: A vertex v of $G \setminus A$ such that for every set $X \subseteq V(G)$ with $\mathsf{bid}_{G \setminus A, W}(X) \leq k$ and $|A \setminus X| \leq a$, it holds that $G \setminus X \in \mathsf{exc}(\mathcal{F})$ if and only if $G \setminus (X \setminus v) \in \mathsf{exc}(\mathcal{F})$.*
Moreover, $f_7(a, \ell_{\mathcal{F}}, q, k) = \mathcal{O}(k \cdot (f_{\mathsf{ul}}(16a + 12\ell_{\mathcal{F}}))^3 + q)$, where f_{ul} is the function of the Unique Linkage Theorem (see [33]) and $f_8(a, \ell_{\mathcal{F}}) = a + \ell_{\mathcal{F}} + 3$, and this algorithm runs in time $\mathcal{O}(n + m)$.

We can prove that both k-apex sets and k-elimination sets have small bidimensionality. If, for every k-apex set S, $G \setminus S \in \mathsf{exc}(\mathcal{F})$ if and only if $G \setminus (S \setminus v) \in \mathsf{exc}(\mathcal{F})$, then it is straightforward to see that v is irrelevant for $\mathcal{F}$-M-Deletion. It is slightly less trivial to prove that, for each k-elimination set S, we can find some superset $X \supseteq S$ of small bidimensionality such that a similar statement holds. Additional details are available in the full paper.

Therefore we can recursively solve the problems on the instance $(G \setminus v, k)$.

If no flatness pair whose compass has bounded treewidth was found, then we are in **Case 3b**. In this case, inspired by [42] and [51], we use the following result of [52] that basically says that if there is a big enough flat wall W and an apex set A' of $a_{\mathcal{F}}$ vertices that are all adjacent to many bags of a canonical partition of W, then each k-apex set or k-elimination set intersects A'.

▶ **Proposition 12** ([52]). *There exist three functions $f_9, f_{10}, f_{11} : \mathbb{N}^3 \to \mathbb{N}$, such that if G is a graph, $k \in \mathbb{N}$, A is a subset of $V(G)$, $(W, \mathfrak{R})$ is a flatness pair of $G \setminus A$ of height at least $f_9(a_{\mathcal{F}}, s_{\mathcal{F}}, k)$, $\tilde{\mathcal{Q}}$ is a W-canonical partition of $G \setminus A$, A' is a subset of vertices of A that are adjacent, in G, to vertices of at least $f_{10}(a_{\mathcal{F}}, s_{\mathcal{F}}, k)$ many $f_{11}(a_{\mathcal{F}}, s_{\mathcal{F}}, k)$-internal bags of $\tilde{\mathcal{Q}}$, and $|A'| \geq a_{\mathcal{F}}$, then for every set $X \subseteq V(G)$ such that $G \setminus X \in \mathsf{exc}(\mathcal{F})$ and $\mathsf{bid}_{G \setminus A, W}(X) \leq k$, it holds that $X \cap A' \neq \emptyset$. Moreover, $f_9(a, s, k) = \mathcal{O}(2^a \cdot s^{5/2} \cdot k^{5/2})$, $f_{10}(a, s, k) = \mathcal{O}(2^a \cdot s^3 \cdot k^3)$, and $f_{11}(a, s, k) = \mathcal{O}((a^2 + k) \cdot s)$, where $a = a_{\mathcal{F}}$ and $s = s_{\mathcal{F}}$.*

For the $\mathcal{F}$-M-Deletion problem, if we find such a set A', then we can branch by guessing which vertex $v \in A'$ belongs to a k-apex set and recursively solving $(G \setminus v, k-1)$. Given that A' has size $a_{\mathcal{F}}$ and that k decreases after each guess, this step is applied at most $a_{\mathcal{F}}^k$ times.

However, for $\mathcal{F}$-M-Elimination Distance, k does not decrease, given that the size of a k-elimination set may not depend on k. Thus, this step may be done $a_{\mathcal{F}}^n$ times, which does not give an FPT-algorithm. To circumvent this problem, we propose two alternatives:

Option 1: The first alternative is to only use **Case 3a**. This is possible given that $(K_{s_{\mathcal{F}}+k}, k)$ is a no-instance of both problems. Thus, when using the algorithms `Grasped-or-Flat` and `Clique-or-twFlat`, we force the outcome to be an apex set A and a flatness pair of $G \setminus A$. However, the bound on the size of A now depends on k, and thus, so does the variable a in the input of the algorithm `Homogeneous`. This explains the triple-exponential parametric dependence on k in Theorem 2. Interestingly, a precise analysis of the time complexity shows that if $a_{\mathcal{F}} = 1$, i.e., when $\mathcal{F}$ contains an apex graph, the parametric dependence is only double-exponential on k (cf. Theorem 2).

Option 2: The second alternative is to restrict ourselves to the case where $a_{\mathcal{F}} = 1$. Thus, in **Case 3b**, we find a vertex v that belongs to every k-elimination set. There is no need to branch, and this step is done at most n times. However, the fact that the time complexity of this step is quadratic in n explains the cubic complexity of the algorithm in Theorem 3.

It remains to show that if no flatness pair whose compass has bounded treewidth was found, then we can find a flatness pair and a set A' satisfying the conditions of Proposition 12. To do so, using flow techniques, we find the set A of vertices with sufficiently many internally-disjoint paths to W, independently from one another. If this set is too large, we can safely declare a no-instance. Otherwise, we extend the canonical partition of W and just check whether $a_{\mathcal{F}}$ vertices of A are adjacent to many vertices of this new canonical partition. If this happens, then we can safely use Proposition 12. The second main improvement with respect to the algorithm in [51] is the new argument that the extension of the canonical partition of W can be done in a totally arbitrary manner. The quadratic complexity of this step stems from the search for internally-disjoint paths for every vertex of the input graph.

4 Vertex deletion to a minor-closed graph class

In this section we prove our main result for the $\mathcal{F}$-M-Deletion problem, namely Theorem 1.

All the propositions necessary for this proof have already been stated in Section 3, aside from the following result proved in [52] that intuitively states that, given a canonical partition $\tilde{\mathcal{Q}}$ of a flat wall $(W, \mathfrak{R})$ of big enough height, we can find a "packing" of subwalls of W that are inside some central part of W and such that the vertex set of every bag of $\tilde{\mathcal{Q}}$ intersects the vertices of at most one of these walls.

▶ **Proposition 13** ([52]). *There exists a function $f_{12} : \mathbb{N}^3 \to \mathbb{N}$ such that if $p, l \in \mathbb{N}_{\geq 1}$, $r \in \mathbb{N}_{\geq 3}$ is an odd integer, G is a graph, $(W, \mathfrak{R})$ is a flatness pair of G of height at least $f_{12}(l, r, p)$, and $\tilde{\mathcal{Q}}$ is a W-canonical partition of G, then there is a collection $\mathcal{W} = \{W^1, \ldots, W^l\}$ of r-subwalls of W such that*

- *for every $i \in [l]$, $\bigcup \mathsf{influence}_{\mathfrak{R}}(W^i)$ is a subgraph of $\bigcup\{Q \mid Q$ is a p-internal bag of $\tilde{\mathcal{Q}}\}$ and*
- *for every $i, j \in [l]$, with $i \neq j$, there is no internal bag of $\tilde{\mathcal{Q}}$ that contains vertices of both $V(\bigcup \mathsf{influence}_{\mathfrak{R}}(W^i))$ and $V(\bigcup \mathsf{influence}_{\mathfrak{R}}(W^j))$.*

Moreover, $f_{12}(l, r, p) = \mathcal{O}(\sqrt{l} \cdot r + p)$ and $\mathcal{W}$ can be constructed in time $\mathcal{O}(n + m)$.

4.1 Description of the algorithm for $\mathcal{F}$-M-Deletion

Our algorithm for $\mathcal{F}$-M-Deletion has three steps. In Step 1, either we can easily conclude with a positive or a negative answer (**Cases 1 and 2**) or we find a big wall. If we can find a large flat wall of bounded treewidth inside this wall, then we go to Step 2 and find an irrelevant vertex (**Case 3a**). Otherwise, we proceed to Step 3 where, by using flow techniques, we find a set of vertices that intersects every solution, and we branch on this set or we report a negative answer (**Case 3b**). The correctness of the algorithm is not trivial and will be justified in Subsection 4.2.

Given a non-negative integer x, we denote by $\mathsf{odd}(x)$ the smallest odd number that is not smaller than x. We define the following constants.

$$
\begin{aligned}
a &= f_3(s_{\mathcal{F}} + a_{\mathcal{F}} - 1), & b &= f_3(s_{\mathcal{F}}),\\
q &= f_{10}(a_{\mathcal{F}}, s_{\mathcal{F}}, k), & p &= f_{11}(a_{\mathcal{F}}, s_{\mathcal{F}}, k),\\
l &= (q-1)\cdot(k+b), & r_6 &= f_7(a+b, \ell_{\mathcal{F}}, 3, k)\\
d &= f_8(a+b, \ell_{\mathcal{F}}) & r_5 &= f_5(r_6, a+b, a+b, d),\\
t &= f_4(s_{\mathcal{F}})\cdot r_5, & r_4 &= \mathsf{odd}(t+3),\\
r_3 &= f_{12}(a_{\mathcal{F}}, r_4, 1), & r_2 &= \mathsf{odd}(2 + f_2(s_{\mathcal{F}} + a_{\mathcal{F}} - 1)\cdot r_3),\\
r_2' &= \mathsf{odd}(\max\{f_9(a_{\mathcal{F}}, s_{\mathcal{F}}, k), f_{12}(l+1, r_2, p)\}), & r_1 &= \mathsf{odd}(f_2(s_{\mathcal{F}})\cdot r_2' + k).
\end{aligned}
$$

Note that $r_6 = \mathcal{O}_{\ell_{\mathcal{F}}}(k)$, $r_5, r_4, r_3, r_2, t = \mathcal{O}_{\ell_{\mathcal{F}}}(k^c)$, and $r_2', r_1 = \mathcal{O}_{\ell_{\mathcal{F}}}(k^{c+2})$ where $c = f_6(a+b, a+b, d)$. Recall from Section 2 that we may assume that G has $\mathcal{O}_{s_{\mathcal{F}}}(k\sqrt{\log k}\cdot n)$ edges.

Step 1. Run the algorithm `Find-Wall` from Proposition 6 with input (G, r_1, k) and, in time $2^{\mathcal{O}_{\ell_{\mathcal{F}}}(r_1^2 + (k+r_1)\log(k+r_1))}\cdot n = 2^{\mathcal{O}_{\ell_{\mathcal{F}}}(k^{2(c+2)})}\cdot n$,

- either report a no-instance, or
- conclude that $\mathsf{tw}(G) \leq f_1(s_{\mathcal{F}})\cdot r_1 + k$ and solve $\mathcal{F}$-M-Deletion in time $2^{\mathcal{O}_{\ell_{\mathcal{F}}}((r_1+k)\log(r_1+k))}\cdot n = 2^{\mathcal{O}_{\ell_{\mathcal{F}}}(k^{c+2}\cdot\log k)}\cdot n$ using the algorithm of Proposition 7, or
- obtain an r_1-wall W_1 of G.

If the output of Proposition 6 is an r_1-wall W_1, consider all the $\binom{r_1}{r_2}^2 = 2^{\mathcal{O}_{\ell_{\mathcal{F}}}(k^c\log k)}$ r_2-subwalls of W_1. For each one of them, say W_2, let W_2^* be the central (r_2-2)-subwall of W_2 and let D_{W_2} be the graph obtained from G after removing the perimeter of W_2 and taking the connected component containing W_2^*. Run the algorithm `Grasped-or-Flat` of Proposition 8 with input $(D_{W_2}, r_3, s_{\mathcal{F}} + a_{\mathcal{F}} - 1, W_2^*)$. This can be done in time $\mathcal{O}_{s_{\mathcal{F}}}(k\sqrt{\log k}\cdot n)$.

If for some of these subwalls the result is a set $A \subseteq V(D_{W_2})$ with $|A| \leq a$ and a flatness pair $(W_3, \mathfrak{R}_3)$ of $D_{W_2}\setminus A$ of height r_3 then, as in Proposition 13, compute a W_3-canonical partition $\tilde{\mathcal{Q}}$ of $D_{W_2}\setminus A$ and a collection $\mathcal{W} = \{W^1, ..., W^{a_{\mathcal{F}}}\}$ of r_4-subwalls of W_3 such that for every $i \in [a_{\mathcal{F}}]$, $\bigcup \mathsf{influence}_{\mathfrak{R}_3}(W^i)$ is a subgraph of $\bigcup\{Q \mid Q \text{ is a } p\text{-internal bag of } \tilde{\mathcal{Q}}\}$ and for every $i, j \in [a_{\mathcal{F}}]$, with $i \neq j$, there is no internal bag of $\tilde{\mathcal{Q}}$ that contains vertices of both $V(\bigcup \mathsf{influence}_{\mathfrak{R}_3}(W^i))$ and $V(\bigcup \mathsf{influence}_{\mathfrak{R}_3}(W^j))$. This can be done in time $\mathcal{O}_{s_{\mathcal{F}}}(k\sqrt{\log k}\cdot n)$.

For $i \in [a_{\mathcal{F}}]$, let W^{i*} be the central (r_4-2)-subwall of W^i and let D_{W^i} be the graph obtained from D_{W_2} after removing A and the perimeter of W^i and taking the connected component containing W^{i*}. Run the algorithm `Clique-or-twFlat` of Proposition 9 with input $(D_{W^i}, r_5, s_{\mathcal{F}})$. This takes time $2^{\mathcal{O}_{\ell_{\mathcal{F}}}(r_5^2)}\cdot n = 2^{\mathcal{O}_{\ell_{\mathcal{F}}}(k^{2c})}\cdot n$. If for one of these subwalls the result is a set A' of size at most b and a regular flatness pair $(W_5, \mathfrak{R}_5)$ of $D_{W^i}\setminus A'$ of height r_5 whose $\mathfrak{R}_5$-compass has treewidth at most t, then we proceed to Step 2.

If, for every flatness pair $(W_3, \mathfrak{R}_3)$ and for every $i \in [a_{\mathcal{F}}]$, the result is a report that $K_{s_{\mathcal{F}}}$ is a minor of D_{W^i}, then we proceed to Step 3.

Step 2 (irrelevant vertex case). We obtain a 7-tuple $\mathfrak{R}'_5$ by adding all vertices of $G \setminus V(\mathsf{Compass}_{\mathfrak{R}_5}(W_5))$ to the set in the first coordinate of $\mathfrak{R}_5$, such that $(W_5, \mathfrak{R}'_5)$ is a regular flatness pair of $G \setminus (A \cup A')$ whose $\mathfrak{R}'_5$-compass has treewidth at most t. We apply the algorithm `Homogeneous` of Proposition 10 with input $(r_6, a+b, a+b, d, t, G, A \cup A', W_5, \mathfrak{R}'_5)$, which outputs, in time $2^{\mathcal{O}_{\ell_{\mathcal{F}}}(t \log t + k \log k)} \cdot n = 2^{\mathcal{O}_{\ell_{\mathcal{F}}}(k^c \log k)} \cdot n$, a flatness pair $(W_6, \mathfrak{R}_6)$ of $G \setminus (A \cup A')$ of height r_6 that is d-homogeneous with respect to $2^{A \cup A'}$ and is a W^*-tilt of $(W_5, \mathfrak{R}'_5)$ for some subwall W^* of W_5. We apply the algorithm `Find-Irrelevant-Vertex` of Proposition 11 with input $(k, a+b, G, A \cup A', W_6, \mathfrak{R}_6)$, which outputs, in time $\mathcal{O}(n+m) = \mathcal{O}_{\ell_{\mathcal{F}}}(k\sqrt{\log k} \cdot n)$, a vertex v such that (G, k) and $(G \setminus v, k)$ are equivalent instances of $\mathcal{F}$-M-Deletion. Then the algorithm runs recursively on the equivalent instance $(G \setminus v, k)$.

Step 3 (branching case). Consider all the r'_2-subwalls of W_1, which are at most $\binom{r_1}{r'_2}^2 = 2^{\mathcal{O}_{\ell_{\mathcal{F}}}(k^{c+2} \log k)}$ many, and for each of them, say W'_2, compute its canonical partition $\mathcal{Q}$. Then, contract each bag Q of $\mathcal{Q}$ to a single vertex v_Q, and add a new vertex v_{all} and make it adjacent to all v_Q's. In the resulting graph G', for every vertex y of $G \setminus V(W'_2)$, check, using a path augmentation algorithm [13], whether there are q internally vertex-disjoint paths from v_{all} to y in time $\mathcal{O}(q \cdot m) = \mathcal{O}_{\ell_{\mathcal{F}}}(k^4\sqrt{\log k} \cdot n)$. Let $\tilde{A}$ be the set of all such y's.

If $|\tilde{A}| < a_{\mathcal{F}}$, then report a no-instance.

If $a_{\mathcal{F}} \leq |\tilde{A}| \leq k+b$, then consider all the $\binom{|\tilde{A}|}{a_{\mathcal{F}}} = 2^{\mathcal{O}_{\ell_{\mathcal{F}}}(\log k)}$ subsets of $\tilde{A}$ of size $a_{\mathcal{F}}$. For each one of them, say A^*, construct $\tilde{\mathcal{Q}}$ by enhancing $\mathcal{Q}$ on $G \setminus A^*$. Then, we distinguish two cases depending on whether for every A^* all its vertices are adjacent to vertices of q p-internal bags of $\tilde{\mathcal{Q}}$.

If each vertex of A^* is adjacent to vertices of q p-internal bags of $\tilde{\mathcal{Q}}$, then (due to Proposition 12) A^* should intersect every solution of $\mathcal{F}$-M-Deletion for the instance (G, k). Therefore, the algorithm runs recursively on each instance $(G \setminus y, k-1)$ for $y \in A^*$. If one of them is a yes-instance with $(k-1)$-apex set S of $G \setminus y$, then (G, k) is a yes-instance with k-apex set $S \cup \{y\}$ of G. If all of them are no-instances, then report a no-instance. This concludes the case where each vertex of A^* is adjacent to vertices of q p-internal bags of $\tilde{\mathcal{Q}}$.

If for every subset A^* of $\tilde{A}$ of size $a_{\mathcal{F}}$, there is a vertex of A^* that is not adjacent to vertices of q p-internal bags of the given $\tilde{\mathcal{Q}}$, then report a no-instance. This concludes the case that $a_{\mathcal{F}} \leq |\tilde{A}| \leq k+b$.

If for every wall, $|\tilde{A}| > k+b$, then report that (G, k) is a no-instance of $\mathcal{F}$-M-Deletion.

Notice that Step 3, when applied, takes time $2^{\mathcal{O}_{\ell_{\mathcal{F}}}(k^{c+2} \log k)} \cdot n^2$, because we apply the flow algorithms to each of the $2^{\mathcal{O}_{\ell_{\mathcal{F}}}(k^{c+2} \log k)}$ r'_2-subwalls and for each vertex of G. However, the search tree created by the branching algorithm has at most $a_{\mathcal{F}}$ branches and depth at most k. So Step 3 cannot be applied more than $a_{\mathcal{F}}{}^k$ times during the course of the algorithm. Since Step 1 runs in time $2^{\mathcal{O}_{\ell_{\mathcal{F}}}(k^{2(c+2)})} \cdot n$, Step 2 runs in time $2^{\mathcal{O}_{\ell_{\mathcal{F}}}(k^{2c})} \cdot n$, and both may be applied at most n times, the claimed time complexity follows: the algorithm runs in time $2^{\mathcal{O}_{\ell_{\mathcal{F}}}(k^{2(c+2)})} \cdot n^2$.

4.2 Correctness of the algorithm

Suppose first that (G, k) is a yes-instance and let S be a k-apex set of G. The application of the algorithm `Find-Wall` of Proposition 6 with input (G, r_1, k) either returns a report that $\mathsf{tw}(G) \leq f_1(s_{\mathcal{F}}) \cdot r_1 + k$ or returns an r_1-wall. In the first case, i.e., if $\mathsf{tw}(G) \leq f_1(s_{\mathcal{F}}) \cdot r_1 + k$,

the application of the algorithm of Proposition 7 correctly outputs a k-apex set of G. We will focus on the latter case, i.e., where the algorithm `Find-Wall` returns an r_1-wall of G, say W_1. Since $r_1 \geq f_2(s_{\mathcal{F}}) \cdot r_2' + k$, there is an $(f_2(s_{\mathcal{F}}) \cdot r_2')$-subwall of W_1, say W_1^*, that does not contain vertices of S. Since $G \setminus S$ does not contain $K_{s_{\mathcal{F}}}$ as a minor, there is no model of $K_{s_{\mathcal{F}}}$ grasped by W_1^* and therefore, due to Proposition 8 with input $(G \setminus S, r_2', s_{\mathcal{F}}, W_1^*)$, we know that there is a set $B \subseteq V(G \setminus S)$, with $|B| \leq b$, and a flatness pair $(W_2', \mathfrak{R}_2')$ of $G \setminus (S \cup B)$ of height r_2' such that W_2' is a W''-tilt of some subwall W'' of W_1^*.

Let $\mathcal{Q}$ be the canonical partition of W_2'. Let G' be the graph obtained by contracting each bag Q of $\mathcal{Q}$ to a single vertex v_Q, and adding a new vertex v_{all} and making it adjacent to all v_Q's. Let $\tilde{A}$ be the set of vertices y of $G \setminus V(W_2')$ such that there are q internally vertex-disjoint paths from v_{all} to y in G'. We claim that $\tilde{A} \subseteq S \cup B$. To show this, we first prove that, for every $y \notin S \cup B$, the maximum number of internally vertex-disjoint paths from v_{all} to y in G' is $k + b + 4$. Indeed, if y is a vertex in the $\mathfrak{R}_2'$-compass of W_2', there are at most $k + b$ such paths that intersect the set $S \cup B$ and at most four paths that do not intersect $S \cup B$ (in the graph $G' \setminus (S \cup B)$) due to the fact that $(W_2', \mathfrak{R}_2')$ is a flatness pair of $G \setminus (S \cup B)$. If y is not a vertex in the $\mathfrak{R}_2'$-compass of W_2', then, since by the definition of flatness pairs the perimeter of W_2' together with the set $S \cup B$ separate y from the $\mathfrak{R}_2'$-compass of W_2', every collection of internally vertex-disjoint paths from v_{all} to y in G' should intersect the set $\{v_{Q_{\text{ext}}}\} \cup S \cup B$, where Q_{ext} is the external bag of $\mathcal{Q}$. Therefore, in both cases, if $y \notin S \cup B$, the maximum number of internally vertex-disjoint paths from v_{all} to y in G' is $k + b + 4$. Since $k + b + 4 < q$, we have that $y \notin \tilde{A}$. Hence, $\tilde{A} \subseteq S \cup B$ and therefore $|\tilde{A}| \leq k + b$. Hence, if (G, k) is a yes-instance we cannot have that $|\tilde{A}| > k + b$, so the algorithm correctly reports a no-instance at the end of Step 3.

Let $\tilde{\mathcal{Q}}$ be a W_2'-canonical partition of $G \setminus (S \cup B)$ obtained by enhancing $\mathcal{Q}$ on $G \setminus (S \cup B)$. Let $\tilde{A}'$ be the set of vertices in $S \cup B$ that are adjacent to vertices of at least q p-internal bags of $\tilde{\mathcal{Q}}$ (recall that $\tilde{A}$ is the set of vertices in $S \cup B$ that are adjacent to vertices of at least q internal bags of $\tilde{\mathcal{Q}}$). Note that $\tilde{A}' \subseteq \tilde{A}$ and therefore $|\tilde{A}'| \leq |\tilde{A}|$.

If $|\tilde{A}'| < a_{\mathcal{F}}$, then at most $a_{\mathcal{F}} - 1$ vertices of $S \cup B$ are adjacent to vertices of at least q p-internal bags of $\tilde{\mathcal{Q}}$. This means that the p-internal bags of $\tilde{\mathcal{Q}}$ that contain vertices adjacent to some vertex of $(S \cup B) \setminus \tilde{A}'$ are at most $(q-1) \cdot (k+b) = l$.

Consider a family $\mathcal{W} = \{W^1, \ldots, W^{l+1}\}$ of $l+1$ r_2-subwalls of W_2' such that for every $i \in [l+1]$, $\bigcup \mathsf{influence}_{\mathfrak{R}_2'}(W^i)$ is a subgraph of $\bigcup\{Q \mid Q \text{ is a } p\text{-internal bag of } \tilde{\mathcal{Q}}\}$ and for every $i, j \in [l+1]$, with $i \neq j$, there is no internal bag of $\tilde{\mathcal{Q}}$ that contains vertices of both $V(\bigcup \mathsf{influence}_{\mathfrak{R}_2'}(W^i))$ and $V(\bigcup \mathsf{influence}_{\mathfrak{R}_2'}(W^j))$. The existence of $\mathcal{W}$ follows from Proposition 13 and the fact that $r_2' \geq f_{12}(l+1, r_2, p)$.

The fact that the p-internal bags of $\tilde{\mathcal{Q}}$ that contain vertices adjacent to some vertex of $(S \cup B) \setminus \tilde{A}'$ are at most l implies that there exists an $i \in [l+1]$ such that no vertex of $V(\bigcup \mathsf{influence}_{\mathfrak{R}_2'}(W^i))$ is adjacent, in G, to a vertex in $(S \cup B) \setminus \tilde{A}'$. Let $W_2 := W^i$, let W_2^* be the central $(r_2 - 2)$-subwall of W_2, and let D_{W_2} be the graph obtained from G by removing the perimeter of W_2 and taking the connected component that contains W_2^*. Since no vertex of $V(\bigcup \mathsf{influence}_{\mathfrak{R}_2'}(W^i))$ is adjacent, in G, to a vertex in $(S \cup B) \setminus \tilde{A}'$, any path in D_{W_2} going from a vertex of W_2^* to a vertex in S must intersect a vertex of $\tilde{A}'$. Thus, there is no model of $K_{s_{\mathcal{F}}+a_{\mathcal{F}}-1}$ grasped by W_2^* in D_{W_2}, because otherwise $K_{s_{\mathcal{F}}}$ would be a minor of $G \setminus S$. So, by applying the algorithm `Grasped-or-Flat` of Proposition 8 with input $(D_{W_2}, r_3, s_{\mathcal{F}} + a_{\mathcal{F}} - 1, W_2^*)$, since $r_2 - 2 \geq f_2(s_{\mathcal{F}} + a_{\mathcal{F}} - 1) \cdot r_3$, we should find a set $A \subseteq V(D_{W_2})$ with $|A| \leq a$ and a flatness pair $(W_3, \mathfrak{R}_3)$ of $D_{W_2} \setminus A$ of height r_3, such that W_3 is a tilt of some subwall $\tilde{W}_3$ of W_2.

Let $\tilde{\mathcal{Q}}'$ be a W_3-canonical partition of $D_{W_2} \setminus A$. Let $\mathcal{W}' = \{W^1, ..., W^{a_{\mathcal{F}}}\}$ be a collection of r_4-subwalls of W_3 such that for every $i \in [a_{\mathcal{F}}]$, $\bigcup \mathsf{influence}_{\mathfrak{R}_3}(W^i)$ is a subgraph of $\bigcup\{Q \mid Q \text{ is an internal bag of } \tilde{\mathcal{Q}}'\}$ and for every $i, j \in [a_{\mathcal{F}}]$, with $i \neq j$, there is no internal bag of $\tilde{\mathcal{Q}}'$ that contains vertices of both $V(\bigcup \mathsf{influence}_{\mathfrak{R}_3}(W^i))$ and $V(\bigcup \mathsf{influence}_{\mathfrak{R}_3}(W^j))$. Since $|\tilde{A}'| < a_{\mathcal{F}}$, there is an $i \in [a_{\mathcal{F}}]$ such that $V(\bigcup \mathsf{influence}_{\mathfrak{R}_3}(W^i))$ does not intersect $\tilde{A}'$. The existence of $\mathcal{W}'$ follows from Proposition 13 and the fact that $r_3 \geq f_{12}(a_{\mathcal{F}}, r_4, 1)$.

Let $W_4 := W^i$. Let W_4^* be the central $(r_4 - 2)$-subwall of W_4 and let D_{W_4} be the graph obtained from D_{W_2} after removing A and the perimeter of W_4 and taking the connected component containing W_4^*. Observe that any path between a vertex of S and a vertex of $V(\bigcup \mathsf{influence}_{\mathfrak{R}_3}(W_4))$ in D_{W_2} intersects $\tilde{A}'$. Since $\tilde{A}'$ does not intersect $V(\bigcup \mathsf{influence}_{\mathfrak{R}_3}(W_4))$, it implies that $\tilde{A}'$ does not intersect D_{W_4}, and thus $S \cap D_{W_4} = \emptyset$. Therefore, D_{W_4} is a minor of $G \setminus S$ and $K_{s_{\mathcal{F}}}$ is not a minor of D_{W_4}. Moreover, W_4^* is a wall of D_{W_4} of height $r_4 - 2 \geq t + 1$, so $\mathsf{tw}(D_{W_4}) > t = f_4(s_{\mathcal{F}}) \cdot r_5$. Therefore, by applying the algorithm `Clique-or-twFlat` of Proposition 9 with input $(D_{W_4}, r_5, s_{\mathcal{F}})$, we should obtain a set A' of size at most b and a regular flatness pair $(W_5, \mathfrak{R}_5)$ of $D_{W_4} \setminus A'$ of height r_5 whose $\mathfrak{R}_5$-compass has treewidth at most t. All this is checked in Step 1, and thus, the algorithm should run Step 2.

If $|\tilde{A}'| \geq a_{\mathcal{F}}$, then, due to Proposition 12 and the fact that $r_2' \geq f_9(a_{\mathcal{F}}, s_{\mathcal{F}}, k)$, for any set $X \subseteq V(G)$ such that $\mathsf{bid}_{G\setminus(S\cup B),W_2'}(X) \leq k$ and such that $G \setminus X \in \mathsf{exc}(\mathcal{F})$, it holds that $X \cap \tilde{A}' \neq \emptyset$. In particular, for any k-apex set S', $\mathsf{bid}_{G\setminus(S\cup B),W_2'}(S') \leq |S'| \leq k$, and thus $S' \cap \tilde{A}' \neq \emptyset$. Thus, there is a vertex $y \in \tilde{A}'$ such that $(G \setminus y, k-1)$ is a yes-instance. Hence, if the algorithm runs Step 3, it finds a vertex $y \in \tilde{A}'$ such that $(G \setminus y, k-1)$ is a yes-instance.

Note that the enhancement $\tilde{\mathcal{Q}}$ of the canonical partition $\mathcal{Q}$ is not unique. In particular, $\tilde{A}'$ depends on $\tilde{\mathcal{Q}}$. However, as long as there is such a $\tilde{\mathcal{Q}}$ such that $|\tilde{A}'| < a_{\mathcal{F}}$, the algorithm finds the wanted flatness pair $(W_4, \mathfrak{R}_4)$ in Step 1 and then runs Step 2. Hence, if (G, k) is a yes-instance, the algorithm runs Step 3 only if for all such $\tilde{A}'$, $|\tilde{A}'| \geq a_{\mathcal{F}}$. Note that, since $|\tilde{A}| \geq |\tilde{A}'|$, in this case we have that, for all such $\tilde{A}'$, $|\tilde{A}| \geq a_{\mathcal{F}}$. This justifies the arbitrary canonical partition enhancement in Step 3 and the fact that, if $|\tilde{A}| < a_{\mathcal{F}}$ in Step 3, then the algorithm reports a no-instance.

Let us now show the correctness of Step 2, and for this we do not suppose anymore that (G, k) is a yes-instance since the argument is the same for both types of instances. Suppose that the algorithm finds in Step 1 a set A' of size at most b and a regular flatness pair $(W_5, \mathfrak{R}_5)$ of $D_{W_4} \setminus A'$ of height r_5 whose $\mathfrak{R}_5$-compass has treewidth at most t. We obtain a 7-tuple $\mathfrak{R}_5'$ by adding all vertices of $G \setminus V(\mathsf{Compass}_{\mathfrak{R}_5}(W_5))$ to the set in the first coordinate of $\mathfrak{R}_5$. Since $(W_5, \mathfrak{R}_5)$ is a regular flatness pair of $D_{W_4} \setminus A'$ and since the vertices added in $\mathfrak{R}_5'$ are either in A, or adjacent at most to the perimeter of W_4, then $(W_5, \mathfrak{R}_5')$ is a regular flatness pair of $G \setminus (A \cup A')$. Since $\mathsf{Compass}_{\mathfrak{R}_5}(W_5) = \mathsf{Compass}_{\mathfrak{R}_5'}(W_5)$, $\mathsf{Compass}_{\mathfrak{R}_5'}(W_5)$ has treewidth at most t. Thus, if we apply the algorithm `Homogeneous` of Proposition 10 with input $(r_6, a+b, a+b, d, t, G, A \cup A', W_5, \mathfrak{R}_5')$, we obtain a flatness pair $(W_6, \mathfrak{R}_6)$ of $G \setminus (A \cup A')$ of height r_6 that is d-homogeneous with respect to $2^{A \cup A'}$ and is a W^*-tilt of $(W_5, \mathfrak{R}_5')$ for some subwall W^* of W_5. Since $|A \cup A'| \leq a + b$, for any set $X \subseteq V(G)$, $|A \setminus X| \leq a + b$. Since $G \setminus S \in \mathsf{exc}(\mathcal{F})$ and $\mathsf{bid}_{G\setminus(A\cup A'),W_6}(S) \leq |S| \leq k$, by applying the algorithm `Find-Irrelevant Vertex` of Proposition 11 with input $(k, a+b, G, A \cup A', W_6, \mathfrak{R}_6)$, we obtain a vertex v such that $G \setminus S \in \mathsf{exc}(\mathcal{F})$ if and only if $G \setminus (S \setminus v) \in \mathsf{exc}(\mathcal{F})$. It follows that (G, k) and $(G \setminus v, k)$ are indeed equivalent instances of $\mathcal{F}$-M-Deletion.

We now suppose that (G, k) is a no-instance. In the beginning of Step 1, the algorithm either reports a no-instance or finds a wall. In the latter case, the algorithm either goes to Step 2 or Step 3. If it runs Step 2, the previous paragraph justifies that the algorithm

finds a vertex v such that $(G \setminus v, k)$ is a no-instance. If the algorithm runs Step 3, then it either reports a no-instance or recursively runs on instances $(G \setminus y, k-1)$. If $(G \setminus y, k-1)$ is yes-instance, then so is (G, k). Thus, $(G \setminus y, k-1)$ is a no-instance for every considered vertex y and the algorithm always reports a no-instance. Hence, Theorem 1 follows.

5 Concluding remarks

For a minor-closed graph class $\mathcal{G}$, we proved that Vertex Deletion to $\mathcal{G}$ can be solved in time $2^{\mathsf{poly}(k)} \cdot n^2$ and that Elimination Distance to $\mathcal{G}$ can be solved in time $2^{2^{2^{\mathsf{poly}(k)}}} \cdot n^2$, and in time $2^{2^{c \cdot k^2 \log k}} \cdot n^2$ and $2^{\mathsf{poly}(k)} \cdot n^3$ in the case where the obstruction set of $\mathcal{G}$ contains an apex-graph. Here the degree of poly and c heavily depend on the size of the obstructions of $\mathcal{G}$. An open question is whether $\mathsf{poly}(k)$ could be replaced by $c \cdot k^d$ for some constant c depending on $\mathcal{G}$ and some *universal* constant d (independent of $\mathcal{G}$). We tend to believe that this dependence on $\mathcal{G}$ in the exponent of the polynomial is unavoidable, at least if we want to use the irrelevant vertex technique, and specially our definition of homogeneity.

On the other hand, we are not aware, for any of the two considered problems, of any lower bound, assuming the Exponential Time Hypothesis [25], stronger than $2^{o(k)} \cdot n^{\mathcal{O}(1)}$, which follows quite easily from known results for Vertex Cover. Proving stronger lower bounds seems to be quite challenging.

Another open problem is whether it is possible to drop the time complexity of Elimination Distance to $\mathcal{G}$ to $2^{\mathsf{poly}(k)} \cdot n^2$ for *every* minor-closed graph class $\mathcal{G}$. We tend to believe that this should be possible. However, it seems to require to use branching ingeniously and, in particular, to find equivalent instances of Elimination Distance to $\mathcal{G}$ with a decreasing value of k.

As for the polynomial running time of our FPT-algorithms, a priori, nothing prevents the existence of algorithms running in *linear time*, although we are quite far from achieving this. Kawarabayashi [30] presented such a linear FPT-algorithm for the Planarization problem, heavily relying on the embedding on the resulting planar graph. Extending this technique to general minor-closed classes would require a very compact encoding of the (entangled) structure of minor-free graphs [48] that would be possible to handle in linear time.

References

1 Isolde Adler, Martin Grohe, and Stephan Kreutzer. Computing excluded minors. In *Proc. of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 641–650, 2008. URL: http://dl.acm.org/citation.cfm?id=1347082.1347153.

2 Akanksha Agrawal, Lawqueen Kanesh, Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Deleting, Eliminating and Decomposing to Hereditary Classes Are All FPT-Equivalent. In *Proc. of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1976–2004, 2022. doi:10.1137/1.9781611977073.79.

3 Akanksha Agrawal, Lawqueen Kanesh, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. An FPT algorithm for elimination distance to bounded degree graphs. In *Proc. of the 38th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 187 of *LIPIcs*, pages 5:1–5:11, 2021. doi:10.4230/LIPIcs.STACS.2021.5.

4 Akanksha Agrawal and M. S. Ramanujan. On the parameterized complexity of clique elimination distance. In *Proc. of the 15th International Symposium on Parameterized and Exact Computation (IPEC)*, volume 180 of *LIPIcs*, pages 1:1–1:13, 2020. doi:10.4230/LIPIcs.IPEC.2020.1.

5 Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos. A complexity dichotomy for hitting connected minors on bounded treewidth graphs: the chair and the banner draw the boundary. In *Proc. of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 951–970, 2020. doi:10.1137/1.9781611975994.57.

6 Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos. Hitting minors on bounded treewidth graphs. I. General upper bounds. *SIAM Journal on Discrete Mathematics*, 34(3):1623–1648, 2020. doi:10.1137/19M1287146.

7 Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos. Hitting minors on bounded treewidth graphs. II. Single-exponential algorithms. *Theoretical Computer Science*, 814:135–152, 2020. doi:10.1016/j.tcs.2020.01.026.

8 Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos. Hitting minors on bounded treewidth graphs. III. Lower bounds. *Journal of Computer and System Sciences*, 109:56–77, 2020. doi:10.1016/j.jcss.2019.11.002.

9 Hans L. Bodlaender, John R. Gilbert, Ton Kloks, and Hjálmtyr Hafsteinsson. Approximating treewidth, pathwidth, and minimum elimination tree height. In *Proc. of the 17th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 570 of *LNCS*, pages 1–12, 1991. doi:10.1007/3-540-55121-2_1.

10 Jannis Bulian and Anuj Dawar. Graph isomorphism parameterized by elimination distance to bounded degree. *Algorithmica*, 75(2):363–382, 2016. doi:10.1007/s00453-015-0045-3.

11 Jannis Bulian and Anuj Dawar. Fixed-parameter tractable distances to sparse graph classes. *Algorithmica*, 79(1):139–158, 2017. doi:10.1007/s00453-016-0235-7.

12 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

13 Reinhard Diestel. *Graph Theory*, volume 173. Springer-Verlag, 5th edition, 2017. doi:10.1007/978-3-662-53622-3.

14 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

15 Zdeněk Dvořák, Archontia C. Giannopoulou, and Dimitrios M. Thilikos. Forbidden graphs for tree-depth. *European Journal of Combinatorics*, 33(5):969–979, 2012. doi:10.1016/j.ejc.2011.09.014.

16 Eduard Eiben, Robert Ganian, Thekla Hamm, and O-joung Kwon. Measuring what matters: A hybrid approach to dynamic programming with treewidth. *Journal of Computer and System Sciences*, 121:57–75, 2021. doi:10.1016/j.jcss.2021.04.005.

17 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. doi:10.1007/3-540-29953-X.

18 Fedor V. Fomin, Petr A. Golovach, and Dimitrios M. Thilikos. Parameterized complexity of elimination distance to first-order logic properties. *ACM Transactions on Computational Logic*, 23(3):17:1–17:35, 2022. doi:10.1145/3517129.

19 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar $\mathcal{F}$-deletion: Approximation, kernelization and optimal FPT algorithms. In *Proc. of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 470–479, 2012. doi:10.1109/FOCS.2012.62.

20 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Hitting topological minors is FPT. In *Proc. of the 52nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 1317–1326, 2020. doi:10.1145/3357713.3384318.

21 Archontia C. Giannopoulou, O-joung Kwon, Jean-Florent Raymond, and Dimitrios M. Thilikos. Lean tree-cut decompositions: Obstructions and algorithms. In *Proc. of the 36th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 126 of *LIPIcs*, pages 32:1–32:14, 2019. doi:10.4230/LIPIcs.STACS.2019.32.

22 Archontia C. Giannopoulou, Michal Pilipczuk, Jean-Florent Raymond, Dimitrios M. Thilikos, and Marcin Wrochna. Linear kernels for edge deletion problems to immersion-closed graph classes. In *Proc. of the 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 80 of *LIPIcs*, pages 57:1–57:15, 2017. doi:10.4230/LIPIcs.ICALP.2017.57.

23 Archontia C. Giannopoulou, Michal Pilipczuk, Jean-Florent Raymond, Dimitrios M. Thilikos, and Marcin Wrochna. Cutwidth: Obstructions and algorithmic aspects. *Algorithmica*, 81(2):557–588, 2019. doi:10.1007/s00453-018-0424-7.

24 Jiong Guo, Falk Hüffner, and Rolf Niedermeier. A structural view on parameterizing problems: Distance from triviality. In *Proc. of the 1st International Workshop on Parameterized and Exact Computation (IWPEC)*, volume 3162 of *LNCS*, pages 162–173, 2004. doi:10.1007/978-3-540-28639-4_15.

25 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

26 Bart M. P. Jansen, Jari J. H. de Kroon, and Michał Włodarczyk. Vertex deletion parameterized by elimination distance and even less. In *Proc. of the 53rd Annual ACM-SIGACT Symposium on Theory of Computing (STOC)*, pages 1757–1769, 2021. doi:10.1145/3406325.3451068.

27 Bart M. P. Jansen, Daniel Lokshtanov, and Saket Saurabh. A near-optimal planarization algorithm. In *Proc. of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1802–1811, 2014. doi:10.1137/1.9781611973402.130.

28 Mamadou Moustapha Kanté and O-joung Kwon. An upper bound on the size of obstructions for bounded linear rank-width, 2014. arXiv:1412.6201.

29 Mamadou Moustapha Kanté and O-joung Kwon. Linear rank-width of distance-hereditary graphs II. vertex-minor obstructions. *European Journal of Combinatorics*, 74:110–139, 2018. doi:10.1016/j.ejc.2018.07.009.

30 Ken-ichi Kawarabayashi. Planarity allowing few error vertices in linear time. In *Proc. of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 639–648, 2009. doi:10.1109/FOCS.2009.45.

31 Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce A. Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424–435, 2012. doi:10.1016/j.jctb.2011.07.004.

32 Ken-ichi Kawarabayashi, Robin Thomas, and Paul Wollan. A new proof of the flat wall theorem. *Journal of Combinatorial Theory, Series B*, 129:204–238, 2018. doi:10.1016/j.jctb.2017.09.006.

33 Ken-ichi Kawarabayashi and Paul Wollan. A Shorter Proof of the Graph Minor Algorithm: The Unique Linkage Theorem. In *Proc. of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 687–694, 2010. doi:10.1145/1806689.1806784.

34 Eun Jung Kim, Alexander Langer, Christophe Paul, Felix Reidl, Peter Rossmanith, Ignasi Sau, and Somnath Sikdar. Linear kernels and single-exponential algorithms via protrusion decompositions. *ACM Transactions on Algorithms*, 12(2):21:1–21:41, 2016. doi:10.1145/2797140.

35 Eun Jung Kim, Maria J. Serna, and Dimitrios M. Thilikos. Data-compression for parametrized counting problems on sparse graphs. In *Proc. of the 29th International Symposium on Algorithms and Computation (ISAAC)*, volume 123 of *LIPIcs*, pages 20:1–20:13, 2018. doi:10.4230/LIPIcs.ISAAC.2018.20.

36 Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994. doi:10.1007/BFb0045375.

37 Tomasz Kociumaka and Marcin Pilipczuk. Deleting Vertices to Graphs of Bounded Genus. *Algorithmica*, 81(9):3655–3691, 2019. doi:10.1007/s00453-019-00592-7.

38 Jens Lagergren. An upper bound on the size of an obstruction. In *Graph Structure Theory*, volume 147 of *Contemporary Mathematics*, pages 601–621. American Mathematical Society, 1991. doi:10.1090/conm/147/01202.

39 Jens Lagergren. Upper bounds on the size of obstructions and intertwines. *Journal of Combinatorial Theory, Series B*, 73:7–40, 1998. doi:10.1006/jctb.1997.1788.

40 Jens Lagergren and Stefan Arnborg. Finding minimal forbidden minors using a finite congruence. In *Proc. of the 18th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 510 of *LNCS*, pages 532–543, 1991. doi:10.1007/3-540-54233-7_161.

41 John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980. doi:10.1016/0022-0000(80)90060-4.

42 Dániel Marx and Ildikó Schlotter. Obtaining a planar graph by vertex deletion. *Algorithmica*, 62(3-4):807–822, 2012. doi:10.1007/s00453-010-9484-z.

43 Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity – Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-27875-4.

44 Rolf Niedermeier. *Invitation to fixed parameter algorithms*, volume 31. Oxford University Press, 2006. doi:10.1093/ACPROF:OSO/9780198566076.001.0001.

45 Alex Pothen. The complexity of optimal elimination trees. *Technical Report. Pennsylvania State University. Dept. of Computer Science*, 1988. URL: https://www.cs.purdue.edu/homes/apothen/Papers/shortest-etree1988.pdf.

46 Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, and Somnath Sikdar. A faster parameterized algorithm for treedepth. In *Proc. of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 8572 of *LNCS*, pages 931–942, 2014. doi:10.1007/978-3-662-43948-7_77.

47 Neil Robertson and Paul D. Seymour. Graph Minors. XIII. The Disjoint Paths Problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

48 Neil Robertson and Paul D. Seymour. Graph Minors. XVI. Excluding a non-planar graph. *Journal of Combinatorial Theory, Series B*, 89(1):43–76, 2003. doi:10.1016/S0095-8956(03)00042-X.

49 Neil Robertson and Paul D. Seymour. Graph Minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004. doi:10.1016/j.jctb.2004.08.001.

50 Ignasi Sau, Giannos Stamoulis, and Dimitrios M. Thilikos. A more accurate view of the Flat Wall Theorem, 2021. arXiv:2102.06463.

51 Ignasi Sau, Giannos Stamoulis, and Dimitrios M. Thilikos. k-apices of minor-closed graph classes. II. Parameterized algorithms. *ACM Transactions on Algorithms*, 18(3):21:1–21:30, 2022. Short version in *Proc. of the 47th International Colloquium on Automata, Languages and Programming (ICALP), volume 168 of LIPIcs, pages 95:1-95:20, 2020*. doi:10.1145/3519028.

52 Ignasi Sau, Giannos Stamoulis, and Dimitrios M. Thilikos. k-apices of minor-closed graph classes. I. Bounding the obstructions. *Journal of Combinatorial Theory, Series B*, 161:180–227, 2023. doi:10.1016/j.jctb.2023.02.012.

53 Andrew Thomason. The extremal function for complete minors. *Journal of Combinatorial Theory, Series B*, 81(2):318–338, 2001. doi:10.1006/jctb.2000.2013.

Nearly Tight Spectral Sparsification of Directed Hypergraphs

Kazusato Oko
Department of Mathematical Informatics, The University of Tokyo, Japan
Center for Advanced Intelligence Project, RIKEN, Tokyo, Japan

Shinsaku Sakaue
Department of Mathematical Informatics, The University of Tokyo, Japan

Shin-ichi Tanigawa
Department of Mathematical Informatics, The University of Tokyo, Japan

Abstract

Spectral hypergraph sparsification, an attempt to extend well-known spectral graph sparsification to hypergraphs, has been extensively studied over the past few years. For undirected hypergraphs, Kapralov, Krauthgamer, Tardos, and Yoshida (2022) have proved an ε-spectral sparsifier of the optimal $O^*(n)$ size, where n is the number of vertices and O^* suppresses the ε^{-1} and $\log n$ factors. For directed hypergraphs, however, the optimal sparsifier size has not been known. Our main contribution is the first algorithm that constructs an $O^*(n^2)$-size ε-spectral sparsifier for a weighted directed hypergraph. Our result is optimal up to the ε^{-1} and $\log n$ factors since there is a lower bound of $\Omega(n^2)$ even for directed graphs. We also show the first non-trivial lower bound of $\Omega(n^2/\varepsilon)$ for general directed hypergraphs. The basic idea of our algorithm is borrowed from the spanner-based sparsification for ordinary graphs by Koutis and Xu (2016). Their iterative sampling approach is indeed useful for designing sparsification algorithms in various circumstances. To demonstrate this, we also present a similar iterative sampling algorithm for undirected hypergraphs that attains one of the best size bounds, enjoys parallel implementation, and can be transformed to be fault-tolerant.

2012 ACM Subject Classification Theory of computation → Sparsification and spanners

Keywords and phrases Spectral sparsification, (Directed) hypergraphs, Iterative sampling

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.94

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2204.02537

Funding This work was supported by JST PRESTO Grant Number JPMJPR2126, JST ERATO Grant Number JPMJER1903, and JSPS KAKENHI Grant Number 20H05961.

1 Introduction

Graph sparsification is a fundamental idea for developing efficient algorithms and data structures. One of the earliest developments in this context is a cut sparsifier due to Benczúr and Karger [4], which approximately keeps the size of cuts (by adjusting edge weights). Spielman and Teng [24] introduced a generalized notion called a spectral sparsifier, which approximately preserves the spectrum of the Laplacian matrix of a given graph. Since this seminal work, spectral sparsification of graphs has been extensively studied and used in many applications. See, e.g., [27, 26, 23] for more details on spectral graph sparsification.

This paper studies spectral sparsification of undirected/directed hypergraphs. A hypergraph is a standard tool for generalizing graph-theoretic arguments in a set-theoretic setting, and extending a theory for graphs to hypergraphs is a common theoretical interest.

Besides, many hypergraph-based methods [12, 28, 25, 29, 31] have recently been attracting much attention as extensions of graph-based methods, which also increases the demand for advancing the theory of spectral hypergraph sparsification.

An *undirected hypergraph* is defined by a tuple $H = (V, F, z)$, where V is a finite vertex set, F is a set of subsets of V, and $z\colon F \to \mathbb{R}_+$. Each element in F is called a hyperedge and $z_f := z(f)$ is called the weight of $f \in F$ in H. The Laplacian $L_H\colon \mathbb{R}^V \to \mathbb{R}^V$ of H is defined as a nonlinear operator such that

$$x^\top L_H(x) = \sum_{f \in F} z_f \max_{u,v \in f} (x_u - x_v)^2 \quad \text{for all } x \in \mathbb{R}^V.$$

If x is restricted to $\{0,1\}^V$, $x^\top L_H(x)$ represents the cut function of H. In this sense, the above definition gives a proper extension of the ordinary graph Laplacian. (Here, $x^\top L_H(x)$ is an abuse of notation since $L_H(x)$ is not defined uniquely; nevertheless, this notation is widely used in analogy to the case of ordinary graphs.)

A *directed hypergraph* $H = (V, F, z)$ consists of a finite set V, a set F of hyperarcs, and $z\colon F \ni f \mapsto z_f \in \mathbb{R}_+$, where each *hyperarc* $f \in F$ is a pair $(t(f), h(f))$ of non-empty subsets of V, called the *tail* and the *head* (which may not be disjoint). The Laplacian $L_H\colon \mathbb{R}^V \to \mathbb{R}^V$ of H is defined as a nonlinear operator such that

$$x^\top L_H(x) = \sum_{f \in F} z_f \max_{u \in t(f), v \in h(f)} (x_u - x_v)_+^2 \quad \text{for all } x \in \mathbb{R}^V,$$

where $(\cdot)_+ = \max\{\cdot, 0\}$ (and $(\cdot)_+^2 = (\max\{\cdot, 0\})^2$). If $x \in \{0,1\}^V$, the definition of directed hypergraph Laplacian L_H also captures the cut function of H. Importantly, cut functions of directed hypergraphs can represent a large class of submodular functions [10].[1] Directed hypergraphs are also useful for modeling higher-order directional relations that appear in, e.g., propositional logic [11] and causal inference [14], which have constituted a motivation for studying spectral properties of directed hypergraphs [7].

Given an undirected/directed hypergraph $H = (V, F, z)$ and $\varepsilon \in (0, 1)$, a hypergraph $\tilde{H} = (V, \tilde{F}, \tilde{z})$ is called an *ε-spectral sparsifier* of H if it satisfies $\tilde{F} \subseteq F$ and

$$(1-\varepsilon) x^\top L_H(x) \le x^\top L_{\tilde{H}}(x) \le (1+\varepsilon) x^\top L_H(x) \quad \text{for all } x \in \mathbb{R}^V.$$

One of the big motivations for studying spectral sparsification of directed hypergraphs comes from the connection to the representation of submodular functions. Since such a cut-function representation uses $\Omega(2^{|V|})$ hyperarcs in general, a spectral sparsifier of a directed hypergraph can serve as a compact approximate representation (see the full version [20] for more details).

Soma and Yoshida [22] initiated the study of spectral hypergraph sparsification and gave an algorithm for constructing an ε-spectral sparsifier with $O(n^3 \log n/\varepsilon^2)$ hyperedges, where n is the number of vertices. Unlike ordinary graphs, the hypergraph size can be as large as 2^n (and 4^n if directed). Thus, obtaining a polynomial bound is already nontrivial. For undirected hypergraphs, the result by Soma and Yoshida [22] has been improved to $\tilde{O}(nr^3/\varepsilon^2)$ [3] and to $\tilde{O}(nr/\varepsilon^{O(1)})$ [15],[2] where r denotes the maximum size of a hyperedge in the input hypergraph H and is called the *rank* of H. Kapralov et al. [16] has removed the dependence on r and obtained a nearly linear bound of $\tilde{O}(n/\varepsilon^4)$. Very recently, an improved bound of $\tilde{O}(n/\varepsilon^2)$ has been shown in [13, 18] (concurrently to our work). This upper bound is nearly tight since the $\Omega(n/\varepsilon^2)$ lower bound applies even to ordinary graphs [2, 6].

[1] In fact, any set function can be represented as a cut function of some directed hypergraph if negative weights are allowed [10].

[2] We use $\tilde{O}$ to hide $\mathrm{poly}(\log(n/\varepsilon))$ factors.

Table 1 Bounds on sparsification of directed hypergraphs. In the time complexity, additive $\mathrm{poly}(n, 1/\varepsilon)$ terms are omitted. Note that Kapralov et al. [15] assume the unweighted case.

Method	Cut/Spectral	Bound	Time complexity
Soma and Yoshida [22]	Spectral	$O(n^3 \log n/\varepsilon^2)$	$O(mr^2)$
Kapralov et al. [15]	Spectral	$O(n^2 r^3 \log^2 n/\varepsilon^2)$	$O(mr^2)$
Rafiey and Yoshida [21]	Cut	$O(n^2 r^2/\varepsilon^2)$	$O(m2^r)$
This paper	Spectral	$O(n^2 \log^3(n/\varepsilon)/\varepsilon^2)$	$O(mr^2)$

As for spectral sparsification of directed hypergraphs, Soma and Yoshida [22] showed that their algorithm is also applicable, and hence the $O(n^3 \log n/\varepsilon^2)$ bound also holds for directed hypergraphs. Later, Kapralov et al. [15] gave an $O(n^2 r^3 \log^2 n/\varepsilon^2)$ bound for unweighted directed hypergraphs, where the rank r is defined by $r = \max_{f \in F}\{|h(f)| + |t(f)|\}$ in the directed case. Recently, for the case of cut sparsification, Rafiey and Yoshida [21] obtained sparsifiers with $O(n^2 r^2/\varepsilon^2)$ hyperarcs.[3] See Table 1. On the other hand, a well-known $\Omega(n^2)$ lower bound for directed graphs [9] is valid for directed hypergraphs. Therefore, a central open question in this context is: *can we obtain an upper bound of $\tilde{O}(n^2/\varepsilon^{O(1)})$ that has no dependence on the rank r?*

1.1 Main Results and Idea

Our main contribution is the first algorithm that constructs an ε-spectral sparsifier with $\tilde{O}(n^2/\varepsilon^2)$ hyperarcs for a directed hypergraph, thus settling the aforementioned question.

▶ **Theorem 1.** *Let $H = (V, F, z)$ be a directed hypergraph with n vertices. For any $\varepsilon \in (0, 1)$, our algorithm (shown in Algorithm 3) returns an ε-spectral sparsifier $\tilde{H} = (V, \tilde{F}, \tilde{z})$ of H such that $|\tilde{F}| = O\left(\frac{n^2}{\varepsilon^2} \log^3 \frac{n}{\varepsilon}\right)$ with probability at least $1 - O\left(\frac{1}{n}\right)$. Its time complexity is $O(mr^2)$ with probability at least $1 - O\left(\frac{1}{n}\right)$, where $m = |F|$ and r is the rank of H.*

This bound improves the previous results and is optimal up to the ε^{-1} and logarithmic factors due to the presence of the $\Omega(n^2)$ lower bound for directed graphs. We prove Theorem 1 in Section 4 by providing a concrete algorithm and its analysis.

A natural next question would be whether the ε^{-1} term can be deleted. Our new lower bound shows that the ε^{-1} term is indeed necessary, and an ε-spectral sparsifier of size $O(n^2)$ may not exist in general, thus complementing our upper bound.

▶ **Theorem 2.** *Let $n \in \mathbb{Z}_{>0}$. For any $\varepsilon \in \left(\frac{1}{4n}, 1\right)$, there is a directed hypergraph $H = (V, F, z)$ with $2n$ vertices, $\Omega\left(\frac{n^2}{\varepsilon}\right)$ hyperarcs, and the rank three that has no sub-hypergraph $\tilde{H} = (V, \tilde{F}, \tilde{z})$ such that $\tilde{F} \subsetneq F$ and $(1-\varepsilon)x^\top L_H(x) \le x^\top L_{\tilde{H}}(x) \le (1+\varepsilon)x^\top L_H(x)$ for all $x \in \{0, 1\}^V$.*

This gives a lower bound even for the case of cut sparsification and is the first nontrivial lower bound for sparsification of directed hypergraphs. We give the proof in the full version [20].

The basic idea of our algorithm for Theorem 1 comes from a spanner-based sparsification method for undirected graphs by Koutis and Xu [17], in contrast to the method of [16] for nearly tight sparsification of undirected hypergraphs. The analysis of [16] uses a technique called *weight assignment* [8], which crucially depends on linear algebraic arguments on the

[3] This bound follows from their general result on sparsification of submodular functions.

linear Laplacian of some underlying undirected graph. *Directed* hypergraphs, however, do not have such convenient underlying *undirected* graphs, and hence their idea cannot be utilized. We thus take an alternative route and use the algorithmic framework of Koutis and Xu [17] – iteratively select important edges and sample the remaining edges. Due to its combinatorial nature, we can analyze errors via combinatorial arguments instead of linear algebraic tools. Although our algorithm is as simple as theirs, our analysis for proving Theorem 1 involves novel techniques. Specifically, while building on a recent chaining-based analysis [15, 16], we develop a completely new *discretization scheme* based on a non-trivial combinatorial observation to obtain the optimal upper bound. See Section 3 for an overview of our analysis.

1.2 Additional Results

We also present the following additional results in the full version [20].

Undirected hypergraph sparsification. The iterative sampling approach mentioned above indeed has much potential in hypergraph sparsification. We exhibit its power by presenting a natural extension of the spanner-based algorithm by Koutis and Xu [17] to undirected hypergraphs. The concept of spanners in graphs can be naturally extended to undirected hypergraphs, and accordingly, Koutis and Xu's algorithm can also be extended to undirected hypergraphs. Based on a result by Bansal et al. [3], we show that the resulting algorithm constructs an ε-spectral sparsifier with $O\left(\frac{nr^3}{\varepsilon^2}\log^2 n\right)$ hyperedges, which is nearly optimal if r is constant and matches the bound of [3] (up to a $\log n$ factor). Moreover, our algorithm inherits advantages of the spanner-based approach in that it can be implemented in parallel [17] and can be converted to be fault-tolerant [32], demonstrating that the iterative sampling approach can enjoy various useful extensions.

Application to learning of submodular functions. A notable application of directed hypergraph sparsification due to [22] is agnostic learning of submodular functions. We apply our method to this setting and obtain an $\tilde{O}\left(\frac{n^3}{\varepsilon^4}+\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right)$ sample complexity bound for agnostic learning of nonnegative hypernetwork-type submodular functions on a ground set of size n, improving the previous $\tilde{O}\left(\frac{n^4}{\varepsilon^4}+\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right)$ bound in [22]. Note that since the rank r of a hypergraph representing a submodular function can be $O(n)$, eliminating the dependence on r in the sparsifier size (i.e., our improvement from [15]) is crucial in this application. It should be mentioned that this application only requires cut sparsifiers. Nevertheless, since our result gives the first near-optimal bound even on the size of cut sparsifiers of directed hypergraphs, this application serves as a good motivation for our result.

1.3 Related Work

Besides the aforementioned application to agnostic learning of submodular functions, there are many other potential applications that involve the quadratic form $x^\top L_H(x)$ (which is sometimes called the *energy* of hypergraphs), e.g., clustering [25], semi-supervised learning [12, 28, 31, 19], and link prediction [29]. For example, Li et al. [19] use the quadratic form as a smoothness regularizer. Our result on spectral sparsification can be useful when dealing with such regularizers on dense directed hypergraphs.

Cohen et al. [9] studied directed graph sparsification under a different definition of approximation based on *Eulerian scaling*. While their definition is compatible with fast Laplacian solvers, how to extend it to directed hypergraphs seems non-trivial. Our definition is based on a general notion called *submodular transformations* [30] and admits a natural interpretation as a generalization of cut sparsification of directed hypergraphs.

2 Preliminaries

We usually denote a directed hypergraph by $H = (V, F, z)$, the numbers of vertices by n, and the numbers of hyperarcs by m. The Laplacian $L_H \colon \mathbb{R}^V \to \mathbb{R}^V$ is defined as a nonlinear operator that satisfies $x^\top L_H(x) = \sum_{f \in F} z_f \max_{u \in t(f), v \in h(f)} (x_u - x_v)_+^2$ for all $x \in \mathbb{R}^V$, where $h(f), t(f) \subseteq V$ are the head and the tail of f, respectively. For each $f \in F$, we denote the contribution of f to $x^\top L_H(x)$ by $Q_H^x(f) = z_f \max_{u \in t(f), v \in h(f)} (x_u - x_v)_+^2$, which we call the *energy* of f. Note that $x^\top L_H(x) = \sum_{f \in F} Q_H^x(f)$ holds. For any subset F' of F, we let $Q_H^x(F') = \sum_{f \in F'} Q_H^x(f)$, i.e., the sum of energies over F'. For a hyperarc $f \in F$, we define its *biclique* as an arc set $C(f) = \{ (u, v) \mid u \in t(f), v \in h(f) \}$. For a subset $F' \subseteq F$, we let $C(F') = \bigcup_{f \in F'} C(f)$. Below, we often take $\operatorname{argmax}_{f \in F'} \zeta(f)$ for a function $\zeta \colon F \to \mathbb{R}$ and a hyperarc subset $F' \subseteq F$. For convenience, we let such argmax (or argmin) operations always return a singleton by using some tie-breaking rule with a pre-defined total order on F. For example, if vertices are labeled by $1, \dots, n$ and each $f \in F$ is labeled by vertices in f, we may use the lexicographical order on F with respect to the labels. Similarly, we break ties when taking argmax/argmin on any $E' \subseteq V \times V$. We will often use the following Chernoff bound.

▶ **Proposition 3** ([1]). *Let $X_1, X_2, \cdots, X_m$ be independent random variables in the range of $[0, a]$. For any $\delta \in [0, 1]$ and $\mu \geq \mathbb{E}[\sum_{i=1}^m X_i]$, we have*

$$\mathbb{P}\left[\left|\sum_{i=1}^m X_i - \mathbb{E}\left[\sum_{i=1}^m X_i\right]\right| > \delta\mu\right] \leq 2\exp\left(-\frac{\delta^2\mu}{3a}\right).$$

3 Technical Overview

Our algorithm is an iterative algorithm whose each step goes as follows: given a hypergraph $H = (V, F, z)$ from a previous iteration, it constructs a set S of heavy hyperarcs, called a *coreset*, which is kept deterministically in this step, and samples the remaining hyperarcs with probability $1/2$, where weights of sampled ones are doubled. This single step yields a hypergraph with fewer hyperarcs, which is taken as input in the next step. We iterate this until a sub-hypergraph of the desired size is obtained. Roughly speaking, the size of the coreset is about $\tilde{O}(n^2/\varepsilon^2)$, and after about $O(\log(m\varepsilon^2/n^2))$ iterations, we obtain a sub-hypergraph of size $\tilde{O}(n^2/\varepsilon^2)$. This algorithmic framework is identical to that of Koutis and Xu [17] for ordinary undirected graph sparsification, which iteratively constructs a bundle of spanners (instead of a coreset) and sample the remaining edges with probability $1/4$.

We then describe how to analyze the sparsification error. Note that if a sub-hypergraph produced in each step is a sparsifier of a hypergraph $H = (V, F, z)$ given from the previous step with a sufficiently large probability, then we can bound the error accumulated over the iterations. Thus, we focus on the analysis of a single step (which is presented in Lemma 6). To bound the sparsification error in $Q_H^x(F) = x^\top L_H(x)$ for all $x \in \mathbb{R}^V$ in each step, we adopted a chaining-type argument [15, 16]; this enables us to derive a desired uniform bound on a continuous domain from a pointwise bound via adaptive scaling of the domain discretization. Here, how to design a discretization scheme crucially affects how sharp the resulting uniform bound is. Therefore, we need to design an appropriate discretization scheme by carefully looking at the structure of directed hypergraphs.

We below sketch our discretization scheme. Inspired by the previous studies [15, 16], we classify hyperarcs $f \in F \setminus S$ based on their energies $Q_H^x(f)$. Here, since the coreset S is always selected, we can exclude it when discussing the following probabilistic arguments. For

each $x \in \mathbb{R}^V$, we consider a partition of $F \setminus S$ into F_i^x $(i \in \mathbb{Z})$ such that each F_i^x consists of hyperarcs f with energies $Q_H^x(f) \approx 2^{-i} Q_H^x(F)$. Then, the Chernoff bound offers the following pointwise guarantee for each $x \in \mathbb{R}^V$:

$$\mathbb{P}\big[|Q_{\tilde{H}}^x(\tilde{F}_i^x) - Q_H^x(F_i^x)| \geq \varepsilon Q_H^x(F)\big] \lesssim \exp\left(-\frac{\varepsilon^2 Q_H^x(F)}{2^{-i} Q_H^x(F)}\right) = \exp\big(-\varepsilon^2 2^i\big),$$

where $\tilde{H}$ is a sparsifier obtained from H and $Q_{\tilde{H}}^x(\tilde{F}_i^x)$ denotes the energy of $\tilde{H}$ with hyperarcs restricted to F_i^x. To obtain a desired uniform bound using this inequality, we need to design a discretization scheme that satisfies the following two requirements:

(R1) the discretization error is $O(\varepsilon)$, and

(R2) the number of possible discretized energies is bounded by about $\exp(\varepsilon^2 2^i)$.

Kapralov et al. [15] obtained such a scheme by looking at underlying clique digraphs. By contrast, we obtain a discretization scheme by directly looking at hypergraphs. This strategy enables us to eliminate the extra r^3 factor in their bound, but it also poses a new challenge.

We explain the challenge when designing such a discretization scheme by directly looking at hypergraphs. Once $x \in \mathbb{R}^V$ is fixed, the number of hyperarcs f with $Q_H^x(f) \approx 2^{-i} Q_H^x(F)$ is bounded by about 2^i; on the other hand, we need to prepare at least $\mathrm{poly}(n, 1/\varepsilon)$ possible discretized energies for each f to satisfy requirement (R1). Thus, naive counting implies that the number of total discretized energies for all $f \in F_i^x$ is $(\mathrm{poly}(n, 1/\varepsilon))^{2^i} \approx \exp(\tilde{O}(2^i))$, which is too large to satisfy requirement (R2). To overcome this problem, we need an additional combinatorial idea: we count the number of discretized energies by focusing on the number of possible critical pairs. We say that $(u, v) \in C(F \setminus S)$ is a *critical pair* of f if $(u, v) = \mathrm{argmax}_{u' \in t(f), v' \in h(f)} (x_{u'} - x_{v'})_+^2$ (see also Figure 1b). Suppose that a lot of hyperarcs in F_i^x share a common critical pair for a given $x \in \mathbb{R}^V$, particularly when F_i^x contains as many as 2^i hyperarcs. Then, since the energy of f is determined by the $(x_u - x_v)_+^2$ value of the critical pair (u, v) of f, we may get a sharper bound on the number of discretized energies by defining a discretization scheme based on $(x_u - x_v)_+^2$ values so that hyperarcs with the same critical pairs share the same discretized energies (up to scaling of weights).

To accomplish the counting based on this idea, we use the existence of a coreset kept in each iteration. As we will see shortly from the definition, a λ-coreset $S \subseteq F$ contains λ heaviest hyperarcs for each $(u, v) \in C(F)$ (see also Figure 1a). Roughly speaking, important properties of λ-coresets are as follows:

(P1) $|S| \leq \lambda n^2$,

(P2) for any fixed $x \in \mathbb{R}^V$, many hyperarcs with large energies are included in S, and

(P3) for any fixed $x \in \mathbb{R}^V$, the number of critical pairs of hyperarcs in F_i^x is at most $2^i/\lambda$.[4]

If we set $\lambda = \tilde{O}(\varepsilon^{-2})$, the size of λ-coreset S is $\tilde{O}(n^2/\varepsilon^2)$ by property (P1), which is small enough that the output size decreases geometrically in each iteration until we obtain an $\tilde{O}(n^2/\varepsilon^2)$ size sparsifier. Property (P2) bounds the range of i such that F_i^x is non-empty. Most importantly, property (P3) implies that if we count possible discretized energies over F_i^x, the total number is at most $(\mathrm{poly}(n, 1/\varepsilon))^{2^i/\lambda} \approx \exp\big(\tilde{O}(\varepsilon^2 2^i)\big)$, satisfying requirement (R2).

In summary, once the coreset is selected, we can categorize the remaining hyperarcs in each F_i^x based on a moderate number of critical pairs, which yields a sharp bound on the number of possible discretized energies of the remaining hyperarcs. This is the key idea of our discretization scheme, which, together with the chaining-type argument, provides the desired uniform bound on the sparsification error.

[4] For ease of exposition, λ is used differently from Section 4. In Section 4.2, we will instead define F_i^x based on $2^{-i} Q_H^x(F)/\lambda$ values and, accordingly, bound the number of critical pairs by 2^i (Lemma 11).

4 Spectral Sparsification of Directed Hypergraphs

We prove Theorem 1 by presenting a concrete algorithm. Section 4.1 presents our algorithm and key lemmas. Section 4.2 focuses on the analysis of a single iteration, and Section 4.3 bounds the overall sparsification error and the resulting sparsifier size, thus proving Theorem 1. Section 4.4 shows the $O(mr^2)$ time complexity bound of our algorithm.

4.1 Algorithm Description

Our algorithm consists of CoresetFinder (Algorithm 1), DH-Onestep (Algorithm 2), and DH-Sparsify (Algorithm 3). DH-Sparsify iteratively calls DH-Onestep, which uses CoresetFinder as a subroutine. We below explain them one by one.

Algorithm 1 CoresetFinder(H, λ): greedy algorithm for coreset construction.

Input: $H = (V, F, z)$ and $\lambda > 0$
Output: $S \subseteq F$
$S \leftarrow \emptyset$ and $S^{uv} \leftarrow \emptyset$ for each $(u, v) \in C(F)$
$A^{uv} \leftarrow \{ f \in F \mid (u, v) \in C(f) \}$ for each $(u, v) \in C(F)$
for each $(u, v) \in C(F)$ **:**
    if $|A^{uv} \setminus S| \geq \lambda$ **:**
        Find the first λ heaviest hyperarcs $f_1^{uv}, f_2^{uv}, \cdots, f_\lambda^{uv} \in A^{uv} \setminus S$
        Add $f_1^{uv}, f_2^{uv}, \cdots, f_\lambda^{uv}$ to S^{uv}
    else
        $S^{uv} \leftarrow A^{uv} \setminus S$
    $S \leftarrow S \cup S^{uv}$
return S

Algorithm 2 DH-Onestep(H, λ): sampling algorithm called in each iteration in Algorithm 3.

Input: $H = (V, F, z)$ and $\lambda > 0$
Output: $\tilde{H} = (V, \tilde{F}, \tilde{z})$
$S \leftarrow$ CoresetFinder(H, λ)
$\tilde{F} \leftarrow S$ and $\tilde{z}_f \leftarrow z_f$ for $f \in S$
for each $f \in F \setminus S$ **:**
    With probability $\frac{1}{2}$, add f to $\tilde{F}$ and set $\tilde{z}_f \leftarrow 2z_f$
return $\tilde{H} = (V, \tilde{F}, \tilde{z})$

The first building block of our algorithm is CoresetFinder(H, λ) given in Algorithm 1. It takes a hypergraph H and a parameter λ as input, constructs a set, S^{uv}, of up to λ hyperarcs for each $(u, v) \in C(F)$, and outputs $S = \bigcup_{(u,v)\in C(F)} S^{uv}$. For each pair (u, v) (in arbitrary order), S^{uv} is obtained by selecting up to the λ heaviest hyperarcs f with $(u, v) \in C(f)$ among those not selected yet. The parameter λ controls the size of output S.

▶ **Lemma 4.** *Let H be a directed hypergraph and λ be a positive integer.* CoresetFinder(H, λ) *returns a set S of at most λn^2 hyperarcs that can be partitioned into disjoints subsets $\{ S^{uv} \mid (u, v) \in C(F) \}$ satisfying the following conditions:*

1. *for any $(u, v) \in C(F)$, every $f \in S^{uv}$ satisfies $(u, v) \in C(f)$,*
2. *if $(u, v) \in C(F \setminus S)$, $|S^{uv}| = \lambda$ holds, and*
3. *for any $(u, v) \in C(F)$, $f \in S^{uv}$, and $f' \in F \setminus S$ such that $(u, v) \in C(f')$, $z_f \geq z_{f'}$ holds.*

Algorithm 3 DH-SPARSIFY(H, ε): iterative algorithm that computes an ε-spectral sparsifier.

Input: $H = (V, F, z)$ with $|V| = n$ and $|F| = m$, and $\varepsilon > 0$
Output: $\tilde{H} = (V, \tilde{F}, \tilde{z})$

$m^* \leftarrow \frac{n^2}{\varepsilon^2} \log^3 \frac{n}{\varepsilon}$ ▷ This is the (asymptotic) target size of the resulting sparsifier.
$T \leftarrow \left\lceil \log_{4/3}\left(\frac{m}{m^*}\right) \right\rceil$
$i \leftarrow 0$, $\tilde{H}_0 = (V, \tilde{F}_0, \tilde{z}_0) \leftarrow H$, and $m_0 \leftarrow |\tilde{F}_0|$
while $i < T$ and $m_i \geq C_2 m^*$ **:** ▷ C_2 is a constant that is explained in Section 4.3.
  $\varepsilon_i \leftarrow \frac{\varepsilon}{4 \log^2_{4/3}\left(\frac{m_i}{m^*}\right)}$ and $\lambda_i \leftarrow \left\lceil \frac{C_1 \log^3 m_i}{\varepsilon_i^2} \right\rceil$ ▷ ε_i is used in the analysis.
  $\tilde{H}_{i+1} = (V, \tilde{F}_{i+1}, \tilde{z}_{i+1}) \leftarrow$ DH-ONESTEP$(\tilde{H}_i, \lambda_i)$
  $m_{i+1} \leftarrow |\tilde{F}_{i+1}|$
  $i \leftarrow i + 1$
$i_{\mathrm{end}} \leftarrow i$ and $\tilde{H} \leftarrow \tilde{H}_{i_{\mathrm{end}}}$
return $\tilde{H} = (V, \tilde{F}, \tilde{z})$

Proof. Since CORESETFINDER(H, λ) constructs S^{uv} for each $(u, v) \in C(F)$ by selecting up to the λ heaviest hyperarcs f with $(u, v) \in C(f)$ among those that have not been selected yet, S^{uv} for $(u, v) \in C(F)$ are mutually disjoint. This also implies $|S| = \sum_{(u,v)\in C(F)} |S^{uv}| \leq \lambda n^2$ and the first and third conditions. After S is constructed, if there is a hyperarc $f' \in F \setminus S$ such that $(u, v) \in C(f')$, then λ hyperarcs must have been added to S^{uv}. Hence $|S^{uv}| = \lambda$ if $(u, v) \in C(F \setminus S)$, implying the second condition. ◀

We call the set S shown in Lemma 4 a *coreset*, which plays a key role in the analysis.

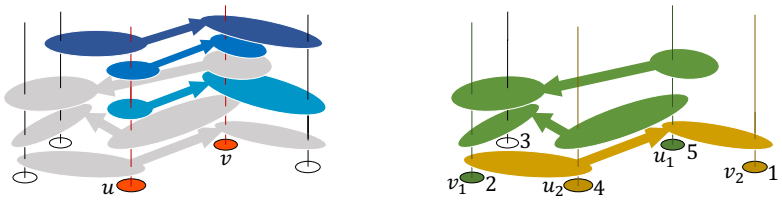
▶ **Definition 5.** *Given a directed hypergraph $H = (V, F, z)$, a subset $S \subseteq F$, and a positive integer λ, we say S is a* λ-coreset *of H if S can be partitioned into disjoints subsets $\{ S^{uv} \mid (u, v) \in C(F) \}$ satisfying the three conditions in the statement of Lemma 4.*

In short, if there is a hyperarc $f' \notin S$ with $(u, v) \in C(f')$, S^{uv} contains (at least) λ hyperarcs that are at least as heavy as $z_{f'}$. Figure 1a illustrates an example of a coreset. We use this coreset as a counterpart of a bundle of spanners in the spanner-based sparsification.

Next, we explain DH-ONESTEP(H, λ) given in Algorithm 2, which is the main subroutine in our algorithm. The algorithm first computes a λ-coreset S by calling CORESETFINDER(H, λ). The hyperarcs in the coreset S are deterministically added to the output. Then, it randomly chooses the remaining hyperarcs with probability $1/2$ and doubles the weights if sampled, thus preserving the expected total weight. The main technical observation is that, under an appropriate choice of λ, the output of DH-ONESTEP(H, λ) is an ε-spectral sparsifier of H. Formally, we can show the following lemma, which is the main technical contribution and will be proved in Section 4.2.

▶ **Lemma 6.** *Let $H = (V, F, z)$ be a directed hypergraph with $|V| = n$ and $|F| = m$. For any $\varepsilon \in (0, 1)$ and $\lambda \geq \frac{C_1 \log^3 m}{\varepsilon^2}$, where C_1 is a sufficiently large constant,* DH-ONESTEP(H, λ) *returns an ε-spectral sparsifier $\tilde{H} = (V, \tilde{F}, \tilde{z})$ of H satisfying $|\tilde{F}| \leq \frac{m}{2} + (3m \log n)^{\frac{1}{2}} + \lambda n^2$ with probability at least $1 - O\left(\frac{1}{n^2}\right)$.*

Finally, we present our sparsification algorithm DH-SPARSIFY(H, ε) in Algorithm 3. In the algorithm description, C_1 denotes the constant given in the statement of Lemma 6, and C_2 is a sufficiently large constant (which we can compute explicitly by carefully expanding the analysis in Section 4.3). The algorithm iteratively calls DH-ONESTEP$(\tilde{H}_i, \lambda_i)$, where $\tilde{H}_i$



(a) Coreset. **(b)** Critical pairs.

Figure 1 Illustration of a coreset and critical pairs on (a part of) a given hypergraph. A circle is a vertex, and a hyperarc is indicated by an arrow and two ellipses representing a head and a tail. A hyperarc contains a vertex if the line originating from the vertex pierces its head or tail. Figure 1a presents an image of a coreset, focusing on a vertex pair (u, v). Suppose that the hyperarcs are aligned in decreasing order of their weights from top to bottom. The blue hyperarcs are the three heaviest ones having u and v as elements of their tails and heads, respectively, and they are included in a subset S^{uv} of a coreset S. We suppose that gray hyperarcs are not in S. While the bottommost gray hyperarc f also satisfies $u \in t(f)$ and $v \in h(f)$, the three blue hyperarcs are heavier than it. Thus, the conditions of the λ-coreset with $\lambda = 3$ are satisfied for (u, v). Figure 1b presents an image of critical pairs of three hyperarcs, which are missed by the coreset S in Figure 1a. Suppose that vertices v have x_v values of 2, 3, 4, 5, and 1 from left to right, respectively, as shown nearby the vertices. Then, the green and yellow hyperarcs have (u_1, v_1) and (u_2, v_2), respectively, as x-ctirical pairs. If the three hyperarcs constitute $F_i^x \subseteq F \setminus S$, we have $E_i^x = \{(u_1, v_1), (u_2, v_2)\}$, and F_i^x is partitioned into F_i^{x,u_1v_1} and F_i^{x,u_2v_2}, shown in green and yellow, respectively.

is the sub-hypergraph obtained in the previous step. Here, the parameter λ_i is defined as in Line 3, which makes $\tilde{H}_{i+1}$ an ε_i-spectral sparsifier of $\tilde{H}_i$ by the condition in Lemma 6.[5] The algorithm repeatedly calls DH-Onestep$(\tilde{H}_i, \lambda_i)$ until the size of $\tilde{H}_i$ becomes $\tilde{O}(n^2/\varepsilon^2)$ or the maximum number of iterations, T, is reached. With this choice of ε_i, we will show that the size of $\tilde{H}_i$ decreases geometrically and that the accumulated sparsification error is bounded by ε. Consequently, the final output is an ε-spectral sparsifier of the desired size, which completes the proof of Theorem 1. We present the analysis in Section 4.3.

4.2 Proof of Lemma 6

We prove Lemma 6, which ensures the correctness of DH-Onestep. In this section, we let $H = (V, F, z)$, $\lambda \geq \frac{C_1 \log^3 m}{\varepsilon^2}$, and $\varepsilon \in (0, 1)$ be as given in the statement of Lemma 6, and let $\tilde{H} = (V, \tilde{F}, \tilde{z})$ be the output of DH-Onestep(H, λ).

To prove Lemma 6, we bound the size and sparsification error of $\tilde{H}$ from above. The former is an easy consequence of the Chernoff bound. We below prove it assuming $m > 12 \log n$; otherwise, an input hypergraph is already sparsified and we do not run DH-Onestep.

▶ **Lemma 7.** *Let $H = (V, F, z)$ be a directed hypergraph with $|V| = n$ and $|F| = m$, and let λ be a positive integer. If $m > 12 \log n$,* DH-Onestep(H, λ) *outputs a sub-hypergraph $\tilde{H} = (V, \tilde{F}, \tilde{z})$ of H satisfying $|\tilde{F}| \leq \frac{m}{2} + (3m \log n)^{\frac{1}{2}} + \lambda n^2$ with probability at least $1 - \frac{2}{n^2}$.*

Proof. Let S be a λ-coreset constructed in Line 2 in DH-Onestep(H, λ). By Lemma 4, S has at most λn^2 hyperarcs. To bound $|\tilde{F} \setminus S|$, for each $f \in F \setminus S$, we let X_f be a random variable that takes 1 if f is sampled and 0 otherwise. Note that $|\tilde{F} \setminus S| = \sum_{f \in F \setminus S} X_f$ holds.

[5] Unlike the existing spanner-based algorithm [17], we need to change ε_i adaptively since fixing $\varepsilon_i = \frac{\varepsilon}{T}$ does not yield a sparsifier of the desired size when the input hypergraph is exponentially large in n.

Since we have $\mathbb{E}\left[\sum_{f\in F\setminus S} X_f\right] = (m-|S|)/2 \le m/2$, for any $t\in(0,1)$, the Chernoff bound (Proposition 3) implies

$$\mathbb{P}\left[\sum_{f\in F\setminus S} X_f - \mathbb{E}\left[\sum_{f\in F\setminus S} X_f\right] > \frac{m}{2}t\right] \le 2\exp\left(-\frac{mt^2}{6}\right).$$

By setting $t = \left(\frac{12\log n}{m}\right)^{\frac{1}{2}}$, which is smaller than 1 due to the lemma assumption, we obtain

$$\mathbb{P}\left[\sum_{f\in F\setminus S} X_f \le \frac{m}{2} + (3m\log n)^{\frac{1}{2}}\right] \ge 1-\frac{2}{n^2}.$$

Thus, we have $|\tilde{F}| = |S| + \sum_{f\in F\setminus S} X_f \le \frac{m}{2} + (3m\log n)^{\frac{1}{2}} + \lambda n^2$ with probability at least $1-\frac{2}{n^2}$. ◀

The rest of this section focuses on showing that $\tilde{H}$ is an ε-spectral sparsifier of H, i.e., $(1-\varepsilon)x^\top L_H(x) \le x^\top L_{\tilde{H}}(x) \le (1+\varepsilon)x^\top L_H(x)$ for any $x\in\mathbb{R}^V$. Since this relation is invariant under scaling of x, it suffices to prove the relation for any x satisfying $x^\top L_H(x) = 1$. Let $\mathbb{S}_H = \left\{ x\in\mathbb{R}^V \;\middle|\; x^\top L_H(x) = 1 \right\}$. A similar normalization is used in [15] with respect to the total energy of the corresponding underlying clique digraphs. By contrast, we directly normalize the total energy of a hypergraph, $x^\top L_H(x)$. This difference is a key to eliminating the extra r^3 factor, while it requires a new discretization scheme, as described later.

Since we analyze the contribution of each hyperarc to the energy of H, it is convenient to use the notation of $Q^x_H(f)$ and $Q^x_H(F')$ for $f\in F$ and $F'\subseteq F$, respectively, defined in Section 2. Our goal is to prove $(1-\varepsilon)Q^x_H(F) \le Q^x_{\tilde{H}}(\tilde{F}) \le (1+\varepsilon)Q^x_H(F)$ for all $x\in\mathbb{S}_H$.

Given $x\in\mathbb{S}_H$ and a λ-coreset $S\subseteq F$, our strategy is to partition $F\setminus S$ into subsets based on the energies and evaluate the error caused by sparsification for each subset. Specifically, we classify hyperarcs $f\in F\setminus S$ into subsets F^x_i defined for each $i\in\mathbb{Z}$ as follows:

$$F^x_i := \left\{ f\in F\setminus S \;\middle|\; Q^x_H(f) \in \left[\frac{1}{2^i\lambda}, \frac{1}{2^{i-1}\lambda}\right) \right\}.$$

We also define $\tilde{F}^x_i := F^x_i \cap \tilde{F}$ for each $i\in\mathbb{Z}$.

Since $Q^x_H(F\setminus S) = \sum_{i\in\mathbb{Z}} Q^x_H(F^x_i)$ and $Q^x_{\tilde{H}}(\tilde{F}\setminus S) = \sum_{i\in\mathbb{Z}} Q^x_{\tilde{H}}(\tilde{F}^x_i)$, our goal is to prove that $|Q^x_{\tilde{H}}(\tilde{F}^x_i) - Q^x_H(F^x_i)|$ is sufficiently small for all $i\in\mathbb{Z}$ and $x\in\mathbb{S}_H$. This is not difficult if i is sufficiently large, as in the following lemma.

▶ **Lemma 8.** *Let $I = \lceil\log_2(9m)\rceil$. For any $x\in\mathbb{S}_H$, $\left|Q^x_H(\cup_{i\ge I+1}F^x_i) - Q^x_{\tilde{H}}\left(\cup_{i\ge I+1}\tilde{F}^x_i\right)\right| \le \frac{\varepsilon}{3}$.*

Proof. Due to the assumption in Lemma 6, $\lambda\varepsilon \ge \frac{C_1\log^3 m}{\varepsilon} \ge 1$ holds for sufficiently large C_1. By the definition of F^x_i, the energy of each hyperarc in $\cup_{i\ge I+1}F^x_i$ is less than $\frac{1}{2^I\lambda}$, which is at most $\frac{\varepsilon}{9m}$ by $I = \lceil\log_2(9m)\rceil$ and $\lambda\varepsilon\ge 1$. Thus, it holds that

$$Q^x_H(\cup_{i\ge I+1}F^x_i) = \sum_{f\in\cup_{i\ge I+1}F^x_i} Q^x_H(f) \le m\cdot\frac{\varepsilon}{9m} \le \frac{\varepsilon}{9}. \tag{1}$$

As for $\tilde{F}\subseteq F$, since the weight of each hyperarc in $\tilde{F}$ is doubled in DH-Onestep, we have

$$Q^x_{\tilde{H}}\left(\cup_{i\ge I+1}\tilde{F}^x_i\right) \le 2\cdot Q^x_H(\cup_{i\ge I+1}F^x_i) \le \frac{2\varepsilon}{9}. \tag{2}$$

Combining eqs. (1) and (2), we obtain the claim. ◀

We then introduce additional definitions for the convenience of describing our discretization scheme and analyzing the sparsification error.

▶ **Definition 9.** *For $x \in \mathbb{S}_H$, we say $(u,v) \in V \times V$ is an x-critical pair of $f \in F$ if we have $(u,v) = \operatorname{argmax}_{(u,v)\in C(f)}(x_u - x_v)_+^2$, breaking ties as in Section 2. For $i \in \mathbb{Z}$ and $x \in \mathbb{S}_H$, let*

$$E_i^x = \{\, (u,v) \in C(F) \mid (u,v) \text{ is an } x\text{-critical pair of some } f \in F_i^x \,\}$$

and, for each $(u,v) \in E_i^x$, let

$$F_i^{x,uv} = \{\, f \in F_i^x \mid (u,v) \text{ is an } x\text{-critical pair of } f \,\}.$$

Note that the collection of $F_i^{x,uv}$ for $(u,v) \in E_i^x$ forms a partition of F_i^x. Figure 1b presents an example of x-critical pairs.

We now discuss how to bound $|Q_{\tilde{H}}^x(\tilde{F}_i^x) - Q_H^x(F_i^x)|$ for i that is not covered in Lemma 8. By using the Chernoff bound, it is easy to evaluate the probability that $|Q_{\tilde{H}}^x(\tilde{F}_i^x) - Q_H^x(F_i^x)|$ is small for each $x \in \mathbb{S}_H$. To convert it to a uniform bound over all $x \in \mathbb{S}_H$, we construct an appropriate discretization scheme, as follows.

Let $\Delta = \frac{\varepsilon}{9m}$. For $(u,v) \in E_i^x$, we define the *discretization width* as $\Delta_i^{uv} \coloneqq \frac{\Delta}{\max_{f\in F_i^{x,uv}} z_f}$. Note that $F_i^{x,uv} \neq \emptyset$ holds for $(u,v) \in E_i$ by the definitions of E_i and $F_i^{x,uv}$, and hence Δ_i^{uv} is well-defined. The denominator plays the role of scaling the width. Given any $x \in \mathbb{S}_H$, we consider discretizing $(x_u - x_v)_+^2$ for each $(u,v) \in E_i^x$, not the energy itself. Specifically, for each $i \in \mathbb{Z}$ and $(u,v) \in E_i^x$, we use $\left\lfloor \frac{(x_u - x_v)_+^2}{\Delta_i^{uv}} \right\rfloor \Delta_i^{uv}$ as a discretized value of $(x_u - x_v)_+^2$. Then, for each $f \in F_i^{x,uv}$ such that $(u,v) \in E_i^x$, we define the *discretized energy* $D_H^x(f)$ by

$$D_H^x(f) \coloneqq z_f \left\lfloor \frac{(x_u - x_v)_+^2}{\Delta_i^{uv}} \right\rfloor \Delta_i^{uv}.$$

It should be noted that the discretized energy of $f \in F_i^{x,uv}$ is defined by first discretizing $(x_u - x_v)_+^2$ and then scaling it by z_f. This somewhat indirect discretization scheme will turn out important when bounding the number of possible discretized energies.

For each sampled hyperarc $f \in F_i^{x,uv} \cap \tilde{F}$ with $(u,v) \in E_i^x$, we define the *discretized energy after sampling* by $D_{\tilde{H}}^x(f) \coloneqq 2D_H^x(f)$. We also let $D_H^x(F_i^x) = \sum_{f\in F_i^x} D_H^x(f)$ and $D_{\tilde{H}}^x(\tilde{F}_i^x) = \sum_{f\in \tilde{F}_i^x} D_{\tilde{H}}^x(f)$. We can ensures that discretization errors are small as follows.

▶ **Lemma 10.** *For any $x \in \mathbb{S}_H$, we have*

$$\sum_{i\in\mathbb{Z}} |D_H^x(F_i^x) - Q_H^x(F_i^x)| \le \frac{\varepsilon}{9} \qquad \text{and} \qquad \sum_{i\in\mathbb{Z}} \left|D_{\tilde{H}}^x(\tilde{F}_i^x) - Q_{\tilde{H}}^x(\tilde{F}_i^x)\right| \le \frac{2\varepsilon}{9}.$$

Proof. Recall that the discretized energy $D_H^x(f)$ of each $f \in F_i^{x,uv}$ is obtained by discretizing $(x_u - x_v)_+^2$ with the width Δ_i^{uv} and scaling it by z_f. Therefore, the discretization error for each f is bounded by $z_f \Delta_i^{uv}$. From the definition of Δ_i^{uv}, we have $z_f \Delta_i^{uv} = z_f \frac{\Delta}{\max_{f\in F_i^{x,uv}} z_f} \le \Delta$. Hence, the total discretization error over all $f \in F \setminus S$ is bounded by $m\Delta$, which is at most $\frac{\varepsilon}{9}$ since $\Delta = \frac{\varepsilon}{9m}$. Thus, we obtain the first inequality. The second inequality follows from the fact that the weights of sampled hyperarcs are doubled. ◀

From Lemma 10, we can bound the sparsification error $|Q_{\tilde{H}}^x(\tilde{F}_i^x) - Q_H^x(F_i^x)|$ for all $x \in \mathbb{S}_H$ by bounding $|D_{\tilde{H}}^x(\tilde{F}_i^x) - D_H^x(F_i^x)|$ for all $x \in \mathbb{S}_H$. Since the number of possible discretized energies is finite, we can use the standard Chernoff bound and union bound to evaluate the sparsification error. Thus, what remains is to prove that the number of discretized energies is

small enough so that we can obtain the desired uniform bound. To this end, we first bound the size of E_i^x and then bound the number of possible discretized values. The following lemma bounds the size of E_i^x, in which the existence of a λ-coreset plays an important role.

▶ **Lemma 11.** *For $i \in \mathbb{Z}$, we have $|E_i^x| < 2^i$.*

Proof. By the definition of E_i^x, for each $(u,v) \in E_i^x$, there is a hyperarc $f^{uv} \in F_i^x \subseteq F \setminus S$ such that (u,v) is an x-critical pair of f^{uv}. Since S is a λ-coreset, S admits a partition $\{ S^{uv} \mid (u,v) \in C(F) \}$ satisfying the three conditions in Lemma 4. Since $f^{uv} \notin S$, the third condition in Lemma 4 implies $z_f \geq z_{f^{uv}}$ for any $f \in S^{uv}$. Hence, for any $f \in S^{uv}$, we have

$$Q_H^x(f^{uv}) = z_{f^{uv}}(x_u - x_v)_+^2 \leq z_f(x_u - x_v)_+^2 \leq \max_{(u',v') \in C(f)} z_f(x_{u'} - x_{v'})_+^2 = Q_H^x(f). \quad (3)$$

Since the second condition in Lemma 4 implies $|S^{uv}| = \lambda$ for $(u,v) \in E_i^x \subseteq C(F \setminus S)$,

$$\begin{aligned} Q_H^x(F) &\geq \sum_{(u,v) \in E_i^x} (Q_H^x(S^{uv}) + Q_H^x(f^{uv})) \quad \text{(since all } S^{uv} \text{ and } f^{uv} \notin S \text{ are disjoint)} \\ &\geq \sum_{(u,v) \in E_i^x} (\lambda + 1) \cdot Q_H^x(f^{uv}) \qquad \text{(by eq. (3) and } |S^{uv}| = \lambda) \\ &\geq \sum_{(u,v) \in E_i^x} (\lambda + 1) \cdot (2^i \lambda)^{-1} \qquad \text{(by } f^{uv} \in F_i^x). \end{aligned}$$

holds, hence $Q_H^x(F) > 2^{-i}|E_i^x|$. Since $Q_H^x(F) = 1$ by $x \in \mathbb{S}_H$, we obtain $|E_i^x| < 2^i$. ◀

From Lemma 11, if $i \leq 0$, we have $|E_i^x| < 2^i \leq 1$, which implies $E_i^x = \emptyset$ and $F_i^x = \emptyset$. Thus, the following corollary holds.

▶ **Corollary 12.** *If $i \leq 0$, we have $F_i^x = \emptyset$.*

Due to Corollary 12 and Lemma 8, we can focus on $i \in \mathbb{Z}$ with $1 \leq i \leq I = \lceil \log_2 9m \rceil$. In this range, we have the following bound on the number of possible discretized values.

▶ **Lemma 13.** *For each positive integer i, let $L_i = \left\{ \left(F_i^x, \{D_H^x(f)\}_{f \in F_i^x} \right) \mid x \in \mathbb{S}_H \right\}$, where $\{D_H^x(f)\}_{f \in F_i^x}$ is the list of the discretized energies over all hyperarcs in F_i^x. If $1 \leq i \leq I = \lceil \log_2 9m \rceil$, we have $|L_i| \leq \left(\frac{648 n^4 m^4}{\lambda \varepsilon} \right)^{2^i}$.*

Since the proof of Lemma 13 is not short, we first complete the proof of Lemma 6 assuming that Lemma 13 is true; then, we prove Lemma 13 in Section 4.2.1.

Proof of Lemma 6. Let $I = \lceil \log_2(9m) \rceil$ as in Lemma 8 and define L_i as in Lemma 13. Fix $i \in \{1, 2, \ldots, I\}$ and consider any element of L_i, which we denote by $(F_i^y, \{D_H^y(f)\}_{f \in F_i^y})$ for some $y \in \mathbb{S}_H$. Since the discretized energy of each hyperarc is obtained by rounding down, we have $D_H^y(f) \leq Q_H^y(f)$. Thus, for every $f \in F_i^y$, it holds that

$$D_H^y(f) \leq Q_H^y(f) < \frac{1}{2^{i-1}\lambda}. \quad (4)$$

For each $f \in F \setminus S$, let X_f be a random variable that takes 1 with probability $1/2$ and 0 otherwise, which represents the randomness of sampling and hence $D_{\tilde{H}}^y(\tilde{F}_i^y) = \sum_{f \in F_i^y} 2X_f D_H^y(f)$. By $D_H^y(f) \leq Q_H^y(f)$ again, we have

$$\mathbb{E}\left[\sum_{f \in F_i^y} 2X_f D_H^y(f) \right] = \sum_{f \in F_i^y} D_H^y(f) = D_H^y(F_i^y) \leq Q_H^y(F_i^y) \leq Q_H^y(F) = 1. \quad (5)$$

Due to eqs. (4) and (5), the Chernoff bound (Proposition 3) with $\mu = 1$, $a = \frac{1}{2^{i-2}\lambda}$, and $\delta = \frac{\varepsilon}{3I}$ implies

$$\mathbb{P}\left[\left|D^y_{\tilde{H}}(\tilde{F}^y_i) - D^y_H(F^y_i)\right| > \frac{\varepsilon}{3I}\right] = \mathbb{P}\left[\left|\sum_{f\in F^y_i} 2X_f D^y_H(f) - \mathbb{E}\left[\sum_{f\in F^y_i} 2X_f D^y_H(f)\right]\right| > \frac{\varepsilon}{3I}\right] \leq 2\exp\left(-\frac{2^i\cdot\varepsilon^2\lambda}{108I^2}\right).$$

This bound is true for each $(F^y_i, \{D^y_H(f)\}_{f\in F^y_i}) \in L_i$, and we can convert it to a uniform bound over all $(F^y_i, \{D^y_H(f)\}_{f\in F^y_i}) \in L_i$ by using Lemma 13 and the union bound as follows:

$$\mathbb{P}\left[\exists (F^y_i, \{D^y_H(f)\}_{f\in F^y_i}) \in L_i, \left|D^y_{\tilde{H}}(\tilde{F}^y_i) - D^y_H(F^y_i)\right| > \frac{\varepsilon}{3I}\right] \leq 2\exp\left(-\frac{2^i\cdot\varepsilon^2\lambda}{108I^2}\right)\cdot\left(\frac{648n^4m^4}{\lambda\varepsilon}\right)^{2^i}.$$

We may assume $nm \geq 648$ (otherwise Lemma 6 is trivial for a sufficiently large C_1). Letting C_1 be sufficiently large, we have $\lambda \geq \frac{C_1\log^3 m}{\varepsilon^2} \geq \frac{108I^2}{\varepsilon^2}(6\log n + 5\log m)$ and $\lambda\varepsilon \geq 1$. Thus, we can further bound the right-hand side from above by

$$2\exp\left(-\frac{2^i\cdot\varepsilon^2\lambda}{108I^2}\right)\cdot\left(n^5m^5\right)^{2^i} \leq 2\exp\left(-2^i\cdot(6\log n + 5\log m)\right)\cdot\left(n^5m^5\right)^{2^i} \leq \frac{2}{n^{2^i}}.$$

Therefore, $\mathbb{P}[\forall (F^y_i, \{D^y_H(f)\}_{f\in F^y_i}) \in L_i, |D^y_{\tilde{H}}(\tilde{F}^y_i) - D^y_H(F^y_i)| \leq \frac{\varepsilon}{3I}] \geq 1 - \frac{2}{n^{2^i}}$ holds. Since $(F^x_i, \{D^x(f)\}_{f\in F^x_i}) \in L_i$ holds for all $x \in \mathbb{S}_H$, we can equivalently rewrite the bound as

$$\mathbb{P}\left[\forall x \in \mathbb{S}_H, \left|D^x_{\tilde{H}}(\tilde{F}^x_i) - D^x_H(F^x_i)\right| \leq \frac{\varepsilon}{3I}\right] \geq 1 - \frac{2}{n^{2^i}}.$$

By the union bound over $1 \leq i \leq I = \lceil\log_2(9m)\rceil$ and $\sum_{i=1}^I \frac{2}{n^{2^i}} \leq \sum_{i=1}^\infty \frac{2}{n^{2i}} \leq \frac{2}{n^2-1} \leq \frac{3}{n^2}$ (for $n \geq 2$), we obtain

$$\mathbb{P}\left[\forall x \in \mathbb{S}_H, \sum_{i=1}^I \left|D^x_{\tilde{H}}(\tilde{F}^x_i) - D^x_H(F^x_i)\right| \leq \frac{\varepsilon}{3}\right] \geq 1 - \frac{3}{n^2}. \tag{6}$$

Thus, for all $x \in \mathbb{S}_H$, we can bound $|x^\top L_{\tilde{H}}(x) - x^\top L_H(x)| = |Q^x_{\tilde{H}}(\tilde{F}) - Q^x_H(F)|$ as follows:

$$\begin{aligned}
&\left|Q^x_{\tilde{H}}(\tilde{F}) - Q^x_H(F)\right| \\
&= \frac{\varepsilon}{3} + \sum_{i=1}^I |Q^x_{\tilde{H}}(\tilde{F}^x_i) - Q^x_H(F^x_i)| \qquad \text{(by Lemma 8 and Corollary 12)} \\
&\leq \frac{\varepsilon}{3} + \sum_{i=1}^I \left[|Q^x_{\tilde{H}}(\tilde{F}^x_i) - D^x_{\tilde{H}}(\tilde{F}^x_i)| + |D^x_{\tilde{H}}(\tilde{F}^x_i) - D^x_H(F^x_i)| + |D^x_H(F^x_i) - Q^x_H(F^x_i)|\right] \\
&\leq \frac{\varepsilon}{3} + \frac{\varepsilon}{9} + \frac{\varepsilon}{3} + \frac{2\varepsilon}{9} \qquad \text{(by Lemma 10 and eq. (6))} \\
&= \varepsilon,
\end{aligned}$$

which holds with probability at least $1 - \frac{3}{n^2}$. Hence, $\tilde{H}$ is an ε-spectral sparsifier of H. Combining this with the size bound in Lemma 7, we obtain Lemma 6. ◀

4.2.1 Proof of Lemma 13

We present the proof of Lemma 13. Our goal is to bound the size of L_i defined in Lemma 13 for $i \in \mathbb{Z}$ with $1 \leq i \leq I = \lceil\log_2 9m\rceil$. To this end, we proceed in two steps: we first bound the number of possible combinations of $(F^x_i, E^x_i, \{F^{x,uv}_i\}_{f\in E^x_i})$ over all $x \in \mathbb{S}_H$, and then bound the number of possible lists $\{D^x_H(f)\}_{f\in F^x_i}$ of discretized energies. For convenience, we define the following notion.

▶ **Definition 14.** *Let $(E, \{f_{uv}\}_{(u,v)\in E}, \pi_E)$ be a tuple such that $E \subseteq V \times V$, $\{f_{uv}\}_{(u,v)\in E}$ is a list of hyperarcs indexed by $(u,v) \in E$, and π_E is a total ordering on E. For $i \in \{1,2,\dots,I\}$, we say $(E, \{f_{uv}\}_{(u,v)\in E}, \pi_E)$ is i-*realized *by $x \in \mathbb{S}_H$ if the following conditions hold:*

1. $E = E_i^x$,
2. $f_{uv} = \operatorname{argmin}_{f\in F_i^{x,uv}} z_f$ *for each* $(u,v) \in E_i^x$, *and*
3. π_E *is the increasing order of the values of* $(x_u - x_v)_+^2$, *i.e.,* (u,v) *is smaller than* (u',v') *in* π_E *if and only if* $(x_u - x_v)_+^2 \le (x_{u'} - x_{v'})_+^2$ *(where the tie-breaking rule explained in Section 2 is used when the equality holds).*

The following lemma says that the i-realizability determines E_i^x, F_i^x, and $F_i^{x,uv}$, implying that we can reduce the problem of counting the number of possible $(F_i^x, E_i^x, \{F_i^{x,uv}\}_{f\in E_i^x})$ to that of counting the number of possible tuples $(E, \{f_{uv}\}_{(u,v)\in E}, \pi_E)$.

▶ **Lemma 15.** *Let $(E, \{f_{uv}\}_{(u,v)\in E}, \pi_E)$ be a tuple as defined in Definition 14 and $x, y \in \mathbb{S}_H$. If both x and y i-realize $(E, \{f_{uv}\}_{(u,v)\in E}, \pi_E)$ and $\bigcup_{j=1}^{i-1} F_j^x = \bigcup_{j=1}^{i-1} F_j^y$ holds, then, for every $(u,v) \in E$, we have $E_i^x = E_i^y$, $F_i^x = F_i^y$, and $F_i^{x,uv} = F_i^{y,uv}$.*

Proof. By the definition of the i-realizability, we have $E_i^x = E = E_i^y$. If we can assume $F_i^{x,uv} = F_i^{y,uv}$ for every $(u,v) \in E$, we have $F_i^x = \bigcup_{(u,v)\in E} F_i^{x,uv} = \bigcup_{(u,v)\in E} F_i^{y,uv} = F_i^y$ since $\{ F_i^{x,uv} \mid (u,v) \in C(F) \}$ and $\{ F_i^{y,uv} \mid (u,v) \in C(F) \}$ are partitions of F_i^x and F_i^y, respectively. Therefore, we below focus on proving $F_i^{x,uv} = F_i^{y,uv}$ for every $(u,v) \in E$.

For a contradiction, suppose $F_i^{x,u_1v_1} \neq F_i^{y,u_1v_1}$ for some $(u_1,v_1) \in E$. Without loss of generality, we assume there is a hyperarc $f^* \in F_i^{x,u_1v_1} \setminus F_i^{y,u_1v_1}$. Since both x and y i-realize $(E, \{f_{uv}\}_{(u,v)\in E}, \pi_E)$ and $(u_1,v_1) \in E$, the second condition of the i-realizability implies

$$\operatorname*{argmin}_{f\in F_i^{x,u_1v_1}} z_f = f_{u_1v_1} = \operatorname*{argmin}_{f\in F_i^{y,u_1v_1}} z_f. \tag{7}$$

In particular, we have $z_{f_{u_1v_1}} \le z_{f^*}$ for $f^* \in F_i^{x,u_1v_1}$. Hence

$$\begin{aligned} Q_H^y(f^*) &= z_{f^*} \max_{(u,v)\in C(f^*)} (y_u - y_v)_+^2 \\ &\ge z_{f_{u_1v_1}} (y_{u_1} - y_{v_1})_+^2 \qquad \big(\text{by } (u_1,v_1) \in C(f^*) \text{ and } z_{f^*} \ge z_{f_{u_1v_1}}\big) \\ &= z_{f_{u_1v_1}} \max_{(u,v)\in C(f_{u_1v_1})} (y_u - y_v)_+^2 \qquad (\text{by } f_{u_1v_1} \in F_i^{y,u_1v_1} \text{ as in eq. (7)}) \\ &\ge \frac{2^{-i}}{\lambda} \qquad (\text{by } f_{u_1v_1} \in F_i^y). \end{aligned}$$

From $Q_H^y(f^*) \ge \frac{2^{-i}}{\lambda}$ and $f^* \in F_i^{x,u_1v_1} \subseteq F \setminus S$, it must hold that $f^* \in \bigcup_{j=1}^{i} F_j^y$. Moreover, since $\bigcup_{j=1}^{i-1} F_j^x = \bigcup_{j=1}^{i-1} F_j^y$ by the lemma assumption and $f^* \notin \bigcup_{j=1}^{i-1} F_j^x$ by $f^* \in F_i^{x,u_1v_1}$, we have $f^* \notin \bigcup_{j=1}^{i-1} F_j^y$, hence $f^* \in F_i^y$. Since the orderings of E with respect to $(x_u - x_v)_+^2$ and $(y_u - y_v)_+^2$ are both equal to π_E and (u_1,v_1) is an x-critical pair of f^*, we have

$$(u_1,v_1) = \operatorname*{argmax}_{(u,v)\in C(f^*)\cap E} (y_u - y_v)_+^2. \tag{8}$$

Since $f^* \in F_i^y$, eq. (8) implies $f^* \in F_i^{y,u_1v_1}$, contradicting the assumption of $f^* \notin F_i^{y,u_1v_1}$. Therefore, $F_i^{x,uv} = F_i^{y,uv}$ holds for every $(u,v) \in E$. ◀

Lemma 15 enables us to bound the number of possible $(F_i^x, E_i^x, \{F_i^{x,uv}\}_{f\in E_i^x})$ for $x \in \mathbb{S}_H$.

▶ **Lemma 16.** *For each $i \ge 1$, $\left|\left\{ (F_i^x, E_i^x, \{F_i^{x,uv}\}_{f\in E_i^x}) \mid x \in \mathbb{S}_H \right\}\right| \le \left(2^i n^2 m\right)^{2^{i+1}}$ holds.*

Proof. First, we suppose that F_j^x for $j = 1, \dots, i-1$ are fixed. Then, due to Lemma 15, we can bound the number of possible combinations of $(F_i^x, E_i^x, \{F_i^{x,uv}\}_{f \in E_i^x})$ for all $x \in \mathbb{S}_H$ by counting the number of possible tuples $(E, \{f_{uv}\}_{(u,v)\in E}, \pi_E)$ that can be i-realized by some $x \in \mathbb{S}_H$. Since $|E| < 2^i$ by Lemma 11, the number of possible E is $\sum_{k=1}^{|E|} \binom{n^2}{k} \le \sum_{k=1}^{2^i-1} \binom{n^2}{k}$. Once E is specified, there are up to m possible choices of f_{uv} for each $(u,v) \in E$. Furthermore, the number of possible total orderings π_E of E is at most $(|E|)! \le (2^i)!$. Thus, the number of possible tuples $(E, \{f_{uv}\}_{(u,v)\in E}, \pi_E)$ that can be i-realized by some $x \in \mathbb{S}_H$ is at most $\left(\sum_{k=1}^{2^i-1} \binom{n^2}{k}\right) \cdot m^{2^i} \cdot (2^i)!$. This is further upper bounded by $\left(2^i n^2 m\right)^{2^i}$ by a simple calculation.

We now remove the assumption that F_j^x for $j = 1, \dots, i-1$ are fixed. By inductively using the above bound in increasing order of j, the number of possible combinations of $(F_i^x, E_i^x, \{F_i^{x,uv}\}_{f\in E_i^x})$ over all $x \in \mathbb{S}_H$ is at most $\prod_{j=1}^{i} \left(2^j n^2 m\right)^{2^j} \le \left(2^i n^2 m\right)^{\sum_{j=1}^{i} 2^j} \le \left(2^i n^2 m\right)^{2^{i+1}}$, thus completing the proof. ◀

We then fix any tuple $(F_i^y, E_i^y, \{F_i^{y,uv}\}_{f\in E_i^y})$ for some representative $y \in \mathbb{S}_H$ and upper bound the number of possible lists of discretized energies, $\{D_H^x(f)\}_{f\in F_i^x}$, over a subspace of $\mathbb{S}_H$ that consists of x with $(F_i^x, E_i^x, \{F_i^{x,uv}\}_{f\in E_i^x}) = (F_i^y, E_i^y, \{F_i^{y,uv}\}_{f\in E_i^y})$.

▶ **Lemma 17.** *Let $i \ge 0$ and fix $y \in \mathbb{S}_H$ arbitrarily. The number of possible lists $\{D_H^x(f)\}_{f\in F_i^x}$ for all $x \in \mathbb{S}_H$ with $(F_i^x, E_i^x, \{F_i^{x,uv}\}_{(u,v)\in E_i^x}) = (F_i^y, E_i^y, \{F_i^{y,uv}\}_{(u,v)\in E_i^y})$ is at most $\left(\frac{9m}{2^{i-2}\lambda\varepsilon}\right)^{2^i}$.*

Proof. Let $x \in \mathbb{S}_H$ satisfy the condition in the lemma statement and fix $(u,v) \in E_i^x$. Since every $f \in F_i^{x,uv} \subseteq F_i^x$ satisfies $z_f (x_u - x_v)_+^2 = Q_H^x(f) < \frac{1}{2^{i-1}\lambda}$, the range of $(x_u - x_v)_+^2$ is restricted to $\left[0, \frac{1}{2^{i-1}\lambda \min_{f\in F_i^{x,uv}} z_f}\right)$. Hence, the number of possible discretized $(x_u - x_v)_+^2$ values, $\left\lfloor (x_u - x_v)_+^2 / \Delta_i^{uv} \right\rfloor \Delta_i^{uv}$, over all $x \in \mathbb{S}_H$ under the lemma condition is at most

$$\frac{1}{\Delta_i^{uv} 2^{i-1}\lambda \min_{f\in F_i^{x,uv}} z_f} = \frac{1}{\Delta 2^{i-1}\lambda} \cdot \frac{\max_{f\in F_i^{x,uv}} z_f}{\min_{f\in F_i^{x,uv}} z_f} \le \frac{1}{\Delta 2^{i-2}\lambda}, \tag{9}$$

where the equality is due to $\Delta_i^{uv} = \Delta / \max_{f\in F_i^{x,uv}} z_f$ and the inequality comes from $z_f (x_u - x_v)_+^2 = Q_H^x(f) \in \left[\frac{1}{2^i\lambda}, \frac{1}{2^{i-1}\lambda}\right)$ for $f \in F_i^{x,uv} \subseteq F_i^x$, i.e., $\max_{f\in F_i^{x,uv}} z_f \le 2 \min_{f\in F_i^{x,uv}} z_f$.

Since the discretized energy of $f \in F_i^{x,uv}$ is defined by $D_H^x(f) = z_f \left\lfloor (x_u - x_v)_+^2 / \Delta_i^{uv} \right\rfloor \Delta_i^{uv}$, fixing the discretization of $(x_u - x_v)_+^2$ determines discretized energies of all $f \in F_i^{x,uv}$. Therefore, the number of possible lists $\{D_H^x(f)\}_{f\in F_i^{x,uv}}$ is also bounded by eq. (9) for each $(u,v) \in E_i^x$. Since $|E_i^x| < 2^i$ by Lemma 11, the number of possible lists $\{D_H^x(f)\}_{f\in F_i^x}$ is at most $\left(\frac{1}{\Delta 2^{i-2}\lambda}\right)^{2^i}$. By substituting $\Delta = \frac{\varepsilon}{9m}$ into it, we obtain the lemma. ◀

We are now ready to prove Lemma 13.

Proof of Lemma 13. We can uniquely specify any element of L_i by first fixing $(F_i^x, E_i^x, \{F_i^{x,uv}\}_{f\in E_i^x})$ and then $\{D_H^x(f)\}_{f\in F_i^x}$. Therefore, we have $|L_i| \le (2^i n^2 m)^{2^{i+1}} \cdot \left(\frac{9m}{2^{i-2}\lambda\varepsilon}\right)^{2^i} = \left(\frac{36 \cdot 2^i n^4 m^3}{\lambda\varepsilon}\right)^{2^i}$ by Lemmas 16 and 17. Combining this with $i \le I = \lceil \log_2 9m \rceil$ completes the proof. ◀

4.3 Proof of Theorem 1

Let $H = (V, F, z)$ be a directed hypergraph with $|V| = n$ and $|F| = m$, $\varepsilon \in (0,1)$, and $\tilde{H} = (V, \tilde{F}, \tilde{z})$ the output of DH-Sparsify(H, ε). Our goal is to prove that $\tilde{H}$ is an ε-spectral sparsifier of H and $|\tilde{F}| = O\left(\frac{n^2}{\varepsilon^2} \log^3 \frac{n}{\varepsilon}\right)$. We here use m^*, T, i_{end}, $(\tilde{H}_i = (V, \tilde{F}_i, \tilde{z}_i), \lambda_i)$, m_i,

and ε_i given in the description of DH-Sparsify(H,ε) (Algorithm 3), where $m^* = \frac{n^2}{\varepsilon^2}\log^3\frac{n}{\varepsilon}$ is the target sparsifier size, $T = \left\lceil \log_{4/3}\left(\frac{m}{m^*}\right)\right\rceil$ is the maximum number of iterations, i_{end} is the number of iterations performed, $(\tilde{H}_i = (V, \tilde{F}_i, \tilde{z}_i), \lambda_i)$ is the input of DH-Onestep at the ith iteration, $m_i = |\tilde{F}_i|$, and $\varepsilon_i = \frac{\varepsilon}{4\log^2_{4/3}\left(\frac{m_i}{m^*}\right)}$, as in Line 3 of Algorithm 3.

We first show that the number of hyperarcs decreases geometrically in each step.

▶ **Lemma 18.** *Let m_i be the number of hyperarcs in $\tilde{H}_i$. Assume $m_i \geq C_2 m^* = C_2\frac{n^2}{\varepsilon^2}\log^3\frac{n}{\varepsilon}$ for a sufficiently large constant C_2. Then, we have $(3m_i\log n)^{\frac{1}{2}} + \lambda_i n^2 \leq \frac{m_i}{4}$.*

Proof. It is easy to show that $(3m_i\log n)^{\frac{1}{2}} \leq \frac{m_i}{8}$ holds if $m_i \geq 192\log n$, which is true if C_2 is sufficiently large. Hence, the desired inequality holds if $\lambda_i n^2 \leq \frac{m_i}{8}$, which we show below.

By Line 3 in Algorithm 3, we have $\varepsilon_i = \frac{\varepsilon}{4\log^2_{4/3}\frac{m_i}{m^*}}$ and $\lambda_i = \left\lceil \frac{C_1\log^3 m_i}{\varepsilon_i^2}\right\rceil$. Hence,

$$\frac{m_i}{8} - \lambda_i n^2 \geq \frac{m_i}{8} - \frac{2500C_1 n^2}{\varepsilon^2}\log^3 m_i \log^4\frac{m_i}{m^*} \qquad (\text{by } 4^2/\log^4(4/3) < 2500).$$

Let $m_i = \alpha m_*$ and $g(\alpha)$ be the right-hand side of the above inequality, which we regard as a function of α. Since $m^* = (n/\varepsilon)^2\log^3(n/\varepsilon)$, we have

$$\begin{aligned} g(\alpha) &= m_*\left(\frac{\alpha}{8} - \frac{2500C_1}{\log^3(n/\varepsilon)}\log^3(\alpha m_*)\log^4\alpha\right) \\ &\geq m_*\left(\frac{\alpha}{8} - \frac{10000C_1}{\log^3(n/\varepsilon)}(\log^3\alpha + \log^3 m_*)\log^4\alpha\right) \quad (\text{by } (a+b)^3 \leq 4(a^3+b^3)) \\ &\geq m_*\left(\frac{\alpha}{8} - 10000C_1\left(\frac{\log^3\alpha}{\log^3(n/\varepsilon)} + 125\right)\log^4\alpha\right) \quad \left(\text{by } m_* = \frac{n^2}{\varepsilon^2}\log^3\frac{n}{\varepsilon} \leq \left(\frac{n}{\varepsilon}\right)^5\right). \end{aligned}$$

Thus, there exists a sufficiently large constant C_2, which is independent of n and ε, such that $g(\alpha) \geq 0$ holds for all $\alpha \geq C_2$. Using this constant C_2, for all $m_i \geq C_2 m^*$, we have $\lambda_i n^2 \leq \frac{m_i}{8}$ as desired. ◀

Proof of Theorem 1. We say DH-Onestep(H_i,λ_i) is *successful* if $\tilde{H}_{i+1}$ is an ε_i-spectral sparsifier of $\tilde{H}_i$ and $m_{i+1} \leq \frac{3}{4}m_i$ holds. DH-Sparsify(H,ε) calls DH-Onestep(H_i,λ_i) only when $m_i \geq C_2 m^*$ and $i \leq T$. Therefore, by Lemmas 6 and 18, with probability at least $1 - O\left(\frac{T}{n^2}\right) \gtrsim 1 - O\left(\frac{1}{n}\right)$, DH-Onestep$(H_i,\lambda_i)$ is successful for all i with $0 \leq i \leq i_{\mathrm{end}}$. Hence, assuming all DH-Onestep(H_i,λ_i) to be successful, we below prove that the output hypergraph $\tilde{H}$ has $O\left(\frac{n^2}{\varepsilon^2}\log^3\frac{n}{\varepsilon}\right)$ hyperarcs and that $\tilde{H}$ is an ε-spectral sparsifier of H.

We first discuss the size of $\tilde{H}$. If $m_i \leq C_2 m^* = \frac{C_2 n^2\log^3(n/\varepsilon)}{\varepsilon^2}$ occurs for some $i \leq T-1$, then m_i gives the size of $\tilde{H}$ by the termination rule of DH-Sparsify, which is already small enough. Hence we below assume $m_i \geq C_2 m^*$ for all $i < T$. Since every DH-Onestep(H_i,λ_i) is successful, $m_{i+1} \leq \frac{3}{4}m_i$ holds for all $i = 0, 1, \cdots, T-1$. Thus, it holds that

$$m_T \leq m\cdot\left(\frac{3}{4}\right)^T \leq m\cdot\left(\frac{3}{4}\right)^{\log_{4/3}\frac{m\varepsilon^2}{n^2\log^3(n/\varepsilon)}} = \frac{n^2\log^3(n/\varepsilon)}{\varepsilon^2}.$$

Therefore, we have $|\tilde{F}| = O\left(\frac{n^2}{\varepsilon^2}\log^3\frac{n}{\varepsilon}\right)$.

We then show that $\tilde{H}$ is an ε-spectral sparsifier of H. Since $\tilde{H}_{i+1}$ is an ε_i-spectral sparsifier of $\tilde{H}_i$ for all $i = 0, 1, \cdots, i_{\mathrm{end}}-1$, the output hypergraph $\tilde{H} = \tilde{H}_{i_{\mathrm{end}}}$ is an $\tilde{\varepsilon}$-spectral sparsifier of H, where

$$\tilde{\varepsilon} = \max\left\{\prod_{i=0}^{i_{\mathrm{end}}-1}(1+\varepsilon_i) - 1, 1 - \prod_{i=0}^{i_{\mathrm{end}}-1}(1-\varepsilon_i)\right\}.$$

A simple calculation yields the following upper bound on $\tilde{\varepsilon}$:

$$\tilde{\varepsilon} \le \sum_{j=1}^{i_{\mathrm{end}}} \sum_{0 \le i_1 < \cdots < i_j \le i_{\mathrm{end}}-1} \varepsilon_{i_1}\varepsilon_{i_2}\cdots\varepsilon_{i_j} \le \sum_{j=1}^{i_{\mathrm{end}}} \left(\sum_{i=0}^{i_{\mathrm{end}}-1} \varepsilon_i \right)^j. \tag{10}$$

Since $m_{i+1} \le \frac{3}{4} m_i$ and $m_{i_{\mathrm{end}}-1} \ge C_2 m^*$, we have $m_{i_{\mathrm{end}}-j} \ge \left(\frac{4}{3}\right)^{j-1} C_2 m^* \ge \left(\frac{4}{3}\right)^j m^*$ for sufficiently large $C_2 \ge \frac{4}{3}$, hence $\log_{4/3}\left(\frac{m_{i_{\mathrm{end}}-j}}{m^*}\right) \ge j$. Using $\sum_{j=1}^{\infty} \frac{1}{j^2} \le \frac{\pi^2}{6}$, we obtain

$$\sum_{i=0}^{i_{\mathrm{end}}-1} \varepsilon_i = \sum_{i=0}^{i_{\mathrm{end}}-1} \frac{\varepsilon}{4 \log_{4/3}^2\left(\frac{m_i}{m^*}\right)} \le \sum_{j=1}^{\infty} \frac{\varepsilon}{4j^2} \le \frac{\varepsilon}{4} \cdot \frac{\pi^2}{6} \le \frac{\varepsilon}{2}.$$

Putting this into the right-hand side of eq. (10), we have

$$\sum_{j=1}^{i_{\mathrm{end}}} \left(\sum_{i=0}^{i_{\mathrm{end}}-1} \varepsilon_i \right)^j \le \sum_{j=1}^{i_{\mathrm{end}}} \left(\frac{\varepsilon}{2}\right)^j \le \frac{\frac{\varepsilon}{2}}{1-\frac{\varepsilon}{2}} \le \varepsilon. \tag{11}$$

By eqs. (10) and (11), $\tilde{H} = \tilde{H}_{i_{\mathrm{end}}}$ is an ε-spectral sparsifier of H.

To conclude, with probability at least $1 - O\left(\frac{1}{n}\right)$, DH-Sparsify$(H, \varepsilon)$ outputs an ε-spectral sparsifier of H with $O\left(\frac{n^2}{\varepsilon^2} \log^3 \frac{n}{\varepsilon}\right)$ hyperarcs. ◀

4.4 Total Time Complexity

We show that our algorithm runs in $O(r^2 m)$ time with probability at least $1 - O(1/n)$.

▶ **Theorem 19.** *For any directed hypergraph $H = (V, F, z)$ with the rank r and m hyperarcs and $\varepsilon \in (0, 1)$,* DH-Sparsify(H, ε) *runs in $O(r^2 m)$ time with probability at least $1 - O(1/n)$.*

Proof. We first discuss the running time of DH-Onestep$(\tilde{H}_i, \lambda_i)$, where $\tilde{H}_i = (V, \tilde{F}_i, \tilde{z}_i)$ and $|\tilde{F}_i| = m_i$. It first constructs a λ_i-coreset by calling CoresetFinder$(\tilde{H}_i, \lambda_i)$. CoresetFinder first constructs $A^{uv} = \{ f \in F \mid C(f) \ni (u, v) \}$ for $(u, v) \in C(F)$, which is done in $O(r^2 m_i)$ time since we have $|C(f)| = O(r^2)$ for each $f \in \tilde{F}_i$. Then, for each $(u, v) \in C(F)$, it selects the λ_i heaviest hyperarcs from $A^{uv} \setminus S$ in $O(|A^{uv} \setminus S|)$ time by using a selection algorithm [5], thus taking $O\left(\sum_{(u,v)\in C(F)} |A^{uv} \setminus S|\right) = O(r^2 m_i)$ time in total. Therefore, CoresetFinder$(\tilde{H}_i, \lambda_i)$ takes $O(r^2 m_i)$ time. After that, DH-Onestep samples the remaining hyperarcs in $O(m_i)$ time. Thus, DH-Onestep$(\tilde{H}_i, \lambda_i)$ takes $O(r^2 m_i)$ time.

We then bound the total time complexity. Since DH-Sparsify(H, ε) calls DH-Onestep$(\tilde{H}_i, \lambda_i)$ for $i = 0, 1, \ldots, T-1$ (or stops earlier), the total time complexity is at most $O\left(r^2 \sum_{i=0}^{T-1} m_i\right)$. From Lemmas 7 and 18, whenever DH-Onestep is called, we have $m_{i+1} \le \frac{3}{4} m_i$ with probability at least $1 - O(1/n^2)$. This implies that $\sum_{i=0}^{T-1} m_i \le m \sum_{i=0}^{T-1} \left(\frac{3}{4}\right)^i \le 4m$ holds with probability at least $1 - O(T/n^2) \gtrsim 1 - O(1/n)$. Therefore, the total time complexity is bounded by $O(r^2 m)$ with probability at least $1 - O(1/n)$. ◀

References

1 Noga Alon and Joel H. Spencer. *The Probabilistic Method*. John Wiley & Sons, 2016.

2 Alexandr Andoni, Jiecao Chen, Robert Krauthgamer, Bo Qin, David P. Woodruff, and Qin Zhang. On sketching quadratic forms. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 311–319, 2016.

3 Nikhil Bansal, Ola Svensson, and Luca Trevisan. New notions and constructions of sparsification for graphs and hypergraphs. In *Proceedings of the 2019 IEEE 60th Annual Symposium on Foundations of Computer Science*, pages 910–928. IEEE, 2019.

4 András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 47–55. ACM, 1996.

5 Manuel Blum, Robert W. Floyd, Vaughan R. Pratt, Ronald L. Rivest, and Robert Endre Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7(4):448–461, 1973.

6 Charles Carlson, Alexandra Kolla, Nikhil Srivastava, and Luca Trevisan. Optimal lower bounds for sketching graph cuts. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2565–2569. SIAM, 2019.

7 T.-H. Hubert Chan, Zhihao Gavin Tang, Xiaowei Wu, and Chenzi Zhang. Diffusion operator and spectral analysis for directed hypergraph Laplacian. *Theoretical Computer Science*, 784:46–64, 2019.

8 Yu Chen, Sanjeev Khanna, and Ansh Nagda. Near-linear size hypergraph cut sparsifiers. In *Proceedings of the 2020 IEEE 61st Annual Symposium on Foundations of Computer Science*, pages 61–72. IEEE, 2020.

9 Michael B. Cohen, Jonathan Kelner, John Peebles, Richard Peng, Anup B. Rao, Aaron Sidford, and Adrian Vladu. Almost-linear-time algorithms for Markov chains and new spectral primitives for directed graphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 410–419. ACM, 2017.

10 Satoru Fujishige and Sachin B. Patkar. Realization of set functions as cut functions of graphs and hypergraphs. *Discrete Mathematics*, 226(1-3):199–210, 2001.

11 Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(2-3):177–201, 1993.

12 Matthias Hein, Simon Setzer, Leonardo Jost, and Syama Sundar Rangapuram. The total variation on hypergraphs - learning on hypergraphs revisited. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

13 Arun Jambulapati, Yang P Liu, and Aaron Sidford. Chaining, group leverage score overestimates, and fast spectral hypergraph sparsification. *arXiv:2209.10539*, 2022.

14 Mohammad Ali Javidian, Zhiyu Wang, Linyuan Lu, and Marco Valtorta. On a hypergraph probabilistic graphical model. *Annals of Mathematics and Artificial Intelligence*, 88(9):1003–1033, 2020.

15 Michael Kapralov, Robert Krauthgamer, Jakab Tardos, and Yuichi Yoshida. Towards tight bounds for spectral sparsification of hypergraphs. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 598–611. ACM, 2021.

16 Michael Kapralov, Robert Krauthgamer, Jakab Tardos, and Yuichi Yoshida. Spectral hypergraph sparsifiers of nearly linear size. In *Proceedings of the 2021 IEEE 62nd Annual Symposium on Foundations of Computer Science*, pages 1159–1170, 2022.

17 Ioannis Koutis and Shen Chen Xu. Simple parallel and distributed algorithms for spectral graph sparsification. *ACM Transactions on Parallel Computing*, 3(2):1–14, 2016.

18 James R. Lee. Spectral hypergraph sparsification via chaining. *arXiv:2209.04539*, 2022.

19 Pan Li, Niao He, and Olgica Milenkovic. Quadratic decomposable submodular function minimization: Theory and practice. *Journal of Machine Learning Research*, 21(106):1–49, 2020.

20 Kazusato Oko, Shinsaku Sakaue, and Shin ichi Tanigawa. Nearly tight spectral sparsification of directed hypergraphs by a simple iterative sampling algorithm. *arXiv:2204.02537*, 2022.

21 Akbar Rafiey and Yuichi Yoshida. Sparsification of decomposable submodular functions. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, volume 36, pages 10336–10344, 2022.

22 Tasuku Soma and Yuichi Yoshida. Spectral sparsification of hypergraphs. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2570–2581. SIAM, 2019.

23 Daniel A. Spielman. Spectral and Algebraic Graph Theory, 2019. (visited on 02/03/2022). URL: `http://cs-www.cs.yale.edu/homes/spielman/sagt/sagt.pdf`.

24 Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011.

25 Yuuki Takai, Atsushi Miyauchi, Masahiro Ikeda, and Yuichi Yoshida. Hypergraph clustering based on pagerank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1970–1978. ACM, 2020.

26 Shang-Hua Teng. Scalable algorithms for data and network analysis. *Foundations and Trends in Theoretical Computer Science*, 12(1-2):1–274, 2016. `doi:10.1561/0400000051`.

27 Nisheeth K. Vishnoi. $Lx = b$. *Foundations and Trends in Theoretical Computer Science*, 8(1-2):1–141, 2013. `doi:10.1561/0400000054`.

28 Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. HyperGCN: A new method of training graph convolutional networks on hypergraphs. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

29 Naganand Yadati, Vikram Nitin, Madhav Nimishakavi, Prateek Yadav, Anand Louis, and Partha Talukdar. NHP: Neural hypergraph link prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1705–1714. ACM, 2020.

30 Yuichi Yoshida. Cheeger inequalities for submodular transformations. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2582–2601. SIAM, 2019.

31 Chenzi Zhang, Shuguang Hu, Zhihao Gavin Tang, and T.-H. Hubert Chan. Re-revisiting learning on hypergraphs: Confidence interval, subgradient method, and extension to multiclass. *IEEE Transactions on Knowledge and Data Engineering*, 32(3):506–518, 2020.

32 Chunjiang Zhu, Sabine Storandt, Kam-Yiu Lam, Song Han, and Jinbo Bi. Improved dynamic graph learning through fault-tolerant sparsification. In *Proceedings of the 36th International Conference on Machine Learning*, pages 7624–7633. PMLR, 2019.

The Communication Complexity of Set Intersection Under Product Distributions

Rotem Oshman
Tel-Aviv University, Israel

Tal Roth
Tel-Aviv University, Israel

Abstract

We consider a multiparty setting where k parties have private inputs $X_1, \ldots, X_k \subseteq [n]$ and wish to compute the intersection $\bigcap_{\ell=1}^{k} X_\ell$ of their sets, using as little communication as possible. This task generalizes the well-known problem of set disjointness, where the parties are required only to determine whether the intersection is empty or not. In the worst-case, it is known that the communication complexity of finding the intersection is the same as that of solving set disjointness, regardless of the size of the intersection: the cost of both problems is $\Omega(n \log k + k)$ bits in the shared blackboard model, and $\Omega(nk)$ bits in the coordinator model.

In this work we consider a realistic setting where the parties' inputs are independent of one another, that is, the input is drawn from a product distribution. We show that this makes finding the intersection significantly easier than in the worst-case: only $\tilde{\Theta}((n^{1-1/k}(\mathrm{H}(S)+1)^{1/k}) + k)$ bits of communication are required, where $\mathrm{H}(S)$ is the Shannon entropy of the intersection S. We also show that the parties do not need to know the exact underlying input distribution; if we are given in advance $O(n^{1/k})$ samples from the underlying distribution μ, we can learn enough about μ to allow us to compute the intersection of an input drawn from μ using expected communication $\tilde{\Theta}((n^{1-1/k}\,\mathbb{E}[|S|]^{1/k}) + k)$, where $|S|$ is the size of the intersection.

2012 ACM Subject Classification Theory of computation → Communication complexity

Keywords and phrases Communication complexity, intersection, set disjointness

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.95

Category Track A: Algorithms, Complexity and Games

Funding Research funded by the Israel Science Foundation, Grant No. 2801/20, and also supported by Len Blavatnik and the Blavatnik Family foundation.

1 Introduction

Communication complexity is concerned with understanding the communication cost of computing on data that is partitioned between $k \geq 2$ parties, with each party holding a private input $\boldsymbol{X}^i$.[1] The parties would like to jointly compute some function $f(\boldsymbol{X}^1, \ldots, \boldsymbol{X}^k)$ of their data, using as little communication as possible. Two models of communication are typically studied: in the *shared blackboard* model, the parties communicate by writing messages on a "board" that all the other parties can read (essentially, they communicate by broadcast); in the *private-channel* model, the parties communicate over private channels. For both models, there is a wealth of protocols and lower bounds characterizing the cost of computing different functions, and obtaining applications in areas ranging from distributed graph algorithms (see [30, 10, 6, 8, 9] and many more), to streaming algorithms (e.g., [1, 3, 15])

[1] This is called *number-in-hand* because each party holds its own private input; in the *number-on-forehead* model, each party can see the inputs of all the other parties, but not its own input. The number-on-forehead model has compelling applications in circuit complexity, but it is not a realistic model of a distributed system.

and beyond. We focus on the *distributional setting*, where the inputs $\boldsymbol{X}^1, \ldots, \boldsymbol{X}^k$ are drawn from a distribution μ, and our goal is to compute $f(\boldsymbol{X}^1, \ldots, \boldsymbol{X}^k)$ with a low error probability over μ.

In this paper we study the cost of computing an intersection: each party holds a set $\boldsymbol{X}^i \subseteq [n]$, and our goal is to output the intersection, $\boldsymbol{S} = \bigcap_{i=1}^n \boldsymbol{X}^i$. This fundamental problem has many applications, including computing joins for distributed databases [10, 16]; computing the Jaccard similarity of data sets, the number of distinct elements, and rarity [10]; and algebraic function computation on reconciled data [19]. Recently this problem also found applications in online advertising [14], notably at Google [18].

Computing the intersection of sets $\boldsymbol{X}^1, \ldots, \boldsymbol{X}^k$ is as at least as hard as solving the *set disjointness* problem, where we are required to determine whether $\cap_{i=1}^k \boldsymbol{X}^i = \emptyset$. Set disjointness is known to require $\Omega(n)$ bits of communication for two-parties [26, 24], $\Omega(n \log k)$ bits of communication for k parties on the shared blackboard [8], and $\Omega(kn)$ bits of communication for k parties with private channels [6]. However, all of these hardness results hold only when the parties' inputs are highly-correlated; if the input is drawn from a *product distribution* (i.e., the parties' inputs are independent of one another), then two-party set disjointness requires only $\Theta(\sqrt{n})$ bits [2, 5], and this was recently extended to the multi-party setting, where it was shown that the communication cost is $\tilde{\Theta}(n^{1-1/k})$ in both the shared blackboard and the coordinator models [12]. We note that all the lower bounds mentioned above hold even for input distributions where the intersection is of small constant size – that is, either the input sets do not intersect at all, or they have a non-empty but constant-sized intersection, and our goal is to distinguish between these two cases.

Our main question in this paper is whether under product distributions we can efficiently compute the full intersection of the parties' inputs, rather than merely determining whether it is empty or not. We show that the answer is *yes*, at least when the intersection is not too large: if the expected intersection size is $\mathbb{E}[\boldsymbol{S}] = s$, then we can compute the full intersection using $\tilde{\Theta}(n^{1-1/k} s^{1/k})$ bits of communication in expectation, in both the shared blackboard and the coordinator models. This can be viewed as a natural extension of the tight bound for set disjointness in the product case, which is $\tilde{\Theta}(n^{1-1/k})$, even when $s = \Theta(1)$ [12]. Our protocol and lower bound bear some similarities to [12]. We generalize our result in two ways, motivated by practical applications.

"Learning" the input distribution. In this scenario, instead of being told the input distribution μ, we are given iid *samples* from μ, and must "learn" whatever we can about μ before running the protocol on the actual input (which is also drawn from μ). Can we learn enough about μ to exploit its product structure, without requiring a prohibitive number of samples? In Section 4.1 we show that the answer is *yes*: $\tilde{O}(n^{1/k})$ sample suffice to learn enough about μ to solve any future instance with optimal communication cost.

▶ **Theorem 1.** *Let $\delta > 0$, and assume we have access to $O\left(n^{1/k} \log(nk/\delta)\right)$ iid samples from an unknown product distribution μ. Then we can construct a zero-error two-round protocol Π for computing the intersection, such that with probability at least $1-\delta$ over the samples, the protocol Π that we constructed has expected communication cost $O\left(kn^{1-1/k}\, \mathbb{E}\left[|\boldsymbol{S}|+1\right]^{1/k} \log n\right)$ on inputs drawn from μ.*

In particular, when $k \geq \log n$, we require only a single sample from μ. This is perhaps surprising, since it is known that in order to fully learn a distribution over kn bits – that is, in order to output a distribution μ' that is ϵ-close to μ in statistical distance – the number of samples required is $\Omega\left(2^{nk}/(nk)\right)$ [28]. The key is that instead of learning the entire distribution, we show that it suffices to estimate the marginal expectation of each input bit, which is a much easier task.

It remains open whether $\Omega(n^{1/k})$ samples are truly necessary to obtain the optimal communication complexity, and more generally, what is the tradeoff between the number of samples we have and the communication complexity we can obtain. However, we show in Section 5 that if we do not have *any* prior information about the distribution μ (i.e., no samples), then the fact that μ is a product distribution is not helpful at all: for any function f, computing f under an unknown product distribution is as hard as computing f under non-product distributions.

Large but predictable intersections. Although we assume that the parties have independent inputs, we do not assume that the elements inside a given party's input are independent of one another: for example, if each party's input is a list of items purchased by some set of customers, then the elements may be highly correlated, as one item purchased by a customer is likely to tell us a lot about other items the same customer is likely to purchase. Correlations between elements can lead to a situation where the intersection is "large but fairly predictable", in the sense that while the intersection S has large expected size, its Shannon entropy $\mathrm{H}(S)$ is much smaller. As an extreme example, if we have two parties with inputs X, Y that are each either $[n]$ or $\emptyset$ with probability $1/2$, then the expected size of the intersection is $n/2$, but its Shannon entropy is only 1.

In Section 4.3 we show that it is not the size of the intersection but its entropy that matters: when the distribution μ is known, we can replace the size $|\boldsymbol{S}|$ of the intersection with its entropy, $\mathrm{H}(\boldsymbol{S})$, and obtain the following.

▶ **Theorem 2.** *Let μ be a product distribution known to all the parties. Then in the coordinator model, there is an $O(\log n)$-round zero-error deterministic protocol for finding the intersection, with expected communication cost at most $O\left(k^2 n^{1-1/k} \left(\mathrm{H}(\boldsymbol{S})+1\right)^{1/k} \log n + k\right)$, where the expectation is with respect to the input distribution μ.*

We remark that for non-product distributions this is not possible: in the hard distribution of Razborov [24] for two-party set disjointness, the intersection has entropy $O(\log n)$, as it is always either empty or contains a single element which is uniformly random over $[n]$. Nevertheless, even determining whether the intersection is empty or not requires $\Omega(n)$ bits of communication, and this of course implies that *finding* the intersection also requires $\Omega(n)$ communication.

Lower bounds. To complement our protocols above, in Section 5 we prove a matching lower bound, up to polylogarithmic factors:

▶ **Theorem 3.** *For every $n, k \in \mathbb{N}$ with $2 \le k \le \log n$, and for every $s \in [1, n/2]$, there exists a product distribution μ over $\{0,1\}^{n \times k}$ such that*

- $\mathbb{E}_\mu [\, |\boldsymbol{S}| \,] = s$,
- $s \le \mathrm{H}(\boldsymbol{S}) \le (s+1)\log n$, *and*
- *Any deterministic protocol for computing the intersection with error probability at most $1/10$ over μ has expected communication complexity $\Omega(n^{1-1/k} s^{1/k}/k^2)$.*

Although the lower bound is stated only for $k \le \log n$ parties, we can "stretch" the lower bound from $k = \log n$ to larger k by generating the inputs of the first $\log n$ parties using the hard distribution from the theorem, and giving the remaining parties the set $[n]$. As a result, for $k > \log n$, we obtain a lower bound of $\tilde{\Omega}(n)$ regardless of the intersection size s, and this is tight up to polylogarithmic factors.

Our lower bound actually applies to a weak output model, where every element of the intersection can be output by a different party: at the end of the protocol, each party ℓ outputs a list of decisions of the form "$i \in \boldsymbol{S}$" or "$i \notin \boldsymbol{S}$". We require that for every coordinate $i \in [n]$, one party must output a decision for i, but the identity of the party that output a decision for i need not be known in advance (that is, it may be a function of the transcript). The party j that outputs a decision for i may rely on its own input $\boldsymbol{X}^j$ when making the decision. This output model is quite weak compared to the standard output model that we assume in our protocols, where the output to the computation must be computable from the transcript of the protocol. Making the lower bound work in this weak model is technically challenging: our lower bound uses information-theoretic arguments, which typically rely on the fact that an *external observer* must learn a lot of information about the inputs, but this is not necessarily true in the weak output model.

We also remark that all of the results discussed up to this point (both upper and lower bounds) assume that the protocol must output the *entire* intersection correctly with high probability: if we output a set that differs from the true intersection in even a single element, this is considered an error. One can also consider a weaker notion, which is more appealing for lower bounds, where for every $i \in [n]$, we only need to determine whether $i \in \boldsymbol{S}$ with good *marginal* error probability, independent of the other elements. This weaker notion is only meaningful when many coordinates have constant probability bounded away from 0 and 1 of being in the intersection, otherwise we can simply guess independently for each coordinate whichever outcome is more likely for that coordinate; e.g., if $\Pr[i \in \boldsymbol{S}] = 1/\sqrt{n}$ for every i, we can guess that the intersection is empty, and still be correct on every element with marginal probability $1 - 1/\sqrt{n}$. However, if every element has probability between $1/3$ and $1/2$ of being in the intersection, then we can also prove a tight lower bound even for the case where the protocol only needs to succeed with good marginal probability on each element (see the full version of this paper for a proof of this theorem).

2 Related Work

Set disjointness has been studied extensively, in many versions and models; we refer to the surveys [11, 27] for more background. The problem of computing the intersection, or of finding an element in the intersection, has also been studied, for two parties [7, 25, 10, 13, 29, 4, 17] and for more than two parties [10, 23]; to our knowledge, all previously mentioned prior work is for either worst-case hardness (that is, a non-distributional setting, where the inputs are chosen adversarially), or for non-product distributions, and is thus not directly relevant to the current paper. In addition, [20, 21, 22] studied a different scenario where two parties wish to compute the bitwise-AND of their input vectors (as well as other functions), assuming the coordinates of the vectors are iid, in the regime where the input length goes to infinity and the error is vanishing. In contrast, here we consider multi-party intersection with a fixed input length and constant error, and we do not assume that the coordinates are iid.

The hardness of set disjointness under product distributions was first studied in [2], which proved a lower bound of $\Omega(\sqrt{n})$ and an upper bound of $O(\sqrt{n}\log n)$ on the communication complexity of the problem. Later, [5] eliminated the log-factor and improved the upper bound to $O(\sqrt{n})$, and showed that in general, when the parties' inputs have mutual information I with one another, the communication cost of set disjointness is $\tilde{\Theta}(\sqrt{n(I+1)})$ (the product case is the case where $I = 0$). It turns out that the techniques of [2, 5] do not scale to more than 2 parties, but in [12], using different techniques, it was shown that $\tilde{\Theta}(n^{1-1/k})$ bits are necessary and sufficient in the k-player setting.

The protocols of [2, 5, 12] for set disjointness share the following feature: at any point in the protocol, if we identify that given what we have learned so far the probability that the inputs are disjoint is bounded by some small threshold ϵ, then we halt and output "not disjoint". If the probability of disjointness is greater than ϵ, we rely on this fact to make progress: in [2, 5], we use it to efficiently sample a large set that is disjoint from one player's input, and those elements are then discarded from consideration; in [12], we exploit the fact that no single element is likely to be in the intersection to show that each element $i \in [n]$ is probably missing from the input of some specific player $p(i)$. We then ask each player j to say only the elements $X_j \cap \{i : p(i) = j\}$, as this set is likely to be small, but at the same time it helps us learn of many elements that are definitely not in the intersection. If we want to find the intersection in full, the basic approach of [12] continues to work if we know that we have a *small* intersection, but it breaks down when the intersection is large.

Our work generalizes the basic approach of [12] to handle larger intersection sizes. This yields a protocol for finding the intersection that depends on the *entropy* of the intersection instead of its expected size (as already noted in the introduction, the former may be much smaller than the latter). We also show that the basic protocol of [12] can be made *robust*, in the sense that the players do not need to know the exact underlying input distribution.

When the number of players is $k \geq \log n$, [12] gives a different protocol that actually finds the entire intersection, and has communication cost $\tilde{O}(n)$. We show that this protocol can be made robust as well, and in fact a *single* sample from the underlying product distribution is enough, with high probability, for the players to be able to successfully execute the protocol.

As for lower bounds, [12] gave a lower bound on finding the intersection under a product distribution, for the case where the expected intersection size is 1 (which coincides with the set disjointness problem), when the transcript reveals the intersection to an *external observer*. In this work, we generalize this lower bound in two ways. First, our lower bound must handle larger intersections, up to linear in n. This large range of intersection sizes implies, naturally, that the lower bound proof must handle both very small and very large probabilities, which requires delicate handling. Secondly, our bound is proven in the weaker model where for each coordinate, one of the players must decide whether this coordinate is in the intersection or not, and the identity of this player may not even be known in advance.

3 Preliminaries

Notation. We use boldface letters to denote random variables. Given a vector v indexed by $\{1, \ldots, m\}$ and a subset of coordinates $J = \{j_1, \ldots, j_\ell\}$, we denote by $v_J = v_{j_1,\ldots,j_\ell}$ coordinates J of v. If $\boldsymbol{A}$ is a random variable and $\mathcal{E}$ is an event, then $\boldsymbol{A}|_{\mathcal{E}}$ denotes the distribution of $\boldsymbol{A}$ conditioned on $\mathcal{E}$.

The input to the k players is denoted $\boldsymbol{X}^1, \ldots, \boldsymbol{X}^k \in \{0,1\}^n$. It is convenient to sometimes view the inputs to the players as sets, and sometimes as the characteristic vectors of their sets. We use $\boldsymbol{X}_i^\ell$ to denote the i-th coordinate of player ℓ's input, when viewed as a characteristic vector. The intersection of the players' inputs is denoted $\boldsymbol{S} = \bigcap_{\ell \in [k]} \boldsymbol{X}^\ell$, and for each $i \in [n]$, $\boldsymbol{S}_i$ is an indicator for the event "$i \in \boldsymbol{S}$". We refer to $\boldsymbol{S}_1, \ldots, \boldsymbol{S}_n$ as the *bits of the intersection*.

We sometimes abuse notation by conflating Bernoulli distributions with their expected value: for example, if the input distribution is μ, we use μ_i^ℓ to denote both the marginal distribution of $\boldsymbol{X}_i^\ell$, and the expected value of $\boldsymbol{X}_i^\ell$.

The shared blackboard model. In this classical model of multiparty communication, we have k players, with private inputs $\boldsymbol{X}^1, \ldots, \boldsymbol{X}^k$. The players communicate by writing on a *shared blackboard* that all players can see. At any point in the protocol, the identity of

the next player to write on the board is determined by the current contents of the board. We refer to the contents of the board as the *transcript* of the protocol, and denote it by the random variable $\boldsymbol{\Pi}$.

The coordinator model. In the coordinator model of multiparty computation, in addition to the k players, we also have a *coordinator*, who has no input. The players communicate with the coordinator over private channels, but players cannot communicate directly with one another. The order of communication is governed by the coordinator, and the *transcript* of the protocol consists of all messages sent and received by the coordinator.

Set intersection. In the k-player set intersection problem, our goal is to compute:

$$\mathrm{INT}_{n,k}(X^1, \ldots, X^k) = \cap_{\ell=1}^{k} X^\ell.$$

Since the intersection can be very large, it is crucial that we do not charge the players for "writing" the output at the end of the protocol. Instead, we assume one of the following two output models:

- In our upper bounds, the output is some predetermined function of the transcript; in other words, an external observer can learn the intersection just by observing the transcript of the protocol, without knowing any of the inputs.
- In our lower bounds, the output is jointly produced by the players, with each player j choosing at the end of the protocol a set of indices I_j, and outputting the bits $\{\boldsymbol{S}_i : i \in I_j\}$. The index set I_j may depend on the transcript of the protocol, but not on player j's input. However, the *values* that player j outputs for the bits $\{\boldsymbol{S}_i : i \in I_j\}$ may depend on player j's input. Thus, an external observer that sees only the transcript of the protocol is able to learn which player will output which bits, but not the values of the output bits. We require that every bit $\boldsymbol{S}_i$ must be output by some player; if more than one player outputs the bit $\boldsymbol{S}_i$, and the players disagree, this is considered an error.

Information theory and entropy. The *Shannon entropy* of a random variable $\boldsymbol{A} \sim \mu$ is given by

$$\mathrm{H}_\mu(\boldsymbol{A}) = \sum_{a \in supp(\mu)} \mu(a) \log \frac{1}{\mu(a)},$$

where $supp(\mu)$ denotes the support of the distribution μ. We omit the subscript μ when the distribution is clear from the context.

Given jointly-distributed variables $(\boldsymbol{A}, \boldsymbol{B}) \sim \mu$, with marginal distributions $\mu_{\boldsymbol{A}}$ and $\mu_{\boldsymbol{B}}$ respectively, the *conditional entropy* of $\boldsymbol{A}$ given $\boldsymbol{B}$ is

$$\mathrm{H}_\mu(\boldsymbol{A}|\boldsymbol{B}) = \mathop{\mathbb{E}}_{b \sim \mu_{\boldsymbol{B}}} \left[\mathrm{H}_{\mu|_{\boldsymbol{B}=b}}(\boldsymbol{A})\right].$$

We sometimes abuse notation by writing $\mathrm{H}_\mu(\boldsymbol{A}|\mathcal{E})$ to denote $\mathrm{H}_{\mu|_{\mathcal{E}}}(\boldsymbol{A})$ (here, $\mathcal{E}$ is an event, not a random variable).

We rely on the following basic properties of the entropy:

1. Entropy is non-negative: $\mathrm{H}(\boldsymbol{A}) \geq 0$.
2. Conditioning does not increase entropy: $\mathrm{H}(\boldsymbol{A}|\boldsymbol{B}) \leq \mathrm{H}(\boldsymbol{A})$.
3. The chain rule for entropy: $\mathrm{H}(\boldsymbol{A}_1, \ldots, \boldsymbol{A}_m) = \sum_{i=1}^{m} \mathrm{H}(\boldsymbol{A}_i|\boldsymbol{A}_{i-1}, \ldots, \boldsymbol{A}_1)$.
4. Subadditivity: $\mathrm{H}(\boldsymbol{A}_1, \ldots, \boldsymbol{A}_m) \leq \sum_{i=1}^{m} \mathrm{H}(\boldsymbol{A}_i)$, with equality iff $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_m$ are independent.

For $p \in [0,1]$, we use $\mathrm{H}(p)$ as short-hand notation for the Shannon entropy of a Bernoulli random variable with probability p of being 1. In our lower bound, we use the following fact:

▶ **Fact 4.** *Let $p \in [0,1]$, Then* $\min\{p, 1-p\} \leq \mathrm{H}(p)$.

To measure the amount of information a protocol reveals about its inputs, we use *mutual information*. The *mutual information* between random variables $\boldsymbol{A}$ and $\boldsymbol{B}$ is given by $\mathsf{I}(\boldsymbol{A}\,;\,\boldsymbol{B}) := H(\boldsymbol{A}) - H(\boldsymbol{A} \mid \boldsymbol{B})$. For random variables $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$, the *conditional mutual information* between $\boldsymbol{A}$ and $\boldsymbol{B}$ given $\boldsymbol{C}$ is $\mathsf{I}(\boldsymbol{A}\,;\,\boldsymbol{B} \mid \boldsymbol{C}) := H(\boldsymbol{A} \mid \boldsymbol{C}) - H(\boldsymbol{A} \mid \boldsymbol{B}, \boldsymbol{C})$.

The following Lemma will be useful in our lower bound:

▶ **Lemma 5.** *Let $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\Pi}$ be random variables, such that $\boldsymbol{A}$ and $\boldsymbol{B}$ are independent. Then* $\mathsf{I}(\boldsymbol{A}\,;\,\boldsymbol{\Pi} \mid \boldsymbol{B}) = \mathsf{I}(\boldsymbol{A}\,;\,\boldsymbol{\Pi}) + \mathsf{I}(\boldsymbol{A}\,;\,\boldsymbol{B} \mid \boldsymbol{\Pi})$.

For an event $\mathcal{E}$, we sometimes abuse notation and denote $\mathsf{I}(\boldsymbol{A}\,;\,\boldsymbol{B} \mid \mathcal{E}) := \mathsf{I}(\boldsymbol{A}|_{\mathcal{E}}\,;\,\boldsymbol{B}|_{\mathcal{E}})$.

To measure the difference between two distributions, we use KL divergence:

▶ **Definition 6** (KL divergence)**.** *For two distributions μ,μ' supported over a set χ, the* KL divergence *of μ from μ' is:*

$$D(\mu \,||\, \mu') := \sum_{x \in \chi} \mu(x) \log \frac{\mu(x)}{\mu'(x)}.$$

We sometimes use $D(p \,||\, p')$ as short-hand notation for the divergence between the Bernoulli distributions with probability p and p' (resp.) of being 1.

KL divergence has the following monotonicity property, which will be useful in our upper bound:

▶ **Lemma 7.** *Let $0 < p < q \leq a < 1/100$ be constants, then $D(p+a \,||\, p) \geq D(q+a \,||\, q)$.*

The proof of Lemma 7 will appear in the full version of this paper.

Our lower bound also uses *Pinsker's inequality*, which asserts that for any $p, p' \in (0,1)$ we have $|p - p'| \leq \sqrt{D(p \,||\, p') \ln 2/2}$.

The mutual information between two variables $\boldsymbol{A}, \boldsymbol{B}$ is equal to the *expected divergence* of $\boldsymbol{A}$'s posterior distribution given $\boldsymbol{B}$, from $\boldsymbol{A}$'s prior distribution (or vice-versa):

▶ **Fact 8.** *For any random variables $\boldsymbol{A}, \boldsymbol{B}$ we have* $\mathsf{I}(\boldsymbol{A}\,;\,\boldsymbol{B}) = \mathbb{E}_{b \sim \boldsymbol{B}}\left[D\left(\boldsymbol{A}|_{\boldsymbol{B}=b} \,||\, \boldsymbol{A}\right)\right]$.

The following technical lemmas will be useful in our lower bound. The first bounds the "difference" between two Bernoulli random variables, in terms of their KL divergence:

▶ **Lemma 9.** *Let p, q be constants in $(0,1)$, and let $\alpha \in (0, 1/2)$, such that $D(q \,||\, p) < p\alpha^2/(4 \ln 2)$. Then we have $q/p \in ((1-\alpha)p, (1+\alpha)p)$.*

In Section 4.2 we use the tight version of the additive Chernoff bound, which is stated in terms of KL divergence: for a sum $\boldsymbol{Y} = \sum_{i=1}^{m} \boldsymbol{Y}_i$ of iid Bernoulli random variables with $\mathbb{E}[\boldsymbol{Y}_i] = p$, we have

$$\Pr\left[\sum_{i=1}^{n} \boldsymbol{Y}_i/n \geq p + \epsilon\right] \leq e^{-D(p+\epsilon \,||\, p)}, \qquad \text{and} \qquad \Pr\left[\sum_{i=1}^{n} \boldsymbol{Y}_i/n \leq p - \epsilon\right] \leq e^{-D(p-\epsilon \,||\, p)}.$$

4 Upper Bounds

In this section we give three upper bounds. The first two address the case where the number of parties is $k \leq \log n$: we first give a protocol with expected communication $\tilde{O}(n^{1-1/k}\,\mathbb{E}\,[\boldsymbol{S}]^{1/k})$, which can also handle input distributions that are known only approximately, and then build on it to construct a protocol with expected communication $\tilde{O}(n^{1-1/k}\mathrm{H}\,(\boldsymbol{S})^{1/k})$, replacing the expected size of the intersection with its entropy. These two protocols can be used in either the shared blackboard model or the coordinator model, since one model can simulate the other with multiplicative cost at most $O(k) = O(\log n)$.

The third protocol is for the case where $k > \log n$, in the coordinator model, which is the harder of the two models when k is large. This final protocol relies on advance access to only a single sample from the input distribution, and computes the intersection with expected communication $\tilde{O}(n + k)$.

Approximate knowledge of a distribution. As explained above, some of our protocols assume that the players do not exactly know the underlying input distribution, and instead are provided advanced access to samples from the distribution. We use these samples to approximate the marginal distribution of every bit $\boldsymbol{X}_i^j$ in the input. It is crucial to allow both multiplicative and additive approximation error, as allowing only one type of error would make it costlier to obtain the approximation (in terms of the number of samples required; see Section 4.2 below).

▶ **Definition 10.** *Let $\epsilon \geq 0, \alpha \in [0,1)$ and let $b \in [0,1]$. We say that a value $a \in [0,1]$ is an* (α,ϵ)-approximation *of b if $(1-\alpha)a - \epsilon \leq b \leq (1+\alpha)a + \epsilon$ and also $(1-\alpha)(1-a) - \epsilon \leq 1 - b \leq (1+\alpha)(1-a) + \epsilon$.*

We extend this definition to a distribution μ over $\{0,1\}^{n\times k}$ by saying that a collection of values $\left(a_i^\ell\right)_{i\in[n],\ell\in[k]} \subseteq [0,1]^{n\times k}$ is an (α,ϵ)-*approximation for the marginals of* μ if a_i^ℓ is an (α,ϵ)-approximation of the marginal μ_i^ℓ for every $i \in [n]$ and $\ell \in [k]$.

4.1 Basic Protocol for Computing Intersections ($k \leq \log n$)

In this section we give our protocol for computing the intersection of the players' inputs assuming approximate knowledge of the marginals of the input distribution. We assume that the number of players is $k \leq \log n$.

▶ **Theorem 11.** *Suppose all players know values $\left(a_i^\ell\right)_{i\in[n],\ell\in[k]}$ that* (α,ϵ)-approximate *the marginals of a product distribution μ. Then there is a zero-error two-round protocol in the coordinator model for finding the intersection, with expected communication cost*

$$O\left(\left(\frac{1+\alpha}{1-\alpha}\right)\left(kn^{1-1/k}\,\mathbb{E}\,[|\boldsymbol{S}|]^{1/k} + 2\epsilon kn\right)\log n + k\right),$$

where the expectation is over the input distribution μ.

From here on, we will refer to this protocol as Π_{base}.

High-level overview. We begin with a high-level overview of Π_{base}, assuming for simplicity that the players know the true marginals $\left(\mu_i^\ell\right)_{i\in[n],\ell\in[k]}$ of the input distribution.

For each coordinate $i \in [n]$, the protocol checks whether i is in the intersection using one of the following two strategies:

- "The 1-strategy": this strategy is appropriate for coordinates i where some player ℓ has a very small probability that $\boldsymbol{X}_i^\ell = 1$. In this case we can make good progress at little expected cost by asking player ℓ to speak up only if it has $\boldsymbol{X}_i^\ell = 1$: with good probability, player ℓ says nothing, and we learn that coordinate i is not in the intersection.
  Concretely, we find the player ℓ that has the smallest probability that $\boldsymbol{X}_i^\ell = 1$, that is, the smallest value of μ_i^ℓ (breaking ties arbitrarily), and ask this player to send index i iff $\boldsymbol{X}_i^\ell = 1$. If player ℓ did not send index i, we learn that $\boldsymbol{X}_i^\ell = 0$, and therefore coordinate i is not in the intersection. However, if player i did send index i, then $\boldsymbol{X}_i^\ell = 1$, and coordinate i might be in the intersection; to check, we simply ask all players $\ell' \neq \ell$ to send $\boldsymbol{X}_i^{\ell'}$, and then we check if they all sent 1.
  The expected communication cost of this strategy is at most $k \log n \min_{\ell \in [k]} \mu_i^\ell$: with probability $1 - \mu_i^\ell$ we have $\boldsymbol{X}_i^\ell = 0$, and in this case no bits are sent.[2] With probability $\min_{\ell \in [k]} \mu_i^\ell$, the player that has the minimum μ_i^ℓ sends index i, and the other players ℓ' follow suit by announcing $\boldsymbol{X}_i^{\ell'}$, for a total cost of at most $k \log n$ bits.
- "The 0-strategy": this strategy is appropriate for coordinates i where all players $\ell \in [k]$ are fairly likely to have $\boldsymbol{X}_i^\ell = 1$ (i.e., μ_i^ℓ is fairly large). In this case, we ask each player ℓ to announce index i iff $\boldsymbol{X}_i^\ell = 0$, and we then know that i is in the intersection iff no player sent index i.
  The expected communication cost of this strategy is $\log n \cdot \sum_{\ell \in [k]} \left(1 - \mu_i^\ell\right)$.

To choose which strategy to pursue for a given coordinate i, we simply compare the expected cost of the two strategies, and choose the strategy with the smaller expected cost; however, since we do not have access to the true marginals $\left(\mu_i^\ell\right)_{i \in [n], \ell \in [k]}$, we use the estimates $\left(a_i^\ell\right)_{i \in [n], \ell \in [k]}$ in their place. Thus, we estimate the cost of the 1-strategy to be $k \min_{\ell \in [k]} a_i^\ell$, and the cost of the 0-strategy to be $\sum_{\ell \in [k]} (1 - a_i^\ell)$ (ignoring the $\log n$ factor), and we choose to follow the 1-strategy for coordinate i iff $k \min_{\ell \in [k]} a_i^\ell < \sum_{\ell \in [k]} (1 - a_i^\ell)$.

We remark that this protocol generalizes a protocol for set disjointness that appeared in [12], but in [12] only the 1-strategy was required, because we could assume that no single coordinate had high probability of being in the intersection – otherwise we could simply guess that the intersection is not empty.

Detailed description of the protocol. The players first partition the coordinates into two sets, I_1 (for the 1-strategy) and I_0 (for the 0-strategy), defined as follows:

$$I_1 := \left\{ i \in [n] \;\middle|\; k \min_{\ell \in [k]} a_i^\ell \leq \sum_{\ell \in [k]} (1 - a_i^\ell) \right\}, \qquad \text{and} \qquad I_0 := [n] \setminus I_1.$$

Note that this is done with no communication, as all players know $(a_i^\ell)_{i \in [n], \ell \in [k]}$.

Next, for any $i \in I_1$, let $owner(i)$ be the player ℓ believed to be most likely to have $i \notin X_i^\ell$ (if there are several such players, we choose the first one):

$$owner(i) := \min \left\{ \ell \in [k] \;\middle|\; a_i^\ell = \min_{m \in [k]} a_i^m \right\}.$$

[2] Technically, players are not allowed to convey information by staying silent. In our implementation below, this is handled by having the players announce all their indices as a set, rather than going over the coordinates one-by-one as we do in our informal overview here. The sets are encoded using a variable-length encoding, and "no bits are sent for coordinate i" technically means that index i does not appear in any player's set.

We partition I_1 into subsets $I_1^1, \ldots, I_1^k$ by owner, with $I_1^\ell := \{i \in I_1 \mid owner(i) = \ell\}$ for each $i \in [k]$. The protocol proceeds as follows.

1. Each player $\ell \in [k]$ announces $\overline{X^\ell} \cap I_0$ and $X^\ell \cap I_1^\ell$.
2. The coordinator can now deduce the intersection in the I_0 coordinates, as it holds that $\cup_{\ell \in [k]} \left(\overline{X^\ell} \cap I_0\right) = I_0 \setminus \cap_{\ell \in [k]} X^\ell$. The coordinator also sets $T := \cup_{\ell \in [k]} \left(X^\ell \cap I_1^\ell\right)$, and announces T to all players.
3. Each player $\ell \in [k]$ sends $X^\ell \cap T$ to the coordinator. The coordinator declares that the intersection in I_1 is $\left(\cap_{\ell \in [k]} X^\ell\right) \cap T$.

Expected communication cost. We prove a tighter bound than the one claimed in Theorem 11, as the tighter bound will be useful to us in Section 4.3:

▷ Claim 12. When executed with an (α, ϵ)-approximation of the marginals of μ, the expected communication cost of Π_{base} is

$$O\left(\left(\frac{1+\alpha}{1-\alpha}\right)\left(k\sum_{i=1}^{n}\min\left\{\mathbb{E}\left[\boldsymbol{S}_i\right]^{1/k}, \left(1-\mathbb{E}\left[\boldsymbol{S}_i\right]\right)^{1/k}\right\} + 2\epsilon k n\right)\log n + k\right). \tag{1}$$

To obtain Theorem 11 from the claim, we apply Hölder's inequality to the inner sum:

$$\sum_{i=1}^{n}\min\left\{\mathbb{E}\left[\boldsymbol{S}_i\right]^{1/k}, \left(1-\mathbb{E}\left[\boldsymbol{S}_i\right]\right)^{1/k}\right\} \le \sum_{i=1}^{n}\mathbb{E}\left[\boldsymbol{S}_i\right]^{1/k} \le n^{1-1/k}\,\mathbb{E}\left[|\boldsymbol{S}|\right]^{1/k}.$$

Plugging this into (1) yields the theorem.

The proof of Claim 12 is given in the full version of this paper, but we give a sketch here. We begin by considering an idealized version of the protocol, where the coordinates are partitioned into subsets J_0, J_1 based on their true marginals (which are not known the players):

$$J_1 := \left\{ i \in [n] \;\middle|\; k\min_{\ell \in [k]}\mathbb{E}\left[\boldsymbol{X}_i^\ell\right] \le \sum_{\ell \in [k]}\left(1-\mathbb{E}\left[\boldsymbol{X}_i^\ell\right]\right)\right\} \qquad \text{and} \qquad J_0 := [n] \setminus J_1.$$

For each $i \in J_1$, the idealized protocol follows the 1-strategy, paying $k\min_{\ell \in [k]}\mathbb{E}\left[\boldsymbol{X}_i^\ell\right]$ in expected communication; for each $i \in J_0$, the idealized protocol follows the 0-strategy, paying $\sum_{\ell \in [k]}\left(1-\mathbb{E}\left[\boldsymbol{X}_i^\ell\right]\right)$ in expected communication.

Due to the way in which J_0, J_1 are defined, we are able to show that the idealized protocol pays an expected cost per coordinate of at most $k\min\left\{\mathbb{E}\left[\boldsymbol{S}_i\right]^{1/k}, \left(1-\mathbb{E}\left[\boldsymbol{S}_i\right]\right)^{1/k}\right\}$:

▶ **Lemma 13.** *Let $i \in [n]$. If $i \in J_0$, then*

$$\sum_{\ell \in [k]}\mathbb{E}\left[1-\boldsymbol{X}_i^\ell\right] \le k\min\left\{E\left[\boldsymbol{S}_i\right]^{1/k}, \left(1-\mathbb{E}\left[\boldsymbol{S}_i\right]\right)^{1/k}\right\},$$

and if $i \in J_1$, then

$$k\min_{\ell \in [k]}\mathbb{E}\left[\boldsymbol{X}_i^\ell\right] \le k\min\left\{E\left[\boldsymbol{S}_i\right]^{1/k}, \left(1-\mathbb{E}\left[\boldsymbol{S}_i\right]\right)^{1/k}\right\}.$$

Proof. Fix a coordinate $i \in [n]$. Observe that since μ is a product distribution,

$$\min_{\ell\in[k]} \mathbb{E}\left[\boldsymbol{X}_i^\ell\right] \le \left(\prod_{\ell\in[k]} \mathbb{E}\left[\boldsymbol{X}_i^\ell\right]\right)^{1/k} = \left(\mathbb{E}\left[\prod_{\ell\in[k]} \boldsymbol{X}_i^\ell\right]\right)^{1/k} = \mathbb{E}\left[\boldsymbol{S}_i\right]^{1/k}.$$

Hence if $\mathbb{E}\left[\boldsymbol{S}_i\right]^{1/k} \le \left(1-\mathbb{E}\left[\boldsymbol{S}_i\right]\right)^{1/k}$ and $i \in J_1$, then we have:

$$\min_{\ell\in[k]} \mathbb{E}\left[\boldsymbol{X}_i^\ell\right] \le \mathbb{E}\left[\boldsymbol{S}_i\right]^{1/k} = \min\left\{E\left[\boldsymbol{S}_i\right]^{1/k}, \left(1-\mathbb{E}\left[\boldsymbol{S}_i\right]\right)^{1/k}\right\}.$$

Similarly, if $\mathbb{E}\left[\boldsymbol{S}_i\right]^{1/k} \le \left(1-\mathbb{E}\left[\boldsymbol{S}_i\right]\right)^{1/k}$ and $i \in J_0$, then we have:

$$\sum_{\ell\in[k]} \mathbb{E}\left[1-\boldsymbol{X}_i^\ell\right] < k \min_{\ell\in[k]} \mathbb{E}\left[\boldsymbol{X}_i^\ell\right] \le k\,\mathbb{E}\left[\boldsymbol{S}_i\right]^{1/k} = k\min\left\{\mathbb{E}\left[\boldsymbol{S}_i\right]^{1/k}, \left(1-\mathbb{E}\left[\boldsymbol{S}_i\right]\right)^{1/k}\right\}.$$

Now, if $\left(1-\mathbb{E}\left[\boldsymbol{S}_i\right]\right)^{1/k} < \mathbb{E}\left[\boldsymbol{S}_i\right]^{1/k}$, i.e., $\mathbb{E}\left[\boldsymbol{S}_i\right] > 1/2$, then observe that this implies that $i \in J_0$, as we have that:

$$1/2 < \mathbb{E}\left[\boldsymbol{S}_i\right] = \mathbb{E}\left[\prod_{\ell\in[k]} \boldsymbol{X}_i^\ell\right] = \prod_{\ell\in[k]} \mathbb{E}\left[\boldsymbol{X}_i^\ell\right] \le \min_{\ell\in[k]} \mathbb{E}\left[\boldsymbol{X}_i^\ell\right],$$

where the last inequality is since all the expectations are upper bounded by 1. This in turn implies that:

$$\sum_{\ell\in[k]} \mathbb{E}\left[1-\boldsymbol{X}_i^\ell\right] \le \frac{k}{2} < k \min_{\ell\in[k]} \mathbb{E}\left[\boldsymbol{X}_i^\ell\right],$$

and hence $i \in J_0$. Now we have that:

$$\begin{aligned}
&\sum_{\ell\in[k]} \mathbb{E}\left[1-\boldsymbol{X}_i^\ell\right] \\
&= k\left(1-(1/k)\sum_{\ell\in[k]} \mathbb{E}\left[\boldsymbol{X}_i^\ell\right]\right) \\
&\le k\left(1-\left(\prod_{\ell\in[k]} \mathbb{E}\left[\boldsymbol{X}_i^\ell\right]\right)^{1/k}\right) && \text{(AM-GM inequality.)} \\
&= k\left(1-\mathbb{E}\left[\boldsymbol{S}_i\right]^{1/k}\right).
\end{aligned}$$

Finally, observe that

$$1-\mathbb{E}\left[\boldsymbol{S}_i\right]^{1/k} \le 1-\mathbb{E}\left[\boldsymbol{S}_i\right] \le \left(1-\mathbb{E}\left[\boldsymbol{S}_i\right]\right)^{1/k} = \min\left\{\mathbb{E}\left[\boldsymbol{S}_i\right]^{1/k}, \left(1-\mathbb{E}\left[\boldsymbol{S}_i\right]\right)^{1/k}\right\}. \quad \blacktriangleleft$$

Since the idealized protocol pays $k\min\left\{\mathbb{E}\left[\boldsymbol{S}_i\right]^{1/k}, \left(1-\mathbb{E}\left[\boldsymbol{S}_i\right]\right)^{1/k}\right\}$ per coordinate i, its total cost matches the bound from Claim 12 with $\alpha = \epsilon = 0$. To prove the claim for the actual protocol, we relate it to the idealized protocol, and show that its communication cost is similar. The key here is to show that even if we "misclassify" a coordinate i by placing it in I_1 when the idealized protocol has it in J_0 or vice-versa, the penalty is not too large: the partition into I_1 vs. I_0 is based on the estimates $\left(a_i^\ell\right)_{i\in[n],\ell\in[k]}$, which are close to the true

marginals $\left(\mu_i^\ell\right)_{i\in[n],\ell\in[k]}$ on which the partition into J_1 vs. J_0 is based. If $i \in I_1$ but $i \in J_0$ or vice-versa, then we must be close to the threshold where one strategy becomes preferable to the other, and therefore it does not matter too much which strategy we choose to pursue. For example, consider the case where $i \in I_1$ but $i \in J_0$. Then our actual protocol communicates $k\,\mathbb{E}\left[\boldsymbol{X}_i^{owner(i)}\right]$ bits in expectation for coordinate i, while an idealized protocol (where the players use J_0, J_1) communicates $\sum_{\ell\in[k]}\mathbb{E}\left[1-\boldsymbol{X}_i^\ell\right]$ bits in expectation. Now, since $i \in I_1$, we have $k\min_{\ell\in[k]} a_i^\ell \le \sum_{\ell\in[k]}(1-a_i^\ell)$. Hence:

$$\begin{aligned}
k\,\mathbb{E}\left[\boldsymbol{X}_i^{owner(i)}\right] &\le k\left((1+\alpha)a_i^{owner(i)}+\epsilon\right) = (1+\alpha)k\min_{\ell\in[k]} a_i^\ell + \epsilon k \\
&\le (1+\alpha)\sum_{\ell\in[k]}(1-a_i^\ell)+\epsilon k \\
&\le (1+\alpha)\sum_{\ell\in[k]}\frac{(1-\mathbb{E}\left[\boldsymbol{X}_i^\ell\right])+\epsilon}{1-\alpha}+\epsilon k \\
&\le \frac{1+\alpha}{1-\alpha}\left(\sum_{\ell\in[k]}\mathbb{E}\left[1-\boldsymbol{X}_i^\ell\right]+2\epsilon k\right).
\end{aligned}$$

Finally, showing a similar bound on the other possible types of misclassification, summing over all coordinates and applying Lemma 13 completes the proof for Claim 12.

4.2 Approximating the Marginals

In this section we show how to compute an $(1/4, n^{1/k})$-approximation of a value $b \in [0,1]$, given access to $\tilde{\Theta}(n^{1/k})$ iid samples from a Bernoulli distribution with probability b of returning 1. We then apply this procedure to obtain the estimates $\left(a_i^\ell\right)_{i\in[n],\ell\in[k]}$ that are used in the protocol of the previous section.

In general, to obtain an (α,ϵ)-approximation of a value $b \in [0,1]$, it suffices to take $\tilde{\Theta}(1/(\alpha^2\epsilon))$ samples from Bernoulli(b). In fact, we can provide a stronger guarantee: with high probability, our estimate $\boldsymbol{a}$ is either

- Purely additive: a $(0,\epsilon)$-approximation of b, that is, $\boldsymbol{a}-\epsilon \le b \le \boldsymbol{a}+\epsilon$; or,
- Purely multiplicative: an $(\alpha,0)$-approximation of b, that is, $(1-\alpha)\boldsymbol{a} \le b \le (1+\alpha)\boldsymbol{a}$ and $(1-\alpha)(1-\boldsymbol{a}) \le 1-b \le (1+\alpha)(1-\boldsymbol{a})$.

However, we do not know in advance (or even in hindsight) whether the estimate that we get will be purely additive or purely multiplicative. We note that if we were to insist on always having a purely additive estimate or on always having a purely multiplicative estimate, then we would require significantly more samples. For example, to obtain an purely additive $\pm\epsilon$ approximation of a value b close to $1/2$, we would require $\Omega(1/\epsilon^2)$ samples rather than $\Omega(1/\epsilon)$, which is important in our case, since ϵ is very small (roughly $n^{-1/k}$); to obtain a purely multiplicative $(1\pm\alpha)$-approximation of a value b close to 0 we require $\Omega(1/b)$ samples (i.e., an unbounded number of samples when b is unknown). Thus, it is important that our protocol can handle the type of estimate that we produce here, which can have both types of approximation error.

Obtaining the estimate is very simple, but the analysis is somewhat delicate:

▶ **Lemma 14.** *For any $\alpha,\epsilon,\delta \in (0,1)$ and $b \in (0,1)$, given access to $O(\frac{1}{\alpha^2\epsilon}\log(1/\delta))$ iid samples of Bernoulli(b), with probability $1-\delta$ we can compute a value a that is either an $(\alpha,0)$-approximation or a $(0,\epsilon)$-approximation to b (or both).*

Proof. Let $m = (100/(\alpha^2\epsilon))\log(4/\delta)$. Given samples $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_m \sim \text{Bernoulli}(b)$, the estimate we output is $\boldsymbol{a} = \sum_{i=1}^{m} \boldsymbol{B}_i/m$. We claim that this estimate (α, ϵ)-approximates b with probability $1-\delta$. We divide into cases based on the values of ϵ and b.

Case 1: $\epsilon > \frac{1}{100}$. In this case we prove that $\boldsymbol{a}$ is a $(0, \epsilon)$-approximation to b with probability $1-\delta$. By the additive Chernoff bound,

$$\Pr\left(\boldsymbol{a} > b + \epsilon\right) \leq e^{-m \cdot D(b+\epsilon \,||\, b)} \leq e^{-m\cdot\epsilon^2} \leq e^{-m\cdot\epsilon/100},$$

where the second step uses Pinsker's inequality, and the last step uses the fact that $\epsilon > 1/100$. Similarly, $\Pr\left(\boldsymbol{a} < b - \epsilon\right) \leq e^{-m\cdot D(b-\epsilon \,||\, b)} \leq e^{-m\cdot\epsilon^2} \leq e^{-m\cdot\epsilon/100}$. Since $m \geq (100/\epsilon)\log(4/\delta)$ we have $e^{-m\cdot\epsilon/100} \leq \delta/4$, so the probability that either $\boldsymbol{a} > b+\epsilon$ or $\boldsymbol{a} < b-\epsilon$ is less than δ. In other words, with probability at least $1-\delta$, we have $\boldsymbol{a} - \epsilon \leq b \leq \boldsymbol{a} + \epsilon$, as required.

Case 2: $\epsilon < b < 1-\epsilon$. In this case we prove that $\boldsymbol{a}$ is an $(\alpha, 0)$-approximation to b with probability $1-\delta$. By the multiplicative Chernoff bound, $\Pr\left(\boldsymbol{a} \notin (1 \pm \alpha/2)b\right) \leq 2e^{-(\alpha/2)^2 bm/3} \leq 2e^{-\alpha^2\epsilon m/12}$. Similarly, $\Pr\left(1-\boldsymbol{a} \notin (1 \pm (\alpha/2))(1-b)\right) \leq 2e^{-(\alpha/2)^2(1-b)m/3} \leq 2e^{-\alpha^2\epsilon m/12}$. Since $m = (100/(\alpha^2\epsilon))\log(4/\delta)$, we have $e^{-\alpha^2\epsilon m/3} \leq \delta/4$, and thus the probability that either $\boldsymbol{a} \notin (1\pm\alpha/2)b$ or $1-\boldsymbol{a} \notin (1\pm\alpha/2)(1-b)$ is at most δ. Note that if $(1-\alpha/2)b \leq \boldsymbol{a} \leq (1+\alpha/2)b$, then we also have $b \leq \boldsymbol{a}/(1-\alpha/2) \leq (1+\alpha)\boldsymbol{a}$ and $b \geq \boldsymbol{a}/(1+\alpha/2) \geq (1-\alpha)\boldsymbol{a}$, as required, and similarly for $1-b$ and $1-\boldsymbol{a}$.

Case 3: $b \leq \epsilon \leq 1/100$ or $1-b \leq \epsilon \leq 1/100$. In this case we prove that $\boldsymbol{a}$ is a $(0, \epsilon)$-approximation to b with probability $1-\delta$. Let us assume that $b \leq \epsilon \leq 1/100$, as the other case is symmetric. First, observe that $\Pr\left(\boldsymbol{a} < b - \epsilon\right) = 0$, as $b - \epsilon < 0$. For the other side, by the additive Chernoff bound, $\Pr\left[\boldsymbol{a} > b + \epsilon\right] \leq e^{-m\cdot D(b+\epsilon \,||\, b)}$. Using Lemma 7 and the fact that $b \leq \epsilon$, we can bound the divergence from below by $D\left(b+\epsilon \,||\, b\right) \geq D\left(2\epsilon \,||\, \epsilon\right)$; and using a technical lemma from [9] and the fact that $\epsilon \leq 1/100$, we have $D\left(2\epsilon \,||\, \epsilon\right) \geq 2\epsilon/10$. All together, we see that $\Pr\left[\boldsymbol{a} > b + \epsilon\right] \leq e^{-2\epsilon m/10} \leq \delta$. ◀

Plugging in $\epsilon = n^{-1/k}$ and $\alpha = 1/4$, we see that $O(n^{1/k})$ samples suffice to estimate a single marginal μ_i^ℓ with sufficient accuracy, and $O(n^{1/k}\log(nk))$ samples suffice to approximate the entire distribution.

4.3 Entropy-Based Protocol ($k \leq \log n$)

In this section we refer to the protocol of Section 4.1 as *the base protocol*, Π_{base}. We show how to build on the base protocol to obtain a better protocol in the case where the intersection has small entropy. For convenience, we describe the new protocol in the shared blackboard model (the protocol can be adapted to the coordinator model with a multiplicative overhead of $O(k) = O(\log n)$ by having the coordinator forward every message to all players).

High-level overview. In this overview we assume for simplicity that $\Pr\left[\boldsymbol{S}_i = 1\right] \leq 1/2$ for each i, which means that $\mathrm{H}(\boldsymbol{S}_i) \geq \mathbb{E}\left[\boldsymbol{S}_i\right]$. This suffices to convey the main ideas; in the actual protocol, we work with $\min\left(\mathbb{E}\left[\boldsymbol{S}_i\right], 1 - \mathbb{E}\left[\boldsymbol{S}_i\right]\right)$ instead of $\mathbb{E}\left[\boldsymbol{S}_i\right]$, and rely on the fact that $\mathrm{H}(p) \geq \min(p, 1-p)$ for every $p \in [0,1]$.

Our protocol is motivated by the observation that the base protocol from Section 4.1 already has the desired communication cost of $\tilde{O}(n^{1-1/k}\mathrm{H}(\boldsymbol{S})^{1/k})$ in the special case where the intersection bits $\boldsymbol{S}_1, \ldots, \boldsymbol{S}_n$ are *independent*: by our assumption that $\mathrm{H}(\boldsymbol{S}_i) \geq \mathbb{E}\left[\boldsymbol{S}_i\right]$ for each i, we can use Claim 12 and Hölder's inequality to see that the base protocol computes

the intersection with expected communication cost $\tilde{O}\left(n^{1-1/k}\left(\sum_{i=1}^{n}\mathrm{H}\left(\boldsymbol{S}_i\right)\right)^{1/k}\right)$. In general, $\sum_{i=1}^{n}\mathrm{H}(\boldsymbol{S}_i)$ can be much greater than $\mathrm{H}(\boldsymbol{S})$ (e.g., if $\boldsymbol{S}_1 = \ldots = \boldsymbol{S}_n$). However, if $\boldsymbol{S}_1, \ldots, \boldsymbol{S}_n$ are independent, then $\sum_{i=1}^{n}\mathrm{H}\left(\boldsymbol{S}_i\right) = \mathrm{H}(\boldsymbol{S})$, and we are done.

What should we do when $\boldsymbol{S}_1, \ldots, \boldsymbol{S}_n$ are not independent? In this case we show that we can *exploit* the correlation between the bits. Given a set of coordinates $I \subseteq [n]$, let us say that the bits of $\boldsymbol{S}_I$ are *nearly-independent* if

$$\sum_{i\in I}\mathrm{H}\left(\boldsymbol{S}_i\right) \leq 2\mathrm{H}\left(\boldsymbol{S}_I\right). \tag{2}$$

Intuitively, (2) requires that the bits of $\boldsymbol{S}_I$ behave "almost as nicely" as independent bits, in that the sum of their marginal entropies is not much greater than their joint entropy (where for truly independent bits these would be equal).

Our protocol finds a maximal subset of coordinates $I \subseteq [n]$ such that the bits $\boldsymbol{S}_I$ are nearly-independent, and uses the base protocol to compute $\boldsymbol{S}_I$. By (2), the communication cost is $\tilde{O}\left(n^{1-1/k}\left(\sum_{i\in I}\mathrm{H}\left(\boldsymbol{S}_i\right)\right)^{1/k}\right) = \tilde{O}\left(n^{1-1/k}\mathrm{H}\left(\boldsymbol{S}_I\right)^{1/k}\right)$. Each remaining coordinate $j \notin I$ is "somewhat dependent" on $\boldsymbol{S}_I$, otherwise we could add j to I and obtain a larger set, contradicting the maximality of I. Intuitively, this means that our uncertainty about $\boldsymbol{S}_j$ should decrease after learning $\boldsymbol{S}_I$, and indeed we can prove that $\mathrm{H}\left(\boldsymbol{S}_j \mid \boldsymbol{S}_I\right) \leq (1/2)\mathrm{H}\left(\boldsymbol{S}_j\right)$ (see Lemma 15 in the next section). We now recurse on the remaining coordinates.

After $O(\log n)$ iterations, for each coordinate j that we have not yet solved, the entropy of $\boldsymbol{S}_j$ is reduced to at most $1/2^{\log n} = 1/n$. We can now afford to simply call the base protocol to solve all the remaining coordinates: if $F \subseteq [n]$ is the set of remaining coordinates, then the cost of solving all of them using the base protocol is roughly $\tilde{O}\left(n^{1-1/k}\left(\sum_{i\in F}\mathrm{H}\left(\boldsymbol{S}_i\right)\right)^{1/k}\right) = \tilde{O}\left(n^{1-1/k}\cdot\left(\sum_{i\in F}(1/n)\right)^{1/k}\right) = \tilde{O}\left(n^{1-1/k}\cdot 1\right)$.

Detailed description of the protocol. Throughout the protocol, the players maintain a subset $J \subseteq [n]$ of coordinates in which the intersection was already computed, and a distribution μ', which is the posterior input distribution given what the protocol has learned so far. All entropies computed during the run of the protocol are with respect to the updated distribution, μ'. The protocol is as follows.

1. Initialize $J \leftarrow \emptyset, \mu' \leftarrow \mu$.
2. Repeat $R = \lceil \log n \rceil$ times, or until $J = [n]$:
   - 2.1. Let $I \subseteq [n] \setminus J$ be a maximal set of nearly-independent coordinates (see (2) above). If there is more than one possible choice for I, we choose the lexicographically-smallest one. This step does not require communication.
   - 2.2. Execute the base protocol Π_{base} on the coordinates of I, using the distribution μ'. Let τ be the transcript of Π_{base}, and let $\mu'|_\tau$ be the distribution μ' conditioned on the event that the transcript of Π_{base} is τ.
   - 2.3. Update $J \leftarrow J \cup I, \mu' \leftarrow \mu'|_\tau$.
3. Finally, if $J \neq [n]$, call the protocol Π_{base} on the remaining coordinates $[n] \setminus J$, using the distribution μ'.

At the end, we output all intersection elements found during any of the calls to Π_{base}.

Expected communication cost. In the analysis we rely on the finer bound given in Claim 12 for the communication cost of Π_{base}. The bound is stated in terms of the expectations $\mathbb{E}\left[\boldsymbol{S}_i\right]$, but since $\mathrm{H}(p) \geq \min\{p, 1-p\}$ for every $p \in [0,1]$, Claim 12 and Hölder's inequality imply that the expected cost of Π_{base} when $\alpha = \epsilon = 0$ is

$$O\left(k\sum_{i=1}^{n}\mathrm{H}(\boldsymbol{S}_i)^{1/k}+k\right)=O\left(kn^{1-1/k}\left(\sum_{i=1}^{n}\mathrm{H}(\boldsymbol{S}_i)\right)^{1/k}\right). \tag{3}$$

Our goal now is essentially to replace the term $\sum_{i=1}^{n}\mathrm{H}(\boldsymbol{S}_i)$ in the bound above by $\mathrm{H}(\boldsymbol{S})$.

The full analysis will be given in the full version of this paper. The main idea is that in every iteration $r \le \lceil \log n \rceil$, if $\boldsymbol{I}_r$ is the set of coordinates on which we call Π_{base} in iteration r, then by choice of $\boldsymbol{I}_r$ we have $\sum_{i\in\boldsymbol{I}_r}\mathrm{H}_{\mu_r}(\boldsymbol{S}_i) \le 2\mathrm{H}_{\mu_r}(\boldsymbol{S}_{\boldsymbol{I}_r})$. Note that the expectation here is taken with respect to the distribution μ_r, which is the input distribution conditioned on the transcript up to iteration r (exclusive). Together with (3), this means that the cost of calling Π_{base} on $\boldsymbol{I}_r$ is $O(kn^{1-1/k}\mathrm{H}_{\mu_r}(\boldsymbol{S}_{\boldsymbol{I}_r})^{1/k})$.

When we reach the last step of the protocol, the set of remaining coordinates may not be nearly-independent. However, we claim that for every coordinate $i \in [n]$, given the transcript of the entire protocol so far, the conditional entropy of $\boldsymbol{S}_i$ is reduced to at most $1/n$. This is because in every iteration, the protocol either determines $\boldsymbol{S}_i$, reducing its entropy to zero, or solves a set of coordinates on which $\boldsymbol{S}_i$ depends strongly, which also reduces its entropy.

▶ **Lemma 15.** *Let $\boldsymbol{\Pi}_{<r}$ denote the transcript of the protocol up to iteration r, exclusive. For every $i \in [n]$ and iteration $r \le R$, $\mathrm{H}_\mu(\boldsymbol{S}_i \mid \boldsymbol{\Pi}_{<r+1}) \le \frac{1}{2}\mathrm{H}_\mu(\boldsymbol{S}_i \mid \boldsymbol{\Pi}_{<r})$.*

Proof. We prove that for every iteration $r \le R$ and transcript $\tau_{<r}$,

$$\mathrm{H}_\mu(\boldsymbol{S}_i \mid \boldsymbol{\Pi}_{<r+1}, \boldsymbol{\Pi}_{<r}=\tau_{<r}) \le \frac{1}{2}\mathrm{H}_\mu(\boldsymbol{S}_i \mid \boldsymbol{\Pi}_{<r}=\tau_{<r}).$$

The lemma then follows by taking the expectation over $\tau_{<r}$.

The transcript $\tau_{<r}$ determines the sets $I_1,\ldots,I_r$ on which Π_{base} is called in every iteration $1,\ldots,r$, as well as $\boldsymbol{S}_{\boldsymbol{I}_1}=S_{I_1},\ldots,\boldsymbol{S}_{\boldsymbol{I}_{r-1}}=S_{I_{r-1}}$. The value of $\boldsymbol{S}_{\boldsymbol{I}_r}$ is not determined by $\tau_{<r}$, but it is determined by $\Pi_{<r+1}$, as it is computed in iteration r itself. Therefore,

$$\mathrm{H}(\boldsymbol{S}_i \mid \boldsymbol{\Pi}_{<r+1}, \tau_{<r}) = \mathrm{H}(\boldsymbol{S}_i \mid \boldsymbol{\Pi}_{<r+1}, \tau_{<r}, \boldsymbol{S}_{I_1},\ldots,\boldsymbol{S}_{I_r}) \le \mathrm{H}(\boldsymbol{S}_i \mid \tau_{<r}, \boldsymbol{S}_{I_1},\ldots,\boldsymbol{S}_{I_r}),$$

where the last step uses the fact that conditioning does not increase entropy.

If there is some iteration $t \le r$ such that $i \in I_t$, then clearly $\mathrm{H}(\boldsymbol{S}_i \mid \tau_{<r}, \boldsymbol{S}_{I_1},\ldots,\boldsymbol{S}_{I_r}) = 0$, and the lemma follows from the non-negativity of entropy. Otherwise, i is not an element of any set $I_1,\ldots,I_r$, and in particular $i \notin I_r$. We claim that $\mathrm{H}(\boldsymbol{S}_i \mid \tau_{<r}, \boldsymbol{S}_{I_r}) \le (1/2)\mathrm{H}(\boldsymbol{S}_i \mid \tau_{<r})$, which proves the claim, as $\mathrm{H}(\boldsymbol{S}_i \mid \tau_{<r}, \boldsymbol{S}_{I_r}) = \mathrm{H}\left(\boldsymbol{S}_i \,\middle|\, \tau_{<r}, \boldsymbol{S}_{I_r}, \boldsymbol{S}_{I_1},\ldots,\boldsymbol{S}_{I_{r-1}}\right)$ and similarly $\mathrm{H}(\boldsymbol{S}_i \mid \tau_{<r}) = \mathrm{H}\left(\boldsymbol{S}_i \,\middle|\, \tau_{<r}, \boldsymbol{S}_{I_1},\ldots,\boldsymbol{S}_{I_{r-1}}\right)$ (as $\boldsymbol{S}_{I_1},\ldots,\boldsymbol{S}_{I_{r-1}}$ are all determined by $\tau_{<r}$).

Suppose for the sake of contradiction that

$$\mathrm{H}(\boldsymbol{S}_i \mid \tau_{<r}, \boldsymbol{S}_{I_r}) > (1/2)\mathrm{H}(\boldsymbol{S}_i \mid \tau_{<r}). \tag{4}$$

Then we can write

$$\begin{aligned}\sum_{j\in I_r\cup\{i\}}\mathrm{H}(\boldsymbol{S}_j \mid \tau_{<r}) &= \mathrm{H}(\boldsymbol{S}_i \mid \tau_{<r}) + \sum_{j\in I_r}\mathrm{H}(\boldsymbol{S}_j \mid \tau_{<r})\\ &\le 2\mathrm{H}(\boldsymbol{S}_i \mid \tau_{<r}, \boldsymbol{S}_{I_r}) + 2\mathrm{H}(\boldsymbol{S}_{I_r} \mid \tau_{<r}) \qquad \text{(by (4) and by choice of } I_r)\\ &= 2\mathrm{H}\left(\boldsymbol{S}_{I_r\cup\{i\}} \,\middle|\, \tau_{<r}\right),\end{aligned}$$

which contradicts the maximality of I_r. ◀

▶ **Corollary 16.** *For every $i \in [n]$ we have $\mathrm{H}_\mu(\boldsymbol{S}_i \mid \boldsymbol{\Pi}_{<R+1}) \le 1/n$.*

By the corollary, we can simply use (3) to bound the cost of calling Π_{base} on all the remaining coordinates by $O\left(kn^{1-1/k} \cdot \left(\sum_{i=1}^{n}(1/n)\right)^{1/k}\right) = O\left(kn^{1-1/k}\right)$. The final step in the proof is to carefully sum the costs of all the iterations, using Hölder's inequality and the chain rule for entropy to obtain the final bound of Theorem 2.

4.4 Upper Bound for Large k

For the case where we have $k \geq \log n$ players, we show that a single sample from the input distribution suffices to later compute the intersection on new inputs with expected communication cost $\tilde{O}(n+k)$. To do so, we modify a protocol from [12].

High-level overview. The protocol from [12] handles coordinates differently based on whether they have a non-negligible probability of being in the intersection or not. Let us say that a coordinate i is *negligible* if $\Pr[\boldsymbol{S}_i = 1] < \delta/n$. For negligible coordinates i, the protocol simply guesses that $\boldsymbol{S}_i = 0$, without trying to actually compute $\boldsymbol{S}_i$. By union bound, this contributes a total of at most $O(\delta)$ to our error probability. For non-negligible coordinates i, it is observed in [12] that since μ is a product distribution, the expected number of players ℓ that have $\boldsymbol{X}_i^\ell = 0$ must be very small; otherwise, $\Pr[\boldsymbol{S}_i = 1] = \prod_{\ell\in[k]} \Pr\left[\boldsymbol{X}_i^\ell = 1\right]$ would be very small, but we assumed that $\Pr[\boldsymbol{S}_i = 1] \geq \delta/n$. This means we can afford to have every player ℓ that has $\boldsymbol{X}_i^\ell = 0$ announce this fact to the coordinator, who then determines that $\boldsymbol{S}_i = 1$ iff no player ℓ announced that $\boldsymbol{X}_i^\ell = 0$.

In our setting we do not know the input distribution exactly, which can lead to two types of mistakes:

- Classifying a coordinate i as negligible when it is in fact non-negligible: we cannot afford to make even one such mistake, because for such coordinates we always output $\boldsymbol{S}_i = 0$, even though there is non-negligible probability that $\boldsymbol{S}_i = 1$. Thus, when we classify a coordinate as negligible, it must truly be negligible under the unknown input distribution.
- Classifying a coordinate i as non-negligible when it is in fact negligible: this type of mistake does not lead to incorrect outputs, but it can increase our expected communication cost, depending on the expected players that have 0 in coordinate i. Unlike the previous case, here we can afford to make some mistakes, but we should avoid classifying a coordinate i as non-negligible if $\sum_{i=1}^{k} \mathbb{E}\left[1 - \boldsymbol{X}_i^\ell\right]$ is large.

We show that when $k \geq \log n$, a single sample from the input distribution suffices to classify coordinates well enough for our purposes. Let $z_i = \sum_{\ell\in[k]}\left(1 - \mathbb{E}\left[\boldsymbol{X}_i^\ell\right]\right)$ be the expected number of zeroes in coordinate $i \in [n]$, under the unknown input distribution. Given one sample $\boldsymbol{A} \sim \mu$, we estimate z_i by $v_i := \sum_{\ell\in[k]}\left(1 - \boldsymbol{A}_i^\ell\right)$. Since $k \geq \log n$, the value $\sum_{\ell\in[k]}\left(1 - \boldsymbol{A}_i^\ell\right)$ is concentrated about its mean, which is z_i. This allows us to simultaneously estimate $z_1, \ldots, z_n$ with enough precision that no non-negligible coordinate is classified as negligible, and at the same time, every coordinate i that is classified as non-negligible has small z_i.

Detailed description of the protocol. As outlined above, we first take a sample $\boldsymbol{A} \sim \mu$, compute the estimates $v_i := \sum_{\ell\in[k]}\left(1 - \boldsymbol{A}_i^\ell\right)$, and then choose the following set of coordinates $N \subseteq [n]$ to classify as non-negligible: $N = \{i \in [n] : v_i \leq \beta \ln(n/\delta)\}$, where $\beta \geq 1$ is a constant whose value will be fixed later.

The protocol then proceeds as follows: given input $\boldsymbol{X} \sim \mu$,

1. For each coordinate $i \notin N$, the coordinator declares that $\boldsymbol{S}_i = 0$, that is, $i \notin \bigcap_{\ell\in[k]} \boldsymbol{X}^\ell$.
2. Simultaneously, each player ℓ sends $\boldsymbol{N}^\ell := N \cap \left([n] \setminus \boldsymbol{X}^\ell\right)$ to the coordinator.
3. The coordinator outputs $N \setminus \bigcup_{\ell\in[k]} \boldsymbol{N}^\ell$ as its estimate for the intersection $\boldsymbol{S}$.

Correctness and expected communication cost. Let $z_i = \sum_{\ell \in [k]} \left(1 - \mathbb{E}\left[\boldsymbol{X}_i^\ell\right]\right)$ be the expected number of zeroes in coordinate $i \in [n]$, and let $\mathcal{E}$ be the event that for every coordinate $i \in [n]$,

- If $z_i < \ln(n/\delta)$ then $i \in N$ (that is, $v_i \leq \beta \ln(n/\delta)$), and
- If $z_i > \beta^2 \ln(n/\delta)$ then $i \notin N$ (that is, $v_i > \beta \ln(n/\delta)$).

Intuitively, $\mathcal{E}$ is the event that we have classified every coordinate "well enough". Using Chernoff, it is easy to see that when β is large enough (say, $\beta \geq 8$), the event $\mathcal{E}$ occurs with probability $\geq 1 - \delta$ over the sample $\boldsymbol{A} \sim \mu$. This implies the correctness of the protocol: whenever $\mathcal{E}$ occurs, every coordinate $i \in [n]$ where $z_i < \beta \ln(n/\delta)$ is identified as non-negligible, $i \in N$. The coordinates in N are correctly solved by the protocol, since every player that has a zero in such a coordinate informs the coordinator. As for coordinates $i \notin N$, these coordinates must have $z_i \geq \beta \ln(n/\delta)$. By Lemma 4 in [12], this implies that $\Pr[\boldsymbol{S}_i = 1] \leq \delta/n$. By union bound, the probability that any such coordinate is in the intersection is at most δ. The error probability of the protocol is therefore bounded by $\Pr\left[\overline{\mathcal{E}}\right] + \Pr[\mathcal{E}] \cdot \delta \leq 2\delta$.

To bound the expected communication, we again condition on the event $\mathcal{E}$, which implies that every coordinate $i \in N$ that we actually solve has $z_i \leq \beta^2 \ln(n/\delta)$. Since $\log n \cdot z_i = \log n \sum_{\ell \in [k]} \left(1 - \mathbb{E}\left[\boldsymbol{X}_i^\ell\right]\right)$ is the expected communication cost of solving i, this means we send an expected $O(\log(n) \log(n/\delta))$ bits per coordinate in N, for a total of $O(n \log(n) \log(n/\delta))$.

5 Lower Bounds

We begin by observing that when we know nothing about the input distribution other than the fact that it is a product distribution, the distributional communication complexity of computing any function f is the same as the worst-case cost. This justifies our need for taking samples from the distribution before constructing the protocols of Sections 4.1, 4.4.

▶ **Observation 17.** *Let f be a function over $\{0,1\}^{n \times k}$, and let C be the worst-case randomized communication complexity[3] of f with error probability $1/3$ (on any input). Let Π be a (possibly randomized) protocol for computing f, such that under* any product distribution *μ over $\{0,1\}^{n \times k}$ we have $\Pr_{\boldsymbol{X} \sim \mu}[\Pi$ correctly computes $f(\boldsymbol{X})] \geq 2/3$. Then there is a product distribution μ such that the expected communication cost of Π under μ is C.*

Proof. For each input $X \in \{0,1\}^{n \times k}$, let μ_X be the product distribution μ_X that assigns to each player ℓ the input X^ℓ (this is a deterministic assignment, but it still qualifies as a product distribution). Because μ_X is a product distribution, and Π can handle any product distribution, when we run Π on inputs drawn from μ_X it has success probability at least $2/3$, which means that it correctly computes $f(X)$ with probability at least $2/3$. Thus, Π is in fact a protocol for computing f with worst-case error probability at most $1/3$ on any input. Therefore there must exist some input X, and hence some product distribution μ_X, on which Π sends C bits in expectation. ◀

Lower bound on the expected communication cost. To prove the lower bound of Theorem 3, we follow the information-theoretic lower bound technique of [12]. We note that the common information-theoretic strategy of using a *direct sum* argument, where we lower-bound

[3] The randomized communication complexity of a function f is the minimum over all protocols that compute f with worst-case error $\leq 1/3$ of the expected number of bits sent in the worst case (i.e., on any input) by the protocol. The error probability and the expectation are taken with respect to the protocol's internal randomness.

the cost of solving each bit of the problem correctly and then sum over the costs, cannot be used in this context: it yields lower bounds on the cost of solving each coordinate with small marginal error, but as we explained in Section 1, computing each bit $\boldsymbol{S}_i$ of the intersection with small marginal error is easy when the marginal probabilities that $\boldsymbol{S}_i = 1$ tend to be small:

▶ **Proposition 18.** *Let $\alpha \in (0,1)$, $\epsilon \in (0,1]$ be constants, $n > (1/\epsilon)^{2/\alpha}$, $k \geq 2$, and let μ be a product distribution over $(\{0,1\}^n)^k$, with expected intersection size $s \leq n^{1-\alpha}$. Then there exists a deterministic protocol that reveals the intersection to an external observer, with per-coordinate error at most ϵ and expected communication cost at most $\tilde{O}\left(n^{(1-\alpha/2)(1-1/k)}(s+1)^{1/k}\right)$.*

Proof. First, observe that the average coordinate $i \in [n]$ has expected intersection size $s/n = n^{-\alpha}$. Denote by I the set of coordinates $i \in [n]$ that have expected intersection size $\mathbb{E}[\boldsymbol{S}_i] > n^{-\alpha/2}$. Note that by Markov's inequality $|I| \leq n^{-\alpha/2} \cdot n = n^{1-\alpha/2}$. Now, if $k \leq \log n$, then the players execute the basic protocol of Theorem 11 on the coordinates of I. Note that the protocol reveals the intersection in coordinates I to an external observer with zero error. Since the overall expected intersection size for the coordinates is at most s, the protocol has expected communication cost $\tilde{O}\left(n^{(1-\alpha/2)(1-1/k)}(s+1)^{1/k}\right)$.

Similarly, if $k > \log n$, the players execute our protocol for $k > \log n$ (described at 4.4) on the coordinates of I. Note that the protocol reveals the intersection to an external observer with per-coordinate error at most $\epsilon/n^{1-\alpha/2} < \epsilon$ and expected communication cost at most $\tilde{O}\left(n^{(1-\alpha/2)(1-1/k)} + k\right)$.

For any remaining coordinate $i \notin I$, the external observer simply declares that $i \notin \cap_{\ell \in [k]} X^\ell$, and has a per-coordinate error at most $n^{-\alpha/2} < \epsilon$. ◀

Fix an expected intersection size s. Our lower bound uses the distribution where each bit of the input has iid probability $(s/n)^{1/k}$ of being 1 (that is, the distribution is also a product distribution over the elements, not just the players). It is not hard to see that this yields the desired expected intersection size of s, and also that the entropy of the intersection is $\tilde{\Theta}(s)$.

Now suppose we are given a protocol Π that sends $o\left(n^{1-1/k}s^{1/k}\right)$ bits in expectation. Following [12], we first show that for the average coordinate $i \in [n]$ and transcript τ of the protocol, for each player ℓ, the distribution of $\boldsymbol{X}_i^\ell$ conditioned on $\boldsymbol{\Pi} = \tau$ is very close to its prior: intuitively, to rule out an event with prior probability p, the protocol must spend $\Omega(p)$ of its information budget; in our case the event is $\boldsymbol{X}_i^\ell = 1$, and $p = (s/n)^{1/k}$. The protocol expends $o\left(n^{1-1/k}s^{1/k}/n\right) = o\left((s/n)^{1/k}\right)$ of its total information budget on the average coordinate, so the event $\boldsymbol{X}_i^\ell = 1$ remains roughly as likely as it was originally.

Consider a specific coordinate $i \in [n]$, and assume that for all the coordinates $j < i$, the bits $\boldsymbol{S}_i$ have been computed correctly; we denote this event by $\mathcal{E}_{<i}$. Since the protocol computes the entire intersection correctly w.h.p., the event $\mathcal{E}_{<i}$ has high probability. Assume w.l.o.g. that player 1 is the one that outputs $\boldsymbol{S}_i$ given the transcript τ: given on the transcript τ, the event $\mathcal{E}$, and the event $\boldsymbol{X}_i^1 = 1$, player 1 must decide whether to output $\boldsymbol{S}_i = 0$ or $\boldsymbol{S}_i = 1$ (if $\boldsymbol{X}_i^1 = 0$ then player 1 knows that $\boldsymbol{S}_i = 0$ and does not need to work to learn the answer). However, we can show that even conditioned on $\tau, \mathcal{E}$, and $\boldsymbol{X}_i^1 = 1$, the distribution of $\boldsymbol{S}_i$ is still very close to its prior, and therefore player 1 has roughly the same uncertainty about whether or not $\boldsymbol{S}_i = 1$ as it had originally, $\mathrm{H}(\boldsymbol{S}_i) = \tilde{\Theta}(s/n)$.

After analyzing the uncertainty about the output in each coordinate $i \in [n]$ conditioned on $\tau, \mathcal{E}$ and the event that the player ℓ deciding this coordinate has $\boldsymbol{X}_i^\ell = 1$, we carefully "collect" these uncertainties and add them up, to show that the players jointly have too much uncertainty about the entire intersection and cannot output it correctly with sufficiently high

probability. We note that unlike [12], in this process we need to handle conditioning on some fairly high-probability events (e.g., the event that $\boldsymbol{X}_i^\ell = 1$ has probability $(s/n)^{1/k}$, which is constant when $s = \Omega(n)$). This has the potential of distorting the distributions we work with by a lot if not handled properly.

References

1 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.

2 László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 337–347, 1986.

3 Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. In *43rd Symposium on Foundations of Computer Science (FOCS 2002)*, pages 209–218, 2002.

4 Anup Bhattacharya, Sourav Chakraborty, Arijit Ghosh, Gopinath Mishra, and Manaswi Paraashar. Disjointness through the lens of vapnik-chervonenkis dimension: Sparsity and beyond. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020*, volume 176 of *LIPIcs*, pages 23:1–23:15, 2020.

5 Ralph Bottesch, Dmitry Gavinsky, and Hartmut Klauck. Correlation in hard distributions in communication complexity. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, volume 40 of *LIPIcs*, pages 544–572, 2015.

6 Mark Braverman, Faith Ellen, Rotem Oshman, Toniann Pitassi, and Vinod Vaikuntanathan. A tight bound for set disjointness in the message-passing model. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*, pages 668–677, 2013.

7 Mark Braverman, Ankit Garg, Denis Pankratov, and Omri Weinstein. From information to exact communication. In *STOC*, pages 151–160. ACM, 2013.

8 Mark Braverman and Rotem Oshman. On information complexity in the broadcast model. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015*, pages 355–364, 2015.

9 Mark Braverman and Rotem Oshman. A rounds vs. communication tradeoff for multi-party set disjointness. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 144–155, 2017.

10 Joshua Brody, Amit Chakrabarti, Ranganath Kondapally, David P. Woodruff, and Grigory Yaroslavtsev. Beyond set disjointness: The communication complexity of finding the intersection. In *Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing*, PODC '14, pages 106–113, 2014.

11 Arkadev Chattopadhyay and Toniann Pitassi. The story of set disjointness. *ACM SIGACT News*, 41(3):59–85, 2010.

12 Nachum Dershowitz, Rotem Oshman, and Tal Roth. The communication complexity of multiparty set disjointness under product distributions. In *STOC*, pages 1194–1207. ACM, 2021.

13 Dmitry Gavinsky. The communication complexity of the inevitable intersection problem. *Chic. J. Theor. Comput. Sci.*, 2020, 2020.

14 Badih Ghazi, Ben Kreuter, Ravi Kumar, Pasin Manurangsi, Jiayu Peng, Evgeny Skvortsov, Yao Wang, and Craig Wright. Multiparty reach and frequency histogram: Private, secure, and practical. *Proc. Priv. Enhancing Technol.*, 2022(1):373–395, 2022. doi:10.2478/popets-2022-0019.

15 André Gronemeier. Asymptotically optimal lower bounds on the nih-multi-party information complexity of the and-function and disjointness. In *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009*, volume 3 of *LIPIcs*, pages 505–516, 2009.

16 Dirk Van Gucht, Ryan Williams, David P. Woodruff, and Qin Zhang. The communication complexity of distributed set-joins with applications to matrix multiplication. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS*, pages 199–212, 2015. doi:10.1145/2745754.2745779.

17 Dawei Huang, Seth Pettie, Yixiang Zhang, and Zhijun Zhang. The communication complexity of set intersection and multiple equality testing. *SIAM J. Comput.*, 50(2):674–717, 2021.

18 Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, Mariana Raykova, David Shanahan, and Moti Yung. On deploying secure computing: Private intersection-sum-with-cardinality. In *IEEE European Symposium on Security and Privacy, EuroS&P*, pages 370–389. IEEE, 2020. doi:10.1109/EuroSP48549.2020.00031.

19 Ivo Kubjas and Vitaly Skachek. Two-party function computation on the reconciled data. In *55th Annual Allerton Conference on Communication, Control, and Computing*, pages 390–396. IEEE, 2017. doi:10.1109/ALLERTON.2017.8262764.

20 Nan Ma and Prakash Ishwar. Two-terminal distributed source coding with alternating messages for function computation. In *2008 IEEE International Symposium on Information Theory*, pages 51–55. IEEE, 2008.

21 Nan Ma and Prakash Ishwar. Infinite-message distributed source coding for two-terminal interactive computing. In *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1510–1517. IEEE, 2009.

22 Nan Ma and Prakash Ishwar. Some results on distributed source coding for interactive function computation. *IEEE Transactions on Information Theory*, 57(9):6180–6195, 2011.

23 Jeff M. Phillips, Elad Verbin, and Qin Zhang. Lower bounds for number-in-hand multiparty communication complexity, made easy. *SIAM J. Comput.*, 45(1):174–196, 2016.

24 Alexander A Razborov. On the distributional complexity of disjointness. In *International Colloquium on Automata, Languages, and Programming*, pages 249–253, 1990.

25 Mert Saglam and Gábor Tardos. On the communication complexity of sparse set disjointness and exists-equal problems. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*, pages 678–687, 2013.

26 Georg Schnitger and Bala Kalyanasundaram. The probabilistic communication complexity of set intersection. In *Proceedings of the Second Annual Conference on Structure in Complexity Theory 1987*, 1987.

27 Alexander A Sherstov. Communication complexity theory: Thirty-five years of set disjointness. In *International Symposium on Mathematical Foundations of Computer Science*, pages 24–43, 2014.

28 Gregory Valiant and Paul Valiant. A CLT and tight lower bounds for estimating entropy. *Electron. Colloquium Comput. Complex.*, TR10-183, 2010. URL: https://eccc.weizmann.ac.il/report/2010/183.

29 Thomas Watson. Communication complexity with small advantage. *Comput. Complex.*, 29(1):2, 2020.

30 David P. Woodruff and Qin Zhang. When distributed computation is communication expensive. In *Distributed Computing: 27th International Symposium, DISC 2013*, pages 16–30, 2013.

An Optimal Separation Between Two Property Testing Models for Bounded Degree Directed Graphs

Pan Peng
School of Computer Science and Technology, University of Science and Technology of China, Hefei, China

Yuyang Wang
School of Computer Science and Technology, University of Science and Technology of China, Hefei, China

Abstract

We revisit the relation between two fundamental property testing models for bounded-degree *directed* graphs: the *bidirectional* model in which the algorithms are allowed to query both the outgoing edges and incoming edges of a vertex, and the *unidirectional* model in which only queries to the outgoing edges are allowed. Czumaj, Peng and Sohler [STOC 2016] showed that for directed graphs with both maximum indegree and maximum outdegree upper bounded by d, any property that can be tested with query complexity $O_{\varepsilon,d}(1)$ in the bidirectional model can be tested with $n^{1-\Omega_{\varepsilon,d}(1)}$ queries in the unidirectional model. In particular, if the proximity parameter ε approaches 0, then the query complexity of the transformed tester in the unidirectional model approaches n. It was left open if this transformation can be further improved or there exists any property that exhibits such an extreme separation.

We prove that testing *subgraph-freeness* in which the subgraph contains k source components, requires $\Omega(n^{1-\frac{1}{k}})$ queries in the unidirectional model. This directly gives the first explicit properties that exhibit an $O_{\varepsilon,d}(1)$ vs $\Omega(n^{1-f(\varepsilon,d)})$ separation of the query complexities between the bidirectional model and unidirectional model, where $f(\varepsilon,d)$ is a function that approaches 0 as ε approaches 0. Furthermore, our lower bound also resolves a conjecture by Hellweg and Sohler [ESA 2012] on the query complexity of testing k-star-freeness.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Graph property testing, Directed graphs, Lower bound, Subgraph-freeness

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.96

Category Track A: Algorithms, Complexity and Games

Funding This work has been supported in part by NSFC grant 62272431 and "the Fundamental Research Funds for the Central Universities".

1 Introduction

Graph property testing is a framework for studying extremely fast (randomized) algorithms for solving a relaxation of classical decision problems on graphs. Given a graph property P, we are interested in designing an algorithm, called a *property tester*, that with high constant probability, accepts any graph G that satisfies P, and rejects any graph that is "far" from satisfying P, i.e., one needs to modify a significant fraction of the representation (e.g., adjacency matrix or adjacency list) of the graph to make it satisfy P. It is assumed that the algorithm is given oracle access to the representation of the graph and the goal of a property tester is to solve the above problem by making as few queries to the oracle as possible. Since the seminal works by Rubinfeld and Sudan [20] (on algebraic property testing)

and Goldreich, Goldwasser and Ron [11] (on combinatorial and graph property testing), a lot of efforts have been made on studying which properties can be tested within a sublinear (e.g., constant) number of queries in several classical models, e.g., the dense graph model [11, 1] and bounded-degree graph model [12]. In particular, we have see a rapid development of property testing on *undirected* graphs in the past two decades. We refer to the recent book [10] for a survey.

On the other hand, we still do not know much about property testing in *directed graphs (digraphs)* so far. Bender and Ron [3] introduced two fundamental models for studying directed graph property testing. The first is called *bidirectional* model, where the algorithm is allowed to query both outgoing and incoming edges of a vertex; the second is called *unidirectional* model, where the algorithm is only allowed to query the outgoing edges, while not incoming edges. The latter model seems more realistic for some applications. For example, consider the webgraphs. It is much easier to query the outgoing edges (u, v) (which corresponds to a hyperlink from webpage u to webpage v) than querying the incoming edges. In this paper, we focus on bounded-degree directed graphs. A digraph G is said to be *d-bounded*, if both the maximum outdegree and maximum indegree of G are upper bounded by d, which is assumed to be a constant.

Bender and Ron gave an algorithm for testing strong connectivity with $\tilde{O}(1/\varepsilon)$ queries in the bidirectional model, and showed that there is a lower bound of $\Omega(\sqrt{n})$ queries for any algorithm with two-sided error[1] in the unidirectional model. Goldreich [9], and Hellweg and Sohler [16] gave a lower bound of $\Omega(n)$ queries for testing strong connectivity with one-sided error in the unidirectional model. The works [9, 16] also gave testers for strong connectivity with $n^{1-1/(d+1/\varepsilon)}$ queries with two-sided error in the unidirectional model. In [16], the authors gave testers for subgraph-freeness with $O(n^{1-\frac{1}{k}})$ queries in the unidirectional model, where k is the number of connected components in the subgraph that have no incoming edges. It is known that a few properties can be tested with a constant number of queries in the bidirectional model, including Eulerianity [18], k-edge connectivity [18, 23, 8], k-vertex connectivity [18, 8].

Towards a deeper understanding of testing properties of bounded degree directed graphs (digraphs), Czumaj, Peng and Sohler [5] studied the relation between these two models and provided a generic transformation that converts testers with constant query complexity in the bidirectional model, to testers with sublinear query complexity in the unidirectional model. Specifically, in [5], it was shown that any property P that can be tested with[2] $q = O_{\varepsilon,d}(1)$ queries in the bidirectional model can be tested with $n^{1-d^{-\Theta(q)}} = n^{1-\Omega_{\varepsilon,d}(1)}$ queries in the unidirectional model (with two-sided error). In particular, if the proximity parameter ε approaches 0, then the query complexity of the transformed tester in the unidirectional model approaches n (as the term $d^{-\Theta(q)}$ approaches 0).

One natural question that is left open is that *is the above the transformation tight*? Or equivalently, *can we achieve a much better query complexity, say $n^{c-\Omega_{\varepsilon,d}(1)}$, in the latter model, for some universal constant $c < 1$*? Indeed, currently, the best known lower bound for this transformation is for testing 3-*star-freeness*, where a 3-star is a 4-vertex directed graph such that there exists one center vertex v, and for any other three vertices u, there is an edge from u to v, and no other edges exist. Hellweg and Sohler [16] have shown that 3-star-freeness can be tested with a constant number of queries in bidirectional model, while

[1] A tester for a property P is said to have *one-sided error* if it accepts every (di)graph satisfying P, and it errs if the graph is far from having P. It is said to have *two-sided error* if it errs in both cases.

[2] Throughout the paper, we use the notation $O_{\varepsilon,d}()$ (resp. $\Omega_{\varepsilon,d}()$) to describe a function in the Big-O (resp. Big-Omega) notation assuming that ε and d are constant.

the query complexity of a tester for this property in the unidirectional model is $\Theta(n^{2/3})$ for any constant $\varepsilon > 0$. Therefore, there is still a significant gap between the upper bound (i.e., $n^{1-\Omega_{\varepsilon,d}(1)}$) in the bidirectional model in the transformation and the current best-known lower bound (i.e., $\Omega(n^{2/3})$).

Before we state our result, we formally introduce the definition of property testing in both directional and unidirectional models. Let $P = (P_n)_{n\in\mathbb{N}}$ be a d-bounded digraph property, where P_n is a property of d-bounded digraphs with n vertices. An n-vertex graph G is said to be ε-far from satisfying P_n if one needs to modify more than εdn edges to make it a d-bounded digraph with property P_n, where $\varepsilon > 0$ is called the proximity parameter. We say that P is *q-query testable* (or that P *can be tested with query complexity* q) if for every n, ε and d, there exists a tester that makes $q = q(n, \varepsilon, d)$ queries and with probability at least $\frac{2}{3}$, accepts any n-vertex d-bounded digraph G satisfying P, and rejects any n-vertex d-bounded digraph G that is ε-far from satisfying P. We call such a tester an *ε-tester* for P.

We show that there exists a property that exhibits an $O_{\varepsilon,d}(1)$ vs $\Omega(n^{1-\Theta_{\varepsilon,d}(1)})$ separation of the query complexities between the bidirectional model and unidirectional model, which implies that the transformation of [5] is essentially tight.

▶ **Theorem 1.** *For any sufficiently small constant $\varepsilon > 0$, there exists a digraph property $P = P_{\varepsilon,d}$ such that P can be tested with $O_{\varepsilon,d}(1)$ queries in the bidirectional model, while any ε-tester for P in the unidirectional model requires $n^{1-f(\varepsilon,d)}$ queries, where $f(\varepsilon, d)$ is a function that approaches 0 when ε approaches 0.*

The above theorem is a direct corollary from the following result regarding testing subgraph-freeness. Let H be a directed graph. A strongly connected component[3] W is called a *source component* of H, if there is no edge from $V(H) \setminus W$ to W. A directed graph H is said to be *weakly connected* if its underlying undirected graph (i.e., the graph that is obtained by ignoring all the directions of the edges) is connected. For example, we note that k-star is just a weakly connected digraph with k source components, where a directed graph H with $k + 1$ vertices is called a *k-star* if there is a vertex v such that each of the other k vertices has exactly one edge pointing to v, and H does not contain any other edges. Let G and H be two directed graphs. The graph G is said to be *H-free* if H does not appear as a subgraph in G. We have the following theorem on testing H-freeness for any (constant-size) H with k source components.

▶ **Theorem 2.** *Let k be any integer such that $k \geq 2$. Let d be some constant. Let H be a weakly connected*[4] *directed graph with k source components. There exists an $\varepsilon_0 = \Theta_{d,k}(1)$ such that any ε_0-tester for testing H-freeness of an n-vertex d-bounded graph requires at least $\Omega(n^{1-\frac{1}{k}})$ queries in the unidirectional model.*

We remark that it has been shown by Hellweg and Sohler [16] that for any H with k source components, H-freeness can be tested with query complexity $O_{\varepsilon,d,k}(1)$ in the bidirectional model, and also can be tested with query complexity $O_{\varepsilon,d,k}(n^{1-\frac{1}{k}})$ in the unidirectional model[5]. Given the above result, we can easily prove Theorem 1.

[3] We call $W \subseteq V(H)$ a *strongly connected component* of H if the subgraph $H[W]$ of H induced by W is strongly connected, and there does not exist any set of vertices $X \subseteq V(H) \setminus W$ such that the subgraph of H induced by $W \cup X$ is strongly connected. That is, the subgraph of $H[W]$ is a strongly connected and maximal.

[4] For graphs H that is not weakly connected, we can handle each of its weakly connected components separately.

[5] On the high level, their algorithms use the following observation: if a bounded-degree directed graph G

Proof of Theorem 1. Let $\eta > 0$ and define property P_η to be the property of being H-free, for any H that is weakly connected digraph with $k = \lceil 1/\eta \rceil$ source components. According to Theorem 2, any ε_0-tester for P_η requires at least $\Omega(n^{1-\frac{1}{k}}) > \Omega(n^{1-\eta})$ queries in the unidirectional model, where $\varepsilon_0 = \varepsilon_0(d, \eta)$ is a function of d, η. Now given any sufficiently small constant $\varepsilon > 0$, let η' be a number satisfying that $\varepsilon = \varepsilon_0(d, \eta')$. Then Theorem 1 follows by taking $P = P_{\eta'}$ and $f(\varepsilon, d) = \eta'$. ◀

Furthermore, it was conjectured in [16] that testing k-star-freeness requires $\Omega(n^{1-\frac{1}{k}})$ queries in the unidirectional model. Since k-star is a directed subgraph with k-source components, our Theorem 2 resolves this conjecture.

1.1 Discussions of previous ideas and our techniques

We first sketch the main ideas of the lower bound for testing 3-star-freeness given by Hellweg and Sohler [16]. Their proof makes use of a problem called *testing* 3-*occurrence-freeness*[6] of a sequence[7]. Let A be a length-n sequence of integers such that each element in A is from $[\ell] := \{1, \cdots, \ell\}$ and occurs at most 3 times. We say A is 3-*occurrence-free* if no integer in A occurs exactly 3 times in A. We say A is ε-*far from being* 3-*occurrence-free* if one needs to change[8] more than εn elements of A to obtain a 3-occurrence-free sequence. [16] gave a local reduction from the problem of testing 3-occurrence-freeness of a sequence to the problem of testing 3-star-freeness. That is, given an instance A with m elements of 3-occurrence-freeness, they constructed a graph G with $\Theta(m)$ vertices, such that

1) if A is 3-occurrence-free, then G is 3-star-free; if A is ε-far from being 3-occurrence-free then G is $\Theta(\varepsilon)$-far from being 3-star-free;

2) every query to G can be answered by performing $O(1)$ queries to A.

To obtain a lower bound for testing 3-occurrence-freeness, [16] constructed two classes $\mathcal{C}_A$, $\mathcal{C}_B$ of length-n sequences such that $\mathcal{C}_A$ is a class of 3-occurrence-free sequences and $\mathcal{C}_B$ is a class of sequences that are $\Omega(1)$-far from being 3-occurrence-free, and the *frequency variables*, denoted by X_A and X_B, of the sequences from these two different classes have 2 proportional moments, i.e.,

$$\frac{\mathbf{E}[X_B]}{\mathbf{E}[X_A]} = \frac{\mathbf{E}[X_B^2]}{\mathbf{E}[X_A^2]}.$$

Then the lower bound $\Omega(n^{2/3})$ for testing 3-occurrence-freeness follows from a lower bound for distinguishing random variables with 2-proportional moments given in [19].

Now we note that to obtain a lower bound for testing H-freeness for any H with k source components, it suffices to give a lower bound for testing k-occurrence-freeness for general k in the way similar as above. That is, we construct two classes $\mathcal{C}_A$, $\mathcal{C}_B$ of length-n sequences such that $\mathcal{C}_A$ is a class of k-occurrence-free sequences and $\mathcal{C}_B$ is a class of sequences that are $\Omega_k(1)$-far from being k-occurrence-free, and the *frequency variables*, denoted by X_A and X_B, of the sequences from these two different classes have $k-1$ proportional moments, i.e.,

$$\frac{\mathbf{E}[X_B]}{\mathbf{E}[X_A]} = \frac{\mathbf{E}[X_B^2]}{\mathbf{E}[X_A^2]} = \cdots = \frac{\mathbf{E}[X_B^{k-1}]}{\mathbf{E}[X_A^{k-1}]}.$$

---

is ε-far from H-freeness, then G contains $\Omega(\varepsilon n)$ vertex-disjoint copies of H. Then in the bidirectional model, one can sample a constant number of vertices and perform BFS from each sampled vertex to find a copy of H; in the unidirectional model, one can sample many edges to see if some copy of H is formed.

[6] In [16], the same problem was called 3-value freeness.

[7] We use "sequence" rather than "multiset" as the position of each element affects our construction.

[8] It is allowed to use integers that are larger than ℓ to change the elements of A.

However, the main difficulty is to construct two classes of sequences satisfying the above equations for general $k \geq 3$, which was also pointed out in [16]. Besides the aforementioned construction in [16] which only works for $k = 3$, we also note that in [19], a special pair of random variables with $k-1$ proportional moments is also constructed (for establishing their lower bound for Distinct-Elements). That is, their random variables take values of the form $(B+3)^i$, for any integers $B > 1, k > 1$ and $i = 0, \ldots, k-1$. This leads to a large gap between the expectations of the corresponding variables. To show a lower bound for testing k-occurrence-freeness, we need to construct random variables taking values $1, 2, \ldots, k$, for any integer $k > 1$. This is more challenging as it corresponds to a much smaller gap (which is arbitrarily close to 1) between the expectations of the corresponding variables (see Lemma 10). To construct such two random variables, we establish some identities related to binomial coefficients, and use them to define two distributions satisfying a number of linear equations which in turn are necessary conditions for two variables having proportional moments.

We then give a local reduction from testing k-occurrence-freeness to testing H-freeness for H with k-source components. The reduction also non-trivially generalizes the one for 3-star-free in [16], as 3-star is a special subgraph with a nice symmetric property, while an arbitrary subgraph H might contain different types of asymmetric structures. Our main idea is as follows. Given a sequence S, we construct a graph G on the fly such that each element in the sequence corresponds to a source component of H in G; an element in S appears k times if and only if a copy of H is added in G. For the latter, we carefully add k source components of H to G and add edges from these components to one center component (which is the rest part of H after removing all the source components). Finally, we show that this construction preserves the distance to the properties and each query to G can be answered by querying at most 1 position in S.

1.2 Other Related work

Ito, Khoury and Newman [17] recently gave a characterization of monotone and hereditary properties that can be tested with constant query complexity and one-sided error in both bounded-degree bidirectional model and bounded-degree unidirectional model. For testing acyclicity in the bidirectional model, Bender and Ron [3] gave a lower bound of $\Omega(n^{1/3})$ queries for algorithms with two-sided error and a lower bound $\Omega(n^{1/2})$ queries for algorithms with one-sided error. The latter lower bound has been improved to $\tilde{\Omega}(n^{5/9})$ queries by Chen, Randolph, Servedio and Sun [4].

In the dense directed graph model (with different types of queries and notion of "ε-far"), Alon and Shapira [2] gave an algorithm with constant query complexity for testing subgraph-freeness.

There exists a class of properties which can be tested with constant number of queries by the so-called *proximity-oblivious testers* [13]. Goldreich and Ron [14] showed that any property that can be tested by a proximity-oblivious tester that makes q uniformly distributed queries with constant detection probability can be tested by a sample-based testers of sample complexity $O(n^{1-1/q})$, where a sample-based tester only samples elements independently from some distribution of the tested object. Building upon [7, 15], Dall'Agnol, Tom and Lachish [6] recently showed that any property that is testable with q queries admits a sample-based tester with sample complexity $n^{1-1/O(q^2 \log^2 q)}$. Their algorithms are defined over a constant-size output alphabet, which is very different from the bounded degree (directed) graph model, in which a super constant alphabet is needed.

Valiant developed a wishful thinking theorem in [22], telling that two distributions whose so-called *k-based moments* have small gap are indistinguishable by k-Poissonized samples. This is a tool for establishing lower bounds of testing symmetric properties on distributions.

On a very high level, both [22] and our work are constructing far distributions with the same collision, while the details for the constructions differ significantly. For example, our proof is built upon Corollary 5.7 of [19], which requires to carefully construct two distributions that have proportional moments. In [22], it is required to construct two distributions whose k-based moments have small gap. It is unclear if two distributions with small gap between k-based moments have proportional moments, or vice versa. In addition, we are using very different properties of Vandermonde matrix from those used in [22].

2 A Lower Bound for Testing k-Occurrence-freeness

In this section, we will prove the lower bound on the query complexity for testing k-occurrence-freeness, which is defined as follows. Given a sequence A of n integers such that each entry of A is from $[n] := \{1, \ldots, n\}$ and each element $i \in [n]$ occurs at most k times, the problem is to distinguish if A is *k-occurrence-free*, i.e., no element occurs in k positions of A, or A is *ε-far from k-occurrence-free*, i.e., more than εn elements of A needs to be changed to make it k-occurrence-free. We assume that the algorithm can query the element (or the value) of any position of the sequence in constant time. The goal is to solve the problem by making as few queries as possible. We will show the following result.

▶ **Theorem 3.** *Any algorithm for testing k-occurrence-freeness with parameter $\varepsilon = \Omega_k(1)$ requires at least $\Omega(n^{1-1/k})$ queries, where n is the length of the input sequence.*

2.1 Basic tools and notions

To prove the above theorem, we will make use of a lower bound by Raskhodnikova et al. [19] for distinguishing two sequences satisfying some property. We first introduce two definitions.

▶ **Definition 4** (Frequency variable)**.** *Let A be a sequence of integers. We define its* frequency variable X_A *as follows. Choose a number uniformly at random from the set of distinct elements that occur in A and then let X_A denote its* frequency[9]*, i.e., the number of times it occurs.*

Take the following sequence $S = \{1, 2, 1, 3, 2, 1, 4\}$ as an example. There are 4 distinct elements (or values) in S: value 1 occurs 3 times, value 2 occurs twice, value 3 and 4 each occurs once. Thus the frequency variable X_S of S satisfies that $\Pr[X_S = 1] = 0.5$, $\Pr[X_S = 2] = 0.25$, $\Pr[X_S = 3] = 0.25$.

▶ **Definition 5** (Proportional moments)**.** *Two random variables X_1 and X_2 are said to have $k-1$* proportional moments*, if $\frac{\mathbf{E}[X_2]}{\mathbf{E}[X_1]} = \frac{\mathbf{E}[X_2^2]}{\mathbf{E}[X_1^2]} = \cdots = \frac{\mathbf{E}[X_2^{k-1}]}{\mathbf{E}[X_1^{k-1}]}$. We say that two sequences have $k-1$ proportional moments if their frequency variables have $k-1$ proportional moments.*

Let P denote a property defined on sequence of integers such that it is invariant under any permutation of indices and values. [19] has shown that any tester for P that makes t queries can be simulated by a *Poisson-s* algorithm that only looks at the histogram of the samples as its input, and $s = O(t)$. Relevant definitions are as follows.

▶ **Definition 6** (*Poisson-s* algorithm)**.** *An algorithm is called a* Poisson-s *algorithm if the number of samples of the algorithm is determined by a Poisson distribution with the expectation s.*

[9] We directly adopt the notion "frequency" from [19].

▶ **Definition 7** (Histogram). *Given a sequence S, the histogram H of S is a function defined as follows:*

$$H(i) := |\{s \in S | s \text{ occurs exactly } i \text{ times in } S\}|$$

In [19], Raskhodnikova et al. proved that if two sequences have $k-1$ proportional moments and $s = o(n^{1-\frac{1}{k}})$, then any *Poisson-s* algorithm can't distinguish their histograms.Formally, based on Lemma 5.3 and Corollary 5.7 in [19], we have the following Lemma.

▶ **Lemma 8** ([19]). *Let X_A and X_B be two random variables with $k-1$ proportional moments. And let D_{X_A} and D_{X_B} be two length-n sequences of integers, whose frequency variables are X_A and X_B, respectively. Let P be a property of sequences that is invariant under permutations of indices and values, and let $\varepsilon > 0$ be a constant.*

1. *If $\mathcal{A}'$ is a tester for P with t queries, i.e., $\mathcal{A}'$ accepts the input sequence that satisfies P with probability at least $\frac{2}{3}$; it rejects any sequence that is ε-far from satisfying P, with probability at least $\frac{2}{3}$.*
   Then there must be a Poisson-s *algorithm $\mathcal{A}$ that gets only the histogram of the samples, where $s = O(t)$, satisfiying the following: if the input sequence satisfies P, $\mathcal{A}$ accepts with probability at least $\frac{2}{3} - o(1)$; if the input sequence is ε-far from satisfying P, $\mathcal{A}$ rejects with probability at least $\frac{2}{3} - o(1)$.*
2. *For any* Poisson-s *algorithm $\mathcal{A}$ with $s = o(n^{1-\frac{1}{k}})$, if $\mathcal{A}$ gets only access to the histogram of samples, then we have*

$$|\Pr[\mathcal{A}(D_{X_A}) = \text{True}] - \Pr[\mathcal{A}(D_{X_B}) = \text{True}]| = o(1).$$

Note that by the above Lemma, for a property P that is invariant under permutation of indices and values, any tester for P can be well simulated by a *Poisson-s* algorithm, which only accesses to the histogram of samples. Thus it suffices to only consider such *Poisson-s* algorithms. Furthermore, if there exist two instances of P with proportional moments, then it is hard to distinguish these two instances, for any *Poisson-s* algorithm that only accesses to the histogram of samples.

2.2 Proof of Theorem 3

Now we give the proof of Theorem 3. We first note that k-occurrence-freeness is a property that is invariant under permutation of indices and values. Suppose that there exist two families of sequence instances, denoted by $\mathcal{C}_A$ and $\mathcal{C}_B$, respectively, such that 1) $\mathcal{C}_A$ and $\mathcal{C}_B$ have $k-1$ proportional moments; 2) sequences in $\mathcal{C}_A$ are k-occurrence-free, and sequences in $\mathcal{C}_B$ are far from k-occurrence-freeness. Now assume that there exist a tester $\mathcal{A}'$ for k-occurrence-freeness with $s = o(n^{1-\frac{1}{k}})$ queries. Then, according to Lemma 8, there must be a *Poisson-s* algorithm $\mathcal{A}$ that gets only access to the histogram of samples. For such algorithm $\mathcal{A}$, we have

$$|\Pr[\mathcal{A}(D_{X_A}) = \text{True}] - \Pr[\mathcal{A}(D_{X_B}) = \text{True}]| = (\frac{2}{3} - o(1)) - (\frac{1}{3} + o(1)) \geq \frac{1}{6},$$

which contradicts to the second part of Lemma 8 and thus implies the $\Omega(n^{1-\frac{1}{k}})$ lower bound. Therefore, to prove Theorem 3, it suffices to construct two families of sequences with the above desired properties.

Proof of Theorem 3. We first construct two classes, denoted by $\mathcal{C}_A, \mathcal{C}_B$, of length-$n$ sequences, such that for any sequences $A \in \mathcal{C}_A$ and $B \in \mathcal{C}_B$, it holds that 1) A is k-occurrence-free and B is ε-far from k-occurrence-free, and 2) the frequency variables X_A, X_B of these two instances A, B have $k-1$ proportional moments.

To do so, we first prove the claim.

▷ Claim 9. It holds that

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & k \\ 1 & 2^2 & \cdots & k^2 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2^{k-1} & \cdots & k^{k-1} \end{pmatrix} \cdot \begin{pmatrix} (-1)^1\binom{k}{1} \\ (-1)^2\binom{k}{2} \\ (-1)^3\binom{k}{3} \\ \vdots \\ (-1)^k\binom{k}{k} \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{1}$$

Proof. We define a sequence of helper functions $f_j(x)$ to prove (1).

$$f_j(x) = \begin{cases} (1+x)^k, & j = 0 \\ x \cdot f_{j-1}^{'}(x), & j = 1, 2, \cdots, k-1 \end{cases} \tag{2}$$

To prove the claim, we note that it suffices to show the following:

$$f_0(-1) \;=\; 1 + \sum_{i=1}^{k} (-1)^{\cdot} \tbinom{k}{j} = 0, \tag{3}$$

$$f_j(-1) \;=\; \sum_{i=1}^{k} i^j \cdot (-1)^i \cdot \tbinom{k}{i} = 0, \text{ for any } j = 1, \ldots, k-1. \tag{4}$$

Note that if the above are true, then each line of Equations (1) holds, which finishes the proof of the claim. In the following, we prove Equations (3) and (4).

Let us first consider the binomial expansion of $f_0(x)$. We have that

$$f_0(x) = (1+x)^k = \sum_{i=0}^{k} x^i \cdot \tbinom{k}{i} = 1 + \sum_{i=1}^{k} x^i \cdot \tbinom{k}{i}. \tag{5}$$

Thus, $f_0(-1) = (1-1)^k = 1 + \sum_{i=1}^{k} (-1)^i \cdot \binom{k}{i} = 0$. That is, Equation (3) holds.

To prove Equation (4), we show that for any $1 \le j \le k-1$, it holds that

(a) $f_j(x) = \sum_{i=1}^{k} i^j \cdot x^i \cdot \binom{k}{i}$,

(b) $f_j(x) = \sum_{i=1}^{j} a_i \cdot x^i \cdot (1+x)^{k-i}$, for some numbers $a_1, \ldots, a_j \ge 0$.

Note that by the above two items, we have that $f_j(-1) = 0 = \sum_{i=1}^{k} i^j \cdot x^i \cdot \binom{k}{i}$, for each $j = 1, \ldots, k-1$, which finishes the proof of Equation (4) (and the claim).

In the following, we prove the above two items (a) and (b) by induction. Consider the case $j = 1$. By definition of function $f_j(x)$ given by (2) and the expansion (5), it holds that

$$f_0'(x) = k \cdot (1+x)^{k-1} = \sum_{i=1}^{k} i \cdot x^{i-1} \cdot \tbinom{k}{i},$$

which implies that

$$f_1(x) = x \cdot f_0^{'}(x) = x \cdot k \cdot (1+x)^{k-1} = \sum_{i=1}^{k} i \cdot x^i \cdot \tbinom{k}{i}$$

Now we assume that the items (a) and (b) hold for $j \leq k-2$, and we prove it for $j+1$.

For item (a), since $f_j(x) = \sum_{i=1}^{k} i^j \cdot x^i \cdot \binom{k}{i}$, we have that $f'_j(x) = \sum_{i=1}^{k} i^{j+1} \cdot x^{i-1} \cdot \binom{k}{i}$. Thus,

$$f_{j+1}(x) = \sum_{i=1}^{k} i^{j+1} \cdot x^i \cdot \binom{k}{i}$$

by Definition (2).

For item (b), since $f_j(x) = \sum_{i=1}^{j} a_i \cdot x^i \cdot (1+x)^{k-i}$, for some numbers $a_1, \ldots, a_j \geq 0$, it holds that

$$f'_j(x) = \sum_{i=1}^{j} (a_i \cdot i \cdot x^{i-1} \cdot (1+x)^{k-i} + a_i \cdot x^i \cdot (k-i) \cdot (1+x)^{k-i-1}).$$

Thus, by Definition (2),

$$f_{j+1}(x) = \sum_{i=1}^{j} (a_i \cdot i \cdot x^i \cdot (1+x)^{k-i} + a_i \cdot x^{i+1} \cdot (k-i) \cdot (1+x)^{k-i-1}) = \sum_{i=1}^{j+1} a'_i \cdot x^i \cdot (1+x)^{k-i},$$

for some numbers $a'_1, \cdots, a'_{j+1} \geq 0$.

Therefore, both items (a) and (b) hold and this finishes the proof the claim. ◁

Now we define two distributions $\mathbf{p}, \mathbf{q}$ over $[k]$ as follows.

1. if k is even, define

$$\mathbf{p}_i = \begin{cases} 0, & \text{if } i \text{ is even} \\ \frac{1}{2^{k-1}} \cdot \binom{k}{i}, & \text{if } i \text{ is odd} \end{cases} \qquad \mathbf{q}_i = \begin{cases} \frac{1}{2^{k-1}-1} \cdot \binom{k}{i}, & \text{if } i \text{ is even} \\ 0, & \text{if } i \text{ is odd} \end{cases}$$

2. if k is odd, define

$$\mathbf{p}_i = \begin{cases} \frac{1}{2^{k-1}-1} \cdot \binom{k}{i}, & \text{if } i \text{ is even} \\ 0, & \text{if } i \text{ is odd} \end{cases} \qquad \mathbf{q}_i = \begin{cases} 0, & \text{if } i \text{ is even} \\ \frac{1}{2^{k-1}} \cdot \binom{k}{i}, & \text{if } i \text{ is odd} \end{cases}$$

Now we show the following Lemma.

▶ **Lemma 10.** *Let $\mathbf{p}, \mathbf{q}$ be defined as above. There exists $d > 0$ such that*

$$\begin{pmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \mathbf{q}_3 \\ \vdots \\ \mathbf{q}_k \end{pmatrix} = d \cdot \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \vdots \\ \mathbf{p}_k \end{pmatrix} + (d-1) \cdot \begin{pmatrix} (-1)^1 \binom{k}{1} \\ (-1)^2 \binom{k}{2} \\ (-1)^3 \binom{k}{3} \\ \vdots \\ (-1)^k \binom{k}{k} \end{pmatrix} \tag{6}$$

Proof. For the case that k is even, we let $d = 1 + \frac{1}{2^{k-1}-1}$.

First note that $\mathbf{p}_k = 0$ and $\mathbf{q}_k = (d-1) \cdot \binom{k}{k}$. Thus, the last equation holds. For even $i \in \{2, 4, \ldots, k\}$, $\mathbf{p}_i = 0$ and $\mathbf{q}_i = d \cdot \mathbf{p}_i + (d-1) \cdot \binom{k}{i}$. For odd $i \in \{1, 3, \ldots, k-1\}$, $\mathbf{p}_i = \frac{d-1}{d} \cdot \binom{k}{i}$ and $\mathbf{q}_i = d \cdot \mathbf{p}_i + (d-1) \cdot (-1) \cdot \binom{k}{i} = 0$. Thus, Equation (6) holds.

For the case that k is odd, we let $d = 1 - \frac{1}{2^{k-1}}$.

Note that $\mathbf{p}_k = 0$ and $\mathbf{q}_k = (1-d) \cdot \binom{k}{k}$. Thus, the last equation holds. For odd $i \in \{1, 3, \ldots, k\}$, $\mathbf{p}_i = 0$ and $\mathbf{q}_i = d \cdot \mathbf{p}_i + (1-d) \cdot \binom{k}{i}$. For even $i \in \{2, 4, \ldots, k-1\}$, $\mathbf{p}_i = \frac{1-d}{d} \cdot \binom{k}{i}$ and $\mathbf{q}_i = d \cdot \mathbf{p}_i + (d-1) \cdot \binom{k}{i} = 0$. Thus, Equation (6) holds. ◀

▶ **Lemma 11.** *Let k be any integer with $k \geq 2$. Let $\mathbf{p}, \mathbf{q}$ be distributions over $[k]$ defined as above. It holds that*

1. $\mathbf{p}_k = 0$ *and* $\mathbf{q}_k \geq \frac{1}{2^k}$*;*
2. *for any two random variables X_A and X_B with distributions $\mathbf{p}$ and $\mathbf{q}$, respectively, it holds that X_A and X_B have $k-1$ proportional moments.*

Proof. The first item follows from the definitions of $\mathbf{p}$ and $\mathbf{q}$.

Now prove the second item. Let $d > 0$ be the number from Lemma 10. We will show that

$$\frac{\mathbf{E}[X_B]}{\mathbf{E}[X_A]} = \frac{\mathbf{E}[X_B^2]}{\mathbf{E}[X_A^2]} = \cdots = \frac{\mathbf{E}[X_B^{k-1}]}{\mathbf{E}[X_A^{k-1}]} = d,$$

or equivalently,

$$\begin{pmatrix} 1 \\ \mathbf{E}[X_B] \\ \mathbf{E}[X_B^2] \\ \vdots \\ \mathbf{E}[X_B^{k-1}] \end{pmatrix} = d \cdot \begin{pmatrix} 1/d \\ \mathbf{E}[X_A] \\ \mathbf{E}[X_A^2] \\ \vdots \\ \mathbf{E}[X_A^{k-1}] \end{pmatrix}. \tag{7}$$

By the definition X_A, it holds that for any $0 \leq i \leq k-1$, $\mathbf{E}[X_A^i] = \sum_{j=1}^{k} p_j \cdot j^i$. That is,

$$\begin{pmatrix} 1 \\ \mathbf{E}[X_A] \\ \mathbf{E}[X_A^2] \\ \vdots \\ \mathbf{E}[X_A^{k-1}] \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & k \\ 1 & 2^2 & \cdots & k^2 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2^{k-1} & \cdots & k^{k-1} \end{pmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \vdots \\ \mathbf{p}_k \end{pmatrix} \tag{8}$$

Similarly, it holds that

$$\begin{pmatrix} 1 \\ \mathbf{E}[X_B] \\ \mathbf{E}[X_B^2] \\ \vdots \\ \mathbf{E}[X_B^{k-1}] \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & k \\ 1 & 2^2 & \cdots & k^2 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2^{k-1} & \cdots & k^{k-1} \end{pmatrix} \begin{pmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \mathbf{q}_3 \\ \vdots \\ \mathbf{q}_k \end{pmatrix} \tag{9}$$

By Equations (8) and (9), we know that to prove Equation (7), it suffices to show that

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & k \\ 1 & 2^2 & \cdots & k^2 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2^{k-1} & \cdots & k^{k-1} \end{pmatrix} \begin{pmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \mathbf{q}_3 \\ \vdots \\ \mathbf{q}_k \end{pmatrix} = d \cdot \begin{pmatrix} 1/d & 1/d & \cdots & 1/d \\ 1 & 2 & \cdots & k \\ 1 & 2^2 & \cdots & k^2 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2^{k-1} & \cdots & k^{k-1} \end{pmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \vdots \\ \mathbf{p}_k \end{pmatrix}. \tag{10}$$

Recall that by Lemma 10, it holds that

$$\begin{pmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \mathbf{q}_3 \\ \vdots \\ \mathbf{q}_k \end{pmatrix} = d \cdot \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \vdots \\ \mathbf{p}_k \end{pmatrix} + (d-1) \cdot \begin{pmatrix} (-1)^1 \binom{k}{1} \\ (-1)^2 \binom{k}{2} \\ (-1)^3 \binom{k}{3} \\ \vdots \\ (-1)^k \binom{k}{k} \end{pmatrix}$$

Substituting $\mathbf{q}_i$ from the above equation to the left hand side of equation (10) gives us that

$$d \cdot \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & k \\ 1 & 2^2 & \cdots & k^2 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2^{k-1} & \cdots & k^{k-1} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \vdots \\ \mathbf{p}_k \end{pmatrix} + (d-1) \cdot \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & k \\ 1 & 2^2 & \cdots & k^2 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2^{k-1} & \cdots & k^{k-1} \end{pmatrix} \cdot \begin{pmatrix} (-1)^1\binom{k}{1} \\ (-1)^2\binom{k}{2} \\ (-1)^3\binom{k}{3} \\ \vdots \\ (-1)^k\binom{k}{i} \end{pmatrix}$$

$$= d \cdot \begin{pmatrix} 1 \\ \mathbf{E}[X_A] \\ \mathbf{E}[X_A^2] \\ \vdots \\ \mathbf{E}[X_A^{k-1}] \end{pmatrix} + (d-1) \cdot \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & k \\ 1 & 2^2 & \cdots & k^2 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2^{k-1} & \cdots & k^{k-1} \end{pmatrix} \cdot \begin{pmatrix} (-1)^1\binom{k}{1} \\ (-1)^2\binom{k}{2} \\ (-1)^3\binom{k}{3} \\ \vdots \\ (-1)^k\binom{k}{i} \end{pmatrix} = d \cdot \begin{pmatrix} 1/d \\ \mathbf{E}[X_A] \\ \mathbf{E}[X_A^2] \\ \vdots \\ \mathbf{E}[X_A^{k-1}] \end{pmatrix},$$

where the last equation follows from Claim 1.

On the other hand, by Equation (8), we know that the right hand side of (10) is,

$$d \cdot \begin{pmatrix} 1/d & 1/d & \cdots & 1/d \\ 1 & 2 & \cdots & k \\ 1 & 2^2 & \cdots & k^2 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2^{k-1} & \cdots & k^{k-1} \end{pmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \vdots \\ \mathbf{p}_k \end{pmatrix} = d \cdot \begin{pmatrix} 1/d \\ \mathbf{E}[X_A] \\ \mathbf{E}[X_A^2] \\ \vdots \\ \mathbf{E}[X_A^{k-1}] \end{pmatrix}.$$

Therefore, Equation (10) holds and thus X_A and X_B have $k-1$ proportional moments. This finishes the proof of the Lemma. ◀

Now we construct class $\mathcal{C}_A$ as follows: $\mathcal{C}_A$ is a class of sequences, and the frequency variable X_A of every sequence A is $\Pr[X_A = i] = p_i$. That is, for every sequence A, the fraction of elements that occur i times is exactly p_i. We can construct $\mathcal{C}_B$ analogously by substituting p_i with q_i.

By construction, sequence A is k-occurrence-free. Consider the sequence B. Suppose that there are l distinct values in B, then at least $q_k \cdot l$ values occur k times in B, which means that B is at least $\frac{q_k \cdot l}{n}$-far from k-occurrence-free. As every value in B occurs in at most k positions, there are at least $\frac{n}{k}$ distinct values, i.e., $l \geq \frac{n}{k}$. Thus, B is at least $\frac{q_k}{k}$-far from k-occurrence-free. According to previous analysis, A and B have $k-1$ proportional moments. The theorem then follows from Lemma 8. ◀

3 A Lower Bound for Testing Subgraph-Freeness

In this section, we give the proof of the lower bound on the query complexity for testing subgraph-freeness, i.e., prove Theorem 2.

Proof of Theorem 2. We give a reduction from the problem of testing k-occurrence of a sequence to the problem of testing H-freeness in the unidirectional model. That is, given an instance of the former problem, i.e., a length-n sequence S such that each element is promised to occur at most k times, we will construct an instance of the H-freeness testing problem, i.e., a directed graph G with $n' = \Theta(n)$ vertices and bounded degree. Then we show that this construction preserves the distances of the properties and any algorithm $\mathcal{A}'$ for testing H-freeness in the unidirectional model can be invoked on G to test if S is

k-occurrence-freeness. In particular, if $\mathcal{A}'$ has query complexity $o(n'^{1-\frac{1}{k}})$, then this implies an algorithm for testing k-occurrence-freeness with query complexity $o(n^{1-\frac{1}{k}})$, contradicting to Theorem 3.

Preprocessing the subgraph H. Since H has k source components, we denote these components by $\{C_1, \cdots, C_k\}$. Note that each C_i is a subgraph of H. We use N_{comp} to denote the maximum number of vertices in $\{C_1, C_2, \cdots, C_k\}$, i.e., $N_{\text{comp}} = \max_{i=1,\cdots,k} |V(C_i)|$ where $V(C)$ denotes the vertex set of the graph C. We use C_0 to denote the subgraph induced by the remainder of vertices in $V(H)$ and we call C_0 the center component of H. Let $N_{\text{center}} = |V(C_0)| = |V(H)| - \sum_{i=1}^{k} |V(C_i)|$. Note that since $C_1, \cdots, C_k$ are source components, by definition, no edge exists between different such components. All the edges leaving C_i (for $i = 1, \cdots, k$) are entering C_0. We can first decompose H into source components and the center component (e.g., by using Tarjan's algorithm [21]), index them, and identify all the edges crossing different components in constant time (as the size of H is constant).

We illustrate such a decomposition of a subgraph $\tilde{H}$ in Figure 1. Note that $\tilde{H}$ has 3 source components and 1 center component (see Figure 2). It can be partitioned into four parts such $V(\tilde{C}_0) = \{v_2, v_7\}$, $V(\tilde{C}_1) = \{v_1\}$, $V(\tilde{C}_2) = \{v_3, v_4, v_5\}$, $V(\tilde{C}_3) = \{v_6\}$ as follows. In this example, $N_{\text{comp}} = 3$, $N_{\text{center}} = 2$.

In the construction of the graph G, we will treat each component C_i, $1 \le i \le k$, as a subgraph with N_{comp} vertices. That is, for each such C_i, we add $(N_{\text{comp}} - |V(C_i)|)$ isolated vertices to C_i to obtain a new component C'_i so that $|V(C'_i)| = N_{\text{comp}}$. We can reassemble these new components $\{C'_1, C'_2, \cdots, C'_k\}$ with C_0 to obtain a graph H'.

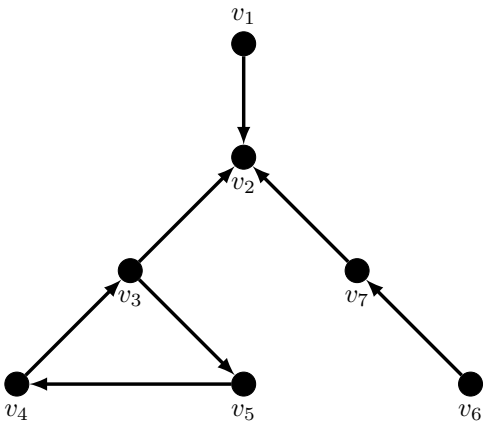
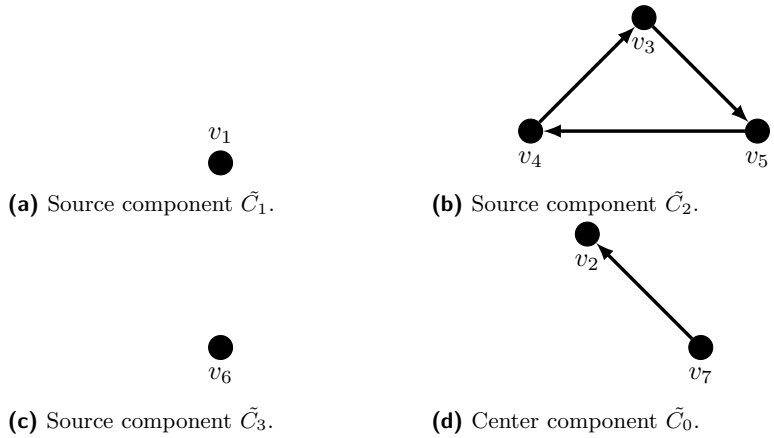


Figure 1 A subgraph $\tilde{H}$.

Now we index each vertex of H' by some integer in $\{1, \cdots, N_{\text{center}} + k \cdot N_{\text{comp}}\}$ as follows. The index set of $V(C'_0)$ is $[1, N_{\text{center}}]$, and the index set of $V(C'_i)$ is $[N_{\text{center}} + (i-1) \cdot N_{\text{comp}} + 1, N_{\text{center}} + i \cdot N_{\text{comp}}]$, for each $1 \le i \le k$. Furthermore, for each component C'_i with $0 \le i \le k$, we sequentially index the vertices using the corresponding index set according to the lexicographical ordering of the vertices in the aforementioned component decomposition.

Now we describe the reduction. Given a length-n sequence S, the directed graph $G = (V, E)$ can be constructed as follows. We first add n disjoint copies of the subgraph C_0 to G. Then we will add n copies of source components and add some edges from source components to some copy of C_0 constructed before. That is, each element in the sequence corresponds to a source component. Note that there are no edges between different copies of source components. The offline construction is formally described as follows.



(a) Source component $\tilde{C}_1$. **(b)** Source component $\tilde{C}_2$.

(c) Source component $\tilde{C}_3$. **(d)** Center component $\tilde{C}_0$.

Figure 2 Decomposing $\tilde{H}$ into 3 source components and 1 center component.

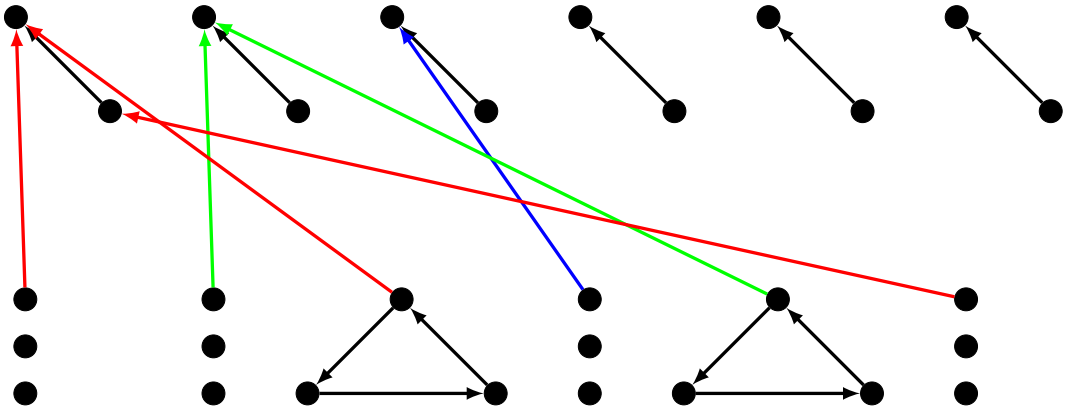


Figure 3 Constructing $\tilde{G}$ from the sequence $\tilde{S}$ and the decomposition of $\tilde{H}$.

Vertex set and vertex indices. We index vertices in G from 1 to $n \cdot (N_{\text{center}} + N_{\text{comp}})$. The vertex set is decomposed into two parts: the *center part* and the *source part*. More precisely, the source part contains n potential source components with vertex indices from 1 to $n \cdot N_{\text{comp}}$, and the center part contains n disjoint copies of the center component C_0 with vertex indices from $n \cdot N_{\text{comp}} + 1$ to $n \cdot (N_{\text{comp}} + N_{\text{center}})$. Furthermore, the vertices in the i-th copy of the source component are indexed from $(i-1) \cdot N_{\text{comp}} + 1$ to $i \cdot N_{\text{comp}}$, while the vertices of the j-th copy of the center component are indexed from $n \cdot N_{\text{comp}} + (j-1) \cdot N_{\text{center}} + 1$ to $n \cdot N_{\text{comp}} + j \cdot N_{\text{center}}$.

Adding components and edges. Add n disjoint copies of C_0 to G. Initialize a size-n array T such that $T_a = 0$ for each $1 \le a \le n$. For each $a = 1, 2, \cdots, n$:

1. let b be the value (or element) of S at position a
2. If b is a new value that algorithm sees for the first time, define an array $R_b = \{1, 2, \cdots, k\}$.
3. Uniformly sample a number t from R_b. Add an copy of C'_t. Ignoring isolated vertices in C'_t, add edges between this copy of C'_t and the b-th copy of C_0 in the same way as the connections between their counterparts in the subgraph H. Delete t from R_b. Set $T_a = t$, i.e., the a-th position of S is mapped to a source component C'_t.

Note that by construction, the graph G is d-bounded, and its maximum (in- or out-) degree the same as the maximum (in- or out-) degree of H.

We give an illustration of the above construction in Figure 3. Given a sequence $\tilde{S} = \{1, 2, 1, 3, 2, 1\}$, and a subgraph $\tilde{H}$ as shown in Figure 1. The graph $\tilde{G}$ from the above reduction is shown Figure 3. In this figure, edges of the same color correspond to positions of the same value (or element) in $\tilde{S}$. For example, the 3 red edges correspond to the 3 occurrences of value 1. Together with the corresponding source and center components, these red edges form an copy of $\tilde{H}$.

Construction on the fly. We show that the above construction of G can be done on the fly and each query to G can be answered by querying at most 1 position in S. More precisely, let $\mathcal{A}'$ be an algorithm for testing H-freeness. When $\mathcal{A}'$ queries the i-th outgoing neighbor of a vertex v, we consider the following cases.

If $v > n \cdot N_{\text{comp}}$, then v belongs to a copy of C_0, then we do not need to query sequence S, and we can simply locate the vertex $v' = (v - n \cdot N_{\text{comp}}) \bmod N_{\text{center}}$ in C_0. And by our index in H', we know the corresponding vertex index in H' is also v'. Then we can check the i-th neighbor of v' in H', denoted by v''. Thus we just return $v - v' + v''$.

If $1 \leq v \leq n \cdot N_{\text{comp}}$, then v belongs to a copy of some source component. Calculate $a = \lceil v/N_{\text{comp}} \rceil$ and query the a-th position of S. Let b denote the query answer. If $T_a = 0$, which means that this element is queried for the first time, uniformly sample a type t from the rest of types R_b for value b, and update $T_a = t$; otherwise simply set $t = T_a$. Note that R and T are maintained as described in the construction. Then calculate $v' = v \bmod N_{\text{comp}}$. Now we know that the queried vertex v corresponds to the v'-th vertex in a C'_t component, which is adjacent to the b-th copy of C_0. We can look up vertex $N_{\text{center}} + (t-1) \cdot N_{\text{comp}} + v'$ in H', which is isomorphic to vertex v in G. We use v'' to denote the i-th neighbor of $N_{\text{center}} + (t-1) \cdot N_{\text{comp}} + v'$ in H'. If v'' belongs to the C'_t part in H, we just return $v - v' + v''$. Otherwise, if v'' belongs to a C_0 part, we return $n \cdot N_{\text{comp}} + (b-1) \cdot N_{\text{center}} + v''$.

Thus, any query for a vertex v with $v > n \cdot N_{\text{comp}}$ can be answered without querying S; query for a vertex v with $1 \leq v \leq n \cdot N_{\text{comp}}$ can be answered by making one query to S.

Note that our construction generates a graph G from a distribution $\mathcal{D} = \{G_1, G_2, \cdots\}$. We will show that if S is k-occurrence-free, then any graph from $\mathcal{D}$ is H-free; if S is far from being k-occurrence-free, then every graph in $\mathcal{D}$ is far from H-freeness.

Preserving the distances. Note that in the above construction, if there exists some value occurring k times in S, then these k occurrences of the same value results in k different source components covering $\{C_1, C_2, \cdots, C_k\}$, and they are adjacent to the same center. That is, each element occurring k times in the sequence result in an occurrence of H in G. For each element occurring less than k times, the center corresponding to this value will be adjacent to less than k source components, which in turn implies that H does not occur in this case. We mention that the auxiliary isolated vertices also do not contribute to any occurrence of H.

Thus, if S is k-occurrence-free, then there can not be any occurrence of H, and thus G must be H-free. If S is ε-far from being k-occurrence-free, then there will be at least εn occurrences of H in G. This implies that G is at least ε'-far from H-freeness, for $\varepsilon' = \frac{\varepsilon n}{d(N_{\text{center}} + N_{\text{comp}})n} = \frac{\varepsilon}{d(N_{\text{center}} + N_{\text{comp}})}$.

Putting things together. Let $\mathcal{A}'$ be an algorithm for testing H-freeness with proximity parameter $\varepsilon = \Theta_{k,d}(1)$. Suppose that the query complexity is $o(n'^{1-\frac{1}{k}})$ on an n'-vertex digraph. Now we invoke the algorithm $\mathcal{A}'$ on the graph G that was constructed as before.

As we have seen, each query in G can be answered by making at most 1 query to the sequence S. Furthermore, if S is k-occurrence-free, then G is H-free and if S is ε-far from being k-occurrence-free, then G is ε'-far from H-free, for $\varepsilon' = \frac{\varepsilon}{d\cdot(N_{\text{center}}+N_{\text{comp}})} = \Theta_{k,d}(1)$. Thus, the algorithm $\mathcal{A}'$, together with the construction, also solves the problem of testing k-occurrence-freeness with $o(n^{1-\frac{1}{k}})$ queries, which contradicts Theorem 3. Thus, the query complexity of $\mathcal{A}'$ is $\Omega(n^{1-\frac{1}{k}})$. This finishes the proof of the theorem. ◀

References

1. Noga Alon, Eldar Fischer, Ilan Newman, and Asaf Shapira. A combinatorial characterization of the testable graph properties: it's all about regularity. *SIAM Journal on Computing*, 39(1):143–167, 2009.
2. Noga Alon and Asaf Shapira. Testing subgraphs in directed graphs. *Journal of Computer and System Sciences*, 69(3):354–382, 2004.
3. Michael A Bender and Dana Ron. Testing properties of directed graphs: acyclicity and connectivity. *Random Structures & Algorithms*, 20(2):184–205, 2002.
4. Xi Chen, Tim Randolph, Rocco A Servedio, and Timothy Sun. A lower bound on cycle-finding in sparse digraphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2936–2952. SIAM, 2020.
5. Artur Czumaj, Pan Peng, and Christian Sohler. Relating two property testing models for bounded degree directed graphs. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 1033–1045, 2016.
6. Marcel Dall'Agnol, Tom Gur, and Oded Lachish. A structural theorem for local algorithms with applications to coding, testing, and privacy. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1651–1665. SIAM, 2021.
7. Eldar Fischer, Oded Lachish, and Yadu Vasudev. Trading query complexity for sample-based testing and multi-testing scalability. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1163–1182. IEEE, 2015.
8. Sebastian Forster, Danupon Nanongkai, Liu Yang, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Computing and testing small connectivity in near-linear time and queries via fast local cut algorithms. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2046–2065. SIAM, 2020.
9. Oded Goldreich. Introduction to testing graph properties. In *Property testing*, pages 105–141. Springer, 2010.
10. Oded Goldreich. *Introduction to property testing*. Cambridge University Press, 2017.
11. Oded Goldreich, Shari Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM (JACM)*, 45(4):653–750, 1998.
12. Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002.
13. Oded Goldreich and Dana Ron. On proximity-oblivious testing. *SIAM Journal on Computing*, 40(2):534–566, 2011.
14. Oded Goldreich and Dana Ron. On sample-based testers. *ACM Transactions on Computation Theory (TOCT)*, 8(2):1–54, 2016.
15. Tom Gur and Oded Lachish. On the power of relaxed local decoding algorithms. *SIAM Journal on Computing*, 50(2):788–813, 2021.
16. Frank Hellweg and Christian Sohler. Property testing in sparse directed graphs: strong connectivity and subgraph-freeness. In *European Symposium on Algorithms*, pages 599–610. Springer, 2012.
17. Hiro Ito, Areej Khoury, and Ilan Newman. On the characterization of 1-sided error strongly testable graph properties for bounded-degree graphs. *computational complexity*, 29(1):1–45, 2020.

18 Yaron Orenstein and Dana Ron. Testing eulerianity and connectivity in directed sparse graphs. *Theoretical Computer Science*, 412(45):6390–6408, 2011.

19 Sofya Raskhodnikova, Dana Ron, Amir Shpilka, and Adam Smith. Strong lower bounds for approximating distribution support size and the distinct elements problem. *SIAM Journal on Computing*, 39(3):813–842, 2009.

20 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

21 Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

22 Paul Valiant. Testing symmetric properties of distributions. *SIAM Journal on Computing*, 40(6):1927, 2011.

23 Yuichi Yoshida and Hiro Ito. Testing k-edge-connectivity of digraphs. *Journal of systems science and complexity*, 23(1):91–101, 2010.

Decidability of Fully Quantum Nonlocal Games with Noisy Maximally Entangled States

Minglong Qin
State Key Laboratory for Novel Software Technology, Nanjing University, China

Penghui Yao
State Key Laboratory for Novel Software Technology, Nanjing University, China
Hefei National Laboratory, 230088, China

Abstract

This paper considers the decidability of fully quantum nonlocal games with noisy maximally entangled states. Fully quantum nonlocal games are a generalization of nonlocal games, where both questions and answers are quantum and the referee performs a binary POVM measurement to decide whether they win the game after receiving the quantum answers from the players. The quantum value of a fully quantum nonlocal game is the supremum of the probability that they win the game, where the supremum is taken over all the possible entangled states shared between the players and all the valid quantum operations performed by the players. The seminal work $\mathrm{MIP}^* = \mathrm{RE}$ [16, 17] implies that it is undecidable to approximate the quantum value of a fully nonlocal game. This still holds even if the players are only allowed to share (arbitrarily many copies of) maximally entangled states. This paper investigates the case that the shared maximally entangled states are noisy. We prove that there is a computable upper bound on the copies of noisy maximally entangled states for the players to win a fully quantum nonlocal game with a probability arbitrarily close to the quantum value. This implies that it is decidable to approximate the quantum values of these games. Hence, the hardness of approximating the quantum value of a fully quantum nonlocal game is not robust against the noise in the shared states.

This paper is built on the framework for the decidability of non-interactive simulations of joint distributions [12, 7, 11] and generalizes the analogous result for nonlocal games in [26]. We extend the theory of Fourier analysis to the space of super-operators and prove several key results including an invariance principle and a dimension reduction for super-operators. These results are interesting in their own right and are believed to have further applications.

2012 ACM Subject Classification Theory of computation → Quantum complexity theory

Keywords and phrases Fully quantum nonlocal games, Fourier analysis, Dimension reduction

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.97

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2211.10613`

Funding This work was supported by National Natural Science Foundation of China (Grant No. 61972191), Innovation Program for Quantum Science and Technology (Grant No. 2021ZD0302900) and the Program for Innovative Talents and Entrepreneur in Jiangsu.
Penghui Yao: Grant No. 61972191, 2021ZD0302900.

Acknowledgements We thank Zhengfeng Ji for helpful discussion. We also thank the anonymous reviewers for their careful reading and many helpful comments and suggestions.

1 Introduction

Nonlocal games are a core model in the theory of quantum computing, which has found wide applications in quantum complexity theory, quantum cryptography, and the foundation of quantum mechanics. A nonlocal game is executed by three parties, a referee and two

non-communicating players, which are usually named Alice and Bob. Before the game starts, the players may share an arbitrary bipartite quantum state. The referee samples a pair of questions and sends each of them to the players, separately. Each player is supposed to reply with a classical answer to the referee. They win the game if the questions and the answers satisfy a given predicate. The distribution of the questions and the predicate is known to the players. The *quantum value* is the supremum of the probability that the players win the game. It is a central topic in quantum computing to understand the computational complexity of computing the quantum value of a nonlocal game. After decades of efforts [6, 21, 20, 14, 15, 24, 9], it has been finally settled by the seminal work $\mathrm{MIP}^* = \mathrm{RE}$ [16, 17], where Ji, Natarajan, Vidick, Wright and Yuen proved that it is undecidable to approximately compute the quantum value of a nonlocal game with constant precision. This result implies that there is no computable upper bound on the preshared entanglement for the players to win the game with a probability close to the quantum value. Otherwise, the probability of success can be obtained by ε-netting all possible quantum strategies and brute-force searching for the optimal value. Ji et al. essentially proved that it is still uncomputable even if the players are only allowed to share (arbitrarily many) EPR states.

In [26], the authors investigated the robustness of the hardness of the nonlocal games under noise. More specifically, they considered a variant of nonlocal games, where the preshared quantum states are corrupted. It is shown that the quantum value of a nonlocal game is computable if the players are allowed to share arbitrarily many copies of *noisy maximally entangled states* (MES). Hence, the hardness of the nonlocal games collapses in the presence of noise from the preshared entangled states.

In this paper, we consider *fully quantum nonlocal games*, which are a broader class of games where both questions and answers are quantum and the predicates are replaced by quantum measurements with binary outcomes: win and loss. More specifically, a fully quantum nonlocal game

$$\mathfrak{G} = \left(\mathcal{P}, \mathcal{Q}, \mathcal{R}, \mathcal{A}, \mathcal{B}, \phi_{\mathsf{in}}^{\mathcal{PQR}}, \left\{P_{\mathsf{win}} = M^{\mathcal{ABR}}, P_{\mathsf{loss}} = \mathbb{1} - M^{\mathcal{ABR}}\right\}\right)$$

consists of a referee and two non-communicating players: Alice and Bob, where $\mathcal{P}, \mathcal{Q}, \mathcal{R}, \mathcal{A}, \mathcal{B}$ are quantum systems, $\phi_{\mathsf{in}}^{\mathcal{PQR}}$ is a tripartite quantum state in $\mathcal{P} \otimes \mathcal{Q} \otimes \mathcal{R}$ and $\{P_{\mathsf{win}}, P_{\mathsf{loss}}\}$ is a measurement acting on $\mathcal{A} \otimes \mathcal{B} \otimes \mathcal{R}$. Alice, Bob, and the referee share the input state $\phi_{\mathsf{in}}^{\mathcal{PQR}}$, where Alice, Bob, and the referee hold $\mathcal{P}, \mathcal{Q}, \mathcal{R}$, respectively, at the beginning of the game. Alice and Bob are supposed to perform quantum operations mapping $\mathcal{P}$ to $\mathcal{A}$ and $\mathcal{Q}$ to $\mathcal{B}$, and then send the quantum states in $\mathcal{A}$ and $\mathcal{B}$ to the referee, respectively. After receiving the quantum messages from the players, the referee performs the POVM measurement $\{P_{\mathsf{win}}, P_{\mathsf{loss}}\}$. Again, the players are allowed to share arbitrary quantum states before the game starts. Both players know the description of ϕ_{in} and the POVM. The quantum value of the game G is the supremum of the probability that the players win the game. The supremum is over all possible preshared quantum states and the quantum operations that can be implemented by both parties. It is not hard to see if $\phi_{\mathsf{in}} = \sum_{x,y} \mu(x,y) |x\rangle\langle x|^{\mathcal{P}} \otimes |y\rangle\langle y|^{\mathcal{Q}} \otimes |xy\rangle\langle xy|^{\mathcal{R}}$ and both P_{win} and P_{loss} are projectors on computational basis, where μ is a bipartite distribution, then it boils down to a nonlocal game.

Fully quantum nonlocal games also capture the complexity class of two-prover one-round quantum multi-prover interactive proof systems $\mathsf{QMIP}(2,1)$. The variants of nonlocal games, where either the questions or the answers are replaced by quantum messages have occurred in much literature [3, 22, 27, 5, 10, 4, 2, 18]. In [3], Buscemi introduced the so-called semi-quantum nonlocal games, which are nonlocal games with quantum questions and classical answers, and proved that semi-quantum nonlocal games can be used to characterize LOSR (local operations and shared randomness) paradigm. Such games are further used to study

the entanglement verification in the subsequent work [4, 2]. In a different context, Regev and Vidick in [27] proposed quantum XOR games, where the questions are quantum and the answers are still classical. In [22], Leung, Toner, and Watrous introduced a communication task: coherent state exchange and its analogue in the setting of nonlocal games, where both questions and answers are quantum. In [10], Fitzsimons and Vidick demonstrated an efficient reduction that transforms a local Hamiltonian into a 5-players nonlocal game allowing classical questions and quantum answers. They showed that approximating the value of this game to a polynomial inverse accuracy is QMA-complete. In [5], Chung, Wu, and Yuen further proved a parallel repetition for nonlocal games where again questions are classical and answers are quantum.

As fully quantum nonlocal games are a generalization of nonlocal games, Ji et al.'s result [16, 17] implies that it is also undecidable to approximately compute the quantum value of a fully quantum nonlocal game, even if they are only allowed to share MESs.

In this paper, we continue the line of research in [26] to investigate whether the hardness of fully quantum nonlocal games can be maintained against the noise. More specifically, we consider the games where the players share arbitrarily many copies of noisy MES's $\psi^{\mathcal{ST}}$. Each $\psi^{\mathcal{ST}}$ is a bipartite state in quantum system $\mathcal{S}\otimes\mathcal{T}$, where Alice and Bob hold $\mathcal{S}$ and $\mathcal{T}$, respectively. The value of a game can be written as

$$\mathrm{val}_Q(\mathfrak{G},\psi)=\lim_{n\to\infty}\max_{\Phi_{\mathsf{Alice}},\Phi_{\mathsf{Bob}}}\mathrm{Tr}\left[P_{\mathsf{win}}\left(\left(\Phi_{\mathsf{Alice}}\otimes\Phi_{\mathsf{Bob}}\right)\left(\phi_{\mathsf{in}}^{\mathcal{PQR}}\otimes\left(\psi^{\mathcal{ST}}\right)^{\otimes n}\right)\right)\right].$$

where the maximum is taken over all quantum operations $\Phi_{\mathsf{Alice}}:\mathcal{P}\otimes\mathcal{S}^{\otimes n}\to\mathcal{A}$ and $\Phi_{\mathsf{Bob}}:\mathcal{Q}\otimes\mathcal{T}^{\otimes n}\to\mathcal{B}$. Noisy MESs were introduced in [26], which will be defined later. They include depolarized EPR states $(1-\varepsilon)\left|\Psi\right\rangle\!\left\langle\Psi\right|+\varepsilon\mathbb{1}/2\otimes\mathbb{1}/2$, where $\varepsilon>0$ and $\left|\Psi\right\rangle=\left(\left|00\right\rangle+\left|11\right\rangle\right)/\sqrt{2}$ is an EPR state. [16, 17] proved that it is undecidable to approximate $\mathrm{val}_Q(\mathfrak{G},\left|\Psi\right\rangle)$ within constant precision.

Main results

In this paper, we prove that it is computable to approximate $\mathrm{val}_Q(\mathfrak{G},\psi)$ within arbitrarily small precision if ψ is a noisy MES.

▶ **Theorem 1** (Main result, informal). *Given integer $m\geq 2$, $\delta\in(0,1)$ and a fully quantum nonlocal game $\mathfrak{G}$, where players are allowed to share arbitrarily many copies m-dimensional noisy MESs ψ, there exists an explicitly computable bound $D=D\left(\varepsilon,\delta,m,\mathfrak{G}\right)$ such that it suffices for the players to share D copies of ψ to achieve the winning probability at least $\mathrm{val}_Q(\mathfrak{G},\psi)-\delta$. Thus it is feasible to approximate the quantum value of the game $(\mathfrak{G},\psi)$ to arbitrarily precision.*

As mentioned above, the class of noisy MESs includes $(1-\varepsilon)\left|\Psi\right\rangle\!\left\langle\Psi\right|+\varepsilon\mathbb{1}/2\otimes\mathbb{1}/2$, where $\varepsilon>0$ and Ψ is an EPR state. It is as hard as Halting problem to approximate $\mathrm{val}_Q(\mathfrak{G},\left|\Psi\right\rangle)$ proved by [16, 17]. Therefore, our result implies that the hardness of fully quantum nonlocal games is also not robust against the noise in the preshared states.

This result generalizes [26] where the authors proved that it is feasible to approximate the values when both questions and answers are classical. Both works are built on the series of works for the decidability of *non-interactive simulations of joint distributions* [12, 11, 7]. In the setting of non-interactive simulations of joint distributions, two non-communicating players Alice and Bob are provided a sequence of independent samples $(x_1,y_1),(x_2,y_2),\ldots$ from a joint distribution μ, where Alice observes $x_1,x_2,\ldots$ and Bob observes $y_1,y_2,\ldots$. The question is to decide what joint distribution ν Alice and Bob can sample. The research on this problem has a long history and fruitful results (see, for example [19] and the references

therein). The quantum analogue was first studied by Delgosha and Beigi [8], which is referred to as *local state transformation*. The decidability of local state transformation is still widely open. In this work, we prove that the local state transformation is decidable when the source states are noisy MESs.

1.1 Contributions

The main contribution in this paper is developing a Fourier-analytic framework for the study of the space of super-operators. Here we list some conceptual or technical contributions, which are believed to be interesting in their own right and have further applications in quantum information science.

1. Analysis in the space of super-operators.
   The space of super-operators is difficult to understand in general. In this paper, we make a crucial observation that the quantum value of a fully quantum nonlocal game can be reformulated in terms of the *Choi representations* of the adjoint maps of the quantum operations. Instead of the space of super-operators, we apply Fourier analysis to the space spanned by those Choi representations. Then we prove an invariance principle for super-operators as well as a dimension reduction for quantum operations, which generalize the analogous results in [26].
   Our understanding of Fourier analysis in the space of super-operators is still very limited, although Boolean analysis has been studied extensively in both mathematics and theoretical computer science for decades. The approach taken in this paper may pave the way for the theory of Fourier analysis in the space of super-operators.
2. Invariance principle for super-operators.
   The classical invariance principle is a central limit theorem for polynomials [23], which asserts that the distribution of a low-degree and flat polynomial with random inputs uniformly drawn from $\{\pm 1\}^n$ is close to the distribution which is obtained by replacing the inputs with i.i.d. standard normal distributions. Here a polynomial is flat means that no variable has high influence on the value of the polynomial. In [26], the authors established an invariance principle for matrix spaces. This paper further proves an invariance principle for super-operators. This is essential to reduce the number of shared noisy MESs.
3. Dimension reduction for quantum operations.
   An important step in our proof is a dimension reduction for quantum operations, which enables us to reduce the dimensions of both players' quantum operations. It, in turn, reduces the number of noisy MESs shared between the players. Dimension reductions for quantum operations are usually difficult and sometimes even impossible [13, 28]. In this paper, we prove a dimension reduction via an invariance principle for super-operators and the dimension reduction for polynomials in Gaussian spaces [11]. we adopt the techniques in [11] with a delicate analysis. It leads to an exponential upper bound in the main theorem. which also improves the doubly exponential upper bound in [26].

1.2 Comparison with [26]

In [26], the authors applied Fourier analysis to the Hilbert space where both players' measurements stay, and proved hypercontractive inequalities, quantum invariance principles and dimension reductions for matrices and random matrices. In a fully quantum nonlocal game, both players perform quantum operations. Hence, a natural approach is to further extend the framework in [12, 26] to the space of super-operators.

The first difficulty occurs as the answers are quantum. In [26], the authors applied the framework to each pair of POVM elements (one from Alice and one from Bob). Further taking a union bound, the result concludes. Hence, it suffices to work on the space where the POVM elements stay, which is a tensor product of identical Hilbert spaces. This approach fails when considering fully quantum nonlocal games as the answers are quantum. Hence, we need to have a convenient representation of super-operators to work on. It is known that there are several equivalent representations of super-operators [29]. In this paper, we choose the Choi representations of super-operators, which view a super-operator as an operator in the tensor product of the input space and the output space. Hence, the underlying Hilbert space is a tensor product of a number of identical Hilbert spaces and the output Hilbert space. Thus, the analysis in [26] cannot be generalized here directly.

The second difficulty occurs as the questions are quantum. In [26], the authors essentially proved an upper bound on the number of noisy MESs for each pair of inputs. If the precision of the approximation is good enough, then we can obtain an upper bound for all inputs again by a union bound because the questions are finite in a nonlocal game. This argument cannot be directly generalized to fully nonlocal games as the questions are the marginal state of the input state with Alice and Bob. Fortunately, this difficulty can be avoided as the input state is in a bounded-dimensional space and thus it suffices to prove the theorem for each basis element from a properly chosen basis in the space, and then take a union bound.

The last difficulty is that the rounding argument in [26] does not apply to fully quantum nonlocal games. In the end of the construction, the new super-operators are no longer valid quantum operations. In [26], the construction gives a number of Hermitian operators in the end. The rounding argument proves that it is possible to round these Hermitian operators to valid POVMs with small deviation. For fully quantum nonlocal games we need a new rounding argument which is able to round super-operators to valid quantum operations with small deviation in the end of the construction.

1.3 Proof overview

The proof is built on the framework in [12, 11, 7] for the decidability of non-interactive simulation of joint distributions. To explain the high-level idea of our proof, we start with the decidability of a particular task of local state transformation. Then we explain how to generalize it to nonlocal games.

Local state transformation

We are interested in the decidability of the following local state transformation problem.

Given $\delta > 0$, a bipartite state σ and a noisy MES ψ, suppose Alice and Bob share arbitrarily many copies of ψ.

- **Yes**. Alice and Bob are able to jointly generate a bipartite state σ' using only local operations such that σ' is δ-close to σ, i.e., $\|\sigma - \sigma'\|_1 \leq \delta$.
- **No**. Any quantum state σ' that Alice and Bob can jointly generate using only local operations is 2δ-far from σ, i.e., $\|\sigma - \sigma'\|_1 \geq 2\delta$.

As there is no upper bound on the number of copies of ψ, the decidability of this question is unclear. If it were proved that any quantum operation could be simulated by a quantum operation in a bounded dimension, then the problem would be decidable as we could search

all possible quantum operations in a bounded-dimensional space via an ε-net and brute force. More specifically, suppose Alice and Bob share n copies of noisy MESs ψ and they perform quantum operations Φ_{Alice} and Φ_{Bob}. For any precision parameter $\delta \in (0,1)$, we need to construct quantum operations $\widetilde{\Phi_{\mathsf{Alice}}}$ and $\widetilde{\Phi_{\mathsf{Bob}}}$ acting on D copies of ψ, where D is independent of n, such that

$$\left(\Phi_{\mathsf{Alice}} \otimes \Phi_{\mathsf{Bob}}\right)\left(\psi^{\otimes n}\right) \approx \left(\widetilde{\Phi_{\mathsf{Alice}}} \otimes \widetilde{\Phi_{\mathsf{Bob}}}\right)\left(\psi^{\otimes D}\right). \tag{1}$$

To explain the high-level ideas, we assume that ψ is a 2-qubit quantum state for simplicity. Let $\{\mathcal{X}_a\}_{a\in\{0,1,2,3\}}$ be an orthonormal basis in the space of 2×2 matrices. We observe that the left hand side of Equation (1) is determined by the following 4^{2n} values:

$$\left\{\mathrm{Tr}\left[\left(\mathcal{X}_a \otimes \mathcal{X}_b\right)\left(\left(\Phi_{\mathsf{Alice}} \otimes \Phi_{\mathsf{Bob}}\right)\left(\psi^{\otimes n}\right)\right)\right]\right\}_{a,b\in\{0,1,2,3\}^n},$$

where $\mathcal{X}_a = \mathcal{X}_{a_1} \otimes \cdots \otimes \mathcal{X}_{a_n}$. Notice that

$$\mathrm{Tr}\left[\left(\mathcal{X}_a \otimes \mathcal{X}_b\right)\left(\left(\Phi_{\mathsf{Alice}} \otimes \Phi_{\mathsf{Bob}}\right)\left(\psi^{\otimes n}\right)\right)\right] = \mathrm{Tr}\left[\left(\left(\Phi_{\mathsf{Alice}}\right)^*\left(\mathcal{X}_a\right) \otimes \left(\Phi_{\mathsf{Bob}}\right)^*\left(\mathcal{X}_b\right)\right)\left(\psi^{\otimes n}\right)\right],$$

where $\left(\Phi_{\mathsf{Alice}}\right)^*$ and $\left(\Phi_{\mathsf{Bob}}\right)^*$ are the adjoints of Φ_{Alice} and Φ_{Bob}, respectively. Hence, Equation (1) is equivalent to

$$\mathrm{Tr}\left[\left(\left(\Phi_{\mathsf{Alice}}\right)^*\left(\mathcal{X}_a\right) \otimes \left(\Phi_{\mathsf{Bob}}\right)^*\left(\mathcal{X}_b\right)\right)\psi^{\otimes n}\right] \approx \mathrm{Tr}\left[\left(\left(\widetilde{\Phi_{\mathsf{Alice}}}\right)^*\left(\mathcal{X}_a\right) \otimes \left(\widetilde{\Phi_{\mathsf{Bob}}}\right)^*\left(\mathcal{X}_b\right)\right)\psi^{\otimes D}\right]. \tag{2}$$

Equation (2) resembles the setting considered in [26]. It is proved in [26] that for any POVM $\{M_i \otimes N_j\}_{i,j}$ acting on $\psi^{\otimes n}$, there exists POVM $\left\{M'_i \otimes N'_j\right\}_{i,j}$ acting on $\psi^{\otimes D}$ such that

$$\mathrm{Tr}\left[\left(M_i \otimes N_j\right)\psi^{\otimes n}\right] \approx \mathrm{Tr}\left[\left(M'_i \otimes N'_j\right)\psi^{\otimes D}\right],$$

for all i,j. However, $\left(\Phi_{\mathsf{Alice}}\right)^*\left(\mathcal{X}_a\right)$ and $\left(\Phi_{\mathsf{Bob}}\right)^*\left(\mathcal{X}_b\right)$ are not positive. It is even not clear how to characterize $\left(\Phi_{\mathsf{Alice}}\right)^*\left(\mathcal{X}_a\right)$ and $\left(\Phi_{\mathsf{Bob}}\right)^*\left(\mathcal{X}_b\right)$ for valid quantum operations Φ_{Alice} and Φ_{Bob}. Thus we cannot directly apply the results in [26]. Instead of working on each of $\left(\Phi_{\mathsf{Alice}}\right)^*\left(\mathcal{X}_a\right)$ and $\left(\Phi_{\mathsf{Bob}}\right)^*\left(\mathcal{X}_b\right)$, we work on the Choi representations $J\left(\left(\Phi_{\mathsf{Alice}}\right)^*\right)$ and $J\left(\left(\Phi_{\mathsf{Bob}}\right)^*\right)$, which include the information of $\left(\Phi_{\mathsf{Alice}}\right)^*\left(\mathcal{X}_a\right)$ and $\left(\Phi_{\mathsf{Bob}}\right)^*\left(\mathcal{X}_b\right)$ for all a,b. One more advantage of Choi representations is that we have a neat characterization of the Choi representations of quantum operations. Thus it is more convenient to bound the deviations of the intermediate super-operators from valid quantum operations throughout the construction. We consider the Fourier expansions of $J\left(\left(\Phi_{\mathsf{Alice}}\right)^*\right)$ and $J\left(\left(\Phi_{\mathsf{Bob}}\right)^*\right)$, and reduce the dimensions of the super-operators via the framework for the decidability of non-interactive simulations of joint distributions in [12, 11, 7, 26]. To this end, we prove an invariance principle for super-operators, and combine it with the dimension reduction for polynomials in Gaussian spaces [11]. There are several prerequisites for the invariance principle. Firstly, the Choi representation should have low degree. Secondly, all but a constant number of systems are of low influence, that is, all but a constant number of subsystems do not influence the super-operators much. The construction takes several steps to adjust the Fourier coefficients of $J\left(\left(\Phi_{\mathsf{Alice}}\right)^*\right)$ and $J\left(\left(\Phi_{\mathsf{Bob}}\right)^*\right)$ to meet those prerequisites. Meanwhile, the new super-operators still need to be close to valid quantum operations so that the value of the game does not change much. Once these prerequisites are satisfied, the basis elements in those subsystems with low influence are replaced by properly chosen Gaussian variables, which only causes a small deviation by the invariance principle.

Each step is sketched as follows.

1. **Smoothing**

   This step is aimed to obtain bounded-degree approximations of $J\left((\Phi_{\mathsf{Alice}})^*\right)$ and $J\left((\Phi_{\mathsf{Bob}})^*\right)$. We apply a noise operator Δ_γ for some $\gamma \in (0,1)$ defined in Definition 10 to both $J\left((\Phi_{\mathsf{Alice}})^*\right)$ and $J\left((\Phi_{\mathsf{Bob}})^*\right)$ on the input spaces. Note that both Choi representations are positive operators. After smoothing the operation and truncating the high-degree parts, we get bounded-degree approximations $M^{(1)}$ and $N^{(1)}$, of $J\left((\Phi_{\mathsf{Alice}})^*\right)$ and $J\left((\Phi_{\mathsf{Bob}})^*\right)$, respectively. Though the bounded-degree approximations may no longer be positive, the deviation can be proved to be small.

2. **Regularity**

   This step is aimed to prove that the number of subsystems having high influence is bounded. The influence of a subsystem of a multipartite Hermitian operator is defined in Definition 3. Informally speaking, the influence measures how much the subsystem can affect the operator. For a bounded operator, the total influence, which is the summation of the influences of all subsystems, is upper bounded by the degree of the operator. This is a generalization of a standard result in Boolean analysis. Note that we have bounded-degree approximations after the first step. The desired result follows by a Markov inequality.

3. **Invariance principle**

   In this step, we use correlated Gaussian variables to substitute the basis elements in all the subsystems with low influence in $M^{(1)}$ and $N^{(1)}$, after which we get random operators $\mathbf{M}^{(2)}$ and $\mathbf{N}^{(2)}$, whose Fourier coefficients are low-degree multilinear polynomials in Gaussian variables. We also need to prove that, $\mathbf{M}^{(2)}$ and $\mathbf{N}^{(2)}$ are close to positive operators in expectation.

4. **Dimension reduction**

   This step is aimed to reduce the number of Gaussian variables. After applying a dimension reduction to $\mathbf{M}^{(2)}$ and $\mathbf{N}^{(2)}$, we get random operators $\mathbf{M}^{(3)}$ and $\mathbf{N}^{(3)}$ containing a bounded number of Gaussian random variables. Unlike [26], we get an upper bound independent of the number of quantum subsystems via a more delicate analysis. However, the Fourier coefficients of $\mathbf{M}^{(3)}$ and $\mathbf{N}^{(3)}$ are no longer low-degree polynomials after the dimension reduction.

5. **Smooth random operators**

   The remaining steps are mainly concerned with removing the Gaussian variables. This step is aimed to get low-degree approximations of the Fourier coefficients of $\mathbf{M}^{(3)}$ and $\mathbf{N}^{(3)}$. We apply the Ornstein-Uhlenbeck operator (aka noise operators in Gaussian space) to the Gaussian variables in $\mathbf{M}^{(3)}$ and $\mathbf{N}^{(3)}$ and truncate the high-degree parts to get $\mathbf{M}^{(4)}$ and $\mathbf{N}^{(4)}$. We should note that the Fourier coefficients of $\mathbf{M}^{(4)}$ and $\mathbf{N}^{(4)}$ are polynomials, but not multilinear.

6. **Multilinearization**

   This step is aimed to get multilinear approximations of the Fourier coefficients of $\mathbf{M}^{(4)}$ and $\mathbf{N}^{(4)}$. To this end, We apply the multilinearization lemma in [11] to get random operators $\mathbf{M}^{(5)}$ and $\mathbf{N}^{(5)}$. Now we are ready to use the invariance principle again to convert random operators back to operators.

7. **Invariance to operators**

   In this step we substitute the Gaussian variables with properly chosen basis elements, to get operators $M^{(6)}$ and $N^{(6)}$, which have a bounded number of quantum subsystems. Again, we need to apply a quantum invariance principle to ensure that $M^{(6)}$ and $N^{(6)}$ are close to positive operators.

8. **Rounding**
   We now have operators $M^{(6)}$ and $N^{(6)}$ that are close to positive operators. The last thing to do is to round them to the Choi representations of the adjoints of some quantum operations. After the rounding, the whole construction is done.

2 Preliminary

Given $n \in \mathbb{Z}_{>0}$, let $[n]$ and $[n]_{\geq 0}$ represent the sets $\{1, \ldots, n\}$ and $\{0, \ldots, n-1\}$, respectively. For all $a \in \mathbb{Z}_{\geq 0}^n$, we define $|a| = |\{i : a_i > 0\}|$. In this paper, the lower-cased letters in bold $\mathbf{g}, \mathbf{h}, \ldots$ are reserved for random variables, and the capital letters in bold $\mathbf{M}, \mathbf{N}$ are reserved for random operators.

2.1 Quantum mechanics

We denote the set of Hermitian operators in a quantum system $\mathcal{S}$ by $\mathcal{H}_\mathcal{S}$. The identity operator is denoted by $\mathbb{1}_\mathcal{S}$. We use the shorthand $\mathcal{SA}$ to represent $\mathcal{S} \otimes \mathcal{A}$. The Hermitian space of the composition of n Hermitian space $\mathcal{H}_\mathcal{S}$ is denoted by $\mathcal{H}_\mathcal{S}^{\otimes n}$, or $\mathcal{H}_\mathcal{S}^n$ for short.

Given quantum systems $\mathcal{S}, \mathcal{A}$, let $\mathcal{L}(\mathcal{S}, \mathcal{A})$ denote the set of all linear maps from $\mathcal{M}_\mathcal{S}$ to $\mathcal{M}_\mathcal{A}$ A quantum operation from the input system $\mathcal{S}$ to the output system $\mathcal{A}$ is represented by a CPTP (completely positive and trace preserving) map $\Phi \in \mathcal{L}(\mathcal{S}, \mathcal{A})$. We define $\psi^\mathcal{S} = \mathrm{Tr}_\mathcal{A}\, \psi^{\mathcal{SA}}$ to represent the state obtained by tracing out system $\mathcal{A}$ from $\psi^{\mathcal{SA}}$.

For a given map $\Phi \in \mathcal{L}(\mathcal{S}, \mathcal{A})$, the adjoint of Φ is defined to be the unique map $\Phi^* \in \mathcal{L}(\mathcal{A}, \mathcal{S})$ that satisfies

$$\mathrm{Tr}\ \Phi^*(Q)^\dagger P = \mathrm{Tr}\ Q^\dagger \Phi(P) \quad \text{for all } P \in \mathcal{L}(\mathcal{S}) \text{ and } Q \in \mathcal{L}(\mathcal{A}). \tag{3}$$

Given $\Psi \in \mathcal{L}(\mathcal{A}, \mathcal{S})$, the Choi representation of Ψ is a linear map $J : \mathcal{L}(\mathcal{A}, \mathcal{S}) \to \mathcal{H}(\mathcal{SA})$ defined as follows:

$$J(\Psi) = \sum_a \Psi\left(\widetilde{\mathcal{A}_a}\right) \otimes \widetilde{\mathcal{A}_a}, \tag{4}$$

where $\widetilde{\mathcal{A}_a} = \mathcal{A}_a / \sqrt{|\mathcal{A}|}$[1], and $\left\{\mathcal{A}_a : a \in \left[|\mathcal{A}|^2\right]_{\geq 0}\right\}$ is an orthonormal basis in $\mathcal{A}$. J is a linear bijection. Ψ can be recovered from its Choi representation $J(\Psi)$ as follows.

$$\Psi(P) = \mathrm{Tr}_\mathcal{A}\left(J(\Psi)\left(\mathbb{1}_\mathcal{S} \otimes P^\dagger\right)\right). \tag{5}$$

▶ **Fact 2.** *$\Phi \in \mathcal{L}(\mathcal{S}, \mathcal{A})$ is a quantum operation if and only if $J(\Phi^*) \geq 0$ and $\mathrm{Tr}_\mathcal{A} J(\Phi^*) = \mathbb{1}_\mathcal{S}$.*

2.2 Fourier analysis in Gaussian space

Given $n \in \mathbb{Z}_{>0}$, let γ_n represent a standard n-dimensional normal distribution. A function $f : \mathbb{R}^n \to \mathbb{R}$ is in $L^2(\mathbb{R}, \gamma_n)$ if $\int_{\mathbb{R}^n} f(x)^2 \gamma_n(\mathrm{d}x) < \infty$.

All the functions involved in this paper are in $L^2(\mathbb{R}, \gamma_n)$. We equip $L^2(\mathbb{R}, \gamma_n)$ with an inner product $\langle f, g \rangle_{\gamma_n} = \mathbb{E}_{x \sim \gamma_n}[f(x)g(x)]$.

[1] The denominator is because of the demoninator in the definition of the inner product $\frac{1}{\mathsf{s}}\mathrm{Tr}\ P^\dagger Q$.

The set of *Hermite polynomials* forms an orthonormal basis in $L^2(\mathbb{R}, \gamma_1)$ with respect to the inner product $\langle \cdot, \cdot \rangle_{\gamma_1}$. The Hermite polynomials $H_r : \mathbb{R} \to \mathbb{R}$ for $r \in \mathbb{Z}_{\geq 0}$ are defined as

$$H_0(x) = 1; H_1(x) = x; H_r(x) = \frac{(-1)^r}{\sqrt{r!}} e^{x^2/2} \frac{\mathrm{d}^r}{\mathrm{d}x^r} e^{-x^2/2}.$$

For any $\sigma \in (\sigma_1, \ldots, \sigma_n) \in \mathbb{Z}^n_{\geq 0}$, define $H_\sigma : \mathbb{R}^n \to \mathbb{R}$ as $H_\sigma(x) = \prod_{i=1}^n H_{\sigma_i}(x_i)$.

The set $\left\{H_\sigma : \sigma \in \mathbb{Z}^n_{\geq 0}\right\}$ forms an orthonormal basis in $L^2(\mathbb{R}, \gamma_n)$. Every function $f \in L^2(\mathbb{R}, \gamma_n)$ has an *Hermite expansion* as

$$f(x) = \sum_{\sigma \in \mathbb{Z}^n_{\geq 0}} \widehat{f}(\sigma) \cdot H_\sigma(x),$$

where $\widehat{f}(\sigma)$'s are the *Hermite coefficients* of f, which can be obtained by $\widehat{f}(\sigma) = \langle H_\sigma, f \rangle_{\gamma_n}$.

We say $f \in L^2(\mathbb{R}, \gamma_n)$ is *multilinear* if $\widehat{f}(\sigma) = 0$ for $\sigma \notin \{0,1\}^n$.

▶ **Definition 3.** *The* influence *of the i-th coordinate(variable) on f, denoted by* $\mathrm{Inf}_i(f)$*, is defined by*

$$\mathrm{Inf}_i(f) = \mathop{\mathbb{E}}_{\mathbf{x} \sim \gamma_n} \left[\mathrm{Var}_{\mathbf{x}'_i \sim \gamma_1} [f(\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, \mathbf{x}'_i, \mathbf{x}_{i+1}, \ldots \mathbf{x}_n)] \right].$$

The following fact summarizes some basic properties of variance and influence.

▶ **Fact 4** ([25, Proposition 8.16 and Proposition 8.23])**.** *Given $f \in L^2(\mathbb{R}, \gamma_n)$, it holds that*

1. $\mathrm{Var}[f] = \sum_{\sigma \neq 0^n} \widehat{f}(\sigma)^2 \leq \sum_\sigma \widehat{f}(\sigma)^2 = \|f\|_2^2$.
2. $\mathrm{Inf}_i(f) = \sum_{\sigma_i \neq 0} \widehat{f}(\sigma)^2 \leq \mathrm{Var}[f]$.

2.3 Fourier analysis in matrix space

Given $1 \leq m, p \leq \infty$, and $H \in \mathcal{H}_m$, the p-norm of H is defined to be

$$\|H\|_p = \left(\mathrm{Tr} \left(H^2 \right)^{p/2} \right)^{1/p}.$$

The *normalized p-norm* of H is defined as

$$|||H|||_p = \left(\frac{1}{m} \mathrm{Tr} \left(H^2 \right)^{p/2} \right)^{1/p}.$$

Given $P, Q \in \mathcal{H}_m$, we define an inner product over $\mathbb{R}$:

$$\langle P, Q \rangle = \frac{1}{m} \mathrm{Tr}\, PQ.$$

We need the following particular classes of bases in $\mathcal{H}_m$ on which our Fourier analysis is based.

▶ **Definition 5.** *Let $\{\mathcal{B}_i\}_{i \in [m^2]_{\geq 0}}$ be an orthonormal basis in $\mathcal{H}_m$ over $\mathbb{R}$. We say $\{\mathcal{B}_i\}_{i \in [m^2]_{\geq 0}}$ is a standard orthonormal basis if $\mathcal{B}_0 = \mathbb{1}_m$.*

▶ **Fact 6.** *Let $\{\mathcal{B}_i\}_{i=0}^{m^2-1}$ be a standard orthonormal basis in $\mathcal{H}_m$. Then the set $\left\{\mathcal{B}_\sigma = \otimes_{i=1}^n \mathcal{B}_{\sigma_i} : \sigma \in [m^2]^n_{\geq 0}\right\}$ is a standard orthonormal basis in $\mathcal{H}_m^{\otimes n}$.*

Given a standard orthonormal basis $\{\mathcal{B}_i\}_{i=0}^{m^2-1}$ in $\mathcal{H}_m$, every $H \in \mathcal{H}_m^{\otimes n}$ has a Fourier expansion:

$$H = \sum_{\sigma \in [m^2]_{\geq 0}^n} \widehat{H}(\sigma) \mathcal{B}_\sigma,$$

where $\widehat{H}(\sigma) \in \mathbb{R}$ are the Fourier coefficients. The basic properties of $\widehat{H}(\sigma)$'s are summarized in the following fact, which can be easily derived from the orthonormality of $\{\mathcal{B}_\sigma\}_{\sigma \in [m^2]_{\geq 0}^n}$.

▶ **Fact 7** ([26, Fact 2.11]). *Given a standard orthonormal basis $\{\mathcal{B}_i\}_{i \in [m^2]_{\geq 0}}$ in $\mathcal{H}_m$ and $M, N \in \mathcal{H}_m$, it holds that*

1. $\langle M, N \rangle = \sum_\sigma \widehat{M}(\sigma) \widehat{N}(\sigma)$.
2. $\|\|M\|\|_2^2 = \langle M, M \rangle = \sum_\sigma \widehat{M}(\sigma)^2$.
3. $\langle \mathbb{1}_m, M \rangle = \widehat{M}(0)$.

▶ **Definition 8.** *Let $\mathcal{B} = \{\mathcal{B}_i\}_{i \in [m^2]_{\geq 0}}$ be a standard orthonormal basis in $\mathcal{H}_m$, $P, Q \in \mathcal{H}_m^{\otimes n}$*

1. *The degree of P is defined to be* $\deg P = \max\left\{|\sigma| : \widehat{P}(\sigma) \neq 0\right\}$.
2. *For any $i \in [n]$, the influence of i-th coordinate is defined to be*

$$\mathrm{Inf}_i(P) = \|\|P - \mathbb{1}_m \otimes \mathrm{Tr}_i P\|\|_2^2,$$

   where $\mathbb{1}_m$ is in the i'th quantum system, and Tr_i means tracing out the i'th system.
3. *The total influence of P is defined to be* $\mathrm{Inf}(P) = \sum_i \mathrm{Inf}_i(P)$.

▶ **Fact 9** ([26, Lemma 2.16]). *Given $P \in \mathcal{H}_m^{\otimes n}$, a standard orthonormal basis $\mathcal{B} = \{\mathcal{B}_i\}_{i \in [m^2]_{\geq 0}}$ in $\mathcal{H}_m$, it holds that*

1. $\mathrm{Inf}_i(P) = \sum_{\sigma : \sigma_i \neq 0} \left|\widehat{P}(\sigma)\right|^2$.
2. $\mathrm{Inf}(P) = \sum_\sigma |\sigma| \left|\widehat{P}(\sigma)\right|^2 \leq \deg P \cdot \|\|P\|\|_2^2$.

▶ **Definition 10.** *Given a quantum system $\mathcal{S}$ with dimension $|\mathcal{S}| = \mathsf{s}$, $\gamma \in [0,1]$, the depolarizing operation $\Delta_\gamma : \mathcal{H}_\mathcal{S} \to \mathcal{H}_\mathcal{S}$ is defined as follows. For any $P \in \mathcal{H}_\mathcal{S}$,*

$$\Delta_\gamma(P) = \gamma P + \frac{1-\gamma}{\mathsf{s}} (\mathrm{Tr}\ P) \cdot \mathbb{1}_\mathcal{S}.$$

▶ **Fact 11** ([26, Lemma 3.6 and Lemma 6.1]). *Given $n, m \in \mathbb{Z}_{>0}$, $\gamma \in [0,1]$, a standard orthonormal basis of $\mathcal{H}_m$: $\mathcal{B} = \{\mathcal{B}_i\}_{i=0}^{m^2-1}$, the following holds:*

1. *For any $P \in \mathcal{H}_m^{\otimes n}$ with a Fourier expansion $P = \sum_{\sigma \in [m^2]_{\geq 0}^n} \widehat{P}(\sigma) \mathcal{B}_\sigma$, it holds that*

$$\Delta_\gamma(P) = \sum_{\sigma \in [m^2]_{\geq 0}^n} \gamma^{|\sigma|} \widehat{P}(\sigma) \mathcal{B}_\sigma.$$

2. *For any $P \in \mathcal{H}_m^{\otimes n}$, $\|\|\Delta_\gamma(P)\|\|_2 \leq \|\|P\|\|_2$.*
3. *Δ_γ is a quantum operation.*
4. *For any $d \in \mathbb{Z}_{>0}, P \in \mathcal{H}_m^{\otimes n}$, it holds that $\left\|\left|(\Delta_\gamma(P))^{>d}\right|\right\|_2 \leq \gamma^d \|\|P\|\|_2$.*

▶ **Definition 12** (Maximal correlation). *[1] Given quantum systems $\mathcal{S}, \mathcal{T}$ with dimensions $\mathsf{s} = |\mathcal{S}|$ and $\mathsf{t} = |\mathcal{T}|$, $\psi^{\mathcal{S}\mathcal{T}} \in \mathcal{H}_{\mathcal{S}\mathcal{T}}$ with $\psi^\mathcal{S} = \mathbb{1}_\mathcal{S}/\mathsf{s}, \psi^\mathcal{T} = \mathbb{1}_\mathcal{T}/\mathsf{t}$, the maximal correlation of $\psi^{\mathcal{S}\mathcal{T}}$ is defined to be*

$$\rho\left(\psi^{\mathcal{S}\mathcal{T}}\right) = \sup\left\{ \begin{smallmatrix} \left|\mathrm{Tr}\left((P \otimes Q)\psi^{\mathcal{S}\mathcal{T}}\right)\right| : P \in \mathcal{H}_\mathcal{S}, Q \in \mathcal{H}_\mathcal{T}, \\ \mathrm{Tr}\ P = \mathrm{Tr}\ Q = 0, \|\|P\|\|_2 = \|\|Q\|\|_2 = 1 \end{smallmatrix} \right\}.$$

2.4 Random operators

In this subsection, we introduce random operators defined in [26], which unifies Gaussian variables and operators.

▶ **Definition 13** ([26]). *Given $p, h, n, m \in \mathbb{Z}_{>0}$, we say $\mathbf{P}$ is a random operator if it can be expressed as*

$$\mathbf{P} = \sum_{\sigma \in [m^2]_{\geq 0}^h} p_\sigma(\mathbf{g}) \mathcal{B}_\sigma,$$

where $\{\mathcal{B}_i\}_{i \in [m^2]_{\geq 0}}$ is a standard orthonormal basis in $\mathcal{H}_m$, $p_\sigma : \mathbb{R}^n \to \mathbb{R}$ for all $\sigma \in [m^2]_{\geq 0}^h$ and $\mathbf{g} \sim \gamma_n$. $\mathbf{P} \in L^p\left(\mathcal{H}_m^{\otimes h}, \gamma_n\right)$ if $p_\sigma \in L^p(\mathbb{R}, \gamma_n)$ for all $\sigma \in [m^2]_{\geq 0}^h$.

We say $\mathbf{P}$ is multilinear if $p_\sigma(\cdot)$ is multilinear for all $\sigma \in [m^2]_{\geq 0}^h$.

▶ **Fact 14** ([26, Lemma 2.23]). *Given $n, h, m \in \mathbb{Z}_{>0}$, let $\mathbf{P} \in L^2\left(\mathcal{H}_m^{\otimes h}, \gamma_n\right)$ with an associated vector-valued function p under a standard orthonormal basis. It holds that $\mathbb{E}\left[\|\mathbf{P}\|_2^2\right] = \|p\|_2^2$.*

2.5 Rounding maps

Define a function $\zeta : \mathbb{R} \to \mathbb{R}$ as follows.

$$\zeta(x) = \begin{cases} x^2 & \text{if } x \leq 0 \\ 0 & \text{otherwise} \end{cases}. \tag{6}$$

The function ζ measures the distance between an Hermitian operator and the set of positive semi-definite operators in 2-norm.

▶ **Fact 15** ([26, Lemma 9.1]). *Given $m \in \mathbb{Z}_{>0}$, $H \in \mathcal{H}_m$, it holds that*

$$\operatorname{Tr} \zeta(H) = \min\left\{\|H - X\|_2^2 : X \geq 0\right\}.$$

3 Main results

▶ **Theorem 16.** *Given $\epsilon \in (0,1)$, $n, s \in \mathbb{Z}_{>0}$, and quantum systems $\mathcal{P}, \mathcal{Q}, \mathcal{R}, \mathcal{S}, \mathcal{T}, \mathcal{A}, \mathcal{B}$ with dimensions $\mathsf{p} = |\mathcal{P}|, \mathsf{q} = |\mathcal{Q}|, \mathsf{r} = |\mathcal{R}|, \mathsf{s} = |\mathcal{S}|, \mathsf{t} = |\mathcal{T}|, \mathsf{a} = |\mathcal{A}|, \mathsf{b} = |\mathcal{B}|$. Let $\{\mathcal{A}_a\}_{a \in [\mathsf{a}^2]_{\geq 0}}$, $\{\mathcal{B}_b\}_{b \in [\mathsf{b}^2]_{\geq 0}}$, $\{\mathcal{R}_r\}_{r \in [\mathsf{r}^2]_{\geq 0}}$ be orthonormal bases in $\mathcal{H}_\mathcal{A}$, $\mathcal{H}_\mathcal{B}$ and $\mathcal{H}_\mathcal{R}$, respectively. Let $\psi^{\mathcal{ST}} \in \mathcal{H}_{\mathcal{ST}}$ be a noisy MES with the maximal correlation $\rho = \rho\left(\psi^{\mathcal{ST}}\right) < 1$, which is defined in Definition 12. Let $\phi_{in}^{\mathcal{PQR}} \in \mathcal{H}_{\mathcal{PQR}}$ be an arbitrary tripartite quantum state. Then there exists an explicitly computable $D = D(\rho, \epsilon, s, \mathsf{p}, \mathsf{q}, \mathsf{r}, \mathsf{s}, \mathsf{t}, \mathsf{a}, \mathsf{b})$, such that for all quantum operations $\Phi_{Alice} \in \mathcal{L}\left(\mathcal{S}^n\mathcal{P}, \mathcal{A}\right)$, $\Phi_{Bob} \in \mathcal{L}\left(\mathcal{T}^n\mathcal{Q}, \mathcal{B}\right)$, there exist quantum operations $\widetilde{\Phi_{Alice}} \in \mathcal{L}\left(\mathcal{S}^D\mathcal{P}, \mathcal{A}\right)$, $\widetilde{\Phi_{Bob}} \in \mathcal{L}\left(\mathcal{T}^D\mathcal{Q}, \mathcal{B}\right)$ such that for all $a \in \left[\mathsf{a}^2\right]_{\geq 0}$, $b \in \left[\mathsf{b}^2\right]_{\geq 0}$, $r \in \left[\mathsf{r}^2\right]_{\geq 0}$,* [2]

$$\left|\operatorname{Tr}\left[\left(\Phi_{Alice}^*\left(\widetilde{\mathcal{A}_a}\right) \otimes \Phi_{Bob}^*\left(\widetilde{\mathcal{B}_b}\right) \otimes \widetilde{\mathcal{R}_r}\right)\left(\phi_{in}^{\mathcal{PQR}} \otimes \left(\psi^{\mathcal{ST}}\right)^{\otimes n}\right)\right]\right.$$
$$\left. - \operatorname{Tr}\left[\left(\widetilde{\Phi_{Alice}^*}\left(\widetilde{\mathcal{A}_a}\right) \otimes \widetilde{\Phi_{Bob}^*}\left(\widetilde{\mathcal{B}_b}\right) \otimes \widetilde{\mathcal{R}_r}\right)\left(\phi_{in}^{\mathcal{PQR}} \otimes \left(\psi^{\mathcal{ST}}\right)^{\otimes D}\right)\right]\right| \leq \epsilon.$$

In particular, one may choose

$$D = \exp\left(\operatorname{poly}\left(\mathsf{a}, \mathsf{b}, \mathsf{p}, \mathsf{q}, \mathsf{r}, \log \mathsf{s}, \log \mathsf{t}, \frac{1}{1-\rho}, \frac{1}{\epsilon}\right)\right).$$

[2] Remind that $\widetilde{\mathcal{A}_a} = \mathcal{A}_a/\sqrt{\mathsf{a}}$, $\widetilde{\mathcal{B}_b} = \mathcal{B}_b/\sqrt{\mathsf{b}}$ and $\widetilde{\mathcal{R}_r} = \mathcal{R}_r/\sqrt{\mathsf{r}}$.

▶ **Theorem 17.** *Given parameters $0 < \epsilon, \rho < 1$, and a fully quantum nonlocal game*

$$\mathfrak{G} = \left(\mathcal{P}, \mathcal{Q}, \mathcal{R}, \mathcal{A}, \mathcal{B}, \phi_{in}, \left\{M^{\mathcal{ABR}}, \mathbb{1} - M^{\mathcal{ABR}}\right\}\right),$$

with dimensions $\mathsf{p} = |\mathcal{P}|, \mathsf{q} = |\mathcal{Q}|, \mathsf{r} = |\mathcal{R}|, \mathsf{s} = |\mathcal{S}|, \mathsf{t} = |\mathcal{T}|, \mathsf{a} = |\mathcal{A}|, \mathsf{b} = |\mathcal{B}|$, suppose the two players are restricted to share an arbitrarily finite number of noisy MES states $\psi^{\mathcal{ST}}$, i.e., $\psi^{\mathcal{S}} = \mathbb{1}_{\mathcal{S}}/\mathsf{s}$, $\psi^{\mathcal{T}} = \mathbb{1}_{\mathcal{T}}/\mathsf{t}$ with the maximal correlation $\rho < 1$ as defined in Definition 12. Let $\mathrm{val}_Q(\mathfrak{G}, \psi^{\mathcal{ST}})$ be the supremum of the winning probability that the players can achieve. Then there exists an explicitly computable bound $D = D(\rho, \epsilon, \mathsf{p}, \mathsf{q}, \mathsf{r}, \mathsf{s}, \mathsf{t}, \mathsf{a}, \mathsf{b})$ such that it suffices for the players to share D copies of $\psi^{\mathcal{ST}}$ to achieve the winning probability at least $\mathrm{val}_Q(\mathfrak{G}, \psi^{\mathcal{ST}}) - \epsilon$. In particular, one may choose

$$D = \exp\left(\mathrm{poly}\left(\mathsf{a}, \mathsf{b}, \mathsf{p}, \mathsf{q}, \mathsf{r}, \log \mathsf{s}, \log \mathsf{t}, \frac{1}{1-\rho}, \frac{1}{\epsilon}\right)\right).$$

Proof. To keep the notations short, the superscripts will be omitted whenever it is clear from the context. Suppose the players share n copies of $\psi^{\mathcal{ST}}$ and employ the strategy $(\Phi_{\mathsf{Alice}}, \Phi_{\mathsf{Bob}})$ with the winning probability $\mathrm{val}_Q(\mathfrak{G}, \psi^{\mathcal{ST}})$. We apply Theorem 16 to $(\Phi_{\mathsf{Alice}}, \Phi_{\mathsf{Bob}})$ with $\epsilon \leftarrow \epsilon/(\mathsf{abr})^{3/2}$ to obtain $\left(\widetilde{\Phi_{\mathsf{Alice}}}, \widetilde{\Phi_{\mathsf{Bob}}}\right)$. We claim that the strategy $\left(\widetilde{\Phi_{\mathsf{Alice}}}, \widetilde{\Phi_{\mathsf{Bob}}}\right)$ wins the game with probability at least $\mathrm{val}_Q(\mathfrak{G}, \psi^{\mathcal{ST}}) - \epsilon$.

Let $\{\mathcal{A}_a\}_{a \in [\mathsf{a}^2]_{\geq 0}}$, $\{\mathcal{B}_b\}_{b \in [\mathsf{b}^2]_{\geq 0}}$, $\{\mathcal{R}_r\}_{r \in [\mathsf{r}^2]_{\geq 0}}$ be orthonormal bases in $\mathcal{H}_{\mathcal{A}}$, $\mathcal{H}_{\mathcal{B}}$ and $\mathcal{H}_{\mathcal{R}}$, respectively. From Theorem 16, for all $a \in \left[\mathsf{a}^2\right]_{\geq 0}$, $b \in \left[\mathsf{b}^2\right]_{\geq 0}$, $r \in \left[\mathsf{r}^2\right]_{\geq 0}$, we have

$$\begin{aligned}\Big|\mathrm{Tr}\Big[\Big(\Phi^*_{\mathsf{Alice}}\left(\widetilde{\mathcal{A}_a}\right) \otimes \Phi^*_{\mathsf{Bob}}\left(\widetilde{\mathcal{B}_b}\right) \otimes \widetilde{\mathcal{R}_r}\Big)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes n}\right)\Big] \\ - \mathrm{Tr}\Big[\Big(\widetilde{\Phi^*_{\mathsf{Alice}}}\left(\widetilde{\mathcal{A}_a}\right) \otimes \widetilde{\Phi^*_{\mathsf{Bob}}}\left(\widetilde{\mathcal{B}_b}\right) \otimes \widetilde{\mathcal{R}_r}\Big)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes D}\right)\Big]\Big| \leq \epsilon/(\mathsf{abr})^{3/2}.\end{aligned}$$

By Equation (3), it is equivalent to

$$\begin{aligned}\Big|\mathrm{Tr}\Big[\left(\left(\Phi_{\mathsf{Alice}} \otimes \Phi_{\mathsf{Bob}}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes n}\right)\right)\left(\widetilde{\mathcal{A}_a} \otimes \widetilde{\mathcal{B}_b} \otimes \widetilde{\mathcal{R}_r}\right)\Big] \\ -\mathrm{Tr}\Big[\left(\left(\widetilde{\Phi_{\mathsf{Alice}}} \otimes \widetilde{\Phi_{\mathsf{Bob}}}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes D}\right)\right)\left(\widetilde{\mathcal{A}_a} \otimes \widetilde{\mathcal{B}_b} \otimes \widetilde{\mathcal{R}_r}\right)\Big]\Big| \leq \epsilon/(\mathsf{abr})^{3/2}.\end{aligned}$$

We finally get the desired result:

$$\begin{aligned}&\left|\mathrm{Tr}\left[M^{\mathcal{ABR}}\left(\left(\Phi_{\mathsf{Alice}} \otimes \Phi_{\mathsf{Bob}}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes n}\right) - \left(\widetilde{\Phi_{\mathsf{Alice}}} \otimes \widetilde{\Phi_{\mathsf{Bob}}}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes D}\right)\right)\right]\right| \\ \overset{(\star)}{\leq}\ & \left\|M^{\mathcal{ABR}}\right\| \cdot \left\|\left(\Phi_{\mathsf{Alice}} \otimes \Phi_{\mathsf{Bob}}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes n}\right) - \left(\widetilde{\Phi_{\mathsf{Alice}}} \otimes \widetilde{\Phi_{\mathsf{Bob}}}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes D}\right)\right\|_1 \\ \leq\ & (\mathsf{abr})^{1/2}\left\|\left(\Phi_{\mathsf{Alice}} \otimes \Phi_{\mathsf{Bob}}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes n}\right) - \left(\widetilde{\Phi_{\mathsf{Alice}}} \otimes \widetilde{\Phi_{\mathsf{Bob}}}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes D}\right)\right\|_2 \\ =\ & \Bigg(\mathsf{abr}\sum_{a,b,r}\Big(\mathrm{Tr}\Big[\left(\left(\Phi_{\mathsf{Alice}} \otimes \Phi_{\mathsf{Bob}}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes n}\right)\right)\left(\widetilde{\mathcal{A}_a} \otimes \widetilde{\mathcal{B}_b} \otimes \widetilde{\mathcal{R}_r}\right)\Big] \\ & -\mathrm{Tr}\Big[\left(\left(\widetilde{\Phi_{\mathsf{Alice}}} \otimes \widetilde{\Phi_{\mathsf{Bob}}}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes D}\right)\right)\left(\widetilde{\mathcal{A}_a} \otimes \widetilde{\mathcal{B}_b} \otimes \widetilde{\mathcal{R}_r}\right)\Big]\Big)^2\Bigg)^{1/2} \leq \epsilon,\end{aligned}$$

where $(\star)$ is by Hölder's inequality. ◀

3.1 Notations and setup

The proof of Theorem 16 involves a number of notations. To keep the proof succinct, we introduce the setup and the notations that are used frequently in the rest of the paper.

▶ **Setup 18.** *Given quantum systems* $\mathcal{P}, \mathcal{Q}, \mathcal{R}, \mathcal{S}, \mathcal{T}, \mathcal{A}, \mathcal{B}$ *with dimensions*

$$\mathsf{p} = |\mathcal{P}|, \mathsf{q} = |\mathcal{Q}|, \mathsf{r} = |\mathcal{R}|, \mathsf{s} = |\mathcal{S}|, \mathsf{t} = |\mathcal{T}|, \mathsf{a} = |\mathcal{A}|, \mathsf{b} = |\mathcal{B}|,$$

let $\phi_{in}^{\mathcal{PQR}}$ *be the input state in* $\mathcal{P} \otimes \mathcal{Q} \otimes \mathcal{R}$ *shared among Alice, Bob and the referee, where Alice, Bob and the referee hold* $\mathcal{P}$, $\mathcal{Q}$ *and* $\mathcal{R}$, *respectively. Let* $\psi^{\mathcal{ST}} \in \mathcal{H}_{\mathcal{ST}}$ *be the noisy MES shared between Alice and Bob, where Alice has* $\mathcal{S}$ *and Bob has* $\mathcal{T}$. *Let* $\rho < 1$ *be the maximal correlation of* $\psi^{\mathcal{ST}}$. *Let* $\mathcal{A}$ *and* $\mathcal{B}$ *be the answer registers of Alice and Bob, respectively.*

Let $\{\mathcal{S}_s\}_{s\in[\mathsf{s}^2]_{\geq 0}}, \{\mathcal{T}_t\}_{t\in[\mathsf{t}^2]_{\geq 0}}$ *be standard orthonormal bases in* $\mathcal{H}_{\mathcal{S}}, \mathcal{H}_{\mathcal{T}}$, *respectively. Let* $\{\mathcal{A}_a\}_{a\in[\mathsf{a}^2]_{\geq 0}}, \{\mathcal{B}_b\}_{b\in[\mathsf{b}^2]_{\geq 0}}, \{\mathcal{P}_p\}_{p\in[\mathsf{p}^2]_{\geq 0}}, \{\mathcal{Q}_q\}_{q\in[\mathsf{q}^2]_{\geq 0}}, \{\mathcal{R}_r\}_{r\in[\mathsf{r}^2]_{\geq 0}}$ *be orthonormal bases (not necessary to be standard orthonormal) in* $\mathcal{H}_{\mathcal{A}}, \mathcal{H}_{\mathcal{B}}, \mathcal{H}_{\mathcal{P}}, \mathcal{H}_{\mathcal{Q}}, \mathcal{H}_{\mathcal{R}}$, *respectively. For convenience, we denote* $\widetilde{\mathcal{A}_a}$ *to be* $\mathcal{A}_a/\sqrt{\mathsf{a}}$. *The same for* $\widetilde{\mathcal{B}_b}$, $\widetilde{\mathcal{P}_p}$, $\widetilde{\mathcal{Q}_q}$, $\widetilde{\mathcal{R}_r}$.

When we use universal quantifiers, we omit the ranges of the variables whenever they are clear in the context. For example, we say "for all a, b*" to mean "for all* $a \in [\mathsf{a}^2]_{\geq 0}$, $b \in [\mathsf{b}^2]_{\geq 0}$*".*

Given $M \in \mathcal{H}_{\mathcal{S}^n\mathcal{PA}}$, *for all* p, a, *we define* M_a *to be* $\mathrm{Tr}_{\mathcal{A}}\left[\left(\mathbb{1}_{\mathcal{S}^n\mathcal{P}} \otimes \widetilde{\mathcal{A}_a}\right) M\right]$, *and* $M_{p,a}$ *to be* $\mathrm{Tr}_{\mathcal{P}}\left[\left(\mathbb{1}_{\mathcal{S}^n} \otimes \widetilde{\mathcal{P}_p}\right) M_a\right]$. *Similar for* $N, N_b, N_{q,b}$. *In other words,*

$$M = \sum_{a\in[\mathsf{a}^2]_{\geq 0}} M_a \otimes \widetilde{\mathcal{A}_a}, \quad N = \sum_{b\in[\mathsf{b}^2]_{\geq 0}} N_b \otimes \widetilde{\mathcal{B}_b}. \tag{7}$$

and

$$M_a = \sum_{p\in[\mathsf{p}^2]_{\geq 0}} M_{p,a} \otimes \widetilde{\mathcal{P}_p}, \quad N_b = \sum_{q\in[\mathsf{q}^2]_{\geq 0}} N_{q,b} \otimes \widetilde{\mathcal{Q}_q}. \tag{8}$$

3.2 Proof of Theorem 16

Proof of Theorem 16. Let δ, θ be parameters which are chosen later. The proof is composed of several steps.

- **Smoothing**

  We apply a noise operator defined in Definition 10 to $J\left(\Phi^*_{\mathsf{Alice}}\right)$ and $J\left(\Phi^*_{\mathsf{Bob}}\right)$, and truncate the high-degree parts to get $M^{(1)}$ and $N^{(1)}$, respectively. [3] They satisfy the following.

1. For all a, b, $\left\|\left\|M_a^{(1)}\right\|\right\|_2 \leq 1$ and $\left\|\left\|N_b^{(1)}\right\|\right\|_2 \leq 1$, where $M_a^{(1)}$ and $N_b^{(1)}$ are defined in Equation (7).
2. For all a, b, r:

$$\left|\mathrm{Tr}\left[\left(\Phi^*_{\mathsf{Alice}}\left(\widetilde{\mathcal{A}_a}\right) \otimes \Phi^*_{\mathsf{Bob}}\left(\widetilde{\mathcal{B}_b}\right) \otimes \widetilde{\mathcal{R}_r}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes n}\right)\right] - \mathrm{Tr}\left[\left(M_a^{(1)} \otimes N_b^{(1)} \otimes \widetilde{\mathcal{R}_r}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes n}\right)\right]\right| \leq \delta.$$

[3] Readers may refer to the full version for details.

3. For all a, b, p, q, $M^{(1)}_{p,a}$ and $N^{(1)}_{q,b}$ have degree at most d_1, where $M^{(1)}_{p,a}$ and $N^{(1)}_{q,b}$ are defined in Equation (8).
4. $$\frac{1}{\mathsf{s}^n}\mathrm{Tr}\,\zeta\left(M^{(1)}\right) \le \delta \quad \text{and} \quad \frac{1}{\mathsf{t}^n}\mathrm{Tr}\,\zeta\left(N^{(1)}\right) \le \delta,$$
   where ζ is defined in Equation (6).
5. $M^{(1)}_0 = \mathbb{1}_{\mathcal{S}^n\mathcal{P}}/\sqrt{\mathsf{a}}$ and $N^{(1)}_0 = \mathbb{1}_{\mathcal{T}^n\mathcal{Q}}/\sqrt{\mathsf{b}}$.

Here $d_1 = O\left(\frac{\mathsf{a}^2\mathsf{b}^2\mathsf{pq}}{\delta(1-\rho)}\right)$.

- **Regularization**
  We denote $H \subseteq [n]$ to be the set of registers with high influence, satisfying $h = |H| \le d_1(\mathsf{a}+\mathsf{b})/\theta$ such that for all $i \notin H$:
  $$\mathrm{Inf}_i(M) \le \theta, \quad \mathrm{Inf}_i(N) \le \theta.$$
- **Invariance to random operators**
  Substituting the basis elements in the subsystems with low influence in $M^{(1)}$ and $N^{(1)}$, we obtain joint random operators $\mathbf{M}^{(2)}$ and $\mathbf{N}^{(2)}$ satisfying the following.

1. For all a, b, p, q:
   $$\mathbb{E}\left[\left|\left|\left|\mathbf{M}^{(2)}_{p,a}\right|\right|\right|_2^2\right]^{1/2} = \left|\left|\left|M^{(1)}_{p,a}\right|\right|\right|_2 \quad \text{and} \quad \mathbb{E}\left[\left|\left|\left|\mathbf{N}^{(2)}_{q,b}\right|\right|\right|_2^2\right]^{1/2} = \left|\left|\left|N^{(1)}_{q,b}\right|\right|\right|_2.$$
2. For all a, b, r:
   $$\mathbb{E}\left[\mathrm{Tr}\left[\left(\mathbf{M}^{(2)}_a \otimes \mathbf{N}^{(2)}_b \otimes \widetilde{\mathcal{R}_r}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes h}\right)\right]\right] = \mathrm{Tr}\left[\left(M^{(1)}_a \otimes N^{(1)}_b \otimes \widetilde{\mathcal{R}_r}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes n}\right)\right].$$
3. $$\left|\frac{1}{\mathsf{s}^h}\mathbb{E}\left[\mathrm{Tr}\,\zeta\left(\mathbf{M}^{(2)}\right)\right] - \frac{1}{\mathsf{s}^n}\mathrm{Tr}\,\zeta\left(M^{(1)}\right)\right| \le O\left(\mathsf{p}^{10/3}\mathsf{a}^4\left(3^{d_1}\mathsf{s}^{d_1/2}\sqrt{\theta}d_1\right)^{2/3}\right)$$
   and
   $$\left|\frac{1}{\mathsf{t}^h}\mathbb{E}\left[\mathrm{Tr}\,\zeta\left(\mathbf{N}^{(2)}\right)\right] - \frac{1}{\mathsf{t}^n}\mathrm{Tr}\,\zeta\left(N^{(1)}\right)\right| \le O\left(\mathsf{q}^{10/3}\mathsf{b}^4\left(3^{d_1}\mathsf{t}^{d_1/2}\sqrt{\theta}d_1\right)^{2/3}\right).$$
4. $\mathbf{M}^{(2)}_0 = \mathbb{1}_{\mathcal{S}^h\mathcal{P}}/\sqrt{\mathsf{a}}$ and $\mathbf{N}^{(2)}_0 = \mathbb{1}_{\mathcal{T}^h\mathcal{Q}}/\sqrt{\mathsf{b}}$.

- **Dimension Reduction**
  We then reducing the number of Gaussian variables in $\left(\mathbf{M}^{(2)}, \mathbf{N}^{(2)}\right)$ randomly. With probability at least $3/4 - \delta/2 > 0$, we get joint random operators $\left(\mathbf{M}^{(3)}, \mathbf{N}^{(3)}\right)$ such that the following holds:

1. For all a, b, p, q:
   $$\mathbb{E}\left[\left|\left|\left|\mathbf{M}^{(3)}_{p,a}\right|\right|\right|_2^2\right] \le (1+\delta)\,\mathbb{E}\left[\left|\left|\left|\mathbf{M}^{(2)}_{p,a}\right|\right|\right|_2^2\right] \quad \text{and} \quad \mathbb{E}\left[\left|\left|\left|\mathbf{N}^{(3)}_{q,b}\right|\right|\right|_2^2\right] \le (1+\delta)\,\mathbb{E}\left[\left|\left|\left|\mathbf{N}^{(2)}_{q,b}\right|\right|\right|_2^2\right].$$

2. $$\mathop{\mathbb{E}}_{\mathbf{x}}\left[\operatorname{Tr}\zeta\left(\mathbf{M}^{(3)}\right)\right] \le 8\mathop{\mathbb{E}}_{\mathbf{g}}\left[\operatorname{Tr}\zeta\left(\mathbf{M}^{(2)}\right)\right] \text{ and } \mathop{\mathbb{E}}_{\mathbf{y}}\left[\operatorname{Tr}\zeta\left(\mathbf{N}^{(3)}\right)\right] \le 8\mathop{\mathbb{E}}_{\mathbf{h}}\left[\operatorname{Tr}\zeta\left(\mathbf{N}^{(2)}\right)\right].$$

3. For all a, b, r:

$$\left|\mathop{\mathbb{E}}_{\mathbf{x},\mathbf{y}}\left[\operatorname{Tr}\left[\left(\mathbf{M}_a^{(3)}\otimes\mathbf{N}_b^{(3)}\otimes\widetilde{\mathcal{R}_r}\right)\left(\phi_{\mathsf{in}}\otimes\left(\psi^{\mathcal{S}\mathcal{T}}\right)^{\otimes h}\right)\right]\right] - \mathop{\mathbb{E}}_{\mathbf{g},\mathbf{h}}\left[\operatorname{Tr}\left[\left(\mathbf{M}_a^{(2)}\otimes\mathbf{N}_b^{(2)}\otimes\widetilde{\mathcal{R}_r}\right)\left(\phi_{\mathsf{in}}\otimes\left(\psi^{\mathcal{S}\mathcal{T}}\right)^{\otimes h}\right)\right]\right]\right| \le \delta.$$

4. $\mathbf{M}_0^{(3)} = \mathbb{1}_{\mathcal{S}^h\mathcal{P}}/\sqrt{\mathsf{a}}$ and $\mathbf{N}_0^{(3)} = \mathbb{1}_{\mathcal{T}^h\mathcal{Q}}/\sqrt{\mathsf{b}}$.

Here $n_0 = O\left(\frac{(\mathsf{abr})^{12}(\mathsf{pq})^{20}d_1^{O(d_1)}}{\delta^6}\right)$.

Smoothing random operators

To get low-degree approximations of the Fourier coefficients of $\mathbf{M}^{(3)}$ and $\mathbf{N}^{(3)}$, we obtain joint random operators $\left(\mathbf{M}^{(4)}, \mathbf{N}^{(4)}\right)$ satisfying the following.

1. For all a, b, p, q:

$$\deg\left(\mathbf{M}_{p,a}^{(4)}\right) \le d_2 \quad \text{and} \quad \deg\left(\mathbf{N}_{q,b}^{(4)}\right) \le d_2.$$

2. For all a, b, p, q:

$$\mathbb{E}\left[\left\|\left\|\mathbf{M}_{p,a}^{(4)}\right\|\right\|_2^2\right]^{1/2} \le \mathbb{E}\left[\left\|\left\|\mathbf{M}_{p,a}^{(3)}\right\|\right\|_2^2\right]^{1/2} \quad \text{and} \quad \mathbb{E}\left[\left\|\left\|\mathbf{N}_{q,b}^{(4)}\right\|\right\|_2^2\right]^{1/2} \le \mathbb{E}\left[\left\|\left\|\mathbf{N}_{q,b}^{(3)}\right\|\right\|_2^2\right]^{1/2}.$$

3. $$\mathbb{E}\left[\operatorname{Tr}\zeta\left(\mathbf{M}^{(4)}\right)\right] \le \mathbb{E}\left[\operatorname{Tr}\zeta\left(\mathbf{M}^{(3)}\right)\right] + \delta \text{ and } \mathbb{E}\left[\operatorname{Tr}\zeta\left(\mathbf{N}^{(4)}\right)\right] \le \mathbb{E}\left[\operatorname{Tr}\zeta\left(\mathbf{N}^{(3)}\right)\right] + \delta.$$

4. For all a, b, r:

$$\left|\mathbb{E}\left[\operatorname{Tr}\left[\left(\mathbf{M}_a^{(4)}\otimes\mathbf{N}_b^{(4)}\otimes\widetilde{\mathcal{R}_r}\right)\left(\phi_{\mathsf{in}}\otimes\psi^{\otimes h}\right)\right]\right] - \mathbb{E}\left[\operatorname{Tr}\left[\left(\mathbf{M}_a^{(3)}\otimes\mathbf{N}_b^{(3)}\otimes\widetilde{\mathcal{R}_r}\right)\left(\phi_{\mathsf{in}}\otimes\psi^{\otimes h}\right)\right]\right]\right| \le \delta.$$

5. $\mathbf{M}_0^{(4)} = \mathbb{1}_{\mathcal{S}^h\mathcal{P}}/\sqrt{\mathsf{a}}$ and $\mathbf{N}_0^{(4)} = \mathbb{1}_{\mathcal{T}^h\mathcal{Q}}/\sqrt{\mathsf{b}}$.

Here $d_2 = O\left(\frac{\mathsf{a}^2\mathsf{b}^2\mathsf{pq}}{\delta(1-\rho)}\right)$.

Multilinearization

Suppose that

$$\mathbf{M}_{p,a}^{(4)} = \sum_{s\in[\mathsf{s}^2]_{\ge 0}^h} m_{s,p,a}^{(4)}(\mathbf{x})\,\mathcal{S}_s \quad \text{and} \quad \mathbf{N}_{q,b}^{(4)} = \sum_{t\in[\mathsf{t}^2]_{\ge 0}^h} n_{t,q,b}^{(4)}(\mathbf{y})\,\mathcal{T}_t.$$

To get multilinear approximations of the Fourier coefficients of $\mathbf{M}^{(4)}$ and $\mathbf{N}^{(4)}$, we obtain multilinear random operators $\left(\mathbf{M}^{(5)}, \mathbf{N}^{(5)}\right)$ such that the following holds:

1. For all a, b, p, q, $\mathbf{M}^{(5)}_{p,a}$ and $\mathbf{N}^{(5)}_{q,b}$ are degree-d_2 multilinear random operators.
2. Suppose that

$$\mathbf{M}^{(5)}_{p,a} = \sum_{s \in [\mathsf{s}^2]^h_{\geq 0}} m^{(5)}_{s,p,a}(\mathbf{x})\, \mathcal{S}_s \quad \text{and} \quad \mathbf{N}^{(5)}_{q,b} = \sum_{t \in [\mathsf{t}^2]^h_{\geq 0}} n^{(5)}_{t,q,b}(\mathbf{y})\, \mathcal{T}_t,$$

   where $(\mathbf{x}, \mathbf{y}) \sim \mathcal{G}_\rho^{\otimes n_0 \cdot n_1}$. For all $(i,j) \in [n_0] \times [n_1], a, b, p, q, s, t$,

$$\mathrm{Inf}_{(i-1)n_1+j}\left(m^{(5)}_{s,p,a}\right) \leq \theta \cdot \mathrm{Inf}_i\left(m^{(4)}_{s,p,a}\right) \quad \text{and} \quad \mathrm{Inf}_{(i-1)n_1+j}\left(n^{(5)}_{t,q,b}\right) \leq \theta \cdot \mathrm{Inf}_i\left(n^{(4)}_{t,q,b}\right).$$

3. For all a, b:

$$\mathbb{E}\left[\left|\left|\left|\mathbf{M}^{(5)}_a\right|\right|\right|_2^2\right] \leq \mathbb{E}\left[\left|\left|\left|\mathbf{M}^{(4)}_a\right|\right|\right|_2^2\right] \quad \text{and} \quad \mathbb{E}\left[\left|\left|\left|\mathbf{N}^{(5)}_b\right|\right|\right|_2^2\right] \leq \mathbb{E}\left[\left|\left|\left|\mathbf{N}^{(4)}_b\right|\right|\right|_2^2\right].$$

4.

$$\frac{1}{\mathsf{s}^h}\left|\mathbb{E}\left[\mathrm{Tr}\,\zeta\left(\mathbf{M}^{(5)}\right)\right] - \mathbb{E}\left[\mathrm{Tr}\,\zeta\left(\mathbf{M}^{(4)}\right)\right]\right| \leq \delta$$

   and

$$\frac{1}{\mathsf{t}^h}\left|\mathbb{E}\left[\mathrm{Tr}\,\zeta\left(\mathbf{N}^{(5)}\right)\right] - \mathbb{E}\left[\mathrm{Tr}\,\zeta\left(\mathbf{N}^{(4)}\right)\right]\right| \leq \delta.$$

5. For all a, b, r:

$$\left|\mathbb{E}\left[\mathrm{Tr}\left[\left(\mathbf{M}^{(5)}_a \otimes \mathbf{N}^{(5)}_b \otimes \widetilde{\mathcal{R}_r}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes h}\right)\right]\right] - \mathbb{E}\left[\mathrm{Tr}\left[\left(\mathbf{M}^{(4)}_a \otimes \mathbf{N}^{(4)}_b \otimes \widetilde{\mathcal{R}_r}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes h}\right)\right]\right]\right| \leq \delta.$$

6. $\mathbf{M}^{(5)}_0 = \mathbb{1}_{\mathcal{S}^h \mathcal{P}}/\sqrt{\mathsf{a}}$ and $\mathbf{N}^{(5)}_0 = \mathbb{1}_{\mathcal{T}^h \mathcal{Q}}/\sqrt{\mathsf{b}}$.

Here $n_1 = O\left(\frac{\mathsf{a}^4\mathsf{b}^4\mathsf{p}^2\mathsf{q}^2 d_2^2}{\theta^2}\right)$.

Invariance to operators

Applying item 2 above, Fact 4 and Fact 14, we have

$$\sum_{s,p,a} \mathrm{Inf}_i\left(m^{(5)}_{s,p,a}\right) \leq \theta \cdot \mathsf{p} \cdot \mathsf{a} \cdot \mathbb{E}\left[\left|\left|\left|\mathbf{M}^{(4)}\right|\right|\right|_2^2\right].$$

Similarly, we have

$$\sum_{t,q,b} \mathrm{Inf}_i\left(n^{(5)}_{t,q,b}\right) \leq \theta \cdot \mathsf{q} \cdot \mathsf{b} \cdot \mathbb{E}\left[\left|\left|\left|\mathbf{N}^{(4)}\right|\right|\right|_2^2\right].$$

Let

$$\theta_0 = \max\left\{\theta\,\mathbb{E}\left[\left|\left|\left|\mathbf{M}^{(4)}\right|\right|\right|_2^2\right], \theta\,\mathbb{E}\left[\left|\left|\left|\mathbf{N}^{(4)}\right|\right|\right|_2^2\right]\right\}.$$

Substituting the Gaussian variables in $\left(\mathbf{M}^{(5)}, \mathbf{N}^{(5)}\right)$ with matrix basis elements to get $\left(M^{(6)}, N^{(6)}\right)$ satisfying that:

1. For all a, b, p, q:
$$\left\|\left\|M^{(6)}_{p,a}\right\|\right\|_2 = \mathbb{E}\left[\left\|\left\|\mathbf{M}^{(5)}_{p,a}\right\|\right\|_2^2\right]^{1/2} \quad \text{and} \quad \left\|\left\|N^{(6)}_{q,b}\right\|\right\|_2 = \mathbb{E}\left[\left\|\left\|\mathbf{N}^{(5)}_{q,b}\right\|\right\|_2^2\right]^{1/2}.$$
2. For all a, b, r:
$$\mathrm{Tr}\left[\left(M^{(6)}_a \otimes N^{(6)}_b \otimes \widetilde{\mathcal{R}_r}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes n_0 n_1 + h}\right)\right] = \mathbb{E}\left[\mathrm{Tr}\left[\left(\mathbf{M}^{(5)}_a \otimes \mathbf{N}^{(5)}_b \otimes \widetilde{\mathcal{R}_r}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes h}\right)\right]\right].$$
3. $$\left|\frac{1}{\mathsf{s}^{n_0 n_1 + h}}\mathrm{Tr}\,\zeta\left(M^{(6)}\right) - \frac{1}{\mathsf{s}^h}\mathbb{E}\left[\mathrm{Tr}\,\zeta\left(\mathbf{M}^{(5)}\right)\right]\right| \le O\left(\mathsf{p}^{10/3}\mathsf{a}^4\left(3^{d_2}\mathsf{s}^{d_2/2}\sqrt{\theta_0}d_2\right)^{2/3}\right)$$
and
$$\left|\frac{1}{\mathsf{t}^{n_0 n_1 + h}}\mathrm{Tr}\,\zeta\left(N^{(6)}\right) - \frac{1}{\mathsf{t}^h}\mathbb{E}\left[\mathrm{Tr}\,\zeta\left(\mathbf{N}^{(5)}\right)\right]\right| \le O\left(\mathsf{q}^{10/3}\mathsf{b}^4\left(3^{d_2}\mathsf{t}^{d_2/2}\sqrt{\theta_0}d_2\right)^{2/3}\right).$$
4. $M^{(6)}_0 = \mathbb{1}_{\mathcal{S}^{n_0 n_1 + h}\mathcal{P}}/\sqrt{\mathsf{a}}$ and $N^{(6)}_0 = \mathbb{1}_{\mathcal{T}^{n_0 n_1 + h}\mathcal{Q}}/\sqrt{\mathsf{b}}$.

Rounding

At last, we round $M^{(6)}$ and $N^{(6)}$ to the Choi representations of the adjoints of some quantum operations, $\widetilde{M}$ and $\widetilde{N}$, satisfying

$$\sum_a \left\|\left\|M^{(6)}_a - \widetilde{M_a}\right\|\right\|_2^2 = \mathsf{a}\cdot\left\|\left\|M^{(6)} - \widetilde{M}\right\|\right\|_2^2 \le O\left(\left(\frac{\mathsf{a}^7}{\mathsf{p}\mathsf{s}^{\mathsf{D}}}\mathrm{Tr}\,\zeta\left(M^{(6)}\right)\right)^{1/2}\right), \tag{9}$$

$$\sum_b \left\|\left\|N^{(6)}_b - \widetilde{N_b}\right\|\right\|_2^2 = \mathsf{b}\cdot\left\|\left\|N^{(6)} - \widetilde{N}\right\|\right\|_2^2 \le O\left(\left(\frac{\mathsf{b}^7}{\mathsf{q}\mathsf{t}^{\mathsf{D}}}\mathrm{Tr}\,\zeta\left(N^{(6)}\right)\right)^{1/2}\right). \tag{10}$$

Let $D = h + n_0 n_1$. Then

$$\begin{aligned}
&\left|\mathrm{Tr}\left[\left(M^{(6)}_a \otimes N^{(6)}_b \otimes \widetilde{\mathcal{R}_r} - \widetilde{M_a} \otimes \widetilde{N_b} \otimes \widetilde{\mathcal{R}_r}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes D}\right)\right]\right| \\
\le\ &\left|\mathrm{Tr}\left[\left(M^{(6)}_a \otimes \left(N^{(6)}_b - \widetilde{N_b}\right) \otimes \widetilde{\mathcal{R}_r}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes D}\right)\right]\right| \\
&+ \left|\mathrm{Tr}\left[\left(\left(M^{(6)}_a - \widetilde{M_a}\right) \otimes \widetilde{N_b} \otimes \widetilde{\mathcal{R}_r}\right)\left(\phi_{\mathsf{in}} \otimes \psi^{\otimes D}\right)\right]\right| \\
\le\ &(\mathsf{pq})^{1/2}\left(\left\|\left\|M^{(6)}_a\right\|\right\|_2\left\|\left\|N^{(6)}_b - \widetilde{N_b}\right\|\right\|_2 + \left\|\left\|M^{(6)}_a - \widetilde{M_a}\right\|\right\|_2\left\|\left\|\widetilde{N_b}\right\|\right\|_2\right) \\
\le\ &(\mathsf{pq})^{1/2}\left(\left\|\left\|M^{(6)}_a\right\|\right\|_2\left(\sum_b\left\|\left\|N^{(6)}_b - \widetilde{N_b}\right\|\right\|_2^2\right)^{1/2} + \left(\sum_a\left\|\left\|M^{(6)}_a - \widetilde{M_a}\right\|\right\|_2^2\right)^{1/2}\left\|\left\|\widetilde{N_b}\right\|\right\|_2\right) \\
\overset{(\star)}{\le}\ &\left\|\left\|M^{(6)}_a\right\|\right\|_2 O\left(\left(\frac{\mathsf{b}^7\mathsf{p}^2\mathsf{q}}{\mathsf{t}^D}\mathrm{Tr}\,\zeta\left(N^{(6)}\right)\right)^{1/4}\right) \\
&+ \left\|\left\|\widetilde{N_b}\right\|\right\|_2 O\left(\left(\frac{\mathsf{a}^7\mathsf{p}\mathsf{q}^2}{\mathsf{s}^D}\mathrm{Tr}\,\zeta\left(M^{(6)}\right)\right)^{1/4}\right),
\end{aligned}$$

where $(\star)$ is by Equation (9) and Equation (10).

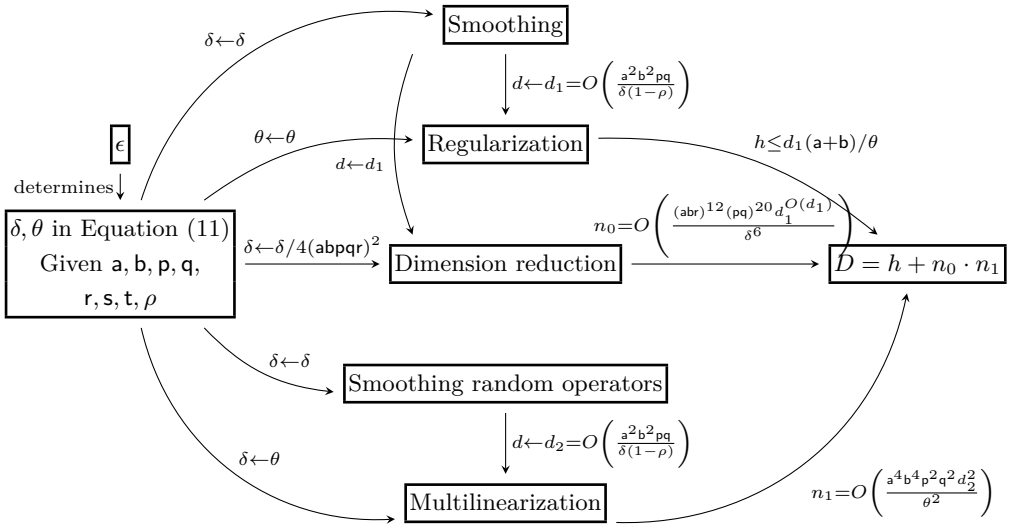


Figure 1 Dependency of parameters in the proof of Theorem 16.

Keeping track of the parameters in the construction, we are able to upper bound $\operatorname{Tr}\zeta\left(M^{(6)}\right)/\mathsf{s}^D$ and $\operatorname{Tr}\zeta\left(N^{(6)}\right)/\mathsf{t}^D$. Finally, by the triangle inequality we are able to upper bound

$$\left|\operatorname{Tr}\left[\left(\Phi^*_{\mathsf{Alice}}\left(\widetilde{\mathcal{A}_a}\right)\otimes\Phi^*_{\mathsf{Bob}}\left(\widetilde{\mathcal{B}_b}\right)\otimes\mathcal{R}_r\right)\left(\phi_{\mathsf{in}}\otimes\psi^{\otimes n}\right)\right]-\operatorname{Tr}\left[\left(\widetilde{M_a}\otimes\widetilde{N_b}\otimes\mathcal{R}_r\right)\left(\phi_{\mathsf{in}}\otimes\psi^{\otimes D}\right)\right]\right|$$

The dependency of the parameters is pictorially described in Figure 1.

We define $\Psi_{\mathsf{Alice}}\in\mathcal{L}\left(\mathcal{A},\mathcal{S}^D\mathcal{P}\right)$, $\Psi_{\mathsf{Bob}}\in\mathcal{L}\left(\mathcal{B},\mathcal{T}^D\mathcal{Q}\right)$ as follows:

$$\Psi_{\mathsf{Alice}}(X)=\operatorname{Tr}_{\mathcal{A}}\left(\widetilde{M}\left(\mathbb{1}_{\mathcal{S}^D\mathcal{P}}\otimes X^\dagger\right)\right),\quad \Psi_{\mathsf{Bob}}(Y)=\operatorname{Tr}_{\mathcal{B}}\left(\widetilde{N}\left(\mathbb{1}_{\mathcal{T}^D\mathcal{Q}}\otimes Y^\dagger\right)\right),$$

just as Equation (5). Let $\widetilde{\Phi_{\mathsf{Alice}}}=\Psi^*_{\mathsf{Alice}}$ and $\widetilde{\Phi_{\mathsf{Bob}}}=\Psi^*_{\mathsf{Bob}}$. Then by Fact 2, $\widetilde{\Phi_{\mathsf{Alice}}}$ and $\widetilde{\Phi_{\mathsf{Bob}}}$ are quantum operations. Furthermore,

$$\begin{aligned}&\operatorname{Tr}\left[\left(\left(\widetilde{\Phi_{\mathsf{Alice}}}\right)^*\left(\widetilde{\mathcal{A}_a}\right)\otimes\left(\widetilde{\Phi_{\mathsf{Bob}}}\right)^*\left(\widetilde{\mathcal{B}_b}\right)\otimes\widetilde{\mathcal{R}_r}\right)\left(\phi_{\mathsf{in}}\otimes\psi^{\otimes D}\right)\right]\\&\qquad=\operatorname{Tr}\left[\left(\widetilde{M_a}\otimes\widetilde{N_b}\otimes\widetilde{\mathcal{R}_r}\right)\left(\phi_{\mathsf{in}}\otimes\psi^{\otimes D}\right)\right].\end{aligned}$$

Choosing

$$\delta=O(\epsilon),\quad \theta=\frac{\epsilon^{12}}{\exp\left(\frac{\mathsf{a}^2\mathsf{b}^2\mathsf{pq}\log\mathsf{s}\log\mathsf{t}}{\epsilon(1-\rho)}\right)},\tag{11}$$

we finally conclude the result. ◀

References

1 Salman Beigi. A new quantum data processing inequality. *Journal of Mathematical Physics*, 54(8):082202, 2013. doi:10.1063/1.4818985.

2 Cyril Branciard, Denis Rosset, Yeong-Cherng Liang, and Nicolas Gisin. Measurement-device-independent entanglement witnesses for all entangled quantum states. *Phys. Rev. Lett.*, 110:060405, February 2013. doi:10.1103/PhysRevLett.110.060405.

3 Francesco Buscemi. All entangled quantum states are nonlocal. *Phys. Rev. Lett.*, 108:200401, May 2012. doi:10.1103/PhysRevLett.108.200401.

4 Eric G. Cavalcanti, Michael J. W. Hall, and Howard M. Wiseman. Entanglement verification and steering when alice and bob cannot be trusted. *Phys. Rev. A*, 87:032306, March 2013. doi:10.1103/PhysRevA.87.032306.

5 Kai-Min Chung, Xiaodi Wu, and Henry Yuen. Parallel Repetition for Entangled k-player Games via Fast Quantum Search. In David Zuckerman, editor, *30th Conference on Computational Complexity (CCC 2015)*, volume 33 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 512–536, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CCC.2015.512.

6 Richard Cleve, Peter Hoyer, Benjamin Toner, and John Watrous. Consequences and limits of nonlocal strategies. In *Proceedings of the 19th IEEE Annual Conference on Computational Complexity*, CCC '04, pages 236–249, Washington, DC, USA, 2004. IEEE Computer Society. doi:10.1109/CCC.2004.9.

7 Anindya De, Elchanan Mossel, and Joe Neeman. Non interactive simulation of correlated distributions is decidable. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 2728–2746, Philadelphia, PA, USA, 2018. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=3174304.3175478.

8 Payam Delgosha and Salman Beigi. Impossibility of local state transformation via hypercontractivity. *Communications in Mathematical Physics*, 332(1):449–476, November 2014. doi:10.1007/s00220-014-2105-y.

9 Joseph Fitzsimons, Zhengfeng Ji, Thomas Vidick, and Henry Yuen. Quantum proof systems for iterated exponential time, and beyond. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, New York, NY, USA, 2019. ACM.

10 Joseph Fitzsimons and Thomas Vidick. A multiprover interactive proof system for the local hamiltonian problem. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, ITCS '15, pages 103–112, New York, NY, USA, 2015. Association for Computing Machinery. doi:10.1145/2688073.2688094.

11 Badih Ghazi, Pritish Kamath, and Prasad Raghavendra. Dimension reduction for polynomials over gaussian space and applications. In *Proceedings of the 33rd Computational Complexity Conference*, CCC '18, pages 28:1–28:37, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CCC.2018.28.

12 Badih Ghazi, Pritish Kamath, and Madhu Sudan. Decidability of non-interactive simulation of joint distributions. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 545–554, Los Alamitos, CA, USA, October 2016. IEEE Computer Society. doi:10.1109/FOCS.2016.65.

13 Aram W. Harrow, Ashley Montanaro, and Anthony J. Short. Limitations on quantum dimensionality reduction. In Luca Aceto, Monika Henzinger, and Jiří Sgall, editors, *Automata, Languages and Programming*, pages 86–97, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

14 Tsuyoshi Ito and Thomas Vidick. A multi-prover interactive proof for NEXP sound against entangled provers. In *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, FOCS '12, pages 243–252, Washington, DC, USA, 2012. IEEE Computer Society. doi:10.1109/FOCS.2012.11.

15 Zhengfeng Ji. Classical verification of quantum proofs. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 885–898, New York, NY, USA, 2016. ACM. doi:10.1145/2897518.2897634.

16 Zhengfeng Ji, Anand Natarajan, Thomas Vidick, John Wright, and Henry Yuen. MIP* = RE. *arXiv preprint*, 2020. arXiv:2001.04383.

17 Zhengfeng Ji, Anand Natarajan, Thomas Vidick, John Wright, and Henry Yuen. Quantum soundness of the classical low individual degree test. *arXiv preprint*, 2020. arXiv:2009.12982.

18 Nathaniel Johnston, Rajat Mittal, Vincent Russo, and John Watrous. Extended non-local games and monogamy-of-entanglement games. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 472(2189):20160003, May 2016. doi:10.1098/rspa.2016.0003.

19 S. Kamath and V. Anantharam. On non-interactive simulation of joint distributions. *IEEE Transactions on Information Theory*, 62(6):3419–3435, June 2016. doi:10.1109/TIT.2016.2553672.

20 J. Kempe, O. Regev, and B. Toner. Unique games with entangled provers are easy. *SIAM Journal on Computing*, 39(7):3207–3229, 2010. doi:10.1137/090772885.

21 Julia Kempe, Hirotada Kobayashi, Keiji Matsumoto, Ben Toner, and Thomas Vidick. Entangled games are hard to approximate. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '08, pages 447–456, Washington, DC, USA, 2008. IEEE Computer Society. doi:10.1109/FOCS.2008.8.

22 Debbie Leung, Ben Toner, and John Watrous. Coherent state exchange in multi-prover quantum interactive proof systems. *Chicago Journal of Theoretical Computer Science*, 2013(11), August 2013.

23 Elchanan Mossel, Ryan O'Donnell, and Krzysztof Oleszkiewicz. Noise stability of functions with low influences: Invariance and optimality. *Annals of Mathematics*, 171:295–341, March 2010.

24 A. Natarajan and J. Wright. NEEXP is contained in MIP. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 510–518, Los Alamitos, CA, USA, November 2019. IEEE Computer Society. doi:10.1109/FOCS.2019.00039.

25 Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, Cambridge, UK, 2013.

26 Minglong Qin and Penghui Yao. Nonlocal games with noisy maximally entangled states are decidable. *SIAM Journal on Computing*, 50(6):1800–1891, 2021.

27 Oded Regev and Thomas Vidick. Quantum XOR games. *ACM Trans. Comput. Theory*, 7(4), August 2015. doi:10.1145/2799560.

28 C. J. Stark and A. W. Harrow. Compressibility of positive semidefinite factorizations and quantum models. *IEEE Transactions on Information Theory*, 62(5):2867–2880, 2016. doi:10.1109/TIT.2016.2538278.

29 John Watrous. *Theory of Quantum Information*. Cambridge University Press, Cambridge, UK, 2018.

Scheduling Under Non-Uniform Job and Machine Delays

Rajmohan Rajaraman
Northeastern University, Boston, MA, USA

David Stalfa
Northeastern University, Boston, MA, USA

Sheng Yang
Shanghai, CN

Abstract

We study the problem of scheduling precedence-constrained jobs on heterogenous machines in the presence of non-uniform job and machine communication delays. We are given a set of n unit size precedence-ordered jobs, and a set of m related machines each with size m_i (machine i can execute at most m_i jobs at any time). Each machine i has an associated in-delay ρ_i^{in} and out-delay ρ_i^{out}. Each job v also has an associated in-delay ρ_v^{in} and out-delay ρ_v^{out}. In a schedule, job v may be executed on machine i at time t if each predecessor u of v is completed on i before time t or on any machine j before time $t - (\rho_i^{\text{in}} + \rho_j^{\text{out}} + \rho_u^{\text{out}} + \rho_v^{\text{in}})$. The objective is to construct a schedule that minimizes makespan, which is the maximum completion time over all jobs.

We consider schedules which allow duplication of jobs as well as schedules which do not. When duplication is allowed, we provide an asymptotic polylog(n)-approximation algorithm. This approximation is further improved in the setting with uniform machine speeds and sizes. Our best approximation for non-uniform delays is provided for the setting with uniform speeds, uniform sizes, and no job delays. For schedules with no duplication, we obtain an asymptotic polylog(n)-approximation for the above model, and a true polylog(n)-approximation for symmetric machine and job delays. These results represent the first polylogarithmic approximation algorithms for scheduling with non-uniform communication delays.

Finally, we consider a more general model, where the delay can be an arbitrary function of the job and the machine executing it: job v can be executed on machine i at time t if all of v's predecessors are executed on i by time $t-1$ or on any machine by time $t - \rho_{v,i}$. We present an approximation-preserving reduction from the Unique Machines Precedence-constrained Scheduling (UMPS) problem, first defined in [15], to this job-machine delay model. The reduction entails logarithmic hardness for this delay setting, as well as polynomial hardness if the conjectured hardness of UMPS holds.

This set of results is among the first steps toward cataloging the rich landscape of problems in non-uniform delay scheduling.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

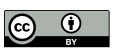
Keywords and phrases Scheduling, Approximation Algorithms, Precedence Constraints, Communication Delay, Non-Uniform Delays

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.98

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2207.13121 [42]

Funding *Rajmohan Rajaraman*: Partially supported by NSF grant CCF-1909363.
David Stalfa: Partially supported by NSF grant CCF-1909363.
Sheng Yang: Work done when the author was at Northwestern University, supported by Samir Khuller's funding.



1 Introduction

With the increasing scale and complexity of scientific and data-intensive computations, it is often necessary to process workloads with many dependent jobs on a network of heterogeneous computing devices with varying computing capabilities and communication delays. For instance, the training and evaluation of neural network models, which involves iterations of precedence constrained jobs, is often distributed over diverse devices such as CPUs, GPUs, or other specialized hardware. This process, commonly referred to as *device placement*, has gained significant interest [18, 21, 32, 33]. Similarly, many scientific workflows are best modeled precedence constrained jobs, and the underlying high-performance computing system as a heterogeneous networked distributed system with communication delays [3, 44, 49].

Optimization problems associated with scheduling under communication delays have been studied extensively, but provably good approximation bounds are few and several challenging open problems remain [1, 4, 14, 23, 26, 34, 35, 37, 39, 40, 43]. With a communication delay, scheduling a set of precedence constrained uniform size jobs on identical machines is already NP-hard [40, 43], and several inapproximability results are known [4, 23]. However, the field is still underexplored and scheduling under communication delay was listed as one of the top ten open problems in scheduling surveys [5, 45]. While there has been progress on polylogarithmic-approximation algorithms for the case of uniform communication delays [16, 26, 29, 31], little is known for more general delay models.

This paper considers the problem of scheduling precedence-constrained jobs on machines connected by a network with *non-uniform* communication delays. In general, the delay incurred in communication between two machines could vary with the machines as well as with the data being communicated, which in turn may depend on the jobs being excuted on the machines. For many applications, however, simpler models suffice. For instance, the machine delays model, where the communication between two machines incurs a delay given by the sum of latencies associated with the two machines, is suitable when the bottleneck is primarily at the machine interfaces. On the other hand, job delays model scenarios where the delay incurred in the communication between two jobs running on two different machines is a function primarily of the two jobs. This is suitable when the communication is data-intensive. Recent work in [15] presents a hardness result for a model in which jobs are given as a DAG and any edge of the DAG separating two jobs running on different machines causes a delay, providing preliminary evidence that obtaining sub-polynomial approximation factors for this model may be intractable. Given polylogarithmic approximations for uniform delays, a natural question is which, if any, non-uniform delay models are tractable.

1.1 Overview of our results

A central contribution of this paper is to explore and catalog a rich landscape of problems in non-uniform delay scheduling. We present polylogarithmic approximation algorithms for several models with non-uniform delays, and a hardness result in the mold of [15] for a different non-uniform delay model. Figure 2 organizes various models in this space, with pointers to results in this paper and relevant previous work.

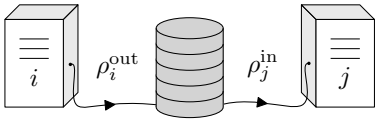


Figure 1 Communicating a result from i to j takes $\rho_i^{\text{out}} + \rho_j^{\text{in}}$ time.

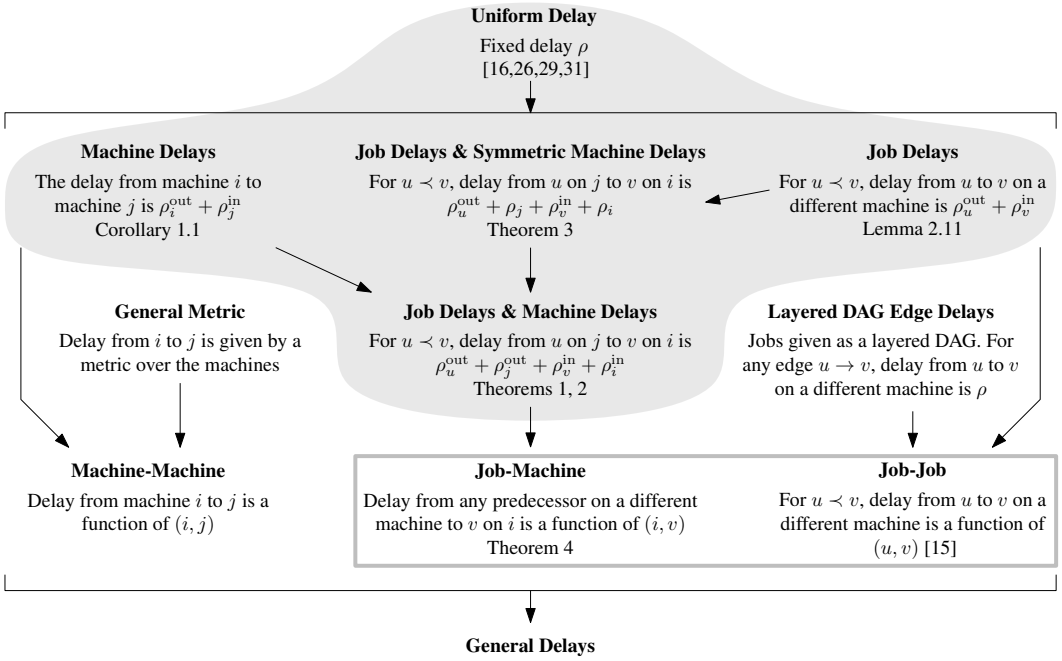


Figure 2 Selection of scheduling models with communication delays. $a \twoheadrightarrow b$ indicates that a is a special case of b. We present approximation algorithms for models with machine delays and job delays, and a hardness of approximation result for the job-machine delays model. Theorems and citations point to results in this paper and in previous work, respectively. Those problems backed in gray are ones for which approximation algorithms are known. Those in the gray box are ones for which hardness results have been proven.

Machine delays and job delays (Section 2). We begin with a natural model where the delay incurred in communication from one machine to another is the sum of delays at the two endpoints. Under machine delays, each machine i has an in-delay ρ_i^{in} and out-delay ρ_i^{out}, and the time taken to communicate a result from i to j is $\rho_i^{\text{out}} + \rho_j^{\text{in}}$. This model, illustrated in Figure 1, is especially suitable for environments where data exchange between jobs occurs via the cloud, an increasingly common mode of operation in modern distributed systems [28,30,50]; ρ_i^{in} and ρ_i^{out} represent the cloud download and upload latencies, respectively, for machine i.

The machine delays model does not account for heterogeneity among jobs, where different jobs may be producing or consuming different amounts of data, which may impact the delay between the processing of one job and that of another dependent job on a different machine. To model this, we allow each job u to have an in-delay ρ_u^{in} and an out-delay ρ_u^{out}.

▶ **Definition 1** (Scheduling under Machine Delays and Job Delays). *We are given as input a set of n precedence ordered jobs and a set of m machines. For any jobs u and v with $u \prec v$, machine i, and time t, u is available to v on i at time t if u is completed on i before time t or on any machine j before time $t - (\rho_j^{\text{out}} + \rho_u^{\text{out}} + \rho_i^{\text{in}} + \rho_v^{\text{in}})$. (This model is illustrated in Figure 3.) If job v is scheduled at time t on machine i, then all of its predecessors must be available to v on i at time t. We define $\rho_{\max} = \max_{x \in V \cup M}\{\rho_x^{\text{in}} + \rho_x^{\text{out}}\}$. The objective is to construct a schedule that minimizes makespan.*

Remark. In our model of Definition 1, communication delay is defined over all pairs of precedence ordered jobs. An alternate model defines communication delay only over those pairs that are adjacent in the job DAG. The two settings differ in general but are equivalent

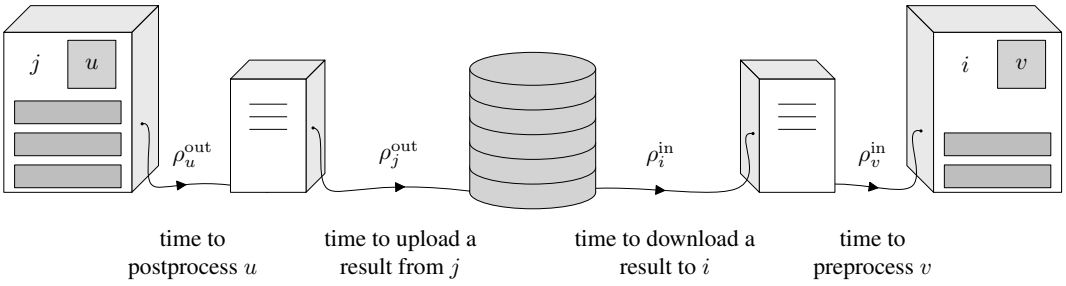


Figure 3 Communicating the result of job u on machine j to execute job v on machine i.

in many scenarios, for instance, when the delays are given by an underlying metric space over the machines, or when communication delays are uniform. The models are equivalent if all delays are machine delays, so our machine delay results hold in the alternate model. The models differ in the presence of general job delays but are equivalent in several special cases, for instance in the setting where the job DAG is transitively closed, which has been extensively studied and proved useful in several important applications [2,19,46]. Transitively closed DAGs capture scenarios where each job may be generating data used by upstream jobs, and an upstream job may need to check the results of any of its predecessors. Examples of such graphs arising in scheduling include interval orders [38], as well as Solution Order Graphs in the context of SAT solvers [8].

We present the first approximation algorithms for scheduling under non-uniform communication delays. In the presence of delays, a natural approach to hide latency and reduce makespan is to duplicate some jobs (for instance, a job that is a predecessor of many other jobs) [1,39]. We consider both schedules that allow duplication (which we assume by default) and those that do not. Our first result is a polylogarithmic asymptotic approximation for scheduling under machine and job delays when duplication is allowed.

▶ **Theorem 1.** There exists a polynomial time algorithm for scheduling unit length, precedence constrained jobs with duplication under machine and job delays, that produces a schedule with makespan $O((\log^9 n)(\text{OPT} + \rho_{\max}))$.

We emphasize that if the makespan of any schedule includes the delays incurred in distributing the problem instance and collecting the output of the jobs, then the algorithm of Theorem 1 is, in fact, a *true polylogarithmic approximation* for makespan. (From a practical standpoint, in order to account for the time incurred to distribute the jobs and collect the results, it is natural to include in the makespan the in- and out-delays of every machine used in the schedule.)

We note that when delays are uniform and duplication is not allowed, it is easy to check if $\text{OPT} < \rho$ since any connected component of the job DAG must be placed on the same machine. This is demonstrated in our true approximation without duplication in Theorem 3. In the presence of duplication, the problem is closely related to the Min k-Union problem, for which conditional hardness proofs are known [12]. This motivates the additive $\rho_{\max}$ in our approximation guarantee.

Related machines and multiprocessors. Theorem 1 is based on a new linear programming framework for addressing non-uniform job and machine delays. We demonstrate the power and flexibility of this approach by incorporating two more aspects of heterogeneity: speed and number of processors. Each machine i has a number m_i of processors and a speed s_i at which each processor processes jobs. We generalize Theorem 1 to obtain the following result.

▶ **Theorem 2.** There exists a polynomial time algorithm for scheduling unit length, precedence constrained jobs with duplication on related multiprocessor machines under machine and job delays, that yields a schedule with makespan $\text{polylog}(n)(\text{OPT} + \rho_{\max})$.

The exact approximation factor obtained depends on the non-uniformity of the particular model. For the most general model we consider in Theorem 2, our proof achieves a $O(\log^{15} n)$ bound. We obtain improved bounds when any of the three defining parameters – size, speed, and delay – are uniform. For instance, we obtain an approximation factor of $O(\log^5 n)$ for scheduling uniform speed and uniform size machines under machine delays alone, i.e., when there are no job delays (Corollary 12 of Section 2). Further, with only job delays and uniform machine delays, we provide a combinatorial asymptotic $O(\log^6 n)$ approximation (Lemma 15 of Section 2) which is improved to an asymptotic $O(\log n)$ approximation if the input contains no out-delays. We note that despite some uniformity, special cases can model certain two-level non-uniform network hierarchies with processors at the leaves, low delays at the first level, and high delays at the second level.

No-duplication schedules. We next consider the problem of designing schedules that do not allow duplication. We obtain a polylogarithmic asymptotic approximation via a reduction to scheduling with duplication. Furthermore, if the delays are symmetric (i.e., $\rho_i^{\text{out}} = \rho_i^{\text{in}}$ for all i, and $\rho_v^{\text{out}} = \rho_v^{\text{in}}$ for all v) we are able to find a *true* polylogarithmic-approximate no-duplication schedule. To achieve this result, we present an approximation algorithm to estimate if the makespan of an optimal no-duplication schedule is at least the delay of any given machine; this enables us to identify machines that cannot communicate in the desired schedule.[1]

▶ **Theorem 3.** There exists a polynomial time algorithm for scheduling unit length, precedence constrained jobs on related multiprocessor machines under machine delays and job delays, which produces a no-duplication schedule with makespan $\text{polylog}(n)(\text{OPT} + \rho_{\max})$. If $\rho_i^{\text{in}} = \rho_i^{\text{out}}$ for all i, then there exists a polynomial time $\text{polylog}(n)$-approximation algorithm for no-duplication schedules.

Pairwise delays. All of the preceding results concern models where the communication associated with a precedence relation $u \prec v$ when u and v are executed on different machines i and j is an *additive* combination of delays at u, v, i, and j. Additive delays are suitable for capturing independent latencies incurred by various components of the system. A more general class of models considers *pairwise* delays where the delay is an *arbitrary function* of i and j (machine-machine), u and v (job-job), or either job and the machine on which it executes (job-machine). The machine-machine delay model captures classic networking scenarios, where the delay across machines is determined by the network links connecting them. Job-job delays model applications where the data that needs to be communicated from one job to another descendant job depends arbitrarily on the two jobs. The job-machine model is well-suited for applications where the delay incurred for communicating the data consumed or produced by a job executing on a machine is an arbitrary function of the size of

[1] We note that the corresponding problem for duplication schedules is a min-max partitioning variant of the Minimum k-Union problem and related to the Min-Max Hypergraph k-Partitioning problem, both of which have been shown to be Densest-k-Subgraph-hard [9, 11]; this might suggest a similar hardness result for deriving a *true* approximation when duplication is allowed.

the data and the bandwidth of the machine. Recent work in [15] shows that scheduling under job-job delays is as hard as the Unique Machine Precedence Scheduling (UMPS) problem, providing preliminary evidence that obtaining sub-polynomial approximation factors may be intractable. We show that UMPS also reduces to scheduling under job-machine delays, suggesting a similar inapproximability for this model.

▶ **Theorem 4 (umps reduces to scheduling under job-machine delays).** There is a polynomial-time approximation-preserving reduction from UMPS to the scheduling precedence constrained jobs under job-machine delays.

1.2 Overview of our techniques

Our approximation algorithms for scheduling under job delays and machine delays (Theorem 1 proved in Section 2) and the generalization to related machines and multiprocessors (Theorem 2 proved in [42]) rely on a framework composed of a carefully crafted linear programming relaxation and a series of reductions that help successively reduce the level of heterogeneity in the problem. While each individual component of the framework refines established techniques or builds on prior work, taken together they offer a flexible recipe for designing approximation algorithms for scheduling precedence-ordered jobs on a distributed system of heterogeneous machines with non-uniform delays. Given the hardness conjectures of [15] for the job-job delay setting (and for the job-machine setting via Theorem 4), we find it surprising that a fairly general model incorporating both job delays and machine delays on related machines is tractable.

Previous results on scheduling under (uniform) communication delays are based on three different approaches: (a) a purely combinatorial algorithm of [26] that works only for uniform delay machines; (b) an LP-based approach of [31] that handles related machines and uniform delays, assuming jobs can be duplicated, and then extends to no-duplication via a reduction; and (c) an approach of [16] based on a Sherali-Adams hierarchy relaxation followed by a semi-metric clustering, which directly tackles the no-duplication model. At a very high level, our main challenge, which is not addressed in any of the previous studies, is to tackle the *multi-dimensional heterogeneity* of the problem space: in the nature of delays (non-uniform values, in- and out-delays, job delays, machine delays) as well as the machines (delay, speed, and size).

We pursue an LP-based framework, which significantly refines the approach of [31]. Their algorithm organizes the computation in phases, each phase corresponding to a (uniform) delay period, and develops a linear program that includes delay constraints capturing when jobs have to be phase-separated and phase constraints bounding the amount of computation within a phase. In non-uniform delay models, the delay constraints for a job v executing on a machine i depend not only on the predecessors of v, but also on the machines on which they may be scheduled. While there is a natural way to account for non-uniform in-delays in the LP, incorporating out-delays or even symmetric delays poses technical difficulties. We overcome this hurdle by first showing that out-delays can be eliminated by suitably adjusting in-delays, at the expense of a polylogarithmic factor in approximation, thus allowing us to focus on in-delays.

Despite the reduction to in-delays, extending the LP of [31] by replacing the uniform delay parameter by the non-uniform delay parameters of our models fails and yields a high integrality gap. This is because their algorithm crucially relies on an ordering of the machines (on the basis of their speeds), which is exploited both in the LP (in the delay and phase constraints) as well as how jobs get assigned and moved in the computation of the final

schedule. Given the multi-dimensional heterogeneity of the problems we study, there is no such natural ordering of the machines. To address the above hurdle, we organize the machines and jobs into groups based on their common characteristics (delay, speed, size), and introduce new variables for assigning jobs to groups without regard to any ordering among them. This necessitates new load and delay constraints and a change in rounding and schedule construction. We now elaborate on these ideas, as we discuss our new framework in more detail.

Reduction to in-delays. The first ingredient of our recipe is an argument that any instance of the problem with machine delays and job delays can be reduced to an instance in which all out-delays are 0, meaning that in the new instance delays depend only on the machine and job receiving the data, at the expense of a polylogarithmic factor in approximation. This reduction is given in Lemma 37 and Algorithm 2 in [42]. To convert from a given schedule with out-delays to one without, we subtract $\rho_i^{\text{out}} + \rho_v^{\text{out}}$ from the execution time of every job v on machine i. However, in order to avoid collisions, we expand the given schedule into phases of different length, organized in particular sequence so that the execution times within each phase may be reduced without colliding with prior phases. This transforms the schedule into one where the in-delay of every machine i is $\rho_i^{\text{in}} + \rho_i^{\text{out}}$ and every job v is $\rho_v^{\text{in}} + \rho_v^{\text{out}}$. This transformation comes at a constant factor cost for machine delays and an $O(\log^2 \rho_{max})$ cost for job delays. A similar procedure converts from an in-delay schedule to one with in- and out-delays, completing the desired reduction.

The linear program (Sections 2.1–2.2). Before setting up the linear program, we partition the machines and the jobs into groups of uniform machines and jobs, respectively; i.e. each machine in a group can be treated as having the same in-delay, speed, and size (to within a constant factor), and each job in a group can be treated as having the same in-delay. The final approximation factor for the most general model grows as K^3 and L, where L is the number of job groups and K is the number of machine groups, which depends on the extent of heterogeneity among the machines. We bound K by $O(\log^3 n)$ in the case when the speeds, sizes, and delays of machines are non-uniform. We emphasize that, even with the machines partitioned in this way, we must carefully design our LP to judiciously distribute jobs among the groups depending on the precedence structure of the jobs and the particular job and machine parameters.

Our LP is inspired by that of [31], though significant changes are necessary to allow for non-uniform delays. The key constraints of each LP are presented below (with the constraints from [31] rewritten to include machine group variables). Here, C^* represents the makespan of the schedule and C_v represents the earliest execution time of job v. $x_{v,k}$ indicates if v is placed on a machine in group $\langle k \rangle$ $(= 1)$ or not $(= 0)$. $z_{u,v,k}$ indicates whether $x_{v,k} = 1$ and $C_v - C_u$ is less the time it takes to communicate the result of u from a different machine. $y_{v,k}$ takes the maximum of $x_{v,k}$ and $\max_u\{z_{v,u,k}\}$ to indicate whether some copy of v is executed on a machine in group $\langle k \rangle$ $(= 1)$ or not $(= 0)$. Other notation used in the linear program is explained in Section 2.

One main difference between our LP and that of [31] is in the constraint that regulates the completion time of precedence ordered jobs in the presence of communication delay.

Delay Constraint in [31]		New Delay Constraint
$C_v \geq C_u + \rho\Big(\sum_{k' \leq k} x_{v,k'} - z_{u,v,k}\Big)$	$\Rightarrow$	$C_v \geq C_u + (\bar{\rho}_k + \bar{\rho}_\ell)(x_{v,k} - z_{u,v,k})$
$\forall u, v, k : u \prec v$		$\forall u, v, k, \ell : u \prec v$ and $v \in [\![\ell]\!]$

The constraint of [31] states that if $u \prec v$ and v is executed on a machine in speed group k, then the completion time of v is at least ρ greater than the completion time of u unless some duplicate of u is executed on group k. The summation over machine groups orders the groups by increasing speeds (similar to [13]). It turns out that the rounding technique which uses this ordering of machine groups, which is used to eliminate a log factor in [13, 31], does not straightforwardly work in our context. The new constraint has an interpretation similar to that of the delay constraint in [31]: if $u \prec v$ and v is executed on delay group k, then the completion time of v is at least the in-delay of k plus the in-delay of v greater than the completion time of u, unless some duplicate of u is also executed on group k. However, in the new constraint, the summation over machine groups has been replaced by a single machine group assignment variable.

The next change to the linear program regards the constraint which governs how many jobs can be duplicated within a communication phase for a single job.

Phase Constraint in [31]			New Phase Constraint	
$\rho \geq \sum_{u \prec v} z_{u,v,k}$	$\forall v, k$	$\Rightarrow$	$(\bar{\rho}_k + \bar{\rho}_\ell) \sum_u z_{u,v,k}$	$\forall v, k, \ell : v \in [\![\ell]\!]$

Both the old and new constraints state that the amount of duplication that can be performed for a single job within a single communication phase on a given group of machines is at most the length of the phase. The new constraint also incorporates the machine and job in-delays.

The final change is to the constraints which lower bound the makespan of the schedule by the total load placed on a single machine.

Load Constraint in [31]			New Load Constraints	
$C^* \cdot \lvert\langle k \rangle\rvert \geq \sum_v x_{v,k}$	$\forall k$	$\Rightarrow$	$C^* \cdot \lvert\langle k \rangle\rvert \geq \sum_v y_{v,k}$	$\forall k$
			$y_{v,k} \geq x_{v,k}$	$\forall v, k$
			$y_{u,k} \geq z_{u,v,k}$	$\forall u, v, k$

Both constraints state that the makespan is at least the total number of jobs placed on any group divided by the size of the group. The old constraint uses $x_{v,k}$ as the sole indicator of whether or not a job is placed on machine group k, and does not need to account for duplicates because of the optimized rounding scheme which utlizes the ordering of job groups by increasing speed. Because the new constraint cannot rely on this ordering, we use the y-variables to account for all duplicates as well.

In [31], the ordering of the groups was leveraged to construct the final schedule by always placing a job on higher capacity groups than the one to which it is assigned by the LP. Since the LP assigns all jobs to some group, we can infer that the total load over all groups does not increase by more than a constant factor. With multidimensional heterogenous machines, there is no clear ordering of machine groups to achieve a similar property (e.g. one set of jobs may be highly parallelizable, while another requires a single fast machine). Using the new LP, our solution is to place all jobs on those groups to which the LP assigns them, along with any predecessors indicated by the z-variables. However, such a construction could vastly exceed the value of the LP unless the load contributed by the z-variables is counted toward the LP makespan. To this end, we introduce the y-variables and associated constraints, which account for this additional, duplicated load. In the most general setting, we also introduce constraints which govern the amount of duplication possible within a single communication phase. These additional constrains model an optimal schedule of the duplicated jobs on the uniform machines within a single group.

Rounding the LP solution and determining final schedule (Sections 2.3–2.4). The next component rounds an optimal LP solution to an integer solution by placing each job on the group for which the job's LP mass is maximized. We also place duplicate predecessors of each job v on its group according to the z-variables for v's predecessors. This indicates a key difference with [31], where the load contributed by duplicates was handled by the ordering of the machines. A benefit of our simple rounding is that it accommodates many different machine and job properties as long as the number of groups can be kept small. Finally, we construct a schedule using the integer LP solution. This subroutine divides the set of jobs assigned to each group into phases and constructs a schedule for each phase by invoking a schedule for the uniform machines case, appending each schedule to the existing schedule for the entire instance.

No-duplication schedules. The proof of the first part of Theorem 3 extends an asympototic polylogarithmic approximation to no-duplication schedules for machine delays and job delays. The theorem follows from the structure of the schedule designed in Theorem 2 and a general reduction in [31] from duplication to no-duplication schedules in the uniform delay case. Avoiding the additive delay penalty of the first part of Theorem 3 to achieve a true approximation is much more difficult. When delays are symmetric (i.e., in-delays equal out-delays), we can distinguish those machines whose delay is low enough to communicate with other machines from those machines with high delay. One of the central challenges is then to distribute jobs among the high-delay machines. We overcome this difficulty by revising the LP in the framework of Theorem 2 to partition the jobs among low- and high-delay machines, and rounding the corresponding solutions separately.

We then must distinguish between those jobs with delay low enough to communicate with other jobs from those with high delay. We note that any predecessor or successor of a high delay job must be executed on the same machine as that job. We leverage this fact to construct our schedule, first placing all high delay jobs with their predecessors and successors on individual machines. We then run our machine and job delay algorithm with the remaining jobs on the low delay machines. This schedule is placed after the execution of the downward closed high-delay components, and before the upward closed high-delay components, ensuring that the schedule is valid.

We note that the design of no-duplication schedules via a reduction to duplication schedules incurs a loss in approximation factor of an additional polylogarithmic factor. While this may not be desirable in a practical implementation, our results demonstrate the flexibility of the approach and highlight its potential for more general delay models.

Hardness for job-machine delay model. The algorithmic framework outlined above incorporates non-uniform job and machine delays that combine additively. It is natural to ask if the techniques extend to other delay combinations or more broadly to pairwise delay models. In the job-machine delay model we study, when a job u executed on machine i precedes job v executed on machine j, then a delay $\rho_{v,j}$ between the two executions is incurred. Our reduction from UMPS to the job-machine delay problem follows the approach of [15] by introducing new jobs with suitable job-machine delay parameters that essentially force each job to be executed on a particular machine. This reduction does not require the flexibility of assigning different delays for different job-job pairs, but it is unclear if the same technique can be applied to machine-machine delay models. Delineating the boundary between tractable models and those for which polylogarithmic approximations violate conjectured complexity lower bounds is a major problem of interest.

1.3 Related work

Precedence constrained scheduling. The problem of scheduling precedence-constrained jobs was initiated in the classic work of Graham who gave a constant approximation algorithm for uniform machines [20]. Jaffe presented an $O(\sqrt{m})$ makespan approximation for the case with related machines [24]. This was improved upon by Chudak and Shmoys who gave an $O(\log m)$ approximation [13], then used the work of Hall, Schulz, Shmoys, and Wein [22] and Queyranne and Sviridenko [41] to generalize the result to an $O(\log m)$ approximation for weighted completion time. Chekuri and Bender [10] proved the same bound as Chudak and Shmoys using a combinatorial algorithm. In subsequent work, Li improved the approximation factor to $O(\log m/\log\log m)$ [27]. The problem of scheduling precedence-constrained jobs is hard to approximate even for identical machines, where the constant depends on complexity assumptions [6, 25, 47]. Also, Bazzi and Norouzi-Fard [7] showed a close connection between structural hardness for k-partite graph and scheduling with precedence constraints.

Precedence constrained scheduling under communication delays. Scheduling under communication delays has been studied extensively [39, 43, 48]. For unit size jobs, identical machines, and unit delay, a (7/3)-approximation is given in [35], and [23] proves the NP-hardness of achieving better than a 5/4-approxmation. Other hardness results are given in [4, 40, 43]. More recently, Davies, Kulkarni, Rothvoss, Tarnawski, and Zhang [16] give an $O(\log\rho\log m)$ approximation in the identical machine setting using an LP approach based on Sherali-Adams hierarchy, which is extended to include related machines in [17]. Concurrently, Maiti, Rajaraman, Stalfa, Svitkina, and Vijayaraghavan [31] provide a polylogarithmic approximation for uniform communication delay with related machines as a reduction from scheduling with duplication. The algorithm of [31] is combinatorial in the case with identical machines.

Davies, Kulkarni, Rothvoss, Sandeep, Tarnawski, and Zhang [15] consider the problem of scheduling precedence-constrained jobs on uniform machine in the presence of non-uniform, job-pairwise communication delays. That is, if $u \prec v$ and u and v are scheduled on different machines, then the time between their executions is at least $\rho_{u,v}$. The authors reduce to this problem from Unique-Machines Precedence-constrained Scheduling (UMPS) in which there is no communication delay, but for each job there is some particular machine on which that job must be placed. The authors show that UMPS is hard to approximate to within a logarithmic factor by a reduction from job-shop scheduling, and conjecture that UMPS is hard to approximate within a polynomial factor.

Precedence constrained scheduling under communication delays with job duplication. Using duplication with communication delay first studied by Papadimitriou and Yannakakis [39], who give a 2-approximation for DAG scheduling with unbounded processors and fixed delay. Improved bounds for infinite machines are given in [1, 14, 36, 37]. Approximation algorithms are given by Munier and Hanen [34, 35] for special cases in which the fixed delay is very small or very large, or the DAG restricted to a tree. The first bounds for a bounded number of machines are given by Lepere and Rapine [26] who prove an asymptotic $O(\log\rho/\log\log\rho)$ approximation. Recent work has extended their framework to other settings: [31] uses duplication to achieve an $O(\log\rho\log m/\log\log\rho)$ approximation for a bounded number of related machines, and Liu, Purohit, Svitkina, Vee, and Wang [29] improve on the runtime of [26] to a near linear time algorithm with uniform delay and identical machines.

1.4 Discussion and open problems

Our results indicate several directions for further work. First, we conjecture that our results extend easily to the setting with non-uniform job sizes. We believe the only barriers to such a result are the techinical difficulties of tracking the completion times of very large jobs that continue executing long after they are placed on a machine. Also, while our approximation ratios are the first polylogarithmic guarantees for scheduling under non-uniform delays, we have not attempted to optimize logarithmic factors. There are obvious avenues for small reductions in our ratio, e.g. the technique used in [26] to reduce the ratio by a factor of $\log \log \rho$. More substantial reduction, however, may require a novel approach. Additionally, in the setting without duplication, we incur even more logarithmic factors owing to our reduction to scheduling with duplication. These factors may be reduced by using a more direct method, possibly extending the LP-hierarchy style approach taken in [16, 17].

Aside from improvements to our current results, our techniques suggest possible avenues to solve related non-uniform delay scheduling problems. A special case of general machine metrics is a machine hierarchy, where machines are given as leaves in a weighted tree. Our incorporation of parallel processors allows our results to apply to a two-level machine hierarch. We would like to explore extensions of our framework to constant-depth hierarchies and tree metrics. More generally, scheduling under metric and general machine-machine delays remains wide open (see Figure 2).

We also believe there are useful analogs to these machine delay models in the job-pairwise regime. A job v with in-delay ρ_v^{in} and out-delay ρ_v^{out} has the natural interpretation of the data required to execute a job, and the data produced by a job. A job tree hierarchy could model the shared libraries required to execute certain jobs: jobs in different subtrees require different resources to execute, and downloading these additional resources incurs a delay. Given the hardness conjectures of [15] and our hardness result for the job-machine delay model, further refining Figure 2 and exploring the tractability boundary would greatly enhance our understanding of scheduling under non-uniform delays.

Finally, recall that our notion of job delays is defined in terms of the precedence relation over the jobs. Another natural notion of job delay may be to consider a DAG defined over the jobs, with a delay incurred only if there is a directed edge $u \to v$ (rather than $u \prec v$). In this setting, while our results do not hold in the presence of general job delays, they do hold for some significant special cases. These include instances where the job DAG is transitively closed, or where job delays are uniform, or where job delays of predecessors are at most that of their successors (i.e. $u \prec v$ implies $\rho_u^{\text{out}} \leq \rho_v^{\text{out}}$ and $\rho_u^{\text{in}} \leq \rho_v^{\text{in}}$), or where there are only machine delays. However, resolving the most general case is an interesting open problem since this family of delay models provides an intuitive and important set of problems.

2 Machine Delays and Job Delays

In this section, we present an asymptotic approximation algorithm for scheduling under machine delays and job delays for unit speed and size machines. As discussed in Section 1.2, we can focus on the setting with no out-delays, at the expense of a polylogarithmic factor in approximation; Lemma 37 of [42] presents the reduction to in-delays. Therefore, in this section, we assume that $\rho_i^{\text{out}} = 0$ for all machines i and $\rho_v^{\text{out}} = 0$ for all jobs v. For convenience, we use ρ_i to denote the in-delay ρ_i^{in} of machine i and ρ_v to denote the in-delay ρ_v^{in} of machine v. Let $\rho_{\max} = \max\{\max_v\{\rho_v\}, \max_i\{\rho_i\}\}$.

2.1 Partitioning machines and jobs into groups

In order to simplify our exposition and analysis, we introduce a new set of machines M' with rounded delays. For each $i \in M$, if $2^{k-1} \leq \rho_i < 2^k$, we introduce $i' \in M'$ with $\rho_{i'} = 2^k$. We then partition M' according to machine delays: machine $i \in M'$ is in $\langle k \rangle$ if $\rho_i = 2^k$; we set $\bar{\rho}_k = 2^k$. We also introduce a new set of jobs V' with rounded delays. For each $v \in V$, if $2^{\ell-1} \leq \rho_v < 2^\ell$, we introduce $v' \in V'$ with $\rho_{v'} = 2^\ell$. We then partition V' according to job delays: job $v \in V'$ is in $[\![\ell]\!]$ if $\rho_v = 2^\ell = \bar{\rho}_\ell$. For the remainder of the section, we work with the machine set M' and the job set V', ensuring that all machines or jobs within a group have identical delays. As shown in the following lemma, this partitioning is at the expense of at most a constant factor in approximation.

▶ **Lemma 2.** *The optimal makespan over the machine set V', M' is no more than a factor of 2 greater than the optimal solution over V, M.*

Proof. Consider any schedule σ on the machine set M. We first show that increasing the delay of each machine by a factor of 2 increases the makespan of the schedule by at most a factor of 2. We define the schedule σ' as follows. For every i, t, if $(i, t) \in \sigma(v)$, then $(i, 2t) \in \sigma'(v)$. It is easy to see that σ' maintains the precedence ordering of jobs, and that the time between the executions of any two jobs has been doubled. Therefore, σ' is a valid schedule with all communication delays doubled, and with the makespan doubled. ◀

We can assume that $\max_k\{\bar{\rho}_k\} \leq n$ since if we ever needed to communicate to a machine with delay greater than n we could schedule everything on a single machine in less time. Therefore, we have $K \leq \log n$ machine groups. Similarly, $\max_\ell\{\bar{\rho}_\ell\} \leq n$, implying that we have $L \leq \log n$ job groups.

2.2 The linear program

In this section, we design a linear program LP_α – Equations (1-11) – parametrized by $\alpha \geq 1$, for machine delays. Following Section 2.1, we assume that the machines and jobs are organized in groups, where each group $\langle k \rangle$ (resp., $[\![\ell]\!]$) is composed of machines (resp., jobs) that have identical delay.

$$C^*_\alpha \geq C_v \qquad \forall v \qquad (1)$$

$$C^*_\alpha \cdot |\langle k \rangle| \geq \sum_v y_{v,k} \qquad \forall k \qquad (2)$$

$$C_v \geq C_u + (\bar{\rho}_k + \bar{\rho}_\ell)(x_{v,k} - z_{u,v,k}) \qquad \forall u, v, k, \ell : u \prec v, v \in [\![\ell]\!] \qquad (3)$$

$$C_v \geq C_u + 1 \qquad \forall u, v : u \prec v \qquad (4)$$

$$\alpha(\bar{\rho}_k + \bar{\rho}_\ell) \geq \sum_u z_{u,v,k} \qquad \forall v, k, \ell : v \in [\![\ell]\!] \qquad (5)$$

$$\sum_k x_{v,k} = 1 \qquad \forall v \qquad (6)$$

$$C_v \geq 0 \qquad \forall v \qquad (7)$$

$$x_{v,k} \geq z_{u,v,k} \qquad \forall u, v, k \qquad (8)$$

$$y_{v,k} \geq x_{v,k} \qquad \forall v, k \qquad (9)$$

$$y_{u,k} \geq z_{u,v,k} \qquad \forall u, v, k \qquad (10)$$

$$z_{u,v,k} \geq 0 \qquad \forall u, v, k \qquad (11)$$

Variables. C^*_α represents the makespan of the schedule. For each job v, C_v represents the earliest completion time of v. For each job v and group $\langle k \rangle$, $x_{v,k}$ indicates whether or not v is first executed on a machine in group $\langle k \rangle$. For each $\langle k \rangle$ and pair of jobs u, v such that $u \prec v$ and $v \in [\![\ell]\!]$, $z_{u,v,k}$ indicates whether v is first executed on a machine in group $\langle k \rangle$ and the earliest execution of u is less that $\bar{\rho}_k + \bar{\rho}_\ell$ time before the execution of v. Intuitively, $z_{u,v,k}$ indicates whether there must be a copy of u executed on the same machine that first

executes v. For each job v and group $\langle k \rangle$, $y_{v,k}$ indicates whether $x_{v,k} = 1$ or $z_{u,v,k} = 1$ for some u; that is, whether or not some copy of v is placed on group $\langle k \rangle$. Constraints (7 - 11) guarantee that all variables are non-negative.

Makespan (2, 1). Constraint 1 states that the makespan is at least the maximum completion time of any job. Constraint 2 states that the makespan is at least the load on any single group.

Delays (3, 5). Constraint 3 states that the earliest completion time of $v \in [\![\ell]\!]$ must be at least $\bar{\rho}_k + \bar{\rho}_\ell$ after the earliest completion time of any predecessor u if v is first executed on a machine in group $\langle k \rangle$ and no copy of u is duplicated on the same machine as v. Constraint 5 limits the amount of duplication that can be done to improve the completion time of any job: if $v \in [\![\ell]\!]$ first executes on a machine in group $\langle k \rangle$ at time t, then the number of predecessors that may be executed in the $\bar{\rho}_k + \bar{\rho}_\ell$ steps preceding t is at most $\bar{\rho}_k$.

The remaining constraints enforce standard scheduling conditions. Constraint 4 states that the completion time of v is at least the completion time of any of its predecessors, and constraint 6 ensures that every job is executed on some group. Constraints 6 and 8 guarantee that $z_{u,v,k} \leq 1$ for all u, v, k. This is an important feature of the LP, since a large z-value could be used to disproportionately reduce the delay between two jobs in constraint 3.

▶ **Lemma 3** (LP_1 is a valid relaxation). *The minimum of C_1^* is at most* OPT.

Proof. Consider an arbitrary schedule σ with makespan C_σ, i.e. $C_\sigma = \max_{v,i,t}\{t : (i,t) \in \sigma(v)\}$.

LP solution. Set $C_1^* = C_\sigma$. For each job v, set C_v to be the earliest completion time of v in σ, i.e. $C_v = \min_{i,t}\{t : (i,t) \in \sigma(v)\}$. Set $x_{v,k} = 1$ if $\langle k \rangle$ is the group that contains the machine on which v first completes (choosing arbitrarily if there is more than one) and 0 otherwise. For u, v, k, set $z_{u,v,k} = 1$ if $u \prec v$, $x_{v,k} = 1$, $v \in [\![\ell]\!]$, and $C_v - C_u < \bar{\rho}_k + \bar{\rho}_\ell$ (0 otherwise). Set $y_{u,k} = \max\{x_{u,k}, \max_v\{z_{u,v,k}\}\}$.

Feasibility. We now establish that the solution defined is feasible. Constraints (1, 7–11) are easy to verify. We now establish constraints (2–5). Consider constraint 2 for fixed group $\langle k \rangle$. $\sum_v y_{v,k}$ is upper bound by the total load Λ on $\langle k \rangle$. The constraint follows from $C_\alpha^* \geq C_\sigma \geq \Lambda/|\langle k \rangle|$.

Consider constraint 3 for fixed u, v, k where $u \prec v$. Let $X = x_{v,k}$ and let $Z = z_{u,v,k}$. If $(X, Z) = (0,0), (0,1)$, or $(1,1)$ then the constraint follows from constraint 4. If $(X, Z) = (1,0)$, then by the assignment of $z_{u,v,k}$ we can infer that $C_v - C_u \geq \bar{\rho}_k + \bar{\rho}_\ell$, which shows the constraint is satisfied.

Consider constraint 5 for fixed v, k. If $x_{v,k} = 0$ then the result follows from the fact that $z_{u,v,k} = 0$ for all u. If $x_{v,k} = 1$, then we can infer that $v \in [\![\ell]\!]$. So, at most $\bar{\rho}_k + \bar{\rho}_\ell$ predecessors of v that can be scheduled in the $\bar{\rho}_k + \bar{\rho}_\ell$ time before C_v, ensuring that the constraint is satisfied. ◀

2.3 Deriving a rounded solution to the linear program

▶ **Definition 4.** (C, x, y, z) *is a* rounded solution *to LP_α if all values of x, y, z are either 0 or 1.*

Let LP_1 be defined over machine groups $\langle 1\rangle, \langle 2\rangle, \ldots, \langle K\rangle$ and job groups $[\![1]\!], [\![2]\!], \ldots, [\![L]\!]$. Given a solution $(\hat{C}, \hat{x}, \hat{y}, \hat{z})$ to LP_1, we construct an integer solution (C, x, y, z) to LP_{2K} as follows. For each v, k, set $x_{v,k} = 1$ if $k = \max_{k'}\{\hat{x}_{v,k'}\}$ (if there is more than one maximizing k, arbitrarily select one); set to 0 otherwise. Set $z_{u,v,k} = 1$ if $x_{v,k} = 1$ and $\hat{z}_{u,v,k} \geq 1/(2K)$; set to 0 otherwise. For all u, k, $y_{u,k} = \max\{x_{u,k}, \max_v\{z_{u,v,k}\}\}$. Set $C_v = 2K \cdot \hat{C}_v$. Set $C^*_{2K} = 2K \cdot \hat{C}^*_1$.

▶ **Lemma 5.** *If $(\hat{C}, \hat{x}, \hat{y}, \hat{z})$ is a valid solution to LP_1, then (C, x, y, z) is a valid solution to LP_{2K}.*

Proof. By constraint (6), $\sum_k \hat{x}_{v,k}$ is at least 1, so $\max_k\{\hat{x}_{v,k}\}$ is at least $1/K$. Therefore, $x_{v,k} \leq K\hat{x}_{v,k}$ for all v and k. Also, $z_{u,v,k} \leq 2K\hat{z}_{u,v,k}$ for any u, v, k by definition. By the setting of C_v for all v, $y_{v,k}$ for all v, k, and C^*_{2K}, it follows that constraints (1, 4-11) of LP_1 imply the respective constraints of LP_{2K}. We first establish constraint (2). For any fixed group $\langle k\rangle$,

$$\begin{aligned} 2K\hat{C}_1 \cdot |\langle k\rangle| &\geq 2K \sum_v \hat{y}_{v,k} = 2K \sum_v \max\{\hat{x}_{v,k}, \max_u\{\hat{z}_{v,u,k}\}\} && \text{by constraints 2, 11 of } \text{LP}_1 \\ &\geq 2K \sum_v \frac{x_{v,k} + \max_u\{z_{v,u,k}\}}{2K} \geq \sum_v y_{v,k} && \text{by definition of } y_{v,k} \end{aligned}$$

which entails constraint (2) by $C^*_{2K} = 2K\hat{C}^*_1$. It remains to establish constraint (3) for fixed u, v, k. We consider two cases. If $x_{v,k} - z_{u,v,k} \leq 0$, then the constraint is trivially satisfied in LP_{2K}. If $x_{v,k} - z_{u,v,k} = 1$, then, by definition of x and z, $\hat{x}_{v,k} - \hat{z}_{u,v,k}$ is at least $1/(2K)$. This entails that $\hat{C}_v \geq \hat{C}_u + ((\bar{\rho}_k + \bar{\rho}_\ell)/2K)$ which establishes constraint (3) of LP_{2K} by definition of C_v and C_u. ◀

▶ **Lemma 6.** $C_{2K} \leq 4K \cdot OPT$.

Proof. Lemma 2 shows that our grouping of machines does not increase the value of the LP by more than a factor of 2. Therefore, by Lemmas 3 and 5, $C_{2K} = 2K \cdot \hat{C}_1 \leq 4K \cdot \text{OPT}$. ◀

2.4 Computing a schedule given an integer solution to the LP

Suppose we are given a partition of M into K groups such that group $\langle k\rangle$ is composed of identical machines (i.e. for all $i, j \in \langle k\rangle$, $\rho_i = \rho_j$). Also, suppose we are given a partition of V into L groups such that group $[\![\ell]\!]$ is composed of jobs with identical in-delay. Finally, we are given a rounded solution (C, x, y, z) to LP_α defined over machine groups $\langle 1\rangle, \ldots, \langle K\rangle$ and job groups $[\![1]\!], \ldots [\![L]\!]$. In this section, we show that we can construct a schedule that achieves an approximation for machine delays in terms of α, K, and L. The combinatorial subroutine that constructs the schedule is defined in Algorithm 1. In the algorithm, we use a subroutine UDPS-Solver for Uniform Delay Precedence-Constrained Scheduling. An $O(\log \rho / \log\log \rho)$-asympototic approximation is given in [26]. For completeness, we use the UDPS-Solver presented and analyzed in [42], which generalizes the algorithm of [26] to incorporate non-uniform machine sizes.

We now describe Algorithm 1 informally. The subroutine takes as input the rounded LP_α solution (C, x, y, z) and initializes an empty schedule σ and global parameters T, θ to 0. For a fixed value of T, we iterate through all machine groups $\langle k\rangle$ and job groups $[\![\ell]\!]$, with decreasing ℓ. For a fixed value of T, k, ℓ, we check if there is some integer d such that $T = d(\bar{\rho}_k + \bar{\rho}_\ell)$. If so, we define $V_{k,\ell,d}$ and $U_{k,\ell,d}$ as in lines 4 and 5. $V_{k,\ell,d}$ represents the set of jobs in $[\![\ell]\!]$ assigned by the LP to machine group $\langle k\rangle$ in a single phase of length $\bar{\rho}_k + \bar{\rho}_\ell$.

Algorithm 1 Machine Delay Scheduling with Duplication.

Init: $\forall v,\ \sigma(v) \leftarrow \varnothing;\ T \leftarrow 0;\ \theta \leftarrow 0$
1 **while** $T \leq C^*_\alpha$ **do**
2 **forall** *machine groups* $\langle k \rangle$ **do**
3 **for** *job group* $[\![\ell]\!] = [\![L]\!]$ *to* $[\![1]\!]$*:* $\exists$ *integer* $d,\ T = d(\bar{\rho}_k + \bar{\rho}_\ell)$ **do**
4 $V_{k,\ell,d} \leftarrow \{v \in [\![\ell]\!] : x_{v,k} = 1 \text{ and } T \leq C_v < T + \bar{\rho}_k + \bar{\rho}_\ell\}$
5 $U_{k,\ell,d} \leftarrow \{u : \exists v \in V_{k,\ell,d},\ u \prec v \text{ and } T \leq C_u < T + \bar{\rho}_k + \bar{\rho}_\ell\}$
6 $\sigma' \leftarrow$ UDPS-Solver on $(V_{k,\ell,d} \cup U_{k,\ell,d}, \langle k \rangle, \bar{\rho}_k + \bar{\rho}_\ell)$
7 $\forall v, i, t,$ if $(i,t) \in \sigma'(v)$ then $\sigma(v) \leftarrow \sigma(v) \cup \{(i, \theta + \bar{\rho}_k + \bar{\rho}_\ell + t)\}$
8 $\theta \leftarrow \theta + 2(\bar{\rho}_k + \bar{\rho}_\ell)$
9 $T \leftarrow T + 1$

$U_{k,\ell,d}$ represents predecessors of $V_{k,\ell,d}$ whose LP completion times are within $\bar{\rho}_k + \bar{\rho}_\ell$ of their successor in $V_{k,\ell,d}$. We then call UDPS-Solver to construct a UDPS schedule σ' on jobs $V_{k,\ell,d} \cup U_{k,\ell,d}$, machines in $\langle k \rangle$, and delay $\bar{\rho}_k + \bar{\rho}_\ell$. We then append σ' to σ. Once all values of k, ℓ have been checked, we increment T and repeat until all jobs are scheduled. The structure of the schedule produced by Algorithm 1 is depicted in Figure 4. Lemma 7 (entailed by Lemma 45 of [42]) provides guarantees for the UDPS-Solver subroutine.

▶ **Lemma 7.** *Let U be a set of η jobs such that for any $v \in U$, $|\{u \in U : u \prec v\}| \leq \alpha\delta$. Given input U, a set of μ identical machines, and delay δ, UDPS-Solver produces, in polynomial time, a valid UDPS schedule with makespan at most $3\alpha\delta \log(\alpha\delta) + (2\eta/\mu)$.*

▶ **Lemma 8.** *Algorithm 1 outputs a valid schedule in polynomial time.*

Proof. It is easy to see that the algorithm runs in polynomial time, and Lemma 7 entails that precedence constraints are obeyed on each machine. Consider a fixed v, k, d such that $v \in V_{k,\ell,d}$. By line 7, we insert a communication phase of length $\bar{\rho}_k + \bar{\rho}_\ell$ before appending the schedule of any set of jobs $V_{k,\ell,d} \cup U_{k,\ell,d}$ on any machine group $\langle k \rangle$. So, by the time Algorithm 1 executes any job in $V_{k,\ell,d}$, every job u such that $C_u < d(\bar{\rho}_k + \bar{\rho}_\ell)$ is available to all machines, including those in group $\langle k \rangle$. So the only predecessors of v left to execute are those jobs in $U_{k,\ell,d}$. Therefore, all communication constraints are satisfied. ◀

▶ **Lemma 9.** *If (C, x, y, z) is a rounded solution to LP_α then Algorithm 1 outputs a schedule with makespan at most $12\alpha \log(\rho_{\max})(KLC^*_\alpha + \rho_{\max}(K + L))$.*

Proof. Fix any schedule σ. Note that the schedule produced by the algorithm executes a single job group on a single machine group at a time. Our proof establishes a bound for the total time spent executing a single job group on a single machine group, then sums this bound over all K machine groups and L job groups.

▷ Claim 10. For any v, u, k, ℓ, d, if $v \in V_{k,\ell,d}$ and $C_v < C_u + (\bar{\rho}_k + \bar{\rho}_\ell)$ then $z_{u,v,k,\ell} = 1$.

Proof. Fix u, v, k, ℓ, d such that $v \in V_{k,\ell,d}$ and $C_v < C_u + (\bar{\rho}_k + \bar{\rho}_\ell)$. By the definition of $V_{k,\ell,d}$, $x_{v,k}$ is 1. By constraint 3, $C_v \geq C_u + \bar{\rho}_k(1 - z_{u,v,k})$, implying that $z_{u,v,k}$ cannot equal 0. Since $z_{u,v,k}$ is either 0 or 1, we have $z_{u,v,k} = 1$. ◁

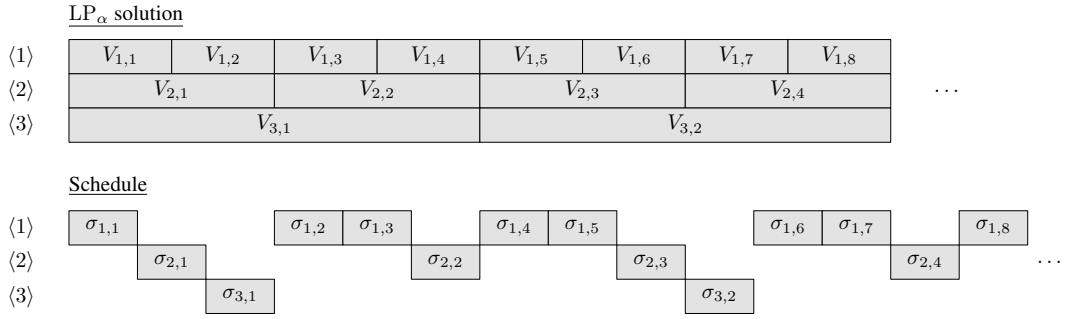


Figure 4 Structure of the schedule produced by Algorithm 1. $\sigma_{k,d}$ denotes a schedule of $V_{k,\ell,d}$ on the machines in group $\langle k \rangle$. The algorithm scans the LP$_\alpha$ solution by increasing time (left to right). At the start of each $V_{k,\ell,d}$, the algorithm constructs a schedule of the set and appends it to the existing schedule.

▷ Claim 11. For any k, ℓ, we show

a $\sum_d |V_{k,\ell,d} \cup U_{k,\ell,d}| \leq C^*_\alpha \cdot |\langle k \rangle|$ and

b for any d and $v \in V_{k,\ell,d}$, the number of v's predecessors in $V_{k,\ell,d} \cup U_{k,\ell,d}$ is at most $\alpha(\bar{\rho}_k + \bar{\rho}_\ell)$.

Proof. Fix k, ℓ. We first prove a. For any v in $V_{k,\ell,d}$ we have $x_{v,k} = 1$ by the definition of $V_{k,\ell,d}$. Consider any u in $U_{k,\ell,d}$. By definition, there exists a $v' \in V_{k,\ell,d}$ such that $x_{v',k} = 1$ and $C_v < C_u + (\bar{\rho}_k + \bar{\rho}_\ell)$; fix such a v'. By claim 10, $z_{u,v',k} = 1$. So, by constraint 10, $y_{v,k} = 1$ for every job $v \in V_{k,\ell,d} \cup U_{k,\ell,d}$. For any $d' \neq d$, $V_{k,\ell,d}$ and $V_{k,\ell,d'}$ are disjoint. So $\sum_d |V_{k,\ell,d} \cup U_{k,\ell,d}|$ is at most the right-hand side of constraint 2, which is at most $C^*_\alpha \cdot |\langle k \rangle|$.

We now prove b. Fix v, d such that $v \in V_{k,\ell,d}$. Consider any u in $V_{k,\ell,d} \cup U_{k,d}$ such that $u \prec v$. By definition of $V_{k,\ell,d}$ and $U_{k,\ell,d}$, $C_v < C_u + (\bar{\rho}_k + \bar{\rho}_\ell)$. By Claim 10, $z_{u,v,k} = 1$. The claim then follows from constraint (5). ◁

By Lemma 7 and Claim 11b, the time spent executing jobs in $[\![\ell]\!]$ on machines in $\langle k \rangle$ is at most

$$\sum_d \left(3\alpha(\bar{\rho}_k + \bar{\rho}_\ell) \log(\alpha(\bar{\rho}_k + \bar{\rho}_\ell)) + \frac{2 \cdot |V_{k,\ell,d} \cup U_{k,\ell,d}|}{|\langle k \rangle|} \right)$$

The summation over the first term is at most $\lceil C^*_\alpha / (\bar{\rho}_k + \bar{\rho}_\ell) \rceil \, 3\alpha(\bar{\rho}_k + \bar{\rho}_\ell) \log(\alpha(\bar{\rho}_k + \bar{\rho}_\ell))$ which is at most $3C^*_\alpha \alpha \log(\alpha(\bar{\rho}_k + \bar{\rho}_\ell)) + 3\alpha(\bar{\rho}_k + \bar{\rho}_\ell) \log(\alpha(\bar{\rho}_k + \bar{\rho}_\ell))$. The summation over the second term is at most $2C^*_\alpha$ by claim 11a. Summing over all K machine groups and L job groups, and considering $K, L \leq \log \rho_{\max}$, the total length of the schedule is at most $12\alpha \log(\rho_{\max})(KLC^*_\alpha + \rho_{\max}(K + L))$. ◀

▶ **Theorem 1** (Job Delays and Machine Delays). *There exists a polynomial time algorithm to compute a valid machine delays and job precedence delays schedule with makespan* $O((\log n)^9(\mathrm{OPT} + \rho_{\max}))$.

Proof. Lemma 5 entails that (C, x, y, z) is a valid solution to LP$_{2K}$. Lemma 6 entails that $C^*_{2K} \leq 4K \cdot \mathrm{OPT}$. With $\alpha = 2K$, Lemma 9 entails that the makespan of our schedule is at most $12\alpha \log(\rho_{\max})(KLC^*_\alpha + \rho_{\max}(K+L)) = 48(\log \rho_{\max})^5 \mathrm{OPT} + 24(\log \rho_{\max})^3 \rho_{\max}$ for the case with no out-delays. By Lemma 37 of [42], the length of our schedule is $O((\log \rho_{\max})^9(\mathrm{OPT} + \rho_{\max})$ The theorem is entailed by $\rho_{\max} \leq n$. This proves the theorem. ◀

▶ **Corollary 12** (Machine Delays). *There exists a polynomial time algorithm to compute a valid machine delays schedule with makespan* $O((\log n)^5 \cdot (OPT + \rho))$.

Proof. Lemma 5 entails that (C, x, y, z) is a valid solution to LP_{2K}. Lemma 6 entails that $C^*_{2K} \leq 4K \cdot \text{OPT}$. With $\alpha = 2K$, Lemma 9 entails that the makespan of our schedule is at most $12\alpha \log(\rho_{\max})(KLC^*_\alpha + \rho_{\max}(K + L)) = 48(\log \rho_{\max})^5 \text{OPT} + 24(\log \rho_{\max})^3 \rho_{\max}$ for the case with no out-delays. By Lemma 41 of [42], the length of our schedule is $O((\log \rho_{\max})^5(\text{OPT} + \rho_{\max})$ The theorem is entailed by $\rho_{\max} \leq n$. ◀

2.5 Combinatorial Algorithm for Uniform Machine Delays

The only noncombinatorial subroutine of our algorithm is solving the linear program. In this section, we describe how to combinatorially construct a rounded solution to LP_1 when machine delays are uniform (i.e. for all i, j, $\rho^{\text{in}}_i = \rho^{\text{out}}_i = \rho^{\text{in}}_j = \rho^{\text{out}}_j$), machine speeds are unit, and machine capacities are unit. We let δ represent the uniform machine delay. By Lemma 37 of [42], we focus on the case where all job out-delays are 0. We let $\rho_v = \delta + \rho^{\text{in}}_v$ for any job v.

Since delays, speeds, and capacities are uniform, there is only one machine group: $\langle 1 \rangle$. Set $x_{v,1} = y_{v,1} = 1$ for all v. For each job v, we define C_v as follows. If v has no predecessors, we set $C_v = 0$. Otherwise, we order v's predecessors such that $C_{u_i} \geq C_{u_{i+1}}$. We define $C_v = \max_{1 \leq i \leq \rho_v}\{C_{u_i} + i\}$. We set $C^* = \max\{n/m, \max_v\{C_v\}\}$. We set $z_{u,v,1} = 1$ if $u \prec v$ and $C_v - C_u < \rho_v$; and set to 0 otherwise.

▶ **Lemma 13.** $C^* \leq OPT$.

Proof. Consider an arbitrary schedule in which t_v is the earliest completion time of any job v. We show that, for any v, $t_v \geq C_v$, which is sufficient to prove the lemma.

We prove the claim by induction on the number of predecessors of v. The claim is trivial if v has no predecessors. Suppose that the claim holds for all of v's predecessors and let $y = \arg\max_{1 \leq i \leq \rho_v}\{C_{u_i} + i\}$. Then $C_v = C_{u_y} + y \leq t_{u_y} + y$ (by IH) $= t_y + |\{u_x : 0 \leq x \leq y\}| \leq t_y + \rho_v$. This entails that all jobs $u_1, \ldots u_y$ must be executed on the same machine as v. Now suppose, for the sake of contradiction, that $t_v < C_v$. Then all jobs $u \in \{u_x : 0 \leq x < y\}|$ must be executed serially in the time $t_v - t_{u_y} < C_v - t_{u_y} = |\{u_x : 0 \leq x \leq y\}|$ which gives us our contradiction. ◀

▶ **Lemma 14.** (C, x, y, z) *is a rounded solution to* LP_1.

Proof. It is easy to see that constraints (1, 2, 3, 4, 6, 7, 8, 9, 10 11) are satisfied by the assignment. So we must only show that constraint (5) is satisfied for fixed v. We can see from the definition of C_v, that maximum number of predecessors u such that $C_v - C_u < \rho_v + \rho$ is at most $\rho_v + \rho$. This proves the lemma. ◀

▶ **Lemma 15** (Combinatorial Algorithm for Job Delays). *There exists a purely combinatorial, polynomial time algorithm to compute a schedule for Job Delays with makespan* $O((\log n)^6(OPT + \max_v\{\rho_v\}))$.

Proof. Lemma 9 entails that the length of the schedule is at most $12(\log \rho_{\max})^2(\text{OPT} + \rho_{\max})$ for the problem with job in-delays. By Lemma 37 of [42] we achieve a makespan of $O((\log \rho_{\max})^6(\text{OPT} + \rho_{\max}))$ for job in- and out-delays. ◀

References

1 Ishfaq Ahmad and Yu-Kwong Kwok. On exploiting task duplication in parallel program scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 9(9):872–892, September 1998. doi:10.1109/71.722221.

2 Adil Amirjanov and Konstantin Sobolev. Scheduling of directed acyclic graphs by a genetic algorithm with a repairing mechanism. *Concurrency and Computation: Practice and Experience*, 29(5):e3954, 2017. e3954 CPE-16-0237.R1. doi:10.1002/cpe.3954.

3 Pau Andrio, Adam Hospital, Javier Conejero, Luis Jordá, Marc Del Pino, Laia Codo, Stian Soiland-Reyes, Carole Goble, Daniele Lezzi, Rosa M Badia, Modesto Orozco, and Josep Gelpi. Bioexcel building blocks, a software library for interoperable biomolecular simulation workflows. *Scientific data*, 6(1):1–8, 2019.

4 Evripidis Bampis, Aristotelis Giannakos, and Jean-Claude König. On the complexity of scheduling with large communication delays. *European Journal of Operational Research*, 94:252–260, 1996.

5 Nikhil Bansal. Scheduling open problems: Old and new. The 13th Workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP 2017), 2017. URL: http://www.mapsp2017.ma.tum.de/MAPSP2017-Bansal.pdf.

6 Nikhil Bansal and Subhash Khot. Optimal long code test with one free bit. *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 453–462, October 2009. doi:10.1109/focs.2009.23.

7 Abbas Bazzi and Ashkan Norouzi-Fard. Towards tight lower bounds for scheduling problems. *Lecture Notes in Computer Science*, pages 118–129, 2015. doi:10.1007/978-3-662-48350-3_11.

8 Gregor Behnke, Daniel Höller, and Susanne Biundo. Bringing order to chaos – a compact representation of partial order in sat-based htn planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7520–7529, July 2019. doi:10.1609/aaai.v33i01.33017520.

9 Karthekeyan Chandrasekaran and Chandra Chekuri. Min-max partitioning of hypergraphs and symmetric submodular functions. In *Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '21, pages 1026–1038, USA, 2021. Society for Industrial and Applied Mathematics.

10 Chandra Chekuri and Michael Bender. An efficient approximation algorithm for minimizing makespan on uniformly related machines. *Journal of Algorithms*, 41(2):212–224, November 2001. doi:10.1006/jagm.2001.1184.

11 Eden Chlamtáč, Michael Dinitz, and Yury Makarychev. Minimizing the union: Tight approximations for small set bipartite vertex expansion. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 881–899. SIAM, 2017.

12 Eden Chlamtáč, Michael Dinitz, and Yury Makarychev. Minimizing the union: Tight approximations for small set bipartite vertex expansion. In *Proceedings of the 2017 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 881–899, 2017.

13 Fabián A. Chudak and David B. Shmoys. Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds. *Journal of Algorithms*, 30(2):323–343, 1999.

14 Sekhar Darbha and Dharma P. Agrawal. Optimal scheduling algorithm for distributed-memory machines. *IEEE Transactions on Parallel and Distributed Systems*, 9:87–95, 1998.

15 Sami Davies, Janardhan Kulkarni, Thomas Rothvoss, Sai Sandeep, Jakub Tarnawski, and Yihao Zhang. On the hardness of scheduling with non-uniform communication delays. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2958–2977, 2022. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611976465.176.

16 Sami Davies, Janardhan Kulkarni, Thomas Rothvoss, Jakub Tarnawski, and Yihao Zhang. Scheduling with communication delays via lp hierarchies and clustering. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 822–833, 2020. doi:10.1109/FOCS46700.2020.00081.

17 Sami Davies, Janardhan Kulkarni, Thomas Rothvoss, Jakub Tarnawski, and Yihao Zhang. Scheduling with communication delays via lp hierarchies and clustering ii: Weighted completion times on related machines. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2958–2977, 2021. doi:10.1137/1.9781611976465.176.

18 Yuanxiang Gao, Li Chen, and Baochun Li. Spotlight: Optimizing device placement for training deep neural networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1676–1684. PMLR, 10–15 July 2018. URL: https://proceedings.mlr.press/v80/gao18a.html.

19 M. R. Garey and David S. Johnson. Scheduling tasks with nonuniform deadlines on two processors. *J. ACM*, 23(3):461–467, 1976. doi:10.1145/321958.321967.

20 Ronald L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17:416–429, 1969.

21 Ubaid Ullah Hafeez, Xiao Sun, Anshul Gandhi, and Zhenhua Liu. Towards optimal placement and scheduling of dnn operations with pesto. In *Proceedings of the 22nd International Middleware Conference*, pages 39–51, 2021.

22 Leslie A. Hall, Andreas S. Schulz, David B. Shmoys, and Joel Wein. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Mathematics of Operations Research*, 22(3):513–544, August 1997. doi:10.1287/moor.22.3.513.

23 J.A. Hoogeveen, Jan Karel Lenstra, and Bart Veltman. Three, four, five, six, or the complexity of scheduling with communication delays. *Operations Research Letters*, 16(3):129–137, 1994. doi:10.1016/0167-6377(94)90024-8.

24 Jeffrey M. Jaffe. Efficient scheduling of tasks without full use of processor resources. *Theoretical Computer Science*, 12(1):1–17, September 1980. doi:10.1016/0304-3975(80)90002-x.

25 Jan Karel Lenstra and A. H. G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26(1):22–35, 1978. doi:10.1287/opre.26.1.22.

26 Renaud Lepere and Christophe Rapine. An asymptotic $\mathcal{O}(\ln \rho / \ln \ln \rho)$-approximation algorithm for the scheduling problem with duplication on large communication delay graphs. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 154–165. Springer, 2002.

27 Shi Li. Scheduling to minimize total weighted completion time via time-indexed linear programming relaxations. In *2017 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 283–294, 2017. doi:10.1109/FOCS.2017.34.

28 Guanfeng Liang and Ulaş C. Kozat. Fast cloud: Pushing the envelope on delay performance of cloud storage with coding. *IEEE/ACM Transactions on Networking*, 22(6):2012–2025, December 2014. doi:10.1109/TNET.2013.2289382.

29 Quanquan C. Liu, Manish Purohit, Zoya Svitkina, Erik Vee, and Joshua R. Wang. Scheduling with communication delay in near-linear time. In *STACS*, 2022.

30 Ashraf Mahgoub, Edgardo Barsallo Yi, Karthick Shankar, Eshaan Minocha, Sameh Elnikety, Saurabh Bagchi, and Somali Chaterji. Wisefuse: Workload characterization and dag transformation for serverless workflows. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 6(2), June 2022. doi:10.1145/3530892.

31 Biswaroop Maiti, Rajmohan Rajaraman, David Stalfa, Zoya Svitkina, and Aravindan Vijayaraghavan. Scheduling precedence-constrained jobs on related machines with communication delay. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 834–845, 2020. doi:10.1109/FOCS46700.2020.00082.

32 Azalia Mirhoseini, Anna Goldie, Hieu Pham, Benoit Steiner, Quoc V. Le, and Jeff Dean. Hierarchical planning for device placement. In *International Conference on Learning Representations*, 2018. URL: https://openreview.net/pdf?id=Hkc-TeZ0W.

33 Azalia Mirhoseini, Hieu Pham, Quoc V. Le, Benoit Steiner, Rasmus Larsen, Yuefeng Zhou, Naveen Kumar, Mohammad Norouzi, Samy Bengio, and Jeff Dean. Device placement optimization with reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, pages 2430–2439, 2017. URL: http://proceedings.mlr.press/v70/mirhoseini17a.html.

34 Alix Munier. Approximation algorithms for scheduling trees with general communication delays. *Parallel Computing*, 25(1):41–48, 1999.

35 Alix Munier and Claire Hanen. Using duplication for scheduling unitary tasks on m processors with unit communication delays. *Theoretical Computer Science*, 178(1):119–127, 1997. doi:10.1016/S0304-3975(97)88194-7.

36 Alix Munier and Jean-Claude König. A heuristic for a scheduling problem with communication delays. *Operations Research*, 45(1):145–147, 1997.

37 Michael A. Palis, Jing-Chiou Liou, and David S. L. Wei. Task clustering and scheduling for distributed memory parallel architectures. *IEEE Transactions on Parallel and Distributed Systems*, 7(1):46–55, 1996.

38 Christos H. Papadimitriou and Mihalis Yannakakis. Scheduling interval-ordered tasks. *SIAM J. Comput.*, 8(3):405–409, 1979. doi:10.1137/0208031.

39 Christos H. Papadimitriou and Mihalis Yannakakis. Towards an architecture-independent analysis of parallel algorithms. *SIAM journal on computing*, 19(2):322–328, 1990.

40 Christophe Picouleau. *Two new NP-complete scheduling problems with communication delays and unlimited number of processors*. Inst. Blaise Pascal, Univ., 1991.

41 Maurice Queyranne and Maxim Sviridenko. Approximation algorithms for shop scheduling problems with minsum objective. *Journal of Scheduling*, 5(4):287–305, 2002. doi:10.1002/jos.96.

42 Rajmohan Rajaraman, David Stalfa, and Sheng Yang. Scheduling under non-uniform job and machine delays, 2022. arXiv:2207.13121.

43 Victor J Rayward-Smith. UET scheduling with unit interprocessor communication delays. *Discrete Applied Mathematics*, 18(1):55–71, 1987.

44 Juan A. Rico-Gallego, Juan C. Díaz-Martín, Ravi Reddy Manumachu, and Alexey L. Lastovetsky. A survey of communication performance models for high-performance computing. *ACM Computing Surveys*, 51(6), January 2019. doi:10.1145/3284358.

45 Petra Schuurman and Gerhard J. Woeginger. Polynomial time approximation algorithms for machine scheduling: ten open problems. *Journal of Scheduling*, 2(5):203–213, 1999.

46 Bastian Seifert, Chris Wendler, and Markus Püschel. Causal fourier analysis on directed acyclic graphs and posets. *CoRR*, abs/2209.07970, 2022. doi:10.48550/arXiv.2209.07970.

47 Ola Svensson. Conditional hardness of precedence constrained scheduling on identical machines. *Proceedings of the 42nd ACM symposium on Theory of computing – STOC '10*, pages 745–754, 2010. doi:10.1145/1806689.1806791.

48 Bart Veltman, B. J. Lageweg, and Jan Lenstra. Multiprocessor scheduling with communication delays. *Parallel Computing*, 16:173–182, 1990.

49 Laurens Versluis, Erwin Van Eyk, and Alexandru Iosup. An analysis of workflow formalisms for workflows with complex non-functional requirements. In *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*, pages 107–112, 2018.

50 Guangyuan Wu, Fangming Liu, Haowen Tang, Keke Huang, Qixia Zhang, Zhenhua Li, Ben Y. Zhao, and Hai Jin. On the performance of cloud storage applications with global measurement. In *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*, pages 1–10, 2016. doi:10.1109/IWQoS.2016.7590449.

Zero-Rate Thresholds and New Capacity Bounds for List-Decoding and List-Recovery

Nicolas Resch
Informatics' Institute, University of Amsterdam, The Netherlands

Chen Yuan
School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China

Yihan Zhang
Institute of Science and Technology Austria, Klosterneuburg, Austria

Abstract

In this work we consider the list-decodability and list-recoverability of arbitrary q-ary codes, for all integer values of $q \geqslant 2$. A code is called $(p, L)_q$-list-decodable if every radius pn Hamming ball contains less than L codewords; $(p, \ell, L)_q$-list-recoverability is a generalization where we place radius pn Hamming balls on every point of a combinatorial rectangle with side length ℓ and again stipulate that there be less than L codewords.

Our main contribution is to precisely calculate the maximum value of p for which there exist infinite families of positive rate $(p, \ell, L)_q$-list-recoverable codes, the quantity we call the *zero-rate threshold*. Denoting this value by p_*, we in fact show that codes correcting a $p_* + \varepsilon$ fraction of errors must have size $O_\varepsilon(1)$, i.e., independent of n. Such a result is typically referred to as a "Plotkin bound." To complement this, a standard random code with expurgation construction shows that there exist positive rate codes correcting a $p_* - \varepsilon$ fraction of errors. We also follow a classical proof template (typically attributed to Elias and Bassalygo) to derive from the zero-rate threshold other tradeoffs between rate and decoding radius for list-decoding and list-recovery.

Technically, proving the Plotkin bound boils down to demonstrating the Schur convexity of a certain function defined on the q-simplex as well as the convexity of a univariate function derived from it. We remark that an earlier argument claimed similar results for q-ary list-decoding; however, we point out that this earlier proof is flawed.

2012 ACM Subject Classification Mathematics of computing → Coding theory

Keywords and phrases Coding theory, List-decoding, List-recovery, Zero-rate thresholds

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.99

Category Track A: Algorithms, Complexity and Games

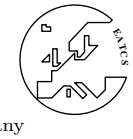
Related Version *Full Version*: `https://arxiv.org/abs/2210.07754` [46]

Funding *Nicolas Resch*: Research supported in part by ERC H2020 grant No.74079 (ALGSTRONGCRYPTO).
Chen Yuan: Research supported in part by the National Key Research and Development Projects under Grant 2022YFA1004900 and Grant 2021YFE0109900, the National Natural Science Foundation of China under Grant 12101403 and Grant 12031011.

Acknowledgements YZ is grateful to Shashank Vatedka, Diyuan Wu and Fengxing Zhu for inspiring discussions.

1 Introduction

Given a code $\mathcal{C} \subset [q]^n$, a fundamental problem of coding-theory is to determine how "well-spread" $\mathcal{C}$ can be if we also insist that $\mathcal{C}$ have large rate $R = \frac{\log_q |\mathcal{C}|}{n}$. The most basic way of quantifying "well-spread" is by insisting that all pairs of codewords are far apart. That is,



we hope that the minimum distance $d := \min\{d_{\mathrm{H}}(\boldsymbol{c}, \boldsymbol{c}') : \boldsymbol{c} \neq \boldsymbol{c}' \in \mathcal{C}\}$ is large, where $d_{\mathrm{H}}(\cdot, \cdot)$ denotes Hamming distance, i.e., the number of coordinates on which the two strings differ. Equivalently, given any word $\boldsymbol{y} \in [q]^n$, we have that $|\mathcal{B}_{\mathrm{H}}(\boldsymbol{y}, r) \cap \mathcal{C}| \leqslant 1$, where $r = \lfloor d/2 \rfloor$ and $\mathcal{B}_{\mathrm{H}}(\boldsymbol{y}, r) = \{\boldsymbol{x} \in [q]^n : d_{\mathrm{H}}(\boldsymbol{x}, \boldsymbol{y}) \leqslant r\}$ denotes the Hamming ball of radius r centered at $\boldsymbol{y}$.

One can naturally relax this requirement to the notion of list-decodability: instead of upper-bounding $|\mathcal{B}_{\mathrm{H}}(\boldsymbol{y}, r) \cap \mathcal{C}|$ by 1, we upper bound it by a larger integer $L - 1$.[1] Equivalently, if we place Hamming balls of radius r on each codeword of $\mathcal{C}$, no vector in $[q]^n$ is covered by L or more balls. If $\mathcal{C}$ satisfies this property we call it $(p, L)_q$-list-decodable. Initially introduced by Elias and Wozencraft in the 1950's [16, 50, 17], this relaxed notion of decoding has been intensively studied in recent years, in part motivated by purely coding-theoretic concerns, but also due to its connections with theoretical computer science more broadly [20, 3, 38, 37, 33, 47].

A further generalization of list-decoding is provided by *list-recoverability*. In this case, one considers tuples of input lists $\boldsymbol{\mathcal{Y}} = (\mathcal{Y}_1, \ldots, \mathcal{Y}_n)$ where each $\mathcal{Y}_i \subset [q]$ is of size at most ℓ, and the requirement is that the number of codewords $\boldsymbol{c}$ satisfying $|\{i \in [n] : c_i \notin \mathcal{Y}_i\}| \leqslant pn$ is at most $L - 1$. Such a code is deemed $(p, \ell, L)_q$-list-recoverable. Note that $(p, 1, L)_q$-list-recoverability is the same as $(p, L)_q$-list-decoding, demonstrating that list-recoverability is a more general notion. While it was originally defined as an abstraction required for the task of uniquely-/list-decoding concatenated codes [21, 22, 23, 24], it has since found myriad further applications in computer science more broadly, e.g., in cryptography [30, 31], randomness extraction [29], hardness amplification [14], group testing [32, 41], streaming algorithms [15], and beyond.

When it comes to list-decoding and list-recovery, the optimal tradeoff between decoding-radius p and rate R is well-understood if one is satisfied with list-sizes $L = O(1)$.[2] That is, there exist $(p, \ell, O(\ell/\varepsilon))_q$-list-recoverable codes of rate $1 - H_{q,\ell}(p) - \varepsilon$ where[3]

$$H_{q,\ell}(p) \coloneqq p \log_q \left(\frac{q - \ell}{p} \right) + (1 - p) \log_q \left(\frac{\ell}{1 - p} \right);$$

conversely, if the rate is at least $1 - H_{q,\ell}(p) + \varepsilon$ then it will not be list-recoverable for any $L = o\left(q^{\varepsilon n}\right)$ [44, Theorem 2.4.12]. (Note that setting $\ell = 1$ recovers the more well-known list-decoding capacity theorem.) While this already provides some "coarse-grained" information concerning the list-decodability/-recoverability of codes, it leaves many questions unanswered.

For example, one can ask about the maximum rate of a $(p, 3)_q$-list-decodable code. That is, what is the maximum rate of a code that never contains more than 2 points from a Hamming ball of radius pn? However, this question as stated appears be quite difficult to solve: any improvement for the special case of $L = 2$ and $q = 2$ would require improving either on the Gilbert-Varshamov bound [19, 48] (on the "possibility" side) or the linear programming bounds [49, 39, 13] (on the "impossibility" side). Unfortunately, despite decades of interest in this basic question hardly any asymptotic improvements on these bounds have been provided in the past fifty years.

Zero-rate thresholds for list-decoding and -recovery. We therefore begin by targeting a more modest question: what is the maximum $p_* = p_*(q, \ell, L)$ such that for any $p < p_*$ there exist infinite families of q-ary $(p, \ell, L)_q$-list-recoverable codes of positive rate? That is,

[1] We find it most convenient to let L denote 1 more than the list-size, which is admittedly nonstandard, but will make our computations much cleaner.

[2] Or indeed, if we insist on L just being subexponential.

[3] For $\ell = 1$, $H_{q,1}$ reduces to the q-ary entropy function denoted by H_q.

imagining the curve describing the achievable tradeoffs with the rate R on the y-axis and decoding radius p on the x-axis, instead of asking to describe this entire curve, we simply seek to determine the point where this curve crosses the x-axis (clearly, this curve is monotonically decreasing).

Over the binary alphabet, setting $\ell = 1$ and $L = 2$ in this question we recover a famous result of Plotkin [42]: the maximum fraction of errors that can be uniquely-decoded by an infinite family of positive rate binary codes is $1/4$. Over general q-ary alphabets, this value is similarly known to be $\frac{q-1}{2q}$ (folklore; see, e.g., [28, Theorem 4.4.1]). The value of $p_*(2, 1, L)$ has been computed by Blinovsky [5] for all L, and is known to be

$$p_*(2, 1, L) = \frac{1}{2} - \frac{\binom{2k}{k}}{2^{2k+1}} \text{ if } L = 2k \text{ or } L = 2k + 1.$$

While this expression is quite impenetrable at first glance, here is a natural probabilistic interpretation: given $x_1, \ldots, x_L \in \{0, 1\}$, let $\mathsf{pl}(x_1, \ldots, x_L)$ denote the number of times the more popular bit appears.[4] We then have

$$p_*(2, 1, L) = 1 - \frac{1}{L} \mathop{\mathbb{E}}_{(X_1,\cdots,X_L)\sim \mathrm{Bern}(1/2)^{\otimes L}} [\mathsf{pl}(X_1, \cdots, X_L)],$$

where the notation $(X_1, \cdots, X_L) \sim \mathrm{Bern}(1/2)^{\otimes L}$ denotes that L independent unbiased bits are sampled.

It is then not difficult to conjecture the value for $p_*(q, \ell, L)$: if $\mathsf{pl}_\ell(x_1, \ldots, x_L)$ denotes the top-ℓ-plurality value of $x_1, \ldots, x_L \in [q]$, i.e., $\mathsf{pl}_\ell(x_1, \ldots, x_L) = \max_{\Sigma\subseteq[q]:|\Sigma|=\ell} |\{i \in [L] : x_i \in \Sigma\}|$, then it should be that

$$p_*(q, \ell, L) = 1 - \frac{1}{L} \mathop{\mathbb{E}}_{(X_1,\cdots,X_L)\sim \mathrm{Unif}([q])^{\otimes L}} [\mathsf{pl}_\ell(X_1, \cdots, X_L)]. \tag{1}$$

This quantity is fairly natural: one can interpret it as the minimum radius of a list-recovery ball (i.e., a set of the form $\{\boldsymbol{v} \in [q]^n : v_i \in \mathcal{Y}_i \text{ for at least } (1-p)n \ i \in [n]\}$) that will contain L codewords in the "typical" case. For the case of $\ell = 1$, i.e., q-ary list-decoding, a proof is claimed in [6, 7]; however, as we outline in Section 3 this proof is flawed. In this work we provide a rigorous derivation of Equation (1) for all values of ℓ, L and q with $1 \leqslant \ell \leqslant q$.

More precisely, we obtain the following results:

- A proof that $(p, \ell, L)_q$-list-recoverable q-ary codes with $p > p_*(q, \ell, L)$ have *constant-size*, i.e., independent of n. This should be interpreted as a generalization of the Plotkin bound [42], which states that binary codes uniquely-decodable from a $1/4 + \varepsilon$ fraction of errors have size at most $O(1/\varepsilon)$. For this reason we call our result a "Plotkin bound for list-recovery."
- Adapting the Elias-Bassalygo argument [4], we subsequently derive upper bounds on the rate of $(p, \ell, L)_q$-list-recoverable q-ary codes when $p < p_*(q, \ell, L)$.
- To complement this, we show that there exist infinite families of positive rate q-ary codes that are $(p, \ell, L)_q$-list-recoverable whenever $p < p_*(q, \ell, L)$. We are therefore justified in calling $p_*(q, \ell, L)$ the zero-rate threshold for list-recovery.

We now describe our techniques in more detail.

[4] We use pl to stand for "plurality". However, we caution that this function does not output a most popular symbol (as is perhaps more in line with the standard meaning of plurality), but the number of $i \in [L]$ for which x_i equals a most popular symbol.

1.1 Our techniques

Schur convexity of the function $f_{q,L,\ell}$. Following prior work [6],[5] our task requires us to answer the following question. Consider the function on distributions P over the alphabet $[q]$ defined as

$$f_{q,L,\ell}(P) \coloneqq \mathop{\mathbb{E}}_{(X_1,\cdots,X_L)\sim P^{\otimes L}} \left[\mathsf{pl}_\ell(X_1,\cdots,X_L)\right].$$

Analogously to before, the notation $(X_1,\cdots,X_L)\sim P^{\otimes L}$ means that L independent samples are taken from the distribution P. A crucial ingredient for deriving the Plotkin bound is a demonstration that this function is minimized by the uniform distribution.

There is a well-studied class of functions on finite distributions with the property that they are minimized by the uniform distribution: *Schur convex* functions. These are the functions that are monotonically-increasing with respect to the *majorization*-ordering, which compares vectors of real numbers by first sorting the vectors in descending order and then checking to see if all the prefix sums of one vector is greater than or equal to the prefix sums of the other. The important detail for us is that the uniform vector $(1/q,\ldots,1/q)\in\mathbb{R}^q$, corresponding to the uniform distribution, is majorized by *every* other vector corresponding to a distribution over $[q]$.

To demonstrate the Schur convexity of this function, we use the Schur-Ostrowski criterion, which states that Schur-convexity is equivalent to the non-negativity of a certain expression involving partial derivatives. Showing that this expression is non-negative boils down to a combinatorial accounting game, where we can show that the positive contributions arising from certain terms exceed the negative contributions arising from others.

Convexity of the univariate function $g_{q,L,\ell}$. Another important technical ingredient that we need for the proof of the Plotkin bound is the convexity of the univariate function

$$g_{q,L,\ell}(w) \coloneqq f_{q,L,\ell}(P_{q,\ell,w}),$$

where the distribution $P_{q,\ell,w}=(p_1,\ldots,p_q)$ is defined as

$$p_i = \begin{cases} \frac{w}{q-\ell} & \text{if } i \leqslant q-\ell \\ \frac{1-w}{\ell} & \text{if } i \geqslant q-\ell+1 \end{cases}.$$

In order to show the function is convex, we prove the second derivative is non-negative. In differentiating, we use the expression for $g_{q,L,\ell}$ in terms of $f_{q,L,\ell}$ and apply the chain rule. Showing the resulting expression is positive is again a sort of combinatorial accounting game: we can show the positive terms contribute more than the negative terms.

Quite interestingly, for $\ell=1$ (i.e., the case relevant for list-decoding) we only prove the convexity of the function $f_{q,1,L}$ on the interval $[0,(q-1)/q]$. Fortunately, as we can also easily show that $g_{q,1,L}$ decreases on the interval $[0,(q-1)/q]$ and then increases on the interval $[(q-1)/q,1]$,[6] convexity of $f_{q,1,L}$ on $[0,(q-1)/q]$ suffices for our purposes. And indeed, this is not an artifact of the proof: Blinovsky had already observed that convexity of $f_{q,1,L}$ does not hold on the entire interval $[0,1]$ [6, 7]. However, for $\ell\geqslant 2$ we obtain that convexity of $f_{q,\ell,L}$ does indeed hold on the entire interval $[0,1]$. We note that the second derivative does behave qualitatively differently, so this is perhaps not too surprising in hindsight; we comment on this further in [46, Remark 5].

---

[5] In fact, [6] only considers list-decoding, so a slight adaptation of this argument is required for list-recovery.
[6] This is in fact an easy corollary of the Schur convexity of $f_{q,1,L}$.

Plotkin bound. Armed with these (Schur-)convexity results, we aim to prove a Plotkin bound for list-decoding/-recovery. That is, if a q-ary code is $(p, \ell, L)_q$-list-recoverable with $p \geqslant p_*(q, \ell, L) + \varepsilon$, how large can the code be? Following the template of the standard argument (although certain subtleties arise when generalizing to list-recovery), we can show that such a code must be of constant size, i.e., independent of n.

Informally, the argument begins with a "preprocessing step" that prunes away some (but, crucially, not too many) codewords and yields a more structured subcode that we can subsequently analyze. The codewords of this subcode are very "balanced" in the sense that all patterns of symbols appear with roughly the same frequency. In particular, every pattern of length t should appear roughly a $1/q^t$ fraction of the time (or the code is very "biased," in which case a separate argument bounds its size).

To analyze this subcode $\mathcal{C}'$ we apply a double-counting argument to the average radius (see [46, Definition 10]) to cover L-subsets (where for list-recoverability, this radius is measured via the distance to a tuple of input lists). The lower bound on this quantity follows quite naturally from the list-decodability/-recoverability of the code, together with the "balancedness" of the subcode. For the upper bound, we compute the radius of an L-subset in terms of the empirical distribution of a coordinate $k \in [n]$, i.e., each $x \in [q]$ is assigned probability mass $P_k(x) = \frac{1}{M} \sum_{\boldsymbol{x} \in \mathcal{C}'} \mathbb{1}\{x_k = x\}$. By the Schur convexity of the function $f_{q,L,\ell}$ and the convexity of the univariate function $g_{q,L,\ell}$, we can bound this in terms of a distribution placing total mass $w \leqslant \frac{q-\ell}{q}$ on the last ℓ elements of $[q]$ and mass $\frac{1-w}{q-\ell}$ on each of the others. The result then follows.

We remark that, due to our use of Ramsey-theoretic arguments, the precise bound we obtain on the code size is quite poor. We have made no effort to optimize this constant. However, we do believe it would be interesting to improve this bound; we discuss this further in Section 4.

Elias-Bassalygo-style bound. After deriving this Plotkin bound, a well-known argument template (typically attributed to Elias and Bassalygo [4]) allows one to derive more general tradeoffs between the rate R and the noise-resilience parameters $(p, \ell, L)_q$. Informally, this proceeds by covering the space $[q]^n$ by a bounded number of list-recovery balls. The radius of these balls is carefully chosen to allow one to apply the Plotkin bound to the subcodes obtained by taking the intersection of the code with these balls. On the other hand, the number of list-recovery balls needed to cover $[q]^n$, known as the covering number, can be sharply estimated. From the above two bounds (the Plotkin bound and the covering number), a bound on the size of the whole code can be derived.

Possibility result: random code with expurgation. To complement the Plotkin bound, we show that if the decoding radius p is less than $p_*(q, \ell, L)$ then there exist infinite families of $(p, \ell, L)_q$-list-recoverable q-ary codes. This justifies our "zero-rate threshold" terminology for $p_*(q, \ell, L)$. The argument is completely standard, obtained by sampling a random code and subsequently expurgating codewords to destroy all size-L lists that can fit into Hamming balls of radius np. In fact, the lower bound on achievable rate is derived from the exact large deviation exponent of a certain quantity known as the average radius (cf. [46, Definition 12]) of a tuple of random vectors. Therefore the bound holds under a stronger notion called average-radius list-recovery: namely, for any subset of L codewords $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_j$ and any tuple of input lists $(\mathcal{Y}_1, \ldots, \mathcal{Y}_n)$, we have

$$\sum_{j=1}^{L} |\{i \in [n] : x_{j,i} \notin \mathcal{Y}_i\}| > Lpn \ .$$

1.2 Discussion on related work

Lower bounds for small q and/or L. For the case of $(p,3)_2$-list-decoding, it was shown in [26, Theorem 6.1] that the *threshold rate*[7] of random binary *linear* codes equals

$$\frac{1}{2}(2 - H_2(3p) - 3p\log_2(3)). \tag{2}$$

The term *threshold* refers to the critical rate below which a random binary linear code is $(p,3)_2$-list-decodable with high probability and above which it is not with high probability. This result was recently extended to the following two cases [45]. For $(p,4)_2$-list-decoding, the threshold rate of random binary linear code is lower bounded by [45, Theorem 1.3]

$$\frac{1}{3}\min_{\substack{x_1,x_2\geqslant 0\\ x_1+2x_2\leqslant 4p\\ x_1+x_2\leqslant 1}} 3 - \eta_2(x_1,x_2) - 2x_1 - x_2\log_2(3). \tag{3}$$

Here we use the notation

$$\eta_q(x_1,\ldots,x_t) \coloneqq \sum_{i=1}^{t} x_i \log_q \frac{1}{x_i} + \left(1 - \sum_{i=1}^{t} x_i\right)\log_q \frac{1}{1-\sum_{i=1}^{t} x_i}$$

for a partial probability vector $(x_1,\ldots,x_t) \in \mathbb{R}^t_{\geqslant 0}$ satisfying $t \leqslant q$ and $x_1 + \cdots + x_t \leqslant 1$. Note that $\eta_2(x) = H_2(x)$, however, this is no longer the case for $q > 2$. Moreover, for $(p,3)_q$-list-decoding, [45, Theorem 1.5] showed that the threshold rate of random linear code is at least

$$\frac{1}{2}\min_{\substack{x_1,x_2\geqslant 0\\ x_1+2x_2\leqslant 3p\\ x_1+x_2\leqslant 1}} 2 - \eta_q(x_1,x_2) - x_1\log_q(3(q-1)) - x_2\log_q(q-1)(q-2). \tag{4}$$

Our general lower bound (cf. Theorem 14) for list-recovery (numerically) matches Equations (2)–(4) upon particularizing the parameters q,ℓ,L suitably. See Figures 1a–1c. It is possible to analytically prove this observation, though we do not pursue it in the current paper. The rationale underlying this phenomenon is that the threshold rate of random linear codes for list-recovery is expected to match the rate achieved by random codes with expurgation (with the notable exception of zero-error list-recovery [25]). This conjecture, in its full generality, remains unproved, although it is partially justified in several recent works [40, 25, 26, 45].

Hash codes. One may note that for $\ell \geqslant 2$, our upper and lower bounds typically exhibit a large gap even at $p = 0$. See Figures 1e–1h. We provide evidence below indicating that closing this gap is in general a rather challenging task and necessarily requires significantly new ideas. Let us focus on the vertical axis $p = 0$, known as *zero-error list-recovery*. We observe that some configurations of q,ℓ,L in this regime encode several longstanding open questions in combinatorics. Indeed, consider $\ell = q-1, L = q$. The $(0,q-1,q)_q$-list recoverability condition can then be written as: for any $\mathcal{Y}_1,\cdots,\mathcal{Y}_n \in \binom{[q]}{q-1}$,

$$|\{\boldsymbol{x} \in \mathcal{C} : |\{j \in [n] : x_j \notin \mathcal{Y}_j\}| = 0\}| \leqslant L-1,$$

[7] We warn the reader not not to confuse this concept with that of the zero-rate threshold.

i.e.,

$$|\{\boldsymbol{x} \in \mathcal{C} : \forall j \in [n],\, x_j \in \mathcal{Y}_j\}| \leqslant L - 1.$$

Taking the contrapositive, we note that this condition is further equivalent to: for any $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_L\} \in \binom{\mathcal{C}}{L}$, there exists $j \in [n]$ such that $|\{x_{1,j}, \cdots, x_{L,j}\}| = q$. In words, for any q-tuple of codewords in a $(0, q-1, q)_q$-list-recoverable code, there must exist one coordinate such that the corresponding q-ary symbols in the tuple are all distinct. Such a code is also known as a *q-hashing* in combinatorics. It is well-known [18, 35] that a probabilistic construction yields such codes of rate[8] at least

$$C_{(0,q-1,q)_q} \geqslant \frac{1}{q-1} \log_q \frac{1}{1 - \frac{q!}{q^q}}. \tag{5}$$

In the same paper [18] also proved an upper bound

$$C_{(0,q-1,q)_q} \leqslant \frac{q!}{q^{q-1}} \log_q(2). \tag{6}$$

Another upper bound

$$C_{(0,q-1,q)_q} \leqslant \log_q \frac{q}{q-1} \tag{7}$$

can be proved using either a double-counting argument (a.k.a. first moment method), or (hyper)graph entropy [35, 36, 34]. Equation (6) is much better than Equation (7) for $q \geqslant 4$. However, the latter bound $\log_3 \frac{3}{2}$ remains the best known for $q = 3$ (called the *trifference problem* by Körner). For larger q, both lower [51] and upper bounds [2, 11, 27, 10, 12] can be improved. However, improving the bound for $q = 3$ is recognized as a formidable challenge. We will show in [46, Remark 9] that our lower bound for list-recovery (cf. Theorem 14) recovers Equation (5) for q-hashing upon setting $\ell = q - 1, L = q$. Furthermore, our upper bound Theorem 16 recovers Equation (7) for q-hashing (cf. [46, Remark 7]).

A generalization of q-hashing known as *(q, L)-hashing* ($q \geqslant L$) can also be cast as zero-error list-recoverable codes with more general values of ℓ, L. Indeed, taking $L = \ell + 1$ and $\ell \leqslant q - 1$, we can write $(0, \ell, \ell + 1)_q$-list-recoverability alternatively as: for any $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{\ell+1}\} \in \binom{\mathcal{C}}{\ell+1}$, there exists $j \in [n]$ such that $|\{x_{1,j}, \cdots, x_{\ell+1,j}\}| = \ell + 1$. This is in turn the precise definition of $(q, \ell + 1)$-hashing. It can be immediately seen that (q, q)-hashing is nothing but q-hashing. The upper and lower bounds in [18] also extend to $(q, \ell + 1)$-hashing and read as follows:

$$\frac{1}{\ell} \log_q \frac{1}{1 - \frac{\binom{q}{\ell+1}(\ell+1)!}{q^{\ell+1}}} \leqslant C_{(0,\ell,\ell+1)_q} \leqslant \frac{\binom{q}{\ell}\ell!}{q^\ell} \log_q(q - \ell + 1). \tag{8}$$

Our lower bound for list-recovery in Theorem 14 also recovers the above lower bound for $(q, \ell + 1)$-hashing by [18] upon setting $L = \ell + 1$ (see [46, Remark 8]). The upper bound was later improved in [36] for $q > L$ using the notion of hypergraph entropy:

$$C_{(0,\ell,\ell+1)_q} \leqslant \min_{0 \leqslant j \leqslant \ell-1} \frac{\binom{q}{j+1}(j+1)!}{q^{j+1}} \log_q \frac{q-j}{\ell-j}, \tag{9}$$

though it coincides with Equation (6) when $\ell = q - 1$. Some improved upper bounds in [27, 12] apply to $(q, \ell + 1)$-hashing as well. To the best of our knowledge, no improvement on lower bounds is known for $\ell < q - 1$.

[8] The bounds in [36, 18] are slightly adjusted so that they are consistent with our definition of code rate which adopts a $\log_q$ normalization (cf. [46, Definition 6]).

Zero-rate thresholds for general adversarial channels. The problem of locating the zero-rate threshold has been addressed in a much more general context [52]. The results in [52] on *general adversarial channel* model can be specialized to the list-recovery setting and read as follows. Given q, p, ℓ, L, define the *confusability set* $\mathcal{K}_{(p,\ell,L)_q}$ as the set of types[9] (cf. [46, Definition 15]) of all "confusable" L-tuple of codewords in the sense that they can fit into a certain list-recovery ball (cf. [46, Definition 3]) of radius np. Specifically,

$$\mathcal{K}_{(p,\ell,L)_q} \coloneqq \left\{ \sum_{\mathcal{Y}\in\binom{[q]}{\ell}} P_{X_1,\cdots,X_L,Y=\mathcal{Y}} \in \Delta\left([q]^L\right) : \begin{array}{c} P_{X_1,\cdots,X_L,Y} \in \Delta\left([q]^L \times \binom{[q]}{\ell}\right) \\ \forall i \in [L], \quad \sum_{\substack{(x,\mathcal{Y})\in[q]\times\binom{[q]}{\ell} \\ x\notin\mathcal{Y}}} P_{X_i,Y}(x,\mathcal{Y}) \leqslant p \end{array} \right\}.$$

In the above definition, we use the notation $\sum_b P_{A,B=b}$ to denote the marginalization of $P_{A,B}$ onto the first variable A, and use $P_{X_i,Y}$ to denote the marginal of $P_{X_1,\cdots,X_i,Y}$ on (X_i, Y). It is not hard to verify that the confusability set is (i) "increasing" in p in the sense that $\mathcal{K}_{(p,\ell,L)_q} \subset \mathcal{K}_{(p',\ell,L)_q}$ if $p \leqslant p'$, and (ii) convex. Define also the convex cone of *completely positive (CP) tensors* of order L, i.e., tensors that can be written as a sum of *element-wise non-negative* rank-one tensors:

$$\mathsf{CP}_q^{\otimes L} \coloneqq \left\{ \sum_{i=1}^{k} \boldsymbol{p}_i^{\otimes L} \in (\mathbb{R}_{\geqslant 0}^q)^{\otimes L} : k \in \mathbb{Z}_{\geqslant 1}, (\boldsymbol{p}_1, \cdots, \boldsymbol{p}_k) \in (\mathbb{R}_{\geqslant 0}^q)^k \right\}.$$

It is proved in [52] that the zero-rate threshold $p_*(q, \ell, L)$ can be expressed as the smallest p such that all completely positive distributions are confusable:

$$p_*(q, \ell, L) = \inf\left\{ p \in [0,1] : \mathsf{CP}_q^{\otimes L} \cap \Delta([q]^L) \subset \mathcal{K}_{(p,\ell,L)_q} \right\}. \tag{10}$$

The above characterization is *single-letter* in the sense that it is independent of the blocklength n. For q, ℓ, L independent of n (which is assumed to be the case in the current paper), the optimization problem on the RHS of Equation (10) can be solved in constant time. However, it does not immediately provide an explicit formula of $p_*(q, \ell, L)$ and analytically solving the optimization problem does not appear easy to the authors. On the other hand, the characterization $p_*(q, \ell, L) = 1 - \frac{1}{L}\mathbb{E}\left[\mathsf{pl}_\ell(X_1, \cdots, X_L)\right]$ (where the expectation is over $(X_1, \cdots, X_L) \sim \mathrm{Unif}([q])^{\otimes L}$, cf. Equation (1)) in this paper can be seen as the explicit solution to the optimization problem, though the way it is obtained is *not* by solving the latter problem per se. Instead, we prove the characterization from the first principle by leveraging specific structures of list-recovery. We hope that our characterization can shed light on the geometry of the high-dimensional polytopes – the confusability set and the set of CP distributions – involved in the characterization in Equation (10).

1.3 Organization

We state our main results in Section 2. We discuss the flaw in Blinovsky's proof in Section 3. We summarize our results and state open problems in Section 4. Additional notation, definitions, preliminary results and missing proofs can be found in [46].

[9] More precisely, the confusability set is the *closure* of the set of types of all confusable codeword tuples, since types are dense in distributions.

2 Main results

2.1 q-ary list-decoding

Define $f_{q,L}\colon \Delta([q]) \to \mathbb{R}_{\geqslant 0}$ as

$$f_{q,L}(P) := \mathop{\mathbb{E}}_{(X_1,\ldots,X_L)\sim P^{\otimes L}} [\mathsf{pl}(X_1,\ldots,X_L)] \tag{11}$$

for $P \in \Delta([q])$.

For $w \in [0,1]$, let $P_{q,w} \in \Delta([q])$ denote the following probability vector:

$$P_{q,w} := \left(\frac{w}{q-1},\ldots,\frac{w}{q-1},1-w\right). \tag{12}$$

Define $g_{q,L}\colon [0,1] \to \mathbb{R}_{\geqslant 0}$ as

$$g_{q,L}(w) := f_{q,L}(P_{q,w}). \tag{13}$$

▶ **Definition 1** (Majorization). *Let $\boldsymbol{a},\boldsymbol{b} \in \mathbb{R}^d$. Let $\boldsymbol{a}^{\downarrow},\boldsymbol{b}^{\downarrow} \in \mathbb{R}^d$ denote the vectors obtained by sorting the elements in $\boldsymbol{a}$ and $\boldsymbol{b}$ in descending order, respectively. We say that $\boldsymbol{a}$* majorizes *$\boldsymbol{b}$, written as $\boldsymbol{a} \succeq \boldsymbol{b}$, if*

$$\sum_{i=1}^{k} a_i^{\downarrow} \geqslant \sum_{i=1}^{k} b_i^{\downarrow}$$

for every $k \in [d]$, and

$$\sum_{i=1}^{d} a_i = \sum_{i=1}^{d} b_i.$$

▶ **Definition 2** (Schur convexity). *A function $f\colon \mathbb{R}^d \to \mathbb{R}$ is called* Schur-convex *if $f(\boldsymbol{x}) \geqslant f(\boldsymbol{y})$ for every $\boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^d$ such that $\boldsymbol{x} \succeq \boldsymbol{y}$ (in the sense of Definition 1).*

▶ **Theorem 3** (Schur convexity of $f_{q,L}$). *For any $q \in \mathbb{Z}_{\geqslant 2}$ and $L \in \mathbb{Z}_{\geqslant 2}$, the function $f_{q,L}\colon \Delta([q]) \to \mathbb{R}_{\geqslant 0}$ defined in Equation (11) is Schur convex.*

Proof. See [46, Sec. 4]. ◀

▶ **Theorem 4** (Convexity of $g_{q,L}$). *For any $q \in \mathbb{Z}_{\geqslant 2}$ and $L \in \mathbb{Z}_{\geqslant 2}$, the function $g_{q,L}\colon [0,1] \to \mathbb{R}_{\geqslant 0}$ defined in Equation (13) is convex in the interval $[0,(q-1)/q]$.*

Proof. See [46, Sec. 5]. ◀

▶ Remark 5. In the binary case (i.e., $q = 2$), understanding the functions $f_{2,L}$ and $g_{2,L}$ is an easier task. In fact, $f_{2,L}$ collapses to a univariate function and coincides with $g_{2,L}$. It can be computed [8, Eqn. (2.15) and (2.16)] that for $L = 2k, 2k+1$,

$$p_*(2,L;w) := 1 - \frac{1}{L} g_{2,L}(w) = \sum_{i=1}^{k} \frac{\binom{2i-2}{i-1}}{i} (w(1-w))^i,$$

and

$$\frac{\partial^2}{\partial w^2} p_*(2,L;w) = -k\binom{2k}{k}(w(1-w))^{k-1}.$$

The concavity (see also [43, Lemma 8]) and monotonicity of $p_*(2,L;w)$ immediately follow. Such explicit computation cannot be performed in the $q > 2$ case (and for list-recovery) and we have to work with summations like in [46, Lemma 14]. Other approaches to arguing monotonicity such as induction [1, Lemma 8(d)] do not seem to work well either for larger q.

As convexity only holds in the interval $[0,(q-1)/q]$, we will also require the following monotonicity properties, which follow easily from the Schur convexity of $f_{q,L}$.

▶ **Lemma 6.** *For any $q \in \mathbb{Z}_{\geqslant 2}$ and $L \in \mathbb{Z}_{\geqslant 2}$, the function $g_{q,L}\colon [0,1] \to \mathbb{R}_{\geqslant 0}$ defined in Equation (13) is non-increasing on $[0,(q-1)/q]$ and non-decreasing on $[(q-1)/q,1]$.*

Proof. See [46, Appendix B]. ◀

Define

$$p_*(q,L;w) \coloneqq 1 - \frac{1}{L} g_{q,L}(w). \tag{14}$$

▶ **Theorem 7** (Plotkin bound for q-ary list-decoding)**.** *Fix any $q \in \mathbb{Z}_{\geqslant 2}$ and $L \in \mathbb{Z}_{\geqslant 2}$. Let $\mathcal{C} \subset [q]^n$ be an arbitrary $(p,L)_q$-list-decodable code with $p = p_*\left(q,L;\frac{q-1}{q}\right) + \tau$ for any constant $\tau \in (0,1)$. Then there exists a constant $M_* = M_*(q,L,\tau)$ independent of n such that $|\mathcal{C}| \leqslant M_*$. As a consequence, in particular, we have*

$$p_*(q,L) \leqslant p_*\left(q,L;\frac{q-1}{q}\right) = 1 - \frac{1}{L} g_{q,L}\left(\frac{q-1}{q}\right).$$

Proof. The proof of this theorem can be found in [46, Sec. 6]. Specifically, a theorem (cf. [46, Theorem 16]) of the above kind will be first proved for *approximately constant-weight* codes in which all codewords have approximately the same Hamming weight. This theorem can then be used to prove Theorem 7 above (see [46, Corollary 18] for a more quantitative version) by partitioning a general (weight-unconstrained) code into a constant number of almost constant-weight subcodes. ◀

The upper bound on the zero-rate threshold in Theorem 7 is in fact sharp. It turns out that positive rate $(p,L)_q$-list-decodable codes exist for any p strictly smaller than the bound $1 - \frac{1}{L} g_{q,L}\left(\frac{q-1}{q}\right)$ in Theorem 7. Indeed, Blinovsky [6] proved the following lower bound on the $(p,L)_q$-list-decoding capacity which remains the best known to date. It can also be implied by our lower bound (Theorem 14 below) for list-recovery upon setting $\ell = 1$.

▶ **Theorem 8** ([6, Sec. 2])**.** *For any $q \in \mathbb{Z}_{\geqslant 2}$, $L \in \mathbb{Z}_{\geqslant 2}$ and $0 \leqslant p < p_*\left(q,L;\frac{q-1}{q}\right)$, the following lower bound on the $(p,L)_q$-list-decoding capacity holds:*

$$C_{(p,L)_q} \geqslant \frac{L}{L-1} - \frac{1}{L-1}\left\{\lambda_* p + \log_q\left[\sum_{\boldsymbol{a} \in \mathcal{A}_{q,L}} \binom{L}{\boldsymbol{a}} \exp_q\left(-\lambda_*\left(1 - \frac{1}{L}\max\{\boldsymbol{a}\}\right)\right)\right]\right\},$$

where $\lambda_* = \lambda_*(q,L,p)$ is the solution to the following equation

$$p = \frac{\sum_{\boldsymbol{a} \in \mathcal{A}_{q,L}} \binom{L}{\boldsymbol{a}} \exp_q\left(-\lambda_*\left(1 - \frac{1}{L}\max\{\boldsymbol{a}\}\right)\right)\left(1 - \frac{1}{L}\max\{\boldsymbol{a}\}\right)}{\sum_{\boldsymbol{a} \in \mathcal{A}_{q,L}} \binom{L}{\boldsymbol{a}} \exp_q\left(-\lambda_*\left(1 - \frac{1}{L}\max\{\boldsymbol{a}\}\right)\right)}.$$

Blinovsky's lower bound is plotted in Figure 1d. It is not hard to verify that the lower bound above vanishes at

$$p = q^{-L} \sum_{\boldsymbol{a} \in \mathcal{A}_{q,L}} \binom{L}{\boldsymbol{a}} \left(1 - \frac{1}{L}\max\{\boldsymbol{a}\}\right),$$

and the corresponding λ_* equals 0.

Theorems 7 and 8 together pin down the exact value of $p_*(q,L)$ shown in the following corollary.

▶ **Corollary 9.** *For any $q \in \mathbb{Z}_{\geqslant 2}$ and $L \in \mathbb{Z}_{\geqslant 2}$, the zero-rate threshold $p_*(q, L)$ for $(p, L)_q$-list-decoding is given by*

$$p_*(q, L) = p_*\left(q, L; \frac{q-1}{q}\right) = 1 - \frac{1}{L} g_{q,L}\left(\frac{q-1}{q}\right) = q^{-L} \sum_{\boldsymbol{a} \in \mathcal{A}_{q,L}} \binom{L}{\boldsymbol{a}} \left(1 - \frac{1}{L}\max\{\boldsymbol{a}\}\right). \tag{15}$$

From now on, we will use $p_*(q, L)$ to denote the RHS of Equation (15).

▶ **Theorem 10** (Elias–Bassalygo bound for q-ary list-decoding)**.** *Fix any $q \in \mathbb{Z}_{\geqslant 2}$, $L \in \mathbb{Z}_{\geqslant 2}$ and $0 \leqslant p < p_*(q, L)$. Then the $(p, L)_q$-list-decoding capacity can be upper bounded as $C_{(p,L)_q} \leqslant 1 - H_q(w_{q,L})$ where $w_{q,L}$ is the solution to the equation $p_*(q, L; w) = p$ in $w \in [0, (q-1)/q]$.*

Proof. The above theorem is implied by [46, Theorem 19] proved in [46, Sec. 7]. The latter theorem shows that for any $(p, L)_q$-list-decodable code $\mathcal{C} \subset [q]^n$ with $p < p_*(q, L)$ and any sufficiently small constant $\tau > 0$, $|\mathcal{C}|$ is at most $B \cdot n^{1.5} \cdot q^{n(1-H_q(w_{q,L,\tau}))}$, where $B = B(q, L, \tau)$ is a constant and $w_{q,L,\tau}$ is the solution to $p_*(q, L; w) = p - \tau$. Taking $\tau \to 0$ and neglecting polynomial factors, we obtain the upper bound on the list-decoding capacity. ◀

The above upper bound is plotted in Figure 1d.

2.2 List-recovery

Define $f_{q,L,\ell} \colon \Delta([q]) \to \mathbb{R}_{\geqslant 0}$ as

$$f_{q,L,\ell}(P) \coloneqq \mathop{\mathbb{E}}_{(X_1,\ldots,X_L) \in P^{\otimes L}} [\mathsf{pl}_\ell(X_1, \ldots, X_L)] \tag{16}$$

for $P \in \Delta([q])$. Define $g_{q,L,\ell} \colon [0, 1] \to \mathbb{R}_{\geqslant 0}$ as

$$g_{q,L,\ell}(w) \coloneqq f_{q,L,\ell}(P_{q,\ell,w}), \tag{17}$$

where the distribution $P_{q,\ell,w} \in \Delta([q])$ is defined as

$$P_{q,\ell,w}(i) = \begin{cases} \frac{w}{q-\ell}, & 1 \leqslant i \leqslant q - \ell \\ \frac{1-w}{\ell}, & q - \ell + 1 \leqslant i \leqslant q \end{cases}. \tag{18}$$

▶ **Theorem 11** (Schur convexity of $f_{q,L,\ell}$)**.** *For any $q \in \mathbb{Z}_{\geqslant 2}$, $L \in \mathbb{Z}_{\geqslant 2}$ and integer $1 \leqslant \ell \leqslant q-1$, the function $f_{q,L,\ell} \colon \Delta([q]) \to \mathbb{R}_{\geqslant 0}$ defined in Equation (16) is Schur convex.*

Proof. See [46, Sec. 8]. ◀

▶ **Theorem 12** (Convexity of $g_{q,L,\ell}$)**.** *For any $q \in \mathbb{Z}_{\geqslant 2}$, $L \in \mathbb{Z}_{\geqslant 2}$ and integer $2 \leqslant \ell \leqslant q-1$, the function $g_{q,L,\ell} \colon \Delta([q]) \to \mathbb{R}_{\geqslant 0}$ defined in Equation (17) is convex in the interval $w \in [0, 1]$.*

Proof. See [46, Sec. 9]. ◀

Define

$$p_*(q, \ell, L; w) \coloneqq 1 - \frac{1}{L} g_{q,L,\ell}(w). \tag{19}$$

▶ **Theorem 13** (Plotkin bound for list-recovery)**.** *Fix any $q \in \mathbb{Z}_{\geqslant 2}$, $L \in \mathbb{Z}_{\geqslant 2}$ and integer $2 \leqslant \ell \leqslant q-1$. Let $\mathcal{C} \subset [q]^n$ be an arbitrary $(p, \ell, L)_q$-list-recoverable code with $p = p_* \left(q, \ell, L; \frac{q-\ell}{q}\right) + \tau$ for any constant $\tau \in (0,1)$. Then there exists a constant $M_* = M_*(q, \ell, \tau)$ independent of n such that $|\mathcal{C}| \leqslant M_*$. This implies, in particular,*

$$p_*(q, \ell, L) \leqslant p_* \left(q, \ell, L; \frac{q-\ell}{q}\right) = 1 - \frac{1}{L} g_{q,L,\ell} \left(\frac{q-\ell}{q}\right).$$

Proof. The proof structure is similar to that of Theorem 7. We first prove the analogous the statement for almost constant-weight codes (in which all codewords have approximately the same list-recovery weight) in [46, Theorem 20] and then pass to general codes by weight partitioning (cf. [46, Corollary 21]). Since the technical proofs bear many similarities to those in the list-decoding case, we only present proof sketches in [46, Sec. 10]. ◀

To complement Theorem 13, we prove in [46, Sec. 12] the following lower bound on the $(p, \ell, L)_q$-list-recovery capacity. To the best of our knowledge, this is the first bound for list-recovery with q, ℓ, L all being *constants* (independent of p and n). We believe that improving it likely requires novel techniques beyond expurgation.

▶ **Theorem 14.** *For any $q \in \mathbb{Z}_{\geqslant 2}$, $L \in \mathbb{Z}_{\geqslant 2}$, integer $2 \leqslant \ell \leqslant q-1$ and $0 \leqslant p < p_* \left(q, \ell, L; \frac{q-\ell}{q}\right)$, the following lower bound on the $(p, \ell, L)_q$-list-recovery capacity holds:*

$$C_{(p,\ell,L)_q} \geqslant \frac{L}{L-1} - \frac{1}{L-1} \left\{ \lambda_* p + \log_q \left[\sum_{\boldsymbol{a} \in \mathcal{A}_{q,L}} \binom{L}{\boldsymbol{a}} \exp_q \left(-\lambda_* \left(1 - \frac{1}{L} \max_\ell \{\boldsymbol{a}\} \right) \right) \right] \right\},$$

where $\lambda_* = \lambda_*(q, \ell, L, p)$ is the solution to the following equation

$$p = \frac{\sum_{\boldsymbol{a} \in \mathcal{A}_{q,L}} \binom{L}{\boldsymbol{a}} \exp_q \left(-\lambda_* \left(1 - \frac{1}{L} \max_\ell \{\boldsymbol{a}\} \right) \right) \left(1 - \frac{1}{L} \max_\ell \{\boldsymbol{a}\} \right)}{\sum_{\boldsymbol{a} \in \mathcal{A}_{q,L}} \binom{L}{\boldsymbol{a}} \exp_q \left(-\lambda_* \left(1 - \frac{1}{L} \max_\ell \{\boldsymbol{a}\} \right) \right)}.$$

Similar to the list-decoding case (Theorem 8), the above lower bound vanishes at

$$p = q^{-L} \sum_{\boldsymbol{a} \in \mathcal{A}_{q,L}} \binom{L}{\boldsymbol{a}} \left(1 - \frac{1}{L} \max_\ell \{\boldsymbol{a}\} \right),$$

and the corresponding λ_* equals 0.

Theorems 13 and 14 jointly determine the value of $p_*(q, \ell, L)$ shown in the corollary below.

▶ **Corollary 15.** *For any $q \in \mathbb{Z}_{\geqslant 2}$, $L \in \mathbb{Z}_{\geqslant 2}$ and integer $2 \leqslant \ell \leqslant q-1$, the zero-rate threshold $p_*(q, \ell, L)$ for $(p, \ell, L)_q$-list-recovery is given by*

$$\begin{aligned} p_*(q, \ell, L) &= p_* \left(q, \ell, L; \frac{q-\ell}{q}\right) = 1 - \frac{1}{L} g_{q,L,\ell} \left(\frac{q-\ell}{q}\right) \\ &= q^{-L} \sum_{\boldsymbol{a} \in \mathcal{A}_{q,L}} \binom{L}{\boldsymbol{a}} \left(1 - \frac{1}{L} \max_\ell \{\boldsymbol{a}\} \right). \end{aligned} \tag{20}$$

From now on, we use $p_*(q, \ell, L)$ to refer to the same quantity as the RHS of Equation (20).

▶ **Theorem 16** (Elias–Bassalygo bound for list-recovery)**.** *Fix any $q \in \mathbb{Z}_{\geqslant 2}$, $L \in \mathbb{Z}_{\geqslant 2}$, integer $2 \leqslant \ell \leqslant q-1$ and $0 \leqslant p < p_*(q, \ell, L)$. Then the $(p, \ell, L)_q$-list-recovery capacity can be upper bounded as $C_{(p,\ell,L)_q} \leqslant 1 - H_{q,\ell}(w_{q,\ell,L})$ where $w_{q,\ell,L}$ is the solution to the equation $p_*(q, \ell, L; w) = p$ in $w \in [0, (q-\ell)/q]$.*

Proof. Parallel to Theorem 10, the above theorem is immediately implied by a finite-blocklength version [46, Theorem 22] (analogous to [46, Theorem 19]) whose full proof is presented in [46, Sec. 11]. ◀

3 Discussion of Blinovsky's results [6, 7]

As mentioned in Section 1, part of the motivation of this work is to fill in the gaps in the proofs in [6, 7] for q-ary list-decoding. We discuss in detail below the issues therein. The main result in [6] is a Plotkin bound (as our Theorem 7) for an arbitrary q-ary list-decodable code $\mathcal{C} \subset [q]^n$. For the sake of brevity, we assume in the proceeding discussion that $\mathcal{C}$ is w-constant weight. Additional bookkeeping is needed to handle small deviations in the weight, as we did in the proof of [46, Theorem 16]. The skeleton of the proof in [6] follows Blinovsky's proof in the *binary* case [5] which we adopt here as well: (i) pass to an (approximately) equi-coupled subcode $\mathcal{C}' = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_M\} \subset \mathcal{C}$ using a Ramsey reduction; (ii) handle asymmetric coupling using Komlós's argument (and its order-L generalization [9]); (iii) prove an upper bound on the size M of the subcode $\mathcal{C}'$ using a double-counting argument. In completing the double-counting argument, one is required to upper bound the average radius (averaged over all L-lists in the subcode) by the zero-rate threshold $p_*(q, L; w) = 1 - \frac{1}{L} g_{q,L}(w)$:

$$\frac{1}{M^L} \sum_{(i_1,\cdots,i_L)\in[M]^L} \overline{\mathrm{rad}}(\boldsymbol{x}_{i_1}, \cdots, \boldsymbol{x}_{i_L}) = \sum_{k=1}^{n} \left(1 - \frac{1}{L} f_{q,L}(P_k)\right) \leqslant n \left(1 - \frac{1}{L} g_{q,L}(w)\right), \quad (21)$$

where $\overline{\mathrm{rad}}$ is defined in [46, Definition 10] and $P_k \in \Delta([q])$ is the empirical distribution of the k-th column of $\mathcal{C}' \in [q]^{M\times n}$. The equality in Equation (21) is by elementary algebraic manipulations (see [46, Eqn. (58)] for details). To show the inequality in Equation (21), we need the following properties of the functions $f_{q,L}$ and $g_{q,L}$:

1. For any $P = (p_1, \cdots, p_q) \in \Delta([q])$, we have $f_{q,L}(P) \geqslant g_{q,L}(1-p_q)$. In words, uniformizing P except one entry will only make $f_{g,L}$ no larger.
2. $g_{q,L}$ is convex as a univariate real-valued function on $[0, (q-1)/q]$.

If these properties hold, one can deduce [46, Eqn. (59) and (61)] from which Equation (21) follows. However, we observe that the proofs in [6, 7] for both properties above are problematic.

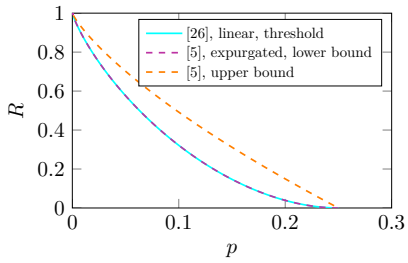
To show Item 1 above, the idea in [6] is to show instead monotonicity of $f_{q,L}$ under the so-called *Robin Hood operation* which averages two distinct entries of P. Specifically, [6] attempts to show

$$f_{q,L}(p_1, \cdots, p_i, \cdots, p_j, \cdots, p_q) \geqslant f_{q,L}\left(p_1, \cdots, \frac{p_i + p_j}{2}, \cdots, \frac{p_i + p_j}{2}, \cdots, p_q\right), \quad (22)$$

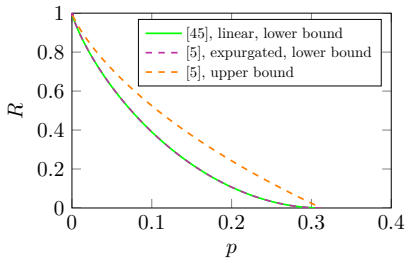
for any $1 \leqslant i < j \leqslant q$. This suffices since a sequence of Robin Hood operations can turn P into $P_{q,1-p_q}$ (defined in Equation (12)). [6] then proceeds to show Equation (22) by checking the derivative of a certain function related to the Robin Hood operation. Specifically, fix $(p_k)_{k\in[q]\setminus\{i,j\}}$ and assume $p_i + p_j = c$ (or equivalently $\sum_{k\in[q]\setminus\{i,j\}} p_i = 1 - c$) for some constant $0 \leqslant c \leqslant 1$. Consider the function $F_{q,L} \colon [0, c] \to \mathbb{R}$ defined as:

$$F_{q,L}(p) = f_{q,L}(p_1, \cdots, p, \cdots, c - p, \cdots, p_q),$$

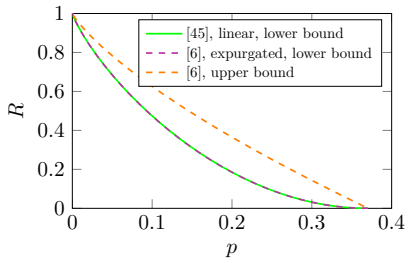
i.e., $f_{q,L}$ evaluated at P with $p_i = p, p_j = c - p$. The proof of Equation (22) is reduced to proving $F'_{q,L}(p) \leqslant 0$ for $p \in [0, c/2]$ and $F'_{q,L}(p) \geqslant 0$ for $p \in [c/2, c]$. If true, it implies that $f_{q,L}(P)$ is minimized at $p_i = p_j = c/2$ with fixed $(p_k)_{k\in[q]\setminus\{i,j\}}$. However, we note that the



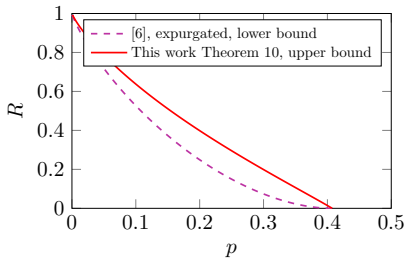
(a) $q = 2, \ell = 1, L = 3$ (list-decoding).



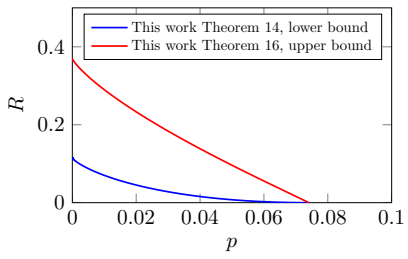
(b) $q = 2, \ell = 1, L = 4$ (list-decoding).



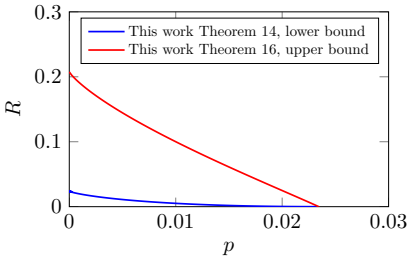
(c) $q = 3, \ell = 1, L = 3$ (list-decoding).



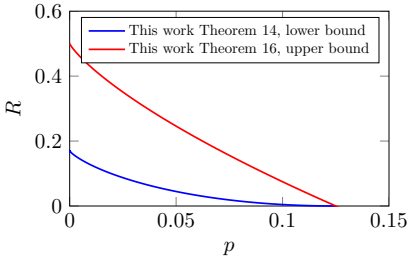
(d) $q = 3, \ell = 1, L = 4$ (list-decoding).



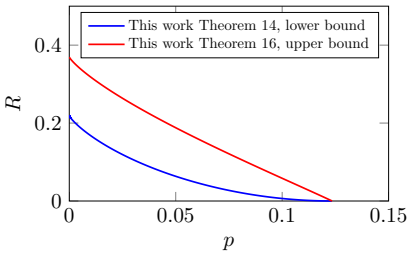
(e) $q = 3, \ell = 2, L = 3$ (list-recovery, 3-hashing).



(f) $q = 4, \ell = 3, L = 4$ (list-recovery, 4-hashing).



(g) $q = 4, \ell = 2, L = 3$ (list-recovery, $(4, 3)$-hashing).



(h) $q = 3, \ell = 2, L = 5$ (list-recovery).

Figure 1 Plots of upper and lower bounds in [5, 6, 26, 45] and this work for various values of $q \geqslant 2, 1 \leqslant \ell \leqslant q - 1, L \geqslant 2$.

expression of $F'_{q,L}(p)$ (see the second displayed equation on page 27 of [6]) is incorrect. Upon correcting it, we do not see an easy way to argue its non-positivity/-negativity. In particular, the claim in [6] that $F'_{q,L}(p)$, as a sum of multiple terms, is *term-wise* non-positive/-negative can be in general falsified by counterexamples.

The proof (attempt) of Item 2 is deferred to a subsequent paper [7]. The methodology thereof is similar to ours, i.e., verifying $g''_{q,L} \geqslant 0$. However, the expression of $g''_{q,L}$ in [7] is not exactly correct (see the first displayed equation on page 36 of [7] and compare it with ours in [46, Eqn. (34)][10]) and we have trouble verifying the case analysis of the values of $G(\cdot)$ (see [46, Eqn. (35)] in our notation, denoted by $\gamma(\cdot)$ in [7]) following that expression.

In contrast to Blinovsky's approach [6, 7], we deduce the monotonicity property of $f_{q,L}$ (cf. Item 1 above) from a stronger property: Schur convexity (cf. Theorem 3). Also, we believe that our proof of the convexity of $g_{q,L}$ (cf. Item 2 above) is cleaner, more transparent and easier to verify. Both results can be extended to list-recovery setting. Another advantage is that the monotonicity property of $g_{q,L}$ (specifically, $g_{q,L}$ is non-increasing in $[0, (q-1)/q]$ and non-decreasing in $[(q-1)/q, 1]$) which is needed in the proof of the Plotkin bound appears to be a simple consequence of the Schur convexity of $f_{q,L}$ (see Lemma 6). In [7], this is proved by checking the first derivative of $g_{q,L}$ which involves somewhat cumbersome calculations and case analysis.

4 Conclusion

In this work, we addressed the basic question of determining the maximum achievable decoding radius for positive rate list-recoverable codes, i.e., we pinned down the list-recovery zero-rate threshold. We then adapted known techniques to show that codes correcting more errors must in fact have *constant* size. Subsequently, we transferred this bound to give upper bounds on the rate of list-recoverable codes for all values of decoding radius.

As we apply general Ramsey-theoretic tools in bounding the size of list-recoverable codes in the zero-rate regime, our dependence on the corresponding parameters is quite poor, and indeed, we made no efforts to optimize these constants. However, for list-decodable binary codes in the zero-rate, a recent work of Alon, Bukh and Polyanskiy [1] derived new (and, in some cases, tight) upper bounds on their size. Obtaining similarly improved size upper bounds for q-ary list-decodable/-recoverable codes in the zero-rate regime therefore appears to be a natural next step.

References

1 Noga Alon, Boris Bukh, and Yury Polyanskiy. List-decodable zero-rate codes. *IEEE Transactions on Information Theory*, 65(3):1657–1667, 2018.

2 Erdal Arikan. Upper bound on the zero-error list-coding capacity. *Information Theory, IEEE Transactions on*, 40:1237–1240, August 1994. doi:10.1109/18.335947.

3 László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Comput. Complex.*, 3:307–318, 1993. doi:10.1007/BF01275486.

4 L. A. Bassalygo. New upper bounds for error-correcting codes. *Probl. of Info. Transm.*, 1:32–35, 1965.

[10] Note that the function considered in [7] is, in our notation, $1 - \frac{1}{L} g_{q,L}(w)$ instead of $g_{q,L}(w)$ per se as considered in Theorem 4.

5 Vladimir M Blinovsky. Bounds for codes in the case of list decoding of finite volume. *Problems of Information Transmission*, 22:7–19, 1986.

6 Vladimir M Blinovsky. Code bounds for multiple packings over a nonbinary finite alphabet. *Problems of Information Transmission*, 41:23–32, 2005.

7 Vladimir M Blinovsky. On the convexity of one coding-theory function. *Problems of Information Transmission*, 44:34–39, 2008.

8 Volodia Blinovsky. *Asymptotic combinatorial coding theory*, volume 415 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, MA, 1997. doi:10.1007/978-1-4615-6193-4.

9 Marco Bondaschi and Marco Dalai. A revisitation of low-rate bounds on the reliability function of discrete memoryless channels for list decoding. *IEEE Transactions on Information Theory*, 68(5):2829–2838, 2022. doi:10.1109/TIT.2022.3145318.

10 Simone Costa and Marco Dalai. New bounds for perfect k-hashing. *CoRR*, abs/2002.11025, 2020. arXiv:2002.11025.

11 M. Dalai, V. G. Carnegie, and J. Radhakrishnan. An improved bound on the zero-error list-decoding capacity of the 4/3 channel. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 1658–1662, June 2017. doi:10.1109/ISIT.2017.8006811.

12 Stefano Della Fiore, Simone Costa, and Marco Dalai. Improved bounds for (b, k)-hashing. *IEEE Transactions on Information Theory*, 68(8):4983–4997, 2022. doi:10.1109/TIT.2022.3167608.

13 Philippe Delsarte. An algebraic approach to the association schemes of coding theory. *Philips Res. Rep. Suppl.*, 10:vi+–97, 1973.

14 Dean Doron, Dana Moshkovitz, Justin Oh, and David Zuckerman. Nearly optimal pseudorandomness from hardness. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1057–1068. IEEE, 2020.

15 Dean Doron and Mary Wootters. High-probability list-recovery, and applications to heavy hitters. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

16 Peter Elias. List decoding for noisy channels. *Wescon Convention Record, Part 2*, pages 94–104, 1957.

17 Peter Elias. Error-correcting codes for list decoding. *IEEE Transactions on Information Theory*, 37(1):5–12, 1991.

18 M. Fredman and J. Komlós. On the size of separating systems and families of perfect hash functions. *SIAM Journal on Algebraic Discrete Methods*, 5(1):61–68, 1984. doi:10.1137/0605009.

19 Edgar N Gilbert. A comparison of signalling alphabets. *The Bell System Technical Journal*, 31(3):504–522, 1952.

20 Oded Goldreich and Leonid A Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 25–32. ACM, 1989.

21 Venkatesan Guruswami and Piotr Indyk. Expander-based constructions of efficiently decodable codes. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 658–667, 2001. doi:10.1109/SFCS.2001.959942.

22 Venkatesan Guruswami and Piotr Indyk. Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 812–821, 2002.

23 Venkatesan Guruswami and Piotr Indyk. Linear time encodable and list decodable codes. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 126–135, 2003.

24 Venkatesan Guruswami and Piotr Indyk. Efficiently decodable codes meeting gilbert-varshamov bound for low rates. In *SODA*, volume 4, pages 756–757. Citeseer, 2004.

25 Venkatesan Guruswami, Ray Li, Jonathan Mosheiff, Nicolas Resch, Shashwat Silas, and Mary Wootters. Bounds for list-decoding and list-recovery of random linear codes. *IEEE Transactions on Information Theory*, 68(2):923–939, 2022. doi:10.1109/TIT.2021.3127126.

26 Venkatesan Guruswami, Jonathan Mosheiff, Nicolas Resch, Shashwat Silas, and Mary Wootters. Threshold rates for properties of random codes. *IEEE Transactions on Information Theory*, 68(2):905–922, 2022. doi:10.1109/TIT.2021.3123497.

27 Venkatesan Guruswami and Andrii Riazanov. Beating Fredman-Komlós for Perfect k-Hashing. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132, pages 92:1–92:14, Dagstuhl, Germany, 2019. doi:10.4230/LIPIcs.ICALP.2019.92.

28 Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential coding theory, January 31, 2022. Draft available at https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/web-coding-book.pdf.

29 Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced expanders and randomness extractors from parvaresh–vardy codes. *Journal of the ACM (JACM)*, 56(4):1–34, 2009.

30 Iftach Haitner, Yuval Ishai, Eran Omri, and Ronen Shaltiel. Parallel hashing via list recoverability. In *Annual Cryptology Conference*, pages 173–190. Springer, 2015.

31 Justin Holmgren, Alex Lombardi, and Ron D Rothblum. Fiat–shamir via list-recoverable codes (or: parallel repetition of gmw is not zero-knowledge). In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 750–760, 2021.

32 Piotr Indyk, Hung Q Ngo, and Atri Rudra. Efficiently decodable non-adaptive group testing. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1126–1142. SIAM, 2010.

33 Jeffrey C Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55(3):414–440, 1997.

34 J. Körner. Coding of an information source having ambiguous alphabet and the entropy of graphs. In *Transactions of the Sixth Prague Conference on Information Theory, Statistical Decision Functions, Random Processes (Tech Univ., Prague, 1971; dedicated to the memory of Antonín Špaček)*, pages 411–425. Academia, Prague, 1973.

35 J. Körner. Fredman-komlós bounds and information theory. *SIAM Journal on Algebraic Discrete Methods*, pages 560–570, 1986.

36 J. Körner and K. Marton. New bounds for perfect hashing via information theory. *European Journal of Combinatorics*, 9(6):523–530, 1988. doi:10.1016/S0195-6698(88)80048-9.

37 Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

38 Richard J Lipton. Efficient checking of computations. In *Proceedings of the 7th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 207–215. Springer, 1990.

39 Robert J. McEliece, Eugene R. Rodemich, Howard Rumsey, Jr., and Lloyd R. Welch. New upper bounds on the rate of a code via the Delsarte-MacWilliams inequalities. *IEEE Trans. Inform. Theory*, IT-23(2):157–166, 1977. doi:10.1109/tit.1977.1055688.

40 Jonathan Mosheiff, Nicolas Resch, Noga Ron-Zewi, Shashwat Silas, and Mary Wootters. LDPC codes achieve list decoding capacity. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science*, pages 458–469. IEEE Computer Soc., Los Alamitos, CA, [2020] ©2020. doi:10.1109/FOCS46700.2020.00050.

41 Hung Q Ngo, Ely Porat, and Atri Rudra. Efficiently decodable error-correcting list disjunct matrices and applications. In *International Colloquium on Automata, Languages, and Programming*, pages 557–568. Springer, 2011.

42 Morris Plotkin. Binary codes with specified minimum distance. *IRE Transactions on Information Theory*, 6(4):445–450, 1960.

43 Yury Polyanskiy. Upper bound on list-decoding radius of binary codes. *IEEE Transactions on Information Theory*, 62(3):1119–1128, 2016.

44 Nicolas Resch. List-decodable codes:(randomized) constructions and applications. *School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep., CMU-CS-20-113*, 2020.

45 Nicolas Resch and Chen Yuan. Threshold rates of code ensembles: Linear is best. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 104:1–104:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.104.

46 Nicolas Resch, Chen Yuan, and Yihan Zhang. Zero-rate thresholds and new capacity bounds for list-decoding and list-recovery. *CoRR*, abs/2210.07754, 2022. doi:10.48550/arXiv.2210.07754.

47 Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.

48 RR Varshamov. Estimate of the number of signals in error correcting codes. *Docklady Akad. Nauk, SSSR*, 117:739–741, 1957.

49 Lloyd R. Welch, Robert J. McEliece, and Howard Rumsey, Jr. A low-rate improvement on the Elias bound. *IEEE Trans. Inform. Theory*, IT-20:676–678, 1974. doi:10.1109/tit.1974.1055279.

50 Jack Wozencraft. List decoding. *Quarter Progress Report*, 48:90–95, 1958.

51 Chaoping Xing and Chen Yuan. Beating the probabilistic lower bound on perfect hashing. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 33–41. SIAM, 2021. doi:10.1137/1.9781611976465.3.

52 Yihan Zhang, Amitalok J. Budkuley, and Sidharth Jaggi. Generalized List Decoding. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 51:1–51:83, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITCS.2020.51.

Convergence of the Number of Period Sets in Strings

Eric Rivals
LIRMM, Université Montpellier, CNRS, France

Michelle Sweering
CWI, Amsterdam, The Netherlands

Pengfei Wang
LIRMM, Université Montpellier, CNRS, France

Abstract

Consider words of length n. The set of all periods of a word of length n is a subset of $\{0, 1, 2, \ldots, n-1\}$. However, any subset of $\{0, 1, 2, \ldots, n-1\}$ is not necessarily a valid set of periods. In a seminal paper in 1981, Guibas and Odlyzko proposed to encode the set of periods of a word into an n long binary string, called an autocorrelation, where a one at position i denotes the period i. They considered the question of recognizing a valid period set, and also studied the number of valid period sets for strings of length n, denoted κ_n. They conjectured that $\ln(\kappa_n)$ asymptotically converges to a constant times $\ln^2(n)$. Although improved lower bounds for $\ln(\kappa_n)/\ln^2(n)$ were proposed in 2001, the question of a tight upper bound has remained open since Guibas and Odlyzko's paper. Here, we exhibit an upper bound for this fraction, which implies its convergence and closes this longstanding conjecture. Moreover, we extend our result to find similar bounds for the number of correlations: a generalization of autocorrelations which encodes the overlaps between two strings.

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics

Keywords and phrases Autocorrelation, period, border, combinatorics, correlation, periodicity, upper bound, asymptotic convergence

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.100

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2209.08926`
Full Version: `https://hal-lirmm.ccsd.cnrs.fr/lirmm-03780386`

Funding *Eric Rivals*: European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 956229.
Michelle Sweering: The Netherlands Organisation for Scientific Research (NWO) through Gravitation-grant NETWORKS-024.002.003.
Pengfei Wang: European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 956229.

1 Introduction

A linear word can overlap itself if one of its prefixes is equal to one of its suffixes. The corresponding prefix (or suffix) is called a border and the shift needed to match the prefix to the suffix is called a period. The dual notions of period and border are critical concepts in word combinatorics: important definitions such as periodic and primitive words, or the normal form of a word rely on them. These concepts play a role in key results of the field like the Critical Factorization Theorem [14]. In computer science, in the field of string algorithms (a.k.a. stringology), pattern matching algorithms heavily exploit borders/periods to optimize the search of occurrences of a word in a text [21]. For clarity, note that the terms *word* and

string both mean a sequence of letters taken from an alphabet. These notions also play a role in statistics. The set of periods of a word controls how two occurrences of the same word can overlap in a text. Hence, the set of periods (or autocorrelation) is a key variable to study the statistics of word occurrences in random texts (waiting time, distance between successive occurrences, etc.) [19]. The notion of autocorrelation has been extended to describe how two distinct words can have overlapping occurrences in the same text. This has been used for instance to study the number of missing words in random texts [16] or to design procedures for testing pseudo-random number generators [15].

An autocorrelation is a binary vector representing the set of periods of a word. The concept of autocorrelation was introduced by Guibas and Odlyzko in [10]. They gave a characterization of autocorrelations and proved the following bounds on κ_n - the cardinality of the set Γ_n of autocorrelations of words of length n.

$$\frac{1}{2\ln(2)} + o(1) \leq \frac{\ln(\kappa_n)}{\ln^2(n)} \leq \frac{1}{2\ln(3/2)} + o(1)$$

They conjectured that $\ln(\kappa_n)$ is asymptotic to a constant times $\ln^2(n)$. Rivals and Rahmann [18], later on studied the combinatorial structure of the set of autocorrelations Γ_n, and improved the lower bound on κ_n as follows:

$$\frac{\ln(\kappa_n)}{\ln^2(n)} \geq \frac{1}{2\ln(2)}\left(1 - \frac{\ln(\ln(n))}{\ln(n)}\right)^2 + \frac{0.4139}{\ln(n)} - \frac{1.47123\ln(\ln(n))}{\ln^2(n)} + O\left(\frac{1}{\ln^2(n)}\right).$$

However, to date, no one has focused on improving the upper bound on κ_n. In this work, we apply the notion of irreducible period sets introduced by Rivals and Rahmann [17, 18] to prove that

$$\frac{\ln(\kappa_n)}{\ln^2(n)} \leq \frac{1}{2\ln(2)} + \frac{3}{2\ln(n)} \quad \forall n \in \mathbb{N}_{\geq 2}.$$

Together with known asymptotic lower bounds [18], we find that

$$\frac{\ln \kappa_n}{\ln^2(n)} \to \frac{1}{2\ln(2)} \quad \text{as} \quad n \to \infty,$$

thus resolving the conjecture of Guibas and Odlyzko.

In their paper about autocorrelations [10], Guibas and Odlyzko also introduced the notion of correlation between words. For two words u and v, the *correlation* of u over v is a binary vector indicating all overlaps between suffixes of u and prefixes of v. In particular, an autocorrelation is the correlation of a word with itself. We show that the number of correlations between two words of length n, denoted by δ_n, has the same asymptotic convergence behaviour as the number of autocorrelations of words of length n, that is

$$\frac{\ln \delta_n}{\ln^2(n)} \to \frac{1}{2\ln(2)} \quad \text{as} \quad n \to \infty.$$

1.1 Related works

Apart from previously cited articles that deal with the combinatorics of period sets, some related works exist in the literature.

For instance, the question of the average period of a random word has been investigated in [13]. Clearly, the number of periods of a word of length n lies between one and n. A recent work exhibits an upper bound on the number of periods of a word as a function of its *initial*

critical exponent – a characteristic of the word related to its degree of periodicity [9], but this has not been used to bound the number of period sets. Last, the combinatorics of period sets has also been investigated in depth for a generalization of the notion of words, called *partial words* [6]. In partial words, some positions may contain a *don't care* symbol, which removes some constraints of equality between positions. To study the combinatorics of period sets, Blanchet-Sadri *et al.* defined weak and strong periods, and proved several important theorems [4], including lower and upper bounds on the number of binary and ternary autocorrelations [6, 5]. However, the cardinality of the family of period sets differs between normal words and partial words, and a tight upper bound for normal words cannot be deduced from that for partial words. Several works investigate sets of words with constraints (either absence or presence) on their mutual overlaps: mutually bordered (overlapping) pairs of words are studied in [8], while methods for constructing a set of mutually unbordered words (also called cross-bifix-free words) are provided in [3, 1, 2].

2 Preliminaries

A string $u = u[0\,..\,n-1] \in \Sigma^n$ is a sequence of n letters over a finite alphabet Σ. For any $0 \le i \le j \le n-1$, we denote the substring starting at position i and ending at position j with $u[i\,..\,j]$. In particular, $u[0\,..\,j]$ denotes a prefix and $u[i\,..\,n-1]$ a suffix of u. Throughout this paper, all our strings and vectors will be zero-indexed.

2.1 Periodicity

In this subsection, we define the concepts of period, period set, basic period, and autocorrelation, and then review some useful results. For the sake of self-containment, we provide in Appendix A the proofs for all lemmas of this subsection.

▶ **Definition 1** (Period). *String $u = u[0\,..\,n-1]$ has a period $p \in \{1, \ldots, n-1\}$ if and only if for any $0 \le i, j \le n-1$ such that $i \equiv j \mod p$, we have $u[i] = u[j]$. Moreover, we consider that $p = 0$ is a period of any string of length n.*

An equivalent definition is the following.

▶ **Definition 2** (Period). *The string $u = u[0\,..\,n-1]$ has period $p \in \{0, 1, \ldots, n-1\}$ if and only if $u[0\,..\,n-p-1] = u[p\,..\,n-1]$, i.e. for all $0 \le i \le n-p-1$, we have $u[i] = u[i+p]$.*

The smallest non-zero period of u is called its *basic period*. The *period set* of a string u is the set of all its periods and is denoted by $P(u)$. We will now list some useful properties about periods, which we will need later on. Their proofs can be found in [10, 14] and in Appendix A.

▶ **Lemma 3.** *Let p be a period of $u \in \Sigma^n$ and $k \in \mathbb{Z}_{\ge 0}$ such that $kp < n$. Then kp is also a period of u.*

▶ **Lemma 4.** *Let p be a period of $u \in \Sigma^n$ and q a period of the suffix $w = u[p\,..\,n-1]$. Then $(p+q)$ is a period of u. Moreover, $(p+kq)$ is also a period of u for all $k \in \mathbb{Z}_{\ge 0}$ with $p + kq < n$.*

▶ **Lemma 5.** *Let p, q be periods of $u \in \Sigma^n$ with $0 \le q \le p$. Then the prefix and the suffix of length $(n-q)$ have the period $(p-q)$.*

▶ **Lemma 6.** *Suppose p is a period of $u \in \Sigma^n$ and there exists a substring v of u of length at least p and with period r, where $r|p$. Then r is also a period of u.*

Table 1 This table illustrates the set of period and the autocorrelation of the word $u = \texttt{abbaabba}$ of length 8. A first copy of the word u is shown on the second line. Another copy of u is displayed on (each) line $(3 + i)$ shifted by i positions to the right, with i going from 0 to 7. If the suffix of the copy of u matches the prefix of the first copy u on line 2, then i is a period, and both the line and the corresponding position/shift (on the first line) are colored in blue. The last column contains the autocorrelation of u, with 1 bits corresponding to periods colored in blue.

pos.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
u	a	b	b	a	a	b	b	a	-	-	-	-	-	-	-	s
u	a	b	b	a	a	b	b	a	-	-	-	-	-	-	-	1
	-	a	b	b	a	a	b	b	a	-	-	-	-	-	-	0
	-	-	a	b	b	a	a	b	b	a	-	-	-	-	-	0
	-	-	-	a	b	b	a	a	b	b	a	-	-	-	-	0
	-	-	-	-	a	b	b	a	a	b	b	a	-	-	-	1
	-	-	-	-	-	a	b	b	a	a	b	b	a	-	-	0
	-	-	-	-	-	-	a	b	b	a	a	b	b	a	-	0
	-	-	-	-	-	-	-	a	b	b	a	a	b	b	a	1

We will also use the famous Fine and Wilf theorem [7], a.k.a. the periodicity lemma, for which a short proof was provided by Halava and colleagues [12].

▶ **Theorem 7** (Fine and Wilf). *Let p, q be periods of $u \in \Sigma^n$. If $n \geq p + q - \gcd(p, q)$, then $\gcd(p, q)$ is a period of u.*

2.2 Autocorrelation

We now give a formal definition of an autocorrelation.

▶ **Definition 8** (Autocorrelation). *For every string $u \in \Sigma^n$, its autocorrelation is the string $s \in \{0, 1\}^n$ such that*

$$s[i] = \begin{cases} 1 & \text{if } i \text{ is a period of } u \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in \{0, \ldots, n-1\}.$$

To illustrate this concept, consider the following example (detailed in Table 1).

▶ **Example 9.** We consider the word $u = \texttt{abbaabba}$ of length 8. Its period set is $P(u) = \{0, 4, 7\}$, its basic period is 4 and its autocorrelation is $s = 10001001$. See Table 1.

Guibas and Odlyzko [10] show that *any alphabet of size at least two will give rise to the same set of correlations* (Corollary 5.1). Autocorrelations have many other useful properties [10, 18]. The most significant one for our work is the following.

▶ **Lemma 10** (Lemma 3.1 [10] and Theorem 1.3 [18]). *If $s \in \{0, 1\}^n$ is an autocorrelation and $s[i] = 1$, then $s[i \,..\, n-1]$ is the autocorrelation of $u[i \,..\, n-1]$*

Proof. Note that $s[i] = 1$ means: i is a period of u. Suppose $s[i + j] = 1$. Then $i + j$ is a period of u. Thus $u[i \,..\, n-1]$ has period $(i + j) - i = j$ by Lemma 5. Conversely, suppose $u[i \,..\, n-1]$ has period $(i + j) - i = j$. Then $i + j$ is a period of u by Lemma 4. Thus $s[i + j] = 1$. Combining these results, we find that $s[i + j] = 1$ if and only of j is a period of $u[i \,..\, n-1]$, and equivalently $s[i \,..\, n-1]$ is the autocorrelation of $u[i \,..\, n-1]$. ◀

2.3 Irreducible Period Set

To prove the upper bound on the number of autocorrelations, we use the notion of irreducible period sets as introduced by Rivals and Rahmann [18]. An irreducible period set is the minimum subset of a period set that determines the period set using the Forward Propagation Rule. Before formally introducing the irreducible period set, we will first explain what forward propagation is.

▶ **Lemma 11** (Forward Propagation Rule). *Let $p \leq q$ be periods of a string u of length n and let $k \in \mathbb{Z}_{\geq 0}$ such that $p + k(q-p) < n$. Then $p + k(q-p)$ is a period of $u[0\,..\,n-1]$.*

Proof. It follows from Lemma 5 that $u[p\,..\,n-1]$ has period $(q-p)$. Applying Lemma 4 we find that $u[0\,..\,n-1]$ has period $p + k(q-p)$ for all $k \in \mathbb{Z}_{\geq 0}$. ◀

The forward closure $FC_n(S)$ of a set $S \subseteq \{0, \ldots, n-1\}$ (not necessarily a period set, typically a subset of one) is the closure of S under the forward propagation rule.

▶ **Definition 12** (Forward Closure). *Let $S \subseteq \{0, \ldots, n-1\}$. Its forward closure $FC_n(S)$ is the minimum superset of S such that for any $p, q \in FC_n(S)$ and $k \geq 0$ with $p \leq q$ and $p + k(q-p) < n$, we have*

$$p + k(q-p) \in FC_n(S).$$

We can now define the irreducible period set.

▶ **Definition 13** (Irreducible Period Set). *Let P be the period set of a string $u \in \Sigma^n$. An irreducible period set of P is a minimal subset $R(P) \subseteq P$ with forward closure P.*

Observe that there exists an irreducible period set for any period set P, because $FC_n(P) = P$ by the forward propagation rule. We will now give a useful characterization of an irreducible period set as the set of periods which are not in the forward closure of the set of all smaller periods. Consequently, every period set has exactly one irreducible period set, whose elements we will call irreducible periods.

Recall that for a given length n, we denote the set of all period sets by Γ_n. Formally stated Γ_n is defined as:

$$\Gamma_n = \{S \subseteq \{0, 1, \ldots, n-1\} : \exists u \in \Sigma^n \text{ such that } P(u) = S\}.$$

As in [18], for a given length n, we denote the set of all irreducible period sets by Λ_n:

$$\Lambda_n = \{T \subseteq \{0, 1, \ldots, n-1\} : \exists u \in \Sigma^n \text{ such that } R(P(u)) = T\}.$$

The bijective relation between period sets and irreducible period sets implies that $|\Gamma_n| = |\Lambda_n|$.

▶ **Lemma 14.** *Let P be the period set of a string $u \in \Sigma^n$ and $R(P)$ an irreducible period set of P. Then*

$$R(P) = \{q \in P \mid q \notin FC_n(P \cap [0, q-1])\}.$$

Proof. Let $p \in P$. We will prove the two alternative cases separately:
(a) $p \notin \{q \in P \mid q \notin FC_n(P \cap [0, q-1])\} \implies p \notin R(P)$ and
(b) $p \in \{q \in P \mid q \notin FC_n(P \cap [0, q-1])\} \implies p \in R(P)$.

(a) Suppose $p \notin \{q \in P \mid q \notin FC_n(P \cap [0, q-1])\}$, or equivalently $p \in FC_n(P \cap [0, p-1])$. Then

$$\begin{aligned} p \in FC_n(P \cap [0,p-1]) &= FC_n(FC_n(R(P)) \cap [0,p-1]) \\ &\subseteq FC_n(FC_n(R(P) \cap [0,p-1])) \\ &= FC_n(R(P) \cap [0,p-1]) \\ &\subseteq FC_n(R(P) \setminus \{p\}). \end{aligned}$$

It follows that $FC_n(R(P) \setminus \{p\}) = FC_n(R(P))$. By minimality of irreducible period sets, we have $p \notin R(P)$.

(b) Suppose on the other hand that $p \notin FC_n(P \cap [0, p-1])$. Then $p \notin FC_n(P \setminus \{p\})$ either. As

$$FC_n(P \setminus \{p\}) \supseteq FC_n(R(P) \setminus \{p\}),$$

then $p \notin FC_n(R(P) \setminus \{p\})$. However, as $p \in P$ and $P = FC_n(R(P))$, it follows that $p \in R(P)$. ◀

3 Asymptotic convergence of κ_n

In this section, we present a new upper bound on κ_n, the number of distinct autocorrelations of strings of length n. Moreover, we shall prove that $\ln(\kappa_n)$ asymptotically converges to $c \cdot \ln^2(n)$, where $c = \frac{1}{2\ln(2)}$.

▶ **Theorem 15** (Upper bound on κ_n)**.** *For all $n \in \mathbb{N}_{\geq 2}$ we have*

$$\frac{\ln(\kappa_n)}{\ln^2(n)} \leq \frac{1}{2\ln(2)} + \frac{3}{2\ln(n)}.$$

Proof. To prove this theorem, we need several lemmas.

▶ **Lemma 16.** *Let $u \in \Sigma^n$ with autocorrelation s, period set P, and irreducible period set $R(P) = \{0 = a_0 < \ldots < a_i < \ldots < a_k < n\}$. Then for all $0 \leq i \leq k$, there exists $q_i \in \{1, \ldots, n - a_i\}$ such that*

1. *$q_i \leq n/2^i$, and*
2. *$a_i + q_i = n$ or $a_i + q_i$ is in the forward closure of $\{a_0, \ldots, a_i\}$.*

Proof. We will prove this by induction.

Basis. By picking $q_0 = n \in \{1, \ldots, n - a_0\}$, we satisfy both $q_0 \leq n/2^0$ and $a_0 + q_0 = n$.

Hypothesis. For some $0 \leq i < k$, there exists a $q_i \in \{1, \ldots, n - a_i\}$ such that

1. $q_i \leq n/2^i$, and
2. $a_i + q_i = n$ or $a_i + q_i$ is in the forward closure of $\{a_0, \ldots, a_i\}$.

Step. We first note that if $n - a_{i+1} \leq n/2^{i+1}$, then we can pick $q_{i+1} = n - a_{i+1}$. Suppose on the other hand that $n - a_{i+1} > n/2^{i+1}$. We distinguish two cases.

- If $a_i + q_i = n$, then

$$\begin{aligned} a_{i+1} - a_i &= (n - a_i) - (n - a_{i+1}) \\ &< n/2^i - n/2^{i+1} \\ &= n/2^{i+1} \\ &< n - a_{i+1}. \end{aligned}$$

Thus, we can pick $q_{i+1} = a_{i+1} - a_i \in \{1, \ldots, n - a_{i+1}\}$, since

1. it satisfies $q_{i+1} \leq n/2^{i+1}$ and
2. $a_{i+1} + q_{i+1} = a_i + 2(a_{i+1} - a_i)$ is in the forward closure of $\{a_0, \ldots, a_{i+1}\}$.

- If $a_i + q_i$ is in the forward closure of $\{a_0, \ldots, a_i\}$, then

$$a_i + \lambda q_i = a_i + \lambda(a_i + q_i - a_i)$$

is in the forward closure of $\{a_0, \ldots, a_i\}$ for all integers $0 \leq \lambda \leq (n-1-a_i)/q_i$. Since a_{i+1} is an irreducible period, there must exist an integer $\lambda_0 \in [0, (n-1-a_i)/q_i]$ such that

$$a_i + \lambda_0 q_i < a_{i+1} < a_i + (\lambda_0 + 1)q_i.$$

In other words, a_{i+1} is comprised between two successive, non-irreducible periods generated from a_i and q_i using the FPR (or $n \leq a_i + (\lambda_0 + 1)q_i$). We pick

$$q_{i+1} = \min(a_{i+1} - (a_i + \lambda_0 q_i), (a_i + (\lambda_0 + 1)q_i) - a_{i+1}, n - a_{i+1})$$

and note that

$$\begin{aligned} q_{i+1} &\leq \frac{a_{i+1} - (a_i + \lambda_0 q_i) + (a_i + (\lambda_0 + 1)q_i) - a_{i+1}}{2} \\ &= q_i/2 \\ &\leq n/2^{i+1}. \end{aligned}$$

It follows that $a_{i+1} + q_{i+1} < n$. Consequently, either $a_{i+1} + q_{i+1} = (a_i + \lambda_0 q_i) + 2(a_{i+1} - (a_i + \lambda_0 q_i))$ or $a_{i+1} + q_{i+1} = a_i + (\lambda_0 + 1)(a_i + q_i - a_i)$. Hence, $a_{i+1} + q_{i+1}$ is in the forward closure of $\{a_0, \ldots, a_{i+1}\}$. Therefore q_{i+1} has all desired properties.

Conclusion. For all $0 \leq i \leq k$, there exists $q_i \in \{1, \ldots, n - a_i\}$ such that

1. $q_i \leq n/2^i$, and
2. $a_i + q_i = n$ or $a_i + q_i$ is in the forward closure of $\{a_0, \ldots, a_i\}$. ◀

▶ **Lemma 17.** *Let $R(P) = \{0 = a_0 < a_1 < \ldots < a_k\}$ be the irreducible period set of a string of length n. Then $k \leq \log_2(n)$.*

Proof. It follows from the Lemma 16 that there exists an integer $q_k \in \{1, \ldots, n - a_k\}$ such that $n/2^k \geq q_k$. Hence $k \leq \log_2(n)$. ◀

To count the number of irreducible period sets, we count the number of possibilities for each a_i with $1 \leq i \leq k$. We know that $a_0 = 0$ is fixed. The other a_i take values in the set $\{1, \ldots, n-1\}$.

▶ **Lemma 18.** *Let $0 \leq i \leq k-1$. Then a_{i+1} can take at most $2^{1-i}n - 1$ possible values given $a_0, \ldots, a_i$.*

Proof. Let q_i be defined as in Lemma 16. We distinguish 3 cases:

1. If $a_{i+1} \leq a_i + q_i$, there are at most $q_i - 1 \leq n/2^i - 1$ possible values for a_{i+1} (note that $a_{i+1} \neq a_i + q_i$, because a_{i+1} cannot be in the forward closure of $\{a_0, \ldots, a_i\}$, nor can it be equal to n).
2. If $a_{i+1} \geq n - q_i$, there are at most $q_i \leq n/2^i$ possible values for a_{i+1}.
3. In the remaining case, $a_{i+1} \in [a_i + q_i + 1, n - q_i - 1]$.

Let us first show that case 3 is impossible. For the sake of contradiction, assume we are in case 3. Since $a_i + q_i < n$, we know that $a_i + q_i$ is in the forward closure of $\{a_0, \ldots, a_i\}$ (by property 2 from Lemma 16). Hence q_i is a period of $u[a_i \,..\, n-1]$. Moreover $a_{i+1} - a_i$ is also a period of $u[a_i \,..\, n-1]$. By the Fine and Wilf theorem, it follows that

(a) either $n - a_i < q_i + (a_{i+1} - a_i) - \gcd(q_i, a_{i+1} - a_i)$

(b) or $\gcd(q_i, a_{i+1} - a_i)$ is a period of $u[a_i \,..\, n-1]$.

We are not in subcase (a) since by hypothesis $a_{i+1} \leq n - q_i - 1$. Suppose we are in subcase (b). Note that $a_i + \gcd(q_i, a_{i+1} - a_i) \leq a_i + q_i < a_{i+1}$ and that a_{i+1} is in the forward propagation of $\{a_0, \ldots, a_i, a_i + \gcd(q_i, a_{i+1} - a_i)\}$. It follows that a_{i+1} is not an irreducible period, which is a contradiction. Therefore both subcases (a) and (b) are impossible.

Summing over cases 1 and 2 (since case 3 is impossible), we conclude that, given $a_0, \ldots, a_i$, there are at most

$$(n/2^i - 1) + n/2^i + 0 = 2^{1-i}n - 1$$

possibilities for a_{i+1}. ◀

Note that the bound of Lemma 18 is not tight: indeed, there are $n-1$ possible values for a_1, while the lemma gives an upper bound of $2n-1$. However, this bound suffices to prove our asymptotic result. Since an autocorrelation is uniquely defined by its irreducible period set, it suffices to count the possible such sets $\{a_0, \ldots, a_k\}$ for all possible values of k. Recall that a_0 is fixed at 0 and that $k \leq \log_2(n)$ by Lemma 17. We thus derive a bound on the total number of autocorrelations by taking the product of all possibilities for a_{i+1} with i going from 0 to $k-1$ and sum this over all integers k from 1 to $\lfloor \log_2(n) \rfloor$, as follows:

$$\begin{aligned}
\kappa_n = |\Gamma_n| = |\Lambda_n| &\leq \sum_{k=1}^{\lfloor \log_2(n) \rfloor} \prod_{i=0}^{k-1} \left(2^{1-i}n - 1\right) \\
&\leq \sum_{k=1}^{\lfloor \log_2(n) \rfloor} \left(\left(2^{2-k}n - 1\right) \prod_{i=0}^{k-2} 2^{1-i}n \right).
\end{aligned}$$

Writing $2^{2-k}n \prod_{i=0}^{k-2} 2^{1-i}n$ and $\prod_{i=0}^{k-2} 2^{1-i}n$ in exponential form, we get

$$\begin{aligned}
\kappa_n \leq \sum_{k=1}^{\lfloor \log_2(n) \rfloor} \Bigg(&\exp\left(\frac{-k(k-3)\ln(2)}{2} + k\ln(n)\right) \\
&- \exp\left(\frac{-(k-1)(k-4)\ln(2)}{2} + (k-1)\ln(n)\right) \Bigg).
\end{aligned}$$

Observe that this is a telescoping sum, so all but two terms cancel out.

$$\kappa_n \leq \exp\left(\frac{-\lfloor \log_2(n) \rfloor (\lfloor \log_2(n) \rfloor - 3)\ln(2)}{2} + \lfloor \log_2(n) \rfloor \ln(n)\right) - 1$$

Since $\frac{d}{dk}\left(\frac{-k(k-3)\ln(2)}{2} + k\ln(n)\right) = \frac{(-2k+3)\ln(2)}{2} + \ln(n)$ is positive for all $k \leq \log_2(n)$, we have

$$\begin{aligned}
\kappa_n &< \exp\left(\frac{\ln(n)(3\ln(2) - \ln(n))}{2\ln(2)} + \frac{\ln^2(n)}{\ln(2)}\right) \\
&= \exp\left(\frac{3\ln(n)}{2} + \frac{\ln^2(n)}{2\ln(2)}\right).
\end{aligned}$$

Taking the natural logarithm of both sides and dividing by $\ln^2(n)$, we get that

$$\frac{\ln(\kappa_n)}{\ln^2(n)} \leq \frac{1}{2\ln(2)} + \frac{3}{2\ln(n)},$$

thereby proving Theorem 15. ◀

▶ **Corollary 19** (Asymptotic Convergence of κ_n). *Let κ_n be the number of autocorrelations of length n. Then*

$$\frac{\ln \kappa_n}{\ln^2(n)} \to \frac{1}{2\ln(2)} \quad \textit{as} \quad n \to \infty.$$

Proof. It follows from Theorem 15 that for $n \in \mathbb{N}_{\geq 2}$

$$\frac{\ln(\kappa_n)}{\ln^2(n)} \leq \frac{1}{2\ln(2)} + \frac{3}{2\ln(n)} = \frac{1}{2\ln(2)} + o(1).$$

The lower bound for κ_n from Theorem 5.1 in [18] indicates that asymptotically

$$\begin{aligned}\frac{\ln(\kappa_n)}{\ln^2(n)} &\geq \frac{1}{2\ln(2)}\left(1 - \frac{\ln(\ln(n))}{\ln(n)}\right)^2 + \frac{0.4139}{\ln(n)} - \frac{1.47123\ln(\ln(n))}{\ln^2(n)} + O\left(\frac{1}{\ln^2(n)}\right) \\ &= \frac{1}{2\ln(2)} - O\left(\frac{\ln(\ln(n))}{\ln(n)}\right).\end{aligned}$$

Combining this lower bound with our upper bound, we obtain

$$\frac{1}{2\ln(2)} - O\left(\frac{\ln\ln n}{\ln n}\right) \quad \leq \quad \frac{\ln \kappa_n}{\ln^2(n)} \quad \leq \quad \frac{1}{2\ln(2)} + o(1).$$

Using the classic *sandwich theorem*, we conclude that

$$\frac{\ln \kappa_n}{\ln^2(n)} \to \frac{1}{2\ln(2)} \quad \text{as} \quad n \to \infty$$

thereby proving the conjecture by Guibas and Odlyzko. ◀

The known values of κ_n are recorded in entry A005434 (see `https://oeis.org/A005434`) of the On-Line Encyclopedia of Integer Sequences [20]. Because the enumeration of Γ_n takes exponential time, the list of κ_n values is limited to a few hundred. In Figure 1, we compare the values of κ_n with the so-called Fröberg lower bound from [18], the upper bound of Guibas and Odlyzko [10], and our new upper bound. The figure illustrates the improvement brought by the new upper bound compared to that given by Guibas and Odlyzko [10]. At $n = 500$, the lower bound, our new upper bound and the values of κ_n clearly differ, meaning the sequences are far from convergence at $n = 500$.

4 Correlation

In this section, we show that the number of correlations between two strings of length n has the same asymptotic convergence behaviour as the number of autocorrelations of strings of length n.

In [11], Guibas and Odlyzko introduced the notion of *correlation* of two strings: it encodes the offset of possible overlaps between these two strings. In [10], the same authors investigate the self-overlaps of a string, which is then encoded in an *autocorrelation*. Before we start, let us define precisely the notion of correlation (which is illustrated in Table 2).

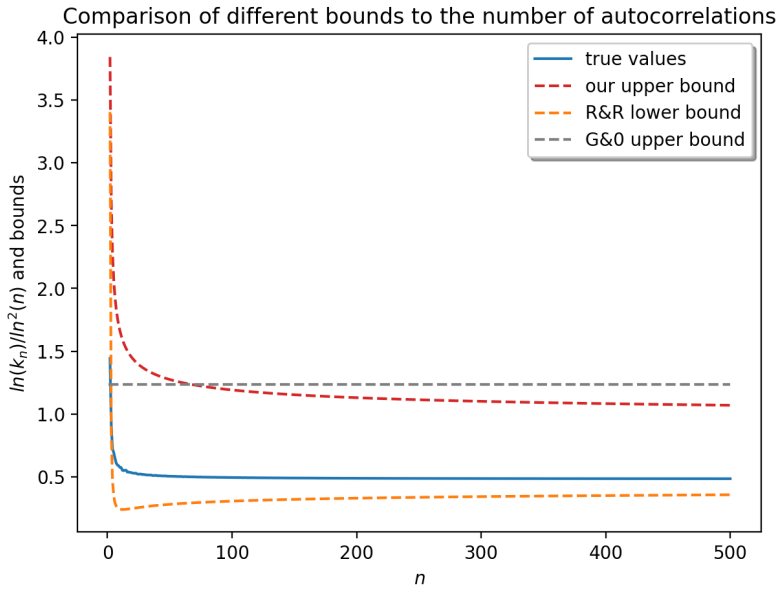


Figure 1 The true values of $\ln k_n / \ln^2(n)$ for $n \leq 500$ are compared to: the upper bound of Guibas & Odlyzko (G&O upper bound) [10], the Fröberg lower bound (R&R lower bound) [18], and our upper bound. Our upper bound seems not so tight: the reason is that n is small, as $\ln 500 \approx 6.2$.

Table 2 The correlation of word $u =$ `aabbaa` over word $v =$ `baabaa` (both of length 6) is $t = 000100$. This table is organized as Table 1 – see the corresponding caption for details.

pos.	0	1	2	3	4	5	6	7	8	9	10	
u	a	a	b	b	a	a	-	-	-	-	-	t
v	b	a	a	b	a	a	-	-	-	-		0
	-	b	a	a	b	a	a	-	-	-	-	0
	-	-	b	a	a	b	a	a	-	-	-	0
	-	-	-	b	a	a	b	a	a	-	-	1
	-	-	-	-	b	a	a	b	a	a	-	0
	-	-	-	-	-	b	a	a	b	a	a	0

▶ **Definition 20** (Correlation). *For every pair of strings $(u, v) \in \Sigma^n \times \Sigma^m$, the correlation of u over v is the vector $t \in \{0,1\}^n$ such that for all $k \in \{0, \ldots, n-1\}$*

$$t[k] = \begin{cases} 1 & \text{if } u[i] = v[j] \text{ for all } i \in \{0,\ldots,n-1\}, j \in \{0,\ldots,m-1\} \\ & \text{with } i = j + k, \\ 0 & \text{otherwise.} \end{cases}$$

Intuitively, we can find correlations as follows. For each index $i \in \{0, \ldots, n-1\}$ we write v below u starting under the ith character of u. Then the ith element of the correlation is 1, if all pairs of characters that are directly above each other match, and 0 otherwise. See Table 2 for an example.

Observe, that if $v \in \Sigma^m$ is longer than $u \in \Sigma^n$, then the correlation of u over v equals the correlation of u over $v[0\,..\,n-1]$. Conversely, any binary vector $t \in \{0,1\}^n$ is the correlation of $u = t \in \{0,1\}^n$ over $v = 1 \in \{0,1\}^1$. Therefore we will restrict ourselves to the interesting case where both strings have the same length.

Let Δ_n be the set of all correlations between two strings of the same length n and let δ_n be the cardinality of Δ_n. We can characterize Δ_n as follows.

▶ **Lemma 21.** *The set of correlations of length n is of the form*

$$\Delta_n = \left\{ 0^{(n-j)} s_j \;\mid\; s_j \in \Gamma_j,\ j \in [0, n] \right\},$$

where Γ_j is the set of autocorrelations of length j.

Proof. Let $t = 0^{(n-j)} s_j$ with s_j the autocorrelation of some string w of length j with $0 \leq j \leq n$. Without loss of generality, w does not start with the letter $\mathtt{a}$. Let $u = \mathtt{a}^{(n-j)} w$ and $v = w\mathtt{b}^{(n-j)}$. Observe that the correlation of u over v is precisely $0^{(n-j)} s_j = t$. Therefore

$$\left\{ 0^{(n-j)} s_j \;\mid\; s_j \in \Gamma_j,\ j \in [0, n] \right\} \subseteq \Delta_n.$$

Conversely, let $u, v \in \Sigma^n$ and let t' be the correlation of u over v. We can write t' in the form $0^{(n-j)} s_j$, where s_j is a binary string starting with 1 (or is empty). If s_j is the empty string, then it is the only autocorrelation of length 0. Otherwise, there is a 1 at position $n - j$, which indicates that $u[n - j \,..\, n - 1] = v[0 \,..\, j - 1]$. Moreover, s_j is the correlation of $u[n - j \,..\, n - 1]$ over v. It follows that s_j is exactly the autocorrelation of $u[n - j \,..\, n - 1] = v[0 \,..\, j - 1]$. Therefore

$$\Delta_n \subseteq \left\{ 0^{(n-j)} s_j \;\mid\; s_j \in \Gamma_j,\ j \in [0, n] \right\}. \qquad \blacktriangleleft$$

In the above characterization, we consider strings over a finite alphabet and found that a correlation depends on some autocorrelation. As it is known that Γ_n is independent of the alphabet size (provided $|\Sigma| > 1$), the reader may wonder whether the number of correlations depends on it. In Appendix B, we show that the set of correlations for equally long strings is independent of the alphabet size, provided that Σ is not unary.

Now we have characterized Δ_n, we can easily deduce its cardinality.

▶ **Lemma 22.** *Let κ_n be the number of autocorrelations of length n and δ_n the number of correlations between two strings of length n. Then*

$$\delta_n = \sum_{j=0}^{n} \kappa_j.$$

Proof. Since autocorrelations do not start with a zero, no two strings of the form $0^{(n-j)} s_j$ with $s_j \in \Gamma_j$ and $j \in [0, n]$ are the same. Therefore

$$\delta_n = |\Delta_n| = \left| \left\{ 0^{(n-j)} s_j \;\mid\; s_j \in \Gamma_j,\ j \in [0, n] \right\} \right| = \sum_{j=0}^{n} |\Gamma_j| = \sum_{j=0}^{n} \kappa_j. \qquad \blacktriangleleft$$

▶ **Theorem 23** (Asymptotic Convergence of δ_n)**.** *Let δ_n be the number of correlations between two strings of length n. Then*

$$\frac{\ln \delta_n}{\ln^2(n)} \to \frac{1}{2\ln(2)} \quad \textit{as} \quad n \to \infty.$$

Proof. From Lemma 18 we know that for all $n \in \mathbb{N}_{\geq 2}$

$$\ln(\kappa_n) \leq \frac{\ln^2(n)}{2\ln(2)} + \frac{3\ln(n)}{2}.$$

It follows that for all $n \in \mathbb{N}_{\geq 2}$ we have

$$\begin{aligned}\frac{\ln(\delta_n)}{\ln^2(n)} &= \ln\left(\sum_{i=0}^{n} \kappa_n\right) / \ln^2(n) \\ &\leq \ln\left(2 + (n-1)\exp\left(\frac{\ln^2(n)}{2\ln(2)} + \frac{3\ln(n)}{2}\right)\right) / \ln^2(n) \\ &\leq \left(\frac{\ln^2(n)}{2\ln(2)} + \frac{3\ln(n)}{2} + \ln(n)\right) / \ln^2(n) \\ &= \frac{1}{2\ln(2)} + o(1) \quad \text{as} \quad n \to \infty.\end{aligned}$$

Conversely, using the fact that $\delta_n \geq \kappa_n$, we find

$$\frac{\ln \delta_n}{\ln^2(n)} \geq \frac{\ln \kappa_n}{\ln^2(n)} = \frac{1}{2\ln(2)} + o(1) \quad \text{as} \quad n \to \infty.$$

Again, by the sandwich theorem we conclude

$$\frac{\ln \delta_n}{\ln^2(n)} \to \frac{1}{2\ln(2)} \quad \text{as} \quad n \to \infty. \qquad \blacktriangleleft$$

References

1. Dragana Bajic and Tatjana Loncar-Turukalo. A simple suboptimal construction of cross-bifix-free codes. *Cryptography and Communications*, 6(6):27–37, 2014. doi:10.1007/s12095-013-0088-8.
2. Stefano Bilotta. Variable-length non-overlapping codes. *IEEE Transactions on Information Theory*, 63(10):6530–6537, 2017. doi:10.1109/TIT.2017.2742506.
3. Stefano Bilotta, Elisa Pergola, and Renzo Pinzani. A new approach to cross-bifix-free sets. *IEEE Transactions on Information Theory*, 58(6):4058–4063, 2012. doi:10.1109/TIT.2012.2189479.
4. Francine Blanchet-Sadri and S. Duncan. Partial words and the critical factorization theorem. *Journal of Combinatorial Theory, Series. A*, 109(2):221–245, 2005. doi:10.1016/j.jcta.2004.09.002.
5. Francine Blanchet-Sadri, Justin Fowler, Joshua D. Gafni, and Kevin H. Wilson. Combinatorics on partial word correlations. *Journal of Combinatorial Theory, Series. A*, 117(6):607–624, 2010. doi:10.1016/j.jcta.2010.03.001.
6. Francine Blanchet-Sadri, Joshua D. Gafni, and Kevin H. Wilson. Correlations of partial words. In Wolfgang Thomas and Pascal Weil, editors, *STACS 2007, 24th Annual Symposium on Theoretical Aspects of Computer Science, Aachen, Germany, February 22-24, 2007, Proceedings*, volume 4393 of *Lecture Notes in Computer Science*, pages 97–108. Springer, 2007. doi:10.1007/978-3-540-70918-3_9.
7. Nathan J. Fine and Herbert S. Wilf. Uniqueness theorems for periodic functions. *Proceedings of the American Mathematical Society*, 16(1):109–114, 1965. doi:10.1090/S0002-9939-1965-0174934-9.
8. Daniel Gabric. Mutual borders and overlaps. *IEEE Transactions on Information Theory*, 68(10):6888–6893, 2022. doi:10.1109/TIT.2022.3167935.
9. Daniel Gabric, Narad Rampersad, and Jeffrey Shallit. An inequality for the number of periods in a word. *International Journal of Foundations of Computer Science*, 32(05):597–614, June 2021. doi:10.1142/s0129054121410094.
10. Leonidas J. Guibas and Andrew M. Odlyzko. Periods in strings. *Journal of Combinatorial Theory, Series. A*, 30:19–42, 1981. doi:10.1016/0097-3165(81)90038-8.

11 Leonidas J. Guibas and Andrew M. Odlyzko. String overlaps, pattern matching, and nontransitive games. *Journal of Combinatorial Theory, Series A*, 30(2):183–208, 1981. doi:10.1016/0097-3165(81)90005-4.

12 Vesa Halava, Tero Harju, and Lucian Ilie. Periods and binary words. *Journal of Combinatorial Theory, Series A*, 89(2):298–303, 2000. doi:10.1006/jcta.1999.3014.

13 Stepan Holub and Jeffrey O. Shallit. Periods and borders of random words. In Nicolas Ollinger and Heribert Vollmer, editors, *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, volume 47 of *LIPIcs*, pages 44:1–44:10. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.STACS.2016.44.

14 M. Lothaire, editor. *Combinatorics on Words*. Cambridge University Press, second edition, 1997.

15 Ora E. Percus and Paula A. Whitlock. Theory and Application of Marsaglia's Monkey Test for Pseudorandom Number Generators. *ACM Transactions on Modeling and Computer Simulation*, 5(2):87–100, April 1995. doi:10.1145/210330.210331.

16 Sven Rahmann and Eric Rivals. On the distribution of the number of missing words in random texts. *Combinatorics, Probability and Computing*, 12(01), January 2003. doi:10.1017/s0963548302005473.

17 Eric Rivals and Sven Rahmann. Combinatorics of Periods in Strings. In F. Orejas, P. Spirakis, and J. van Leuween, editors, *ICALP 2001, Proc. of the 28th International Colloquium on Automata, Languages and Programming, (ICALP), Crete, Greece, July 8-12, 2001*, volume 2076 of *Lecture Notes in Computer Science*, pages 615–626. Springer Verlag, 2001. doi:10.1007/3-540-48224-5_51.

18 Eric Rivals and Sven Rahmann. Combinatorics of periods in strings. *Journal of Combinatorial Theory, Series A*, 104(1):95–113, 2003. doi:10.1016/s0097-3165(03)00123-7.

19 Stéphane Robin, François Rodolphe, and Sophie Schbath. *DNA, Words and Models*. Cambrigde University Press, 2005.

20 Neil J. A. Sloane. The on-line encyclopedia of integer sequences. Published electronically at https://oeis.org, 2022.

21 William F. Smyth. *Computating Pattern in Strings*. Pearson – Addison Wesley, 2003.

A Omitted proofs

▶ **Lemma 3.** *Let p be a period of $u \in \Sigma^n$ and $k \in \mathbb{Z}_{\geq 0}$ such that $kp < n$. Then kp is also a period of u.*

Proof. If $p = 0$ or $k = 0$, the statement trivially holds. Suppose $p \in \{1, \ldots, n-1\}$ and $k > 0$. If $i, j \in \{0, \ldots, n-1\}$ such that $i \equiv j \mod kp$, then we also have $i \equiv j \mod p$, and hence $u[i] = u[j]$ by Definition 1. This shows kp is a period of u by Definition 1. ◀

▶ **Lemma 4.** *Let p be a period of $u \in \Sigma^n$ and q a period of the suffix $w = u[p\,..\,n-1]$. Then $p+q$ is a period of u. Moreover, $p+kq$ is also a period of u for all $k \in \mathbb{Z}_{\geq 0}$ with $p+kq < n$.*

Proof. By Definition 2 of period, the fact that p is a period of u implies $u[0\,..\,n-p-1] = u[p\,..\,n-1]$, while q is a period of w implies $w[0\,..\,n-p-q-1] = w[q\,..\,n-p-1]$. As w is the suffix of u starting at position p, we can combine the above results to find that

$$\begin{aligned} u[0\,..\,n-p-q-1] &= u[p\,..\,n-q-1] = w[0\,..\,n-p-q-1] \\ &= w[q\,..\,n-p-1] = u[p+q\,..\,n-1], \end{aligned}$$

which indicates that $p+q$ is a period of u. Moreover, if $p+iq$ is a period of u for some $i \in \mathbb{N}$, then we can similarly show that $p+(i+1)q$ is also a period of u if $p+(i+1)q < n$. It follows by induction that $p+kq$ is a period of u for all $k \in \mathbb{N}$ with $p+kq < n$. The case $k = 0$ is trivial. ◀

▶ **Lemma 5.** *Let p, q be periods of $u \in \Sigma^n$ with $0 \leq q \leq p$. Then the prefix and the suffix of length $n - q$ have the period $p - q$.*

Proof. Since p, q be periods of $u \in \Sigma^n$ with $0 \leq q \leq p$, we have

$$\begin{aligned} u[0\,..\,n-p-1] &= u[p\,..\,n-1] && \text{(by periodicity } p\text{)} \\ &= u[p-q\,..\,n-q-1] && \text{(by periodicity } q\text{)}. \end{aligned}$$

It follows that $u[0\,..\,n-q-1]$ has period $p-q$. Similarly the suffix of u of length $(n-q)$ also has period $p-q$. ◀

▶ **Lemma 6.** *Suppose p is a period of $u \in \Sigma^n$ and there exists a substring v of u of length at least p and with period r, where $r|p$. Then r is also a period of u.*

Proof. If $p = 0$, then $r = 0$ and the lemma trivially holds.

Otherwise p is non-zero. Let $i, j \in [0, n-1]$ with $i \equiv j \mod r$. We can write $v = u[h\,..\,k]$ with $0 \leq h < k \leq n-1$. Since v has length at least p, there exist $i', j' \in [h, k]$ such that $i \equiv i' \mod p$ and $j \equiv j' \mod p$. By Definition 1 of period, we have $u[i] = u[i']$ and $u[j] = u[j']$. Note that $i' \equiv i \equiv j \equiv j' \mod r$, because $r \mid p$. Applying Definition 1 again, we obtain $u[i'] = u[j']$. It follows that $u[i] = u[i'] = u[j'] = u[j]$. Therefore r is a period of u. ◀

B Independence of alphabet

Guibas and Odlyzko showed that for every autocorrelation, there exists a string over a binary alphabet with that autocorrelation [10]. A nice alternative constructive proof appears in [12]. We will now show that the same holds for arbitrary correlations of equally long strings.

▶ **Corollary 24.** *For any $t \in \Delta_n$, there exist $u, v \in \{\mathtt{a}, \mathtt{b}\}^n$ such that the correlation of u over v is t.*

Proof. Let t be the correlation of u' over v' with $u', v' \in \Sigma^n$. By Lemma 21, we can write $t = 0^{(n-j)}s_j$, where $s_j \in \{0,1\}^j$ is the autocorrelation of $u'[n-j\,..\,n-1] = v'[0\,..\,j-1]$. By the result of Guibas and Odlyzko, we know that there also exists some binary string $w \in \{\mathtt{a}, \mathtt{b}\}^j$ with the same autocorrelation. Without loss of generality, we can assume that w starts with $\mathtt{b}$. It follows that the constructed strings $u = \mathtt{a}^{(n-j)}w$ and $v = w\mathtt{b}^{(n-j)}$, which have a correlation of t by the proof of Lemma 21, use the same binary alphabet. ◀

We conclude that the number of correlations between strings of equal length is alphabet-independent (i.e. every alphabet of size at least 2 gives rise to the same set of correlations).

▶ Remark 25. Such a binary string w can be constructed from $u'[n-j\,..\,n-1]$ in linear time using the algorithm of Halava, Harju and Ilie [12]. Therefore u and v can also be constructed in linear time given u' and v'.

Lasserre Hierarchy for Graph Isomorphism and Homomorphism Indistinguishability

David E. Roberson
Department of Applied Mathematics and Computer Science,
Technical University of Denmark, Copenhagen, Denmark
QMATH, Department of Mathematical Sciences, University of Copenhagen, Denmark

Tim Seppelt
RWTH Aachen University, Germany

Abstract

We show that feasibility of the t^{th} level of the Lasserre semidefinite programming hierarchy for graph isomorphism can be expressed as a homomorphism indistinguishability relation. In other words, we define a class $\mathcal{L}_t$ of graphs such that graphs G and H are not distinguished by the t^{th} level of the Lasserre hierarchy if and only if they admit the same number of homomorphisms from any graph in $\mathcal{L}_t$. By analysing the treewidth of graphs in $\mathcal{L}_t$ we prove that the $3t^{\text{th}}$ level of Sherali–Adams linear programming hierarchy is as strong as the t^{th} level of Lasserre. Moreover, we show that this is best possible in the sense that $3t$ cannot be lowered to $3t-1$ for any t. The same result holds for the Lasserre hierarchy with non-negativity constraints, which we similarly characterise in terms of homomorphism indistinguishability over a family $\mathcal{L}_t^+$ of graphs. Additionally, we give characterisations of level-t Lasserre with non-negativity constraints in terms of logical equivalence and via a graph colouring algorithm akin to the Weisfeiler–Leman algorithm. This provides a polynomial time algorithm for determining if two given graphs are distinguished by the t^{th} level of the Lasserre hierarchy with non-negativity constraints.

2012 ACM Subject Classification Mathematics of computing → Combinatorics; Mathematics of computing → Graph theory

Keywords and phrases Lasserre hierarchy, homomorphism indistinguishability, Sherali–Adams hierarchy, treewidth, semidefinite programming, linear programming, graph isomorphism

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.101

Category Track A: Algorithms, Complexity and Games

Related Version *Preprint*: `https://arxiv.org/abs/2302.10538` [28]

Funding *David E. Roberson*: Supported by the Carlsberg Foundation Young Researcher Fellowship CF21-0682 – "Quantum Graph Theory".
Tim Seppelt: German Research Council (DFG) within Research Training Group 2236 (UnRAVeL)

Acknowledgements Initial discussions for this work took place at Dagstuhl Seminar 22051 "Finite and Algorithmic Model Theory".

1 Introduction

The aim of this paper is to relate two rich sets of tools used to distinguish non-isomorphic graphs: the Lasserre semidefinite programming hierarchy and homomorphism indistinguishability.

Distinguishing non-isomorphic graphs is a ubiquitous problem in the theoretical and practical study of graphs. The ability of certain graph invariants to distinguish graphs has long been a rich area of study, leading to fundamental questions such as the longstanding open problem of whether almost all graphs are determined by their spectrum [35]. In practice, deploying e.g. machine learning architectures powerful enough to distinguish graphs with different features is of great importance [12]. This motivates an in-depth study of the power of various graph invariants and tools used to distinguish graphs.

Among such techniques is the Lasserre semidefinite programming hierarchy [17] which can be used to relax the integer program for graph isomorphism $\mathrm{ISO}(G, H)$, cf. Section 2.4. This yields a sequence of semidefinite programs, i.e. the level-t Lasserre relaxation of $\mathrm{ISO}(G, H)$ for $t \geq 1$, which are infeasible for more and more non-isomorphic graphs as t grows. In [33, 25, 5], it was shown that in general only the level-$\Omega(n)$ Lasserre system of equations can distinguish all non-isomorphic n-vertex graphs. In [4], the Lasserre hierarchy was compared with the Sherali–Adams[1] linear programming hierarchy [32], which is closely related to the Weisfeiler–Leman algorithm [36, 3, 13], the arguably most relevant combinatorial method for distinguishing graphs. It was shown in [4] that there exists a constant c such that, for all graphs G and H, if the level-ct Sherali–Adams relaxation of $\mathrm{ISO}(G, H)$ is feasible then so is the level-t Lasserre relaxation, which in turn implies that the level-t Sherali–Adams relaxation is feasible, cf. [18].

Another set of expressive equivalence relations comparing graphs is given by homomorphism indistinguishability, a notion originating from the study of graph substructure counts. Two graphs G and H are *homomorphism indistinguishable* over a family of graphs $\mathcal{F}$, in symbols $G \equiv_{\mathcal{F}} H$, if the number of homomorphisms from F to G is equal to the number of homomorphisms from F to H for every graph $F \in \mathcal{F}$. The study of this notion began in 1967, when Lovász [19] showed that two graphs G and H are isomorphic if and only if they are homomorphism indistinguishable over all graphs. In recent years, many prominent equivalence relations comparing graphs were characterised as homomorphism indistinguishability relations over restricted graph classes [9, 10, 11, 8, 20, 15, 2, 23, 1, 27, 26]. For example, a folklore result asserts that two graphs have cospectral adjacency matrices iff they are homomorphism indistinguishable over all cycle graphs, cf. [15]. Two graphs are quantum isomorphic iff they are homomorphism indistinguishable over all planar graphs [20]. Furthermore, feasibility of the level-t Sherali–Adams relaxation of $\mathrm{ISO}(G, H)$ has been characterised as homomorphism indistinguishability over all graphs of treewidth at most $t-1$ [3, 13, 10]. In this way, notions from logic [10, 11, 26], category theory [8, 23, 1], algebraic graph theory [9, 15], and quantum groups [20] have been related to homomorphism indistinguishability.

1.1 Contributions

Although feasibility of the level-t Lasserre relaxation of $\mathrm{ISO}(G, H)$ was sandwiched between feasibility of the level-ct and level-t Sherali–Adams relaxation in [4], the constant c remained unknown. In fact, this c is not explicit and depends on the implementation details of an algorithm developed in that paper. Our main result asserts that c can be taken to be three and that this constant is best possible.

▶ **Theorem 1.** *For two graphs G and H and every $t \geq 1$, the following implications hold:*

$$G \simeq_{3t}^{\mathrm{SA}} H \implies G \simeq_{t}^{\mathrm{L}} H \implies G \simeq_{t}^{\mathrm{SA}} H$$

Furthermore, for every $t \geq 1$, there exist graphs G and H such that $G \simeq_{3t-1}^{\mathrm{SA}} H$ and $G \not\simeq_{t}^{\mathrm{L}} H$.

Here, $G \simeq_{t}^{\mathrm{L}} H$ and $G \simeq_{t}^{\mathrm{SA}} H$ denote that the level-t Lasserre relaxation and respectively the level-t Sherali–Adams relaxation of $\mathrm{ISO}(G, H)$ are feasible.

Theorem 1 is proven using the framework of homomorphism indistinguishability. In previous works [9, 22, 15, 26], the feasibility of various systems of equations associated to graphs like the Sherali–Adams relaxation of $\mathrm{ISO}(G, H)$ was characterised in terms of

[1] Following [4], when referring to the Sherali–Adams relaxation of $\mathrm{ISO}(G, H)$ in this article, we do not refer to the original relaxation [32] but to its variant introduced by [3, 13], which corresponds more directly to other graph properties, cf. Theorem 8 and [15].

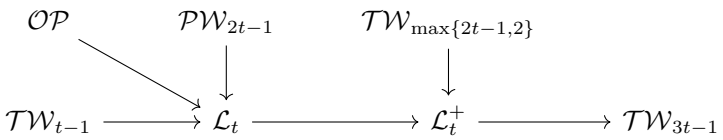


Figure 1 Relationship between $\mathcal{L}_t$, $\mathcal{L}_t^+$, the classes of graphs of bounded treewidth, bounded pathwidth, and the class of outerplanar graphs. An arrow $\mathcal{A} \to \mathcal{B}$ indicates that $\mathcal{A} \subseteq \mathcal{B}$ and thus that $G \equiv_{\mathcal{B}} H$ implies $G \equiv_{\mathcal{A}} H$ for all graphs G and H. For formal statements, see Sections 4.1 and 4.2.

homomorphism indistinguishability over certain graph classes. We continue this line of research by characterising the feasibility of the level-t Lasserre relaxation of $\mathrm{ISO}(G, H)$ by homomorphism indistinguishability of G and H over the novel class of graphs $\mathcal{L}_t$ introduced in Definition 22.

▶ **Theorem 2.** *For every integer $t \geq 1$, there is a minor-closed graph class $\mathcal{L}_t$ of graphs of treewidth at most $3t-1$ such that for all graphs G and H it holds that $G \simeq_t^{\mathrm{L}} H$ if and only if $G \equiv_{\mathcal{L}_t} H$.*

The bound on the treewidth of graphs in $\mathcal{L}_t$ in Theorem 2 yields the upper bound in Theorem 1 given the result of [3, 13, 4, 10] that two graphs G and H satisfy $G \simeq_t^{\mathrm{SA}} H$ if and only if they are homomorphism indistinguishable over the class $\mathcal{TW}_{t-1}$ of graphs of treewidth at most $t-1$. To our knowledge, Theorem 1 is the first result which tightly relates equivalence relations on graphs by comparing the graph classes which characterise them in terms of homomorphism indistinguishability.

Our techniques extend to a stronger version of the Lasserre hierarchy which imposes non-negativity constraints on all variables. Denoting feasibility of the level-t Lasserre relaxation of $\mathrm{ISO}(G, H)$ with non-negativity constraints by $G \simeq_t^{\mathrm{L}^+} H$, we characterise $\simeq_t^{\mathrm{L}^+}$ in terms of homomorphism indistinguishability over the graph class $\mathcal{L}_t^+$, defined in Definition 22 as a super class of $\mathcal{L}_t$. This is in line with previous work in [9, 15], where the feasibility of the level-t Sherali–Adams relaxation of $\mathrm{ISO}(G, H)$ without non-negativity constraints was characterised as homomorphism indistinguishable over the class $\mathcal{PW}_{t-1}$ of graphs of pathwidth at most $t-1$.

▶ **Theorem 3.** *For every integer $t \geq 1$, there is a minor-closed graph class $\mathcal{L}_t^+$ of graphs of treewidth at most $3t-1$ such that for all graphs G and H it holds that $G \simeq_t^{\mathrm{L}^+} H$ if and only if $G \equiv_{\mathcal{L}_t^+} H$.*

Given the aforementioned correspondence between the Sherali–Adams relaxation with and without non-negativity constraints and homomorphism indistinguishability over graphs of bounded treewidth and pathwidth, we conduct a detailed study of the relationship between the class of graphs of bounded treewidth, pathwidth, and the classes $\mathcal{L}_t$ and $\mathcal{L}_t^+$. Their results, depicted in Figure 1, yield independent proofs of the known relations between feasibility of the Lasserre relaxation with and without non-negativity constraints and the Sherali–Adams relaxation with and without non-negativity constraints [5, 4, 15] using the framework of homomorphism indistinguishability.

In the course of proving Theorems 2 and 3, we derive further equivalent characterisations of $\simeq_t^{\mathrm{L}}$ and $\simeq_t^{\mathrm{L}^+}$. These characterisations, which are mostly of a linear algebraic nature, ultimately yield a characterisation of $\simeq_t^{\mathrm{L}^+}$ in terms of a fragment of first-order logic with counting quantifiers and indistinguishability under a polynomial time algorithm akin to

the Weisfeiler–Leman algorithm. In this way, we obtain the following algorithmic result. It implies that *exact* feasibility of the Lasserre semidefinite program with non-negativity constraints can be tested in polynomial time. In general, only the *approximate* feasibility of semidefinite programs can be decided efficiently, e.g. using the ellipsoid method [16, 4].

▶ **Theorem 4.** *Let $t \geq 1$. Given graphs G and H, it can be decided in polynomial time whether $G \simeq_t^{\mathrm{L}^+} H$.*

Finally, for $t = 1$, we show that $\mathcal{L}_1$ and $\mathcal{L}_1^+$ are respectively equal to the class $\mathcal{OP}$ of outerplanar graphs and to the class of graphs of treewidth at most 2. The following Theorem 5 parallels a result of [20] asserting that two graphs G and H are indistinguishable under the 2-WL algorithm iff $G \simeq_1^{\mathrm{L}^+} H$.

▶ **Theorem 5.** *Two graphs G and H satisfy $G \simeq_1^{\mathrm{L}} H$ iff $G \equiv_{\mathcal{OP}} H$.*

1.2 Techniques

In the first part of the paper (Section 3), linear algebraic tools developed in [21, 20] are generalised to yield reformulations of the entire Lasserre hierarchy with and without non-negativity results. Section 4 is concerned with the graph theoretic properties of the graph classes $\mathcal{L}_t$ and $\mathcal{L}_t^+$. For understanding the homomorphism indistinguishability relations over these graph classes, the framework of bilabelled graphs and their homomorphism tensors developed in [22, 15] is used. Despite this, our approach is different from [15, 26] in the sense that here the graph classes $\mathcal{L}_t$ and $\mathcal{L}_t^+$ are inferred from given systems of equations, namely the Lasserre relaxation, rather than that a system of equations is built for a given graph class.

2 Preliminaries

2.1 Linear Algebra

Let $\mathcal{S}_+$ denote the family of real *positive semidefinite matrices*, i.e. of matrices M of the form $M_{ij} = v_i^T v_j$ for vectors $v_1, \dots, v_n$, the *Gram vectors* of M. Write $M \succeq 0$ iff $M \in \mathcal{S}_+$. Let $\mathcal{DNN}$ denote the family of *doubly non-negative matrices*, i.e. of entry-wise non-negative positive semidefinite matrices.

A linear map $\Phi\colon \mathbb{C}^{m\times m} \to \mathbb{C}^{n\times n}$ is *trace-preserving* if $\operatorname{tr}\Phi(X) = \operatorname{tr} X$ for all $X \in \mathbb{C}^{m\times m}$, *unital* if $\Phi(\mathrm{id}_m) = \mathrm{id}_n$, *$\mathcal{K}$-preserving* for a family of matrices $\mathcal{K}$ if $\Phi(K) \in \mathcal{K}$ for all $K \in \mathcal{K}$, *positive* if it is $\mathcal{S}_+$-preserving, i.e. if $\Phi(X)$ is positive semidefinite for all positive semidefinite X, *completely positive* if $\mathrm{id}_r \otimes \Phi$ is positive for all $r \in \mathbb{N}$. The *Choi matrix* of Φ is $C_\Phi = \sum_{i,j=1}^m E_{ij} \otimes \Phi(E_{ij}) \in \mathbb{C}^{mn\times mn}$.

A *tensor* is an element $A \in \mathbb{C}^{n^t \times n^t}$ for some $n, t \in \mathbb{N}$. The symmetric group $\mathfrak{S}_{2t}$ acts on $\mathbb{C}^{n^t\times n^t}$ by permuting the coordinates, i.e. for all $\boldsymbol{u}, \boldsymbol{v} \in [n]^t$, $A^\sigma(\boldsymbol{u}, \boldsymbol{v}) \coloneqq A(\boldsymbol{x}, \boldsymbol{y})$ where $\boldsymbol{x}_i \coloneqq (\boldsymbol{u}\boldsymbol{v})_{\sigma^{-1}(i)}$ and $\boldsymbol{y}_{j-t} \coloneqq (\boldsymbol{u}\boldsymbol{v})_{\sigma^{-1}(j)}$ for all $1 \leq i \leq t < j \leq 2t$.

For two vectors $v, w \in \mathbb{C}^n$, write $v \odot w$ for their *Schur product*, i.e. $(v \odot w)(i) \coloneqq v(i)w(i)$ for all $i \in [n]$.

2.2 Bilabelled Graphs and Homomorphism Tensors

All graphs in this article are undirected, finite, and without multiple edges. A graph is *simple* if it does not contain any loops. A *homomorphism* $h\colon F \to G$ from a graph F to a graph G is a map $V(F) \to V(G)$ such that for all $uv \in E(F)$ it holds that $h(u)h(v) \in E(G)$.

Note that this implies that any vertex in F carrying a loop must be mapped to a vertex carrying a loop in G. Write $\hom(F,G)$ for the number of homomorphisms from F to G. For a family of graphs $\mathcal{F}$ and graphs G and H write $G \equiv_{\mathcal{F}} H$ if G and H are *homomorphism indistinguishable over* $\mathcal{F}$, i.e. $\hom(F,G) = \hom(F,H)$ for all $F \in \mathcal{F}$. Since the graphs G and H into which homomorphisms are counted, are throughout assumed to be simple, looped graphs in $\mathcal{F}$ can generally be disregarded as they do not admit any homomorphisms into simple graphs.

We recall the following definitions from [20, 15]. Let $\ell \geq 1$. An (ℓ,ℓ)*-bilabelled graph* is a tuple $\boldsymbol{F} = (F, \boldsymbol{u}, \boldsymbol{v})$ where F is a graph and $\boldsymbol{u}, \boldsymbol{v} \in V(F)^\ell$. The $\boldsymbol{u}$ are the *in-labelled vertices* of $\boldsymbol{F}$ while the $\boldsymbol{v}$ are the *out-labelled vertices* of $\boldsymbol{F}$. Given a graph G, the *homomorphism tensor* of $\boldsymbol{F}$ for G is $\boldsymbol{F}_G \in \mathbb{C}^{V(G)^\ell \times V(G)^\ell}$ whose $(\boldsymbol{x},\boldsymbol{y})$-th entry is the number of homomorphisms $h\colon F \to G$ such that $h(\boldsymbol{u}_i) = \boldsymbol{x}_i$ and $h(\boldsymbol{v}_i) = \boldsymbol{y}_i$ for all $i \in [\ell]$.

For an (ℓ,ℓ)-bilabelled graph $\boldsymbol{F} = (F, \boldsymbol{u}, \boldsymbol{v})$, write $\operatorname{soe} \boldsymbol{F} \coloneqq F$ for the underlying unlabelled graph of $\boldsymbol{F}$. Write $\operatorname{tr} \boldsymbol{F}$ for the unlabelled graph underlying the graph obtained from $\boldsymbol{F}$ by identifying $\boldsymbol{u}_i$ with $\boldsymbol{v}_i$ for all $i \in [\ell]$. For $\sigma \in \mathfrak{S}_{2t}$, write $\boldsymbol{F}^\sigma \coloneqq (F, \boldsymbol{x}, \boldsymbol{y})$ where $\boldsymbol{x}_i \coloneqq (\boldsymbol{uv})_{\sigma(i)}$ and $\boldsymbol{y}_{j-t} \coloneqq (\boldsymbol{uv})_{\sigma(j)}$ for all $1 \leq i \leq t < j \leq 2t$, i.e. $\boldsymbol{F}^\sigma$ is obtained from $\boldsymbol{F}$ by permuting the labels according to σ. As a special case, define $\boldsymbol{F}^* \coloneqq (F, \boldsymbol{v}, \boldsymbol{u})$ the graph obtained by swapping in- and out-labels.

For two (ℓ,ℓ)-bilabelled graphs $\boldsymbol{F} = (F, \boldsymbol{u}, \boldsymbol{v})$ and $\boldsymbol{F}' = (F', \boldsymbol{u}', \boldsymbol{v}')$, write $\boldsymbol{F} \cdot \boldsymbol{F}'$ for the graph obtained from them by *series composition*. That is, the underlying unlabelled graph of $\boldsymbol{F} \cdot \boldsymbol{F}'$ is the graph obtained from the disjoint union of F and F' by identifying $\boldsymbol{v}_i$ and $\boldsymbol{u}'_i$ for all $i \in [\ell]$. Multiple edges arising in this process are removed. The in-labels of $\boldsymbol{F} \cdot \boldsymbol{F}'$ lie on $\boldsymbol{u}$, the out-labels on $\boldsymbol{v}'$. Moreover, write $\boldsymbol{F} \odot \boldsymbol{F}'$ for the *parallel composition* of $\boldsymbol{F}$ and $\boldsymbol{F}'$. That is, the underlying unlabelled graph of $\boldsymbol{F} \odot \boldsymbol{F}'$ is the graph obtained from the disjoint union of F and F' by identifying $\boldsymbol{u}_i$ with $\boldsymbol{u}'_i$ and $\boldsymbol{v}_i$ with $\boldsymbol{v}'_i$ for all $i \in [\ell]$. Again, multiple edges are dropped. The in-labels of $\boldsymbol{F} \odot \boldsymbol{F}'$ lie on $\boldsymbol{u}$, the out-labels on $\boldsymbol{v}$.

As observed in [20, 15], the benefit of these combinatorial operations is that they have an algebraic counterpart. Formally, for all graphs G and all (ℓ,ℓ)-bilabelled graphs $\boldsymbol{F}, \boldsymbol{F}'$, it holds that $\operatorname{soe} \boldsymbol{F}_G = \hom(\operatorname{soe} \boldsymbol{F}, G)$, $\operatorname{tr} \boldsymbol{F}_G = \hom(\operatorname{tr} \boldsymbol{F}, G)$, $(\boldsymbol{F}_G)^\sigma = (\boldsymbol{F}^\sigma)_G$, $(\boldsymbol{F} \cdot \boldsymbol{F}')_G = \boldsymbol{F}_G \cdot \boldsymbol{F}'_G$, and $(\boldsymbol{F} \odot \boldsymbol{F}')_G = \boldsymbol{F}_G \odot \boldsymbol{F}'_G$.

Slightly abusing notation, we say that two graphs G and H are homomorphism indistinguishable over a family of bilabelled graphs $\boldsymbol{S}$, in symbols $G \equiv_{\mathcal{S}} H$ if G and H are homomorphism indistinguishable over the family $\{\operatorname{soe} \boldsymbol{S} \mid \boldsymbol{S} \in \mathcal{S}\}$ of the underlying unlabelled graphs of the $\boldsymbol{S} \in \mathcal{S}$.

2.3 Pathwidth and Treewidth

▶ **Definition 6.** *Let F and T be graphs. A* T-decomposition of F *is a map $\beta\colon V(T) \to 2^{V(F)}$ such that*

1. $\bigcup_{t \in V(T)} \beta(t) = V(F)$,
2. *for every $e \in E(F)$, there is $t \in V(T)$ such that $e \subseteq \beta(t)$,*
3. *for every $v \in V(F)$, the set of $t \in V(T)$ such that $v \in \beta(t)$ induces a connected component of T.*

The width *of a T-decomposition β is $\max_{t \in V(T)} |\beta(t)| - 1$. For a graph class $\mathcal{T}$, the* $\mathcal{T}$-width of F *is the minimal width of a T-decomposition of F for $T \in \mathcal{T}$.*

The *treewidth* $\operatorname{tw} F$ of a graph F is the minimal width of a T-decomposition of F where T is a tree. Similarly, the *pathwidth* $\operatorname{pw} F$ is the minimal width of a P-decomposition of F where P is a path. For every $t \geq 0$, write $\mathcal{TW}_t$ and $\mathcal{PW}_t$ for the classes of all graphs of treewidth and respectively pathwidth at most t.

2.4 Systems of Equations for Graph Isomorphism

Two simple graphs G and H are isomorphic if and only if there exists a $\{0,1\}$-solution to the system of equations $\mathrm{ISO}(G,H)$ which comprises variables X_{gh} for $gh \in V(G) \times V(H)$ and equations

$$\sum_{h \in V(H)} X_{gh} - 1 = 0 \qquad \text{for all } g \in V(G), \tag{1}$$

$$\sum_{g \in V(G)} X_{gh} - 1 = 0 \qquad \text{for all } h \in V(H), \tag{2}$$

$$X_{gh} X_{g'h'} = 0 \qquad \text{for all } gh, g'h' \in V(G) \times V(H) \text{ s.t. } \mathrm{rel}_G(g,g') \neq \mathrm{rel}_H(h,h'). \tag{3}$$

Here, $\mathrm{rel}_G(g,g') = \mathrm{rel}_H(h',h')$ if and only if both pairs of vertices are adjacent, non-adjacent, or identical.

The Lasserre relaxation of $\mathrm{ISO}(G,H)$ is defined as follows. An element $\{g_1h_1, \dots g_\ell h_\ell\} \in \binom{V(G)\times V(H)}{\ell}$ is a *partial isomorphism* if $g_i = g_j \Leftrightarrow h_i = h_j$ and $g_ig_j \in E(G) \Leftrightarrow h_ih_j \in E(H)$ for all $i,j \in [\ell]$. See also [28] for a comparison to the version used in [4].

▶ **Definition 7.** *Let $t \geq 1$. The* level-t Lasserre relaxation for graph isomorphism *has variables y_I ranging over $\mathbb{R}$ for $I \in \binom{V(G)\times V(H)}{\leq 2t}$. The constraints are*

$$M_t(y) \coloneqq (y_{I\cup J})_{I,J \in \binom{V(G)\times V(H)}{\leq t}} \succeq 0, \tag{4}$$

$$\sum_{h \in V(H)} y_{I \cup \{gh\}} = y_I \text{ for all } I \text{ s.t. } |I| \leq 2t-2 \text{ and all } g \in V(G), \tag{5}$$

$$\sum_{g \in V(G)} y_{I \cup \{gh\}} = y_I \text{ for all } I \text{ s.t. } |I| \leq 2t-2 \text{ and all } h \in V(H), \tag{6}$$

$$y_I = 0 \text{ if } I \text{ s.t. } |I| \leq 2t \text{ is not partial isomorphism} \tag{7}$$

$$y_\emptyset = 1. \tag{8}$$

If the system is feasible for two graphs G and H, write $G \simeq_t^{\mathrm{L}} H$. If the system together with the constraint $y_I \geq 0$ for all $I \in \binom{V(G)\times V(H}{\leq 2t}$ is feasible, write $G \simeq_t^{\mathrm{L}^+} H$.

For a definition of the Sherali–Adams relaxation of $\mathrm{ISO}(G,H)$ in the version used here following [4], the reader is referred to [14, Appendix D.1]. Instead of feasibility of the level-t Sherali–Adams relaxation, one may think of the following equivalent notions:

▶ **Theorem 8** ([4, 10, 6])**.** *Let $t \geq 1$. For graphs G and H, the following are equivalent:*

1. *the level-t Sherali–Adams relaxation of $\mathrm{ISO}(G,H)$ is feasible, i.e. $G \simeq_t^{\mathrm{SA}} H$,*
2. *G and H satisfy the same sentences of t-variable first order logic with counting quantifiers,*
3. *G and H are homomorphism indistinguishable over the graphs of treewidth at most $t-1$,*
4. *G and H are not distinguished by the $(t-1)$-dimensional Weisfeiler–Leman algorithm,*

3 From Lasserre to Homomorphism Tensors

In this section, the tools are developed which will be used to translate a solution to the level-t Lasserre relaxation into a statement on homomorphism indistinguishability. For this purpose, three equivalent characterisations of $\simeq_t^{\mathrm{L}}$ and $\simeq_t^{\mathrm{L}^+}$ are introduced. Theorems 9 and 10 summarise our results. The notions in items 2–4 and the graph classes $\mathcal{L}_t$ and $\mathcal{L}_t^+$ are defined in Sections 3.1, 3.2, 3.4, and 4, respectively. Most of the proofs are of a linear algebraic nature. Graph theoretical repercussions are discussed in Section 4.

▶ **Theorem 9.** *Let $t \geq 1$. For graphs G and H, the following are equivalent:*
1. *the level-t Lasserre relaxation of* $\mathrm{ISO}(G, H)$ *is feasible,*
2. *G and H are level-t $\mathcal{S}_+$-isomorphic,*
3. *there is a level-t $\mathcal{S}_+$-isomorphism map from G to H,*
4. *G and H are partially t-equivalent,*
5. *G and H are homomorphism indistinguishable over $\mathcal{L}_t$.*

▶ **Theorem 10.** *Let $t \geq 1$. For graphs G and H, the following are equivalent:*
1. *the level-t Lasserre relaxation of* $\mathrm{ISO}(G, H)$ *with non-negativity constraints is feasible,*
2. *G and H are level-t $\mathcal{DNN}$-isomorphic,*
3. *there is a level-t $\mathcal{DNN}$-isomorphism map from G to H,*
4. *G and H are t-equivalent,*
5. *G and H are homomorphism indistinguishable over $\mathcal{L}_t^+$.*

Variants of the notions in items 2–4 have already been defined for the case $t = 1$ in [22]. Our contribution amounts to extending these definitions to the entire Lasserre hierarchy. A recurring theme in this context is accounting for additional symmetries. The variables y_I of the Lasserre system of equations, cf. Definition 7, are indexed by sets of vertex pairs rather than by tuples of such. Hence, when passing from such variables to tuple-indexed matrices, one must impose the additional symmetries arising this way. This is formalised at various points using an action of the symmetric group on the axes of the matrices. In the case $t = 1$, such a set up is not necessary since indices I are of size at most 2 and all occurring matrices can be taken to be invariant under transposition.

In the subsequent sections, Theorems 9 and 10 will be proven in parallel. The equivalence of items 1 and 2, 2 and 3, and 3 and 4 are established in Section 3.3, Section 3.2, and Section 3.4, respectively. The statements on homomorphism indistinguishability are proven in Section 4.

3.1 Isomorphism Relaxations via Matrix Families

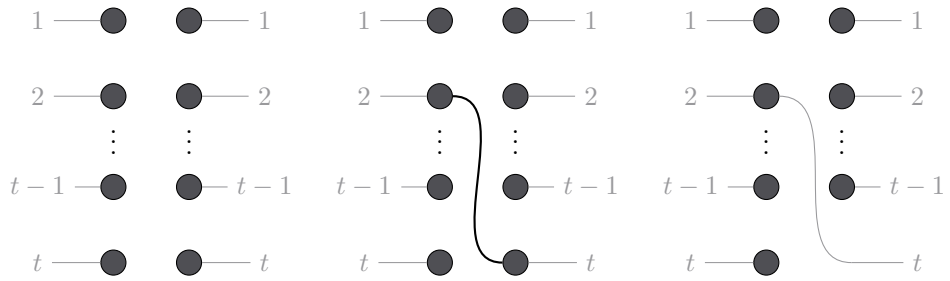
In this section, as a first step towards proving Theorems 9 and 10, the notion of level-t $\mathcal{K}$-isomorphic graphs for arbitrary families of matrices $\mathcal{K}$ is introduced. In [22], level-1 $\mathcal{K}$-isomorphic graphs where studied for various families of matrices $\mathcal{K}$. In this work, the main interest lies on the family of positive semidefinite matrices $\mathcal{S}_+$ and the family of entry-wise non-negative positive semidefinite matrices $\mathcal{DNN}$. Level-t isomorphism for these families is proven to correspond to $\simeq_t^{\mathrm{L}}$ and $\simeq_t^{\mathrm{L}^+}$ respectively, cf. Theorems 16 and 17.

▶ **Definition 11.** *Let $\mathcal{K}$ be a family of matrices. Graphs G and H are said to be* level-t $\mathcal{K}$-isomorphic*, in symbols $G \cong_{\mathcal{K}}^t H$, if there is a matrix $M \in \mathcal{K}$ with rows and columns indexed by $(V(G) \times V(H))^t$ such that for every $g_1h_1 \dots g_th_t, g_{t+1}h_{t+1} \dots g_{2t}h_{2t} \in (V(G) \times V(H))^t$ the following equations hold:*

For every $i \in [2t]$,

$$\sum_{g_i \in V(G)} M_{g_1h_1 \dots g_th_t, g_{t+1}h_{t+1} \dots g_{2t}h_{2t}} = \sum_{h_i \in V(H)} M_{g_1h_1 \dots g_th_t, g_{t+1}h_{t+1} \dots g_{2t}h_{2t}}, \tag{9}$$

$$\sum_{h_1', \dots, h_{2t}' \in V(H)} M_{g_1h_1' \dots g_th_t', g_{t+1}h_{t+1}' \dots g_{2t}h_{2t}'} = 1 = \sum_{g_1', \dots, g_{2t}' \in V(G)} M_{g_1'h_1 \dots g_t'h_t, g_{t+1}'h_{t+1} \dots g_{2t}'h_{2t}}. \tag{10}$$



(a) $\boldsymbol{J}$. (b) $\boldsymbol{A}^{2,2t}$. (c) $\boldsymbol{I}^{2,2t}$.

Figure 2 Examples of the atomic graphs from Definition 13. The gray lines (the *wires* [20]) indicate the in-labels (left) and out-labels (right).

If $\mathrm{rel}_G(g_1,\dots,g_{2t}) \neq \mathrm{rel}_H(h_1,\dots,h_{2t})$ *then*

$$M_{g_1h_1\dots g_th_t,g_{t+1}h_{t+1}\dots g_{2t}h_{2t}} = 0. \tag{11}$$

For all $\sigma \in \mathfrak{S}_{2t}$,

$$M_{g_1h_1\dots g_th_t,g_{t+1}h_{t+1}\dots g_{2t}h_{2t}} = M_{g_{\sigma(1)}h_{\sigma(1)}\cdots g_{\sigma(t)}h_{\sigma(t)},g_{\sigma(t+1)}h_{\sigma(t+1)}\cdots g_{\sigma(2t)}h_{\sigma(2t)}}. \tag{12}$$

Note that for $t = 1$ and any family of matrices $\mathcal{K}$ closed under taking transposes Equation (12) is vacuous.

Systems of equations comparing graphs akin to Equations (9)–(12) were also studied by [15]. Feasibility of such equations is typically invariant under taking the complements of the graphs as remarked below. This semantic property of the relation $\cong^t_{\mathcal{K}}$ is relevant in the context of homomorphism indistinguishability as shown by [30].

▶ Remark 12. For a simple graph G, write $\overline{G}$ for its complement, i.e. $V(\overline{G}) \coloneqq V(G)$ and $E(\overline{G}) \coloneqq \binom{V(G)}{2} \setminus E(G)$. For all graphs G and H and $g_1,\dots,g_{2t} \in V(G)$, $h_1,\dots,h_{2t} \in V(H)$, it holds that

$$\mathrm{rel}_G(g_1,\dots,g_{2t}) = \mathrm{rel}_H(h_1,\dots,h_{2t}) \iff \mathrm{rel}_{\overline{G}}(g_1,\dots,g_{2t}) = \mathrm{rel}_{\overline{H}}(h_1,\dots,h_{2t}).$$

Thus, $G \cong^t_{\mathcal{K}} H$ if and only if $\overline{G} \cong^t_{\mathcal{K}} \overline{H}$ for all families of matrices $\mathcal{K}$ and $t \in \mathbb{N}$.

3.2 Choi Matrices and Isomorphism Maps

In this section, an alternative characterisation for level-t $\mathcal{K}$-isomorphism is given. Intuitively, the indices of the matrix $M \in \mathbb{C}^{(V(G)\times V(H))^t\times(V(G)\times V(H))^t}$ from Definition 11 are regrouped yielding a linear map $\Phi\colon \mathbb{C}^{V(G)^t\times V(G)^t} \to \mathbb{C}^{V(H)^t\times V(H)^t}$. In linear algebraic terms, M is the Choi matrix of Φ. The map Φ will later be interpreted as a function sending homomorphism tensors of (t,t)-bilabelled graphs $\boldsymbol{F}_G \in \mathbb{C}^{V(G)^t\times V(G)^t}$ with respect to G to their counterparts $\boldsymbol{F}_H$ for H.

The most basic bilabelled graphs, so called *atomic* graphs, make their first appearance in Theorem 14. These graphs are used to reformulate Equations (7) and (11). The atomic graphs are also the graphs which the sets $\mathcal{L}_t$ and $\mathcal{L}^+_t$ of Theorems 2 and 3 are generated by, cf. Definition 22. Examples are depicted in Figures 2 and 3.

▶ **Definition 13.** *Let* $t \geq 1$. *A* (t,t)*-bilabelled graph* $\boldsymbol{F} = (F, \boldsymbol{u}, \boldsymbol{v})$ *is* atomic *if all its vertices are labelled. Write* $\mathcal{A}_t$ *for the set of* (t,t)*-bilabelled atomic graphs. Note that the the set of atomic graphs* $\mathcal{A}_t$ *is generated under parallel composition by the graphs*

- $\boldsymbol{J} \coloneqq (J, (1, \dots, t), (t+1, \dots, 2t))$ *with* $V(J) = [2t]$, $E(J) = \emptyset$,
- $\boldsymbol{A}^{ij} \coloneqq (A^{ij}, (1, \dots, t), (t+1, \dots, 2t))$ *with* $V(A^{ij}) = [2t]$, $E(A^{ij}) = \{ij\}$ *for* $1 \leq i < j \leq 2t$,
- $\boldsymbol{I}^{ij}$ *for* $1 \leq i < j \leq 2t$ *which is obtained from* $\boldsymbol{A}^{ij}$ *by contracting the edge* ij.

The following Theorem 14 relates the properties of Φ and M. In Equation (15), J denotes the all-ones matrix of appropriate dimension. Its proof is deferred to the full version [28].

▶ **Theorem 14.** *Let* $t \geq 1$. *Let* G *and* H *be graphs and* $\mathcal{K} \in \{\mathcal{DNN}, \mathcal{S}_+\}$ *be a family of matrices. Let* $\Phi \colon \mathbb{C}^{V(G)^t \times V(G)^t} \to \mathbb{C}^{V(H)^t \times V(H)^t}$ *be a linear map. Then the following are equivalent.*

1. *The Choi matrix* C_Φ *of* Φ *satisfies Equations* (9)–(12) *and* $C_\Phi \in \mathcal{K}$,
2. Φ *is a* level-t $\mathcal{K}$-isomorphism map from G to H, *i.e. it satisfies*

$$\Phi \text{ is completely } \mathcal{K}\text{-preserving}, \tag{13}$$

$$\Phi(\boldsymbol{A}_G \odot X) = \boldsymbol{A}_H \odot \Phi(X) \text{ for all atomic } \boldsymbol{A} \in \mathcal{A}_t \text{ and all } X \in \mathbb{C}^{V(G)^t \times V(G)^t}, \tag{14}$$

$$\Phi(J) = J = \Phi^*(J), \tag{15}$$

$$\Phi(X^\sigma) = \Phi(X)^\sigma \text{ for all } \sigma \in \mathfrak{S}_{2t} \text{ and all } X \in \mathbb{C}^{V(G)^t \times V(G)^t}. \tag{16}$$

3. Φ^* *is a level-*t $\mathcal{K}$*-isomorphism map from* H *to* G.

We remark that Theorem 14 and in particular its Equation (15) has brought us closer to interpreting the Lasserre system of equation from the perspective of homomorphism indistinguishability. As argued in Remark 15, the map Φ, which will be understood as mapping homomorphism tensors $\boldsymbol{F}_G$ to $\boldsymbol{F}_H$, is sum-preserving. Since the sum of the entries of these tensors equals the number of homomorphisms from their underlying unlabelled graphs to G and H, respectively, for establishing a connection between $\mathcal{K}$-isomorphism maps and homomorphism indistinguishability.

▶ Remark 15. If a linear map $\Phi \colon \mathbb{C}^{n \times n} \to \mathbb{C}^{m \times m}$ is such that $J = \Phi^*(J)$ then it is *sum-preserving*, i.e. $\operatorname{soe} X = \operatorname{soe} \Phi(X)$ for all $X \in \mathbb{C}^{n \times n}$. Indeed, $\operatorname{soe} X = \langle X, J \rangle = \langle X, \Phi^*(J) \rangle = \langle \Phi(X), J \rangle = \operatorname{soe} \Phi(X)$ where $\langle A, B \rangle \coloneqq \operatorname{tr}(AB^*)$. In particular, if there is Φ satisfying Equations (14) and (15) for graphs G and H then $|G| = |H|$.

3.3 Connection to Lasserre

By the following Theorems 16 and 17, the notions introduced in Definition 11 and Theorem 14 are equivalent to the object of our main interest, namely feasibility of the level-t Lasserre relaxation with and without non-negativity constraints. Our results extend those of [22, Lemma 9.1] to the entire Lasserre hierarchy. The proofs are deferred to the full version [28].

▶ **Theorem 16.** *Let* $t \geq 1$. *Two graphs* G *and* H *are level-*t $\mathcal{S}_+$*-isomorphic if and only if the level-*t *system of the Lasserre hierarchy for graph isomorphism, i.e. Equations* (4)–(8), *is feasible.*

▶ **Theorem 17.** *Let* $t \geq 1$. *Two graphs* G *and* H *are level-*t $\mathcal{DNN}$*-isomorphic if and only if the level-*t *system of the Lasserre hierarchy for graph isomorphism Equations* (4)–(8) *with the additional constraint* $y_I \geq 0$ *for all* $I \in \binom{V(G) \times V(H)}{\leq 2t}$ *is feasible.*

(a) J. (b) $A^{1,2}$. (c) $I^{1,2}$.

Figure 3 The three atomic graphs in $\mathcal{A}_1$.

3.4 Isomorphisms between Matrix Algebras

To the two reformulations of $\simeq_t^{\mathrm{L}}$ and $\simeq_t^{\mathrm{L}^+}$ from the previous sections, a third characterisation is added in this section. It is shown that two graphs are level-t $\mathcal{S}_+$-isomorphic ($\mathcal{DNN}$-isomorphic) if and only if certain matrix algebras associated to them are isomorphic. These algebras will be identified as the algebras of homomorphism tensors for graphs from the families $\mathcal{L}_t$ and $\mathcal{L}_t^+$. The so called (partially) coherent algebras considered in this section are natural generalisations of the coherent algebra which are well-studied in the context of the 2-dimensional Weisfeiler–Leman algorithm [7].

3.4.1 Partially Coherent Algebras and $\mathcal{S}_+$-Isomorphism Maps

Let $S \subseteq \mathbb{C}^{n^t \times n^t}$. A matrix algebra $\mathcal{A} \subseteq \mathbb{C}^{n^t \times n^t}$ is *S-partially coherent* if it is unital, self-adjoint, contains the all-ones matrix, and is closed under Schur products with any matrix in S. A matrix algebra $\mathcal{A} \subseteq \mathbb{C}^{n^t \times n^t}$ is *self-symmetrical* if for every $A \in \mathcal{A}$ and $\sigma \in \mathfrak{S}_{2t}$ also $A^\sigma \in \mathcal{A}$. Note that for $t = 1$, an algebra $\mathcal{A}$ is self-symmetrical if for all $A \in \mathcal{A}$ also $A^T \in \mathcal{A}$.

▶ **Definition 18.** *Given a graph G, construct its* t-partially coherent algebra $\widehat{\mathcal{A}}_G^t$ *as the minimal self-symmetrical S-partially coherent algebra where S is the set of homomorphism tensors of (t,t)-bilabelled atomic graphs for G.*

Two n-vertex graphs G and H are partially t-equivalent *if there is a* partial t-equivalence*, i.e. a vector space isomorphism $\varphi\colon \widehat{\mathcal{A}}_G^t \to \widehat{\mathcal{A}}_H^t$ such that*

1. $\varphi(M^*) = \varphi(M)^*$ *for all* $M \in \widehat{\mathcal{A}}_G^t$,
2. $\varphi(MN) = \varphi(M)\varphi(N)$ *for all* $M, N \in \widehat{\mathcal{A}}_G^t$,
3. $\varphi(I) = I$, $\varphi(\boldsymbol{A}_G) = \boldsymbol{A}_H$ *for all* $\boldsymbol{A} \in \mathcal{A}_t$, *and* $\varphi(J) = J$,
4. $\varphi(\boldsymbol{A}_G \odot M) = \boldsymbol{A}_H \odot \varphi(M)$ *for all* $\boldsymbol{A} \in \mathcal{A}_t$ *and any* $M \in \widehat{\mathcal{A}}_G^t$.
5. $\varphi(M^\sigma) = \varphi(M)^\sigma$ *for all* $M \in \widehat{\mathcal{A}}_G^t$ *and all* $\sigma \in \mathfrak{S}_{2t}$.

The following Theorem 19 extends [22, Theorem 5.2]. Its proof is deferred to the full version [28].

▶ **Theorem 19.** *Let $t \geq 1$. Two graphs G and H are partially t-equivalent if and only if there is a level-t $\mathcal{S}_+$-isomorphism map from G to H.*

3.4.2 Coherent Algebras and $\mathcal{DNN}$-Isomorphism Maps

A matrix algebra $\mathcal{A} \subseteq \mathbb{C}^{n \times n}$ is *coherent* if it is unital, self-adjoint, contains the all-ones matrix and is closed under Schur products.

For $t = 1$, the 1-adjacency algebra as defined below is equal to the well-studied *adjacency algebra* of a graph G, cf. [7]. The latter is the smallest coherent algebra containing the adjacency matrix of the graph. The former is generated by the homomorphism tensors of $(1,1)$-bilabelled atomic graphs. These graphs are depicted in Figure 3. Their homomorphism tensors are the all-ones matrix, the adjacency matrix of the graph, and the identity matrix.

▶ **Definition 20.** *Let $t \geq 1$. The* t-adjacency algebra $\mathcal{A}_G^t$ *of a graph G is the self-symmetrical coherent algebra generated by the homomorphism tensors of the atomic graphs $\mathcal{A}_t$.*

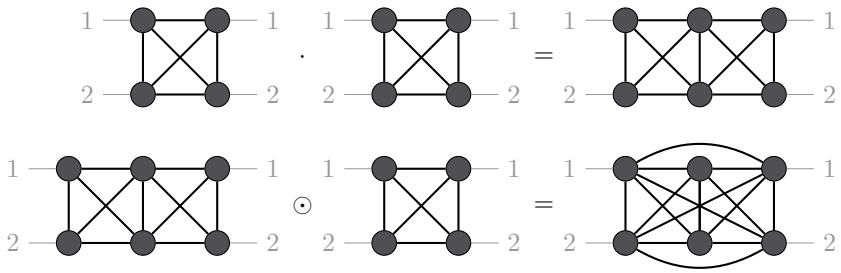


Figure 4 The bilabelled graphs in Observation 23 for $t = 2$.

Two n-vertex graphs G and H are t-equivalent *if there is* t-equivalence*, i.e. a vector space isomorphism $\varphi\colon \mathcal{A}_G^t \to \mathcal{A}_H^t$ such that*

1. $\varphi(M^*) = \varphi(M)^*$ *for all* $M \in \mathcal{A}_G^t$,
2. $\varphi(MN) = \varphi(M)\varphi(N)$ *for all* $M, N \in \mathcal{A}_G^t$,
3. $\varphi(I) = I$, $\varphi(\boldsymbol{A}_G) = \boldsymbol{A}_H$ *for all* $\boldsymbol{A} \in \mathcal{A}_t$, *and* $\varphi(J) = J$,
4. $\varphi(M \odot N) = \varphi(M) \odot \varphi(N)$ *for all* $M, N \in \mathcal{A}_G^t$.
5. $\varphi(M^\sigma) = \varphi(M)^\sigma$ *for all* $M \in \mathcal{A}_G^t$ *and all* $\sigma \in \mathfrak{S}_{2t}$.

The following Theorem 21 extends [22, Theorem 6.3]. Its proof is deferred to the full version [28].

▶ **Theorem 21.** *Let $t \geq 1$. Two graphs G and H are t-equivalent if and only if there is a level-t $\mathcal{DNN}$-isomorphism map from G to H.*

4 Homomorphism Indistinguishability

Using techniques from [15], we finally establish a characterisation of when the level-t Lasserre relaxation of $\mathrm{ISO}(G, H)$ is feasible in terms of homomorphism indistinguishability of G and H. In order to do so, we introduce the graph classes $\mathcal{L}_t$ and $\mathcal{L}_t^+$. In Section 4.1, we relate $\mathcal{L}_t$ and $\mathcal{L}_t^+$ to the classes of graphs of bounded treewidth and pathwidth obtaining the results depicted in Figure 1. In Section 4.2, $\mathcal{L}_1$ and $\mathcal{L}_1^+$ are identified as the classes of outerplanar graphs and graphs of treewidth two, respectively.

▶ **Definition 22.** *Let $t \geq 1$. Write $\mathcal{L}_t^+$ for the class of (t,t)-bilabelled graphs generated by the set of atomic graphs $\mathcal{A}_t$ under parallel composition, series composition, and the action of $\mathfrak{S}_{2t}$ on the labels.*

Write $\mathcal{L}_t \subseteq \mathcal{L}_t^+$ for the class of (t,t)-bilabelled graphs generated by the set of atomic graphs $\mathcal{A}_t$ under parallel composition with graphs from $\mathcal{A}_t$, series composition, and the action of $\mathfrak{S}_{2t}$ on the labels.

Note that the only difference between $\mathcal{L}_t$ and $\mathcal{L}_t^+$ is that $\mathcal{L}_t$ is closed under parallel composition with atomic graphs only. This reflects an observation by [15] relating the closure under arbitrary gluing products to non-negative solutions to systems of equations characterising homomorphism indistinguishability. Intuitively, one may use arbitrary Schur products, the algebraic counterparts of gluing, for a Vandermonde interpolation argument, cf. [14, Appendix B.4].

The following Observation 23 illustrates how the operations in Definition 22 can be used to generate more complicated graphs from the atomic graphs, cf. Figure 4.

▶ **Observation 23.** *Let $t \geq 1$. The class $\mathcal{L}_t$ contains a bilabelled graph whose underlying unlabelled graph is isomorphic to the $3t$-clique K_{3t}.*

Proof. Let $\boldsymbol{E} \coloneqq \bigodot_{1 \leq i < j \leq 2t} \boldsymbol{A}^{ij} \in \mathcal{A}_t$. The graph underlying $\boldsymbol{E} \odot (\boldsymbol{E} \cdot \boldsymbol{E})$ is isomorphic to K_{3t}. ◀

The only missing implications of Theorems 9 and 10 follow from the next two theorems:

▶ **Theorem 24.** *Let $t \geq 1$. Two graphs G and H are homomorphism indistinguishable over $\mathcal{L}_t$ if and only if they are partially t-equivalent.*

▶ **Theorem 25.** *Let $t \geq 1$. Two graphs G and H are homomorphism indistinguishable over $\mathcal{L}_t^+$ if and only if they are t-equivalent.*

For the proofs of Theorems 24 and 25, we extend the framework developed by [15]. In this work, the authors introduced tools for constructing systems of equations characterising homomorphism indistinguishably over classes of labelled graphs. A requirement of these tools is that the graph class in question is *inner-product compatible* [15, Definition 24]. This means that for every two labelled graphs $\boldsymbol{R}$ and $\boldsymbol{S}$ one can write the inner-product of their homomorphism vectors $\boldsymbol{R}_G$ and $\boldsymbol{S}_G$ as the sum-of-entries of some $\boldsymbol{T}_G$ where $\boldsymbol{T}$ is labelled graph from the class. Due to the correspondence between combinatorial operations on labelled graphs and algebraic operations on their homomorphism vectors, cf. Section 2.2, this is equivalent to the graph theoretic assumption that $\operatorname{soe}(\boldsymbol{R} \odot \boldsymbol{S}) = \operatorname{soe}(\boldsymbol{T})$, i.e. the unlabelled graph obtained by unlabelling the gluing product of $\boldsymbol{R}$ and $\boldsymbol{S}$ can be labelled such that the resulting labelled graph is in the class.

We extend this notion to bilabelled graphs. A class of (t,t)-bilabelled graphs $\mathcal{S}$ is said to be *inner-product compatible* if for all $\boldsymbol{R}, \boldsymbol{S} \in \mathcal{S}$ there is a graph $\boldsymbol{T} \in \mathcal{S}$ such that $\operatorname{tr}(\boldsymbol{R} \cdot \boldsymbol{S}^*) = \operatorname{soe}(\boldsymbol{T})$. This definition is inspired by the inner-product on $\mathbb{C}^{n \times n}$ given by $\langle A, B \rangle \coloneqq \operatorname{tr}(AB^*)$.

▶ **Lemma 26.** *Let $t \geq 1$. The classes $\mathcal{L}_t$ and $\mathcal{L}_t^+$ are inner-product compatible.*

Proof. Since $\mathcal{L}_t$ is closed under matrix products and taking transposes, it suffices to show that for every $\boldsymbol{S} \in \mathcal{L}_t$ the graph $\operatorname{tr} \boldsymbol{S}$ is the underlying unlabelled graph of some element of $\mathcal{L}_t$. Indeed, for every (t,t)-bilabelled graphs $\boldsymbol{F}$ it holds that $\operatorname{tr}(\boldsymbol{F}) = \operatorname{soe}(\boldsymbol{I}^{1,t+1} \odot \cdots \odot \boldsymbol{I}^{t,2t} \odot \boldsymbol{F})$ where the $\boldsymbol{I}^{ij}$ are as in Definition 13. Since $\mathcal{L}_t$ is closed under parallel composition with atomic graphs, the claim follows. For $\mathcal{L}_t^+$, an analogous argument yields the claim. ◀

The following Theorem 27, which extends the toolkit for constructing systems of equations characterising homomorphism indistinguishability over families of bilabelled graphs, is the bilabelled analogue of [15, Theorem 13]. Write $\mathbb{C}\mathcal{S}_G \subseteq \mathbb{C}^{V(G)^t \times V(G)^t}$ for the vector space spanned by homomorphism tensors $\boldsymbol{S}_G$ for $\boldsymbol{S} \in \mathcal{S}$.

▶ **Theorem 27.** *Let $t \geq 1$ and $\mathcal{S}$ be an inner-product compatible class of (t,t)-bilabelled graphs containing $\boldsymbol{J}$. For graphs G and H, the following are equivalent:*

1. *G and H are homomorphism indistinguishable over $\mathcal{S}$,*
2. *there exists a sum-preserving vector space isomorphism $\varphi \colon \mathbb{C}\mathcal{S}_G \to \mathbb{C}\mathcal{S}_H$ such that $\varphi(\boldsymbol{S}_G) = \boldsymbol{S}_H$ for all $\boldsymbol{S} \in \mathcal{S}$.*

Theorems 24 and 25 follows from this theorem as described in the full version [28].

4.1 The Classes $\mathcal{L}_t$ and $\mathcal{L}_t^+$ and Graphs of Bounded Treewidth

In this section, the classes $\mathcal{L}_t$ and $\mathcal{L}_t^+$ are compared to the classes of graphs of bounded treewidth and pathwidth. Figure 1 depicts the relationships between these classes. The first result, Lemma 28, gives an upper bound on the treewidth of graphs in $\mathcal{L}_t^+$.

▶ **Lemma 28.** *Let $t \geq 1$. The treewidth of an unlabelled graph F underlying some $\boldsymbol{F} = (F, \boldsymbol{u}, \boldsymbol{v}) \in \mathcal{L}_t^+$ is at most $3t-1$.*

Proof. By structural induction, it is shown that every $\boldsymbol{F} = (F, \boldsymbol{u}, \boldsymbol{v}) \in \mathcal{L}_t^+$ admits a tree decomposition $\beta\colon V(T) \to 2^{V(F)}$ of width at most $3t-1$ such that the labelled vertices $\boldsymbol{u}$ and $\boldsymbol{v}$ lie together in one bag, i.e. there exists $x \in V(T)$ such that $\{\boldsymbol{u}_1, \dots, \boldsymbol{u}_t, \boldsymbol{v}_1, \dots, \boldsymbol{v}_t\} \subseteq \beta(x)$.

This is clearly the case for all $\boldsymbol{F} \in \mathcal{A}_t$. Let $\boldsymbol{F} = (F, \boldsymbol{u}, \boldsymbol{v})$ and $\boldsymbol{F}' = (F', \boldsymbol{u}', \boldsymbol{v}')$ from $\mathcal{L}_t^+$ be given. Suppose there are tree decompositions $\beta\colon V(T) \to 2^{V(F)}$ and $\beta'\colon V(T') \to 2^{V(F')}$ as in the inductive hypothesis. Let $x \in V(T)$ and $x' \in V(T')$ be such that the labelled vertices of $\boldsymbol{F}$ and $\boldsymbol{F}'$ lie in $\beta(x)$ and $\beta'(x')$ respectively. Let S be the tree obtained by taking the disjoint union of T, T', and a fresh vertex y, and connecting x and x' to y.

For the graph $\boldsymbol{F} \cdot \boldsymbol{F}'$, an S-decomposition is given by the function

$$\gamma\colon z \mapsto \begin{cases} \beta(z), & \text{if } z \in V(T), \\ \beta'(z), & \text{if } z \in V(T'), \\ \{\boldsymbol{u}_1, \dots, \boldsymbol{u}_t, \boldsymbol{v}'_1, \dots, \boldsymbol{v}'_t, \boldsymbol{v}_1, \dots, \boldsymbol{v}_t\}, & \text{if } z = y. \end{cases}$$

where one may note that $\boldsymbol{v}_i = \boldsymbol{u}'_i$ for every $i \in [t]$ in $\boldsymbol{F} \cdot \boldsymbol{F}'$. It is easy to check that Definition 6 is satisfied. The decomposition is of width $3t-1$.

For the graph $\boldsymbol{F} \odot \boldsymbol{F}'$, an S-decomposition is given by the function

$$\gamma\colon z \mapsto \begin{cases} \beta(z), & \text{if } z \in V(T), \\ \beta'(z), & \text{if } z \in V(T'), \\ \{\boldsymbol{u}_1, \dots, \boldsymbol{u}_t, \boldsymbol{v}_1, \dots, \boldsymbol{v}_t\}, & \text{if } z = y. \end{cases}$$

where one may note that $\boldsymbol{u}_i = \boldsymbol{u}'_i$ and $\boldsymbol{v}_i = \boldsymbol{v}'_i$ for every $i \in [t]$ in $\boldsymbol{F} \odot \boldsymbol{F}'$. Again, it is easy to check that Definition 6 is satisfied. The decomposition is of width at most $3t-1$. ◀

Lemma 28 in conjunction with Theorems 9 and 10 implies Theorems 2 and 3. As a corollary, this yields the upper bound in Theorem 1. Indeed, by Theorem 8, $G \simeq_t^{\mathrm{SA}} H$ if and only if G and H are homomorphism indistinguishable over the class of graphs of treewidth at most $t-1$. Hence, if $G \simeq_{3t}^{\mathrm{SA}} H$ then $G \simeq_t^{\mathrm{L}^+} H$ and in particular $G \simeq_t^{\mathrm{L}} H$.

It remains to show the lower bound asserted by Theorem 1, i.e. that $3t$ cannot be replaced by $3t-1$ for no $t \geq 1$. To that end, first observe that Observation 23 implies that the bound in Lemma 28 is tight. However, this syntactic property of the graph class $\mathcal{L}_t$ does not suffice to derive the aforementioned semantic property of $\simeq_t^{\mathrm{SA}}$ and $\simeq_t^{\mathrm{L}}$. In fact, it could well be that for all graphs G and H if G and H are homomorphism indistinguishable over the graphs of treewidth at most $3t-2$ also $\hom(K_{3t}, G) = \hom(K_{3t}, H)$ despite that $\operatorname{tw} K_{3t} > 3t-2$. That this does not hold is implied by a conjecture of the first author [27] which asserts that every minor-closed graph class $\mathcal{F}$ which is closed under taking disjoint unions (*union-closed*) is *homomorphism distinguishing closed*, i.e. for all $F \notin \mathcal{F}$ there exist graphs G and H such that $G \equiv_{\mathcal{F}} H$ but $\hom(F, G) \neq \hom(F, H)$. Although being generally open, this conjecture was proven by Neuen [24] for the class of graphs of treewidth at most t for every t. Theorem 29 implies the last assertion of Theorem 1.

▶ **Theorem 29.** *For every $t \geq 1$, there exist graphs G and H such that $G \simeq^{\mathrm{SA}}_{3t-1} H$ and $G \not\simeq^{\mathrm{L}}_{t} H$.*

Proof. Towards a contraction, suppose that $G \simeq^{\mathrm{SA}}_{3t-1} H \implies G \simeq^{\mathrm{L}}_{t} H$ for all graphs G and H. By Theorem 8, $G \simeq^{\mathrm{SA}}_{3t-1} H$ if and only if G and H are homomorphism indistinguishable over the class $\mathcal{TW}_{3t-2}$ of graphs of treewidth at most $3t-2$. By Observation 23 and Theorem 9, if $G \equiv_{\mathcal{TW}_{3t-2}} H$ then $G \equiv_{\mathcal{L}_t} H$ and in particular $\hom(K_{3t}, G) = \hom(K_{3t}, H)$. As shown in [24], the class $\mathcal{TW}_{3t-2}$ is homomorphism distinguishing closed. As $\operatorname{tw} K_{3t} = 3t-1$, it follows that there exist graphs G and H with $G \simeq^{\mathrm{SA}}_{3t-1} H$ and $\hom(K_{3t}, G) \neq \hom(K_{3t}, H)$. In particular, $G \not\simeq^{\mathrm{L}}_{t} H$ by Theorem 9. ◀

It is worth noting that the classes of unlabelled graphs underlying the elements of $\mathcal{L}_t$ and $\mathcal{L}_t^+$ are themselves minor-closed and union-closed. Hence, they are subject to the aforementioned conjecture. Furthermore, by the Robertson–Seymour Theorem and [29], membership in $\mathcal{L}_t$ and $\mathcal{L}_t^+$ can be tested in polynomial time for every fixed $t \geq 1$. The proof of Lemma 30 is deferred to the full version [28].

▶ **Lemma 30.** *Let $t \geq 1$. The class of graphs underlying the elements of $\mathcal{L}_t$ and the class of graphs underlying the elements of $\mathcal{L}_t^+$ are minor-closed and union-closed.*

The remainder of this section is dedicated to some further relations between the classes of graphs of bounded treewidth or pathwidth, $\mathcal{L}_t$, and $\mathcal{L}_t^+$. Note that these facts give independent proofs for the correspondence between the feasibility of the level-t Sherali–Adams relaxation (without non-negativity constraints), which corresponds to homomorphism indistinguishability over graphs of treewidth (pathwidth) at most $t-1$, as proven by [9, 15], and the feasibility of the level-t Lasserre relaxation with and without non-negativity constraints.

First of all, it is easy to see that dropping the semidefiniteness constraint Equation (4) of the level-t Lasserre system of equations turns this system essentially into the level-$2t$ Sherali–Adams system of equations without non-negativity constraints, e.g. as defined in [14, Appendix D.1]. This is paralleled by Lemma 31.

▶ **Lemma 31.** *Let $t \geq 1$. For every graph F with $\operatorname{pw} F \leq 2t-1$, there is a graph $\boldsymbol{F} \in \mathcal{L}_t$ whose underlying unlabelled graph is isomorphic to F.*

Furthermore, one may drop Equation (4) from the level-t Lasserre system of equations with non-negativity constraints to obtain the level-$2t$ Sherali–Adams system of equations in its original form, i.e. with non-negativity constraints. This is paralleled by Lemma 32.

▶ **Lemma 32.** *Let $t \geq 1$. For every graph F with $\operatorname{tw} F \leq 2t-1$, there is a graph $\boldsymbol{F} \in \mathcal{L}_t^+$ whose underlying unlabelled graph is isomorphic to F.*

Since the diagonal entries of a positive semidefinite matrix are necessarily non-negative, Equation (4) implies that any solution (y_I) to the level-t Lasserre system of equations is such that $y_I \geq 0$ for all $I \in \binom{V(G)\times V(H)}{\leq t}$. Hence, such a solution is a solution to the level-t Sherali–Adams system of equations as well. This is paralleled by Lemma 33.

▶ **Lemma 33.** *Let $t \geq 1$. For every graph F with $\operatorname{tw} F \leq t-1$, there is a graph $\boldsymbol{F} \in \mathcal{L}_t$ whose underlying unlabelled graph is isomorphic to F.*

The proofs of Lemmas 31–33 are all by inductively constructing an element of $\mathcal{L}_t^+$ using a tree decomposition of the given graph. They are deferred to the full version [28].

4.2 The Classes $\mathcal{L}_1$ and $\mathcal{L}_1^+$

The classes $\mathcal{L}_1$ and $\mathcal{L}_1^+$ can be identified as the class of outerplanar graphs and as the class of graphs of treewidth at most two, respectively. This yields Theorem 5. Proofs are deferred to the full version [28].

▶ **Proposition 34.** *The class of unlabelled graphs underlying an element of $\mathcal{L}_1^+$ coincides with the class of graphs of treewidth at most two.*

A graph F is *outerplanar* if it does not have K_4 or $K_{2,3}$ as a minor. Equivalent, it is outerplanar if it has a planar drawing such that all its vertices lie on the same face [34].

▶ **Proposition 35.** *The class of unlabelled graphs underlying an element of $\mathcal{L}_1$ coincides with the class of outerplanar graphs.*

As a corollary of Proposition 35, we observe the following:

▶ **Corollary 36.** *If $G \equiv_{\mathcal{L}_1} H$ then G is connected iff H is connected.*

5 Deciding Exact Feasibility of the Lasserre Relaxation with Non-Negativity Constraints in Polynomial Time

This section is dedicated to proving Theorem 4. To that end, it is argued that $\simeq_t^{\mathrm{L}^+}$ has equivalent characterisations in terms of logical equivalence and a colouring algorithm akin to the k-dimensional Weisfeiler–Leman algorithm [36]. This algorithm has polynomial running time. It is defined as follows:

▶ **Definition 37.** *Let $t \geq 1$, define for a graph G, $i \geq 1$, and $\boldsymbol{r}, \boldsymbol{s} \in V(G)^t$*

$$\begin{aligned}
\mathsf{mwl}_G^0(\boldsymbol{rs}) &\coloneqq \mathrm{rel}_G(\boldsymbol{rs}),\\
\mathsf{mwl}_G^{i-1/2}(\boldsymbol{rs}) &\coloneqq \left(\mathsf{mwl}_G^{i-1}(\sigma(\boldsymbol{rs})) \;\middle|\; \sigma \in \mathfrak{S}_{2t}\right),\\
\mathsf{mwl}_G^{i}(\boldsymbol{rs}) &\coloneqq \left(\mathsf{mwl}_G^{i-1/2}(\boldsymbol{rs}), \left\{\!\!\left\{ \left(\mathsf{mwl}_G^{i-1/2}(\boldsymbol{rt}), \mathsf{mwl}_G^{i-1/2}(\boldsymbol{ts})\right) \;\middle|\; \boldsymbol{t} \in V(G)^t \right\}\!\!\right\}\right).
\end{aligned}$$

The mwl_G^i for $i \in \mathbb{N}$ define increasingly fine colourings of $V(G)^{2t}$. Let mwl_G^∞ denote the finest such colouring. Two graphs G and H are not distinguished by the t-dimensional mwl algorithm *if the multisets*

$$\left\{\!\!\left\{ \mathsf{mwl}_G^\infty(\boldsymbol{rs}) \;\middle|\; \boldsymbol{r}, \boldsymbol{s} \in V(G)^t \right\}\!\!\right\} \quad \text{and} \quad \left\{\!\!\left\{ \mathsf{mwl}_H^\infty(\boldsymbol{uv}) \;\middle|\; \boldsymbol{u}, \boldsymbol{v} \in V(H)^t \right\}\!\!\right\}$$

are the same.

Since the finest colouring mwl_G^∞ is reached in $\leq n^{2t} - 1$ iterations for graphs on n vertices, for fixed t, it can be tested in polynomial time whether two graphs are not distinguished by the t-dimensional mwl algorithm. We are about to show that the latter happens if and only if the level-t Lasserre relaxation with non-negative constraints is feasible. As a by-product, we obtain a logical characterisation for this equivalence relation.

▶ **Definition 38.** *For $t \geq 1$, let M^t denote the fragment of first-order logic with counting quantifiers and at most $3t$ variables comprising the following expressions:*

- *$x_i = x_j$ and Ex_ix_j for all $i, j \in [3t]$,*
- *if $\varphi, \psi \in \mathsf{M}^t$ then $\neg\varphi$, $\varphi \wedge \psi$, and $\varphi \vee \psi$ are in M^t,*
- *if $\varphi, \psi \in \mathsf{M}^t$ and $n \in \mathbb{N}$ then $\exists^{\geq n}\boldsymbol{y}.\ \varphi(\boldsymbol{x}, \boldsymbol{y}) \wedge \psi(\boldsymbol{y}, \boldsymbol{z})$ is in M^t. Here, the bold face letters $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$ denote t-tuples of distinct variables.*

The semantic of the quantifier $\exists^{\geq n}\boldsymbol{y}.\ \varphi(\boldsymbol{y})$ is that there exist at least n many t-tuples of vertices from the graph over which the formula is evaluated which satisfy φ. The following Theorem 39 may be thought of as a analogue of Theorem 8 for $\mathcal{L}_t^+$.

▶ **Theorem 39.** *Let $t \geq 1$. For graphs G and H, the following are equivalent:*

1. *G and H are not distinguished by the t-dimensional* mwl *algorithm,*
2. *G and H are homomorphism indistinguishable over $\mathcal{L}_t^+$,*
3. *G and H satisfy the same M^t-sentences.*

The proof of Theorem 39 is deferred to the full version [28]. It is conceptually similar to arguments of [6, 10, 15]. As mentioned above, Theorem 39 implies Theorem 4.

6 Conclusion

We have established a characterisation of the feasibility of the level-t Lasserre relaxation with and without non-negativity constraints of the integer program $\mathrm{ISO}(G, H)$ for graph isomorphism in terms of homomorphism indistinguishability over the graph classes $\mathcal{L}_t$ and $\mathcal{L}_t^+$. By analysing the treewidth of the graphs $\mathcal{L}_t$ and $\mathcal{L}_t^+$ and invoking results from the theory of homomorphism indistinguishability, we have determined the precise number of Sherali–Adams levels necessary such that their feasibility guarantees the feasibility of the level-t Lasserre relaxation. This concludes a line of research brought forward in [4]. For feasibility of the level-t Lasserre relaxation with non-negativity constraints, we have given, besides linear algebraic reformulations generalising the adjacency algebra of a graph, a polynomial time algorithm deciding this property.

Missing in Theorem 1 is a tight lower bound on the number of Lasserre levels necessary to ensure feasibility of a given Sherali–Adams level:

▶ **Question 40.** *Do there exist for every $t \geq 3$ graphs G and H such that $G \simeq_{t-1}^{\mathrm{L}} H$ and $G \not\simeq_t^{\mathrm{SA}} H$?*

Following the path taken in this paper, this question could potentially be resolved in two steps: Firstly, one would need to prove the graph theoretic assertion that the class $\mathcal{L}_t$ does not contain $\mathcal{TW}_t$ for all $t \geq 2$. Secondly, one would need to show that $\mathcal{L}_t$ is homomorphism distinguishing closed or at least that the homomorphism distinguishing closure [27] of $\mathcal{L}_t$ does not contain $\mathcal{TW}_t$ for all $t \geq 2$. Given the means currently available for proving such a statement [27, 24], this would involve giving game characterisations for $\mathcal{L}_t$ (mimicking the robber-cops game for $\mathcal{TW}_t$) and for $\equiv_{\mathcal{L}_t}$ (similar to the bijective $(t+1)$-pebble game for $\mathcal{TW}_t$). For the former, finding analogies to the notions of brambles or heavens seems necessary [31].

Another interesting extension of our work might be an efficient algorithm for computing an explicit partial t-equivalence between two graphs, cf. Definitions 18 and 20, or deciding that no such map exists. This would yield an efficient algorithm for deciding the exact feasibility of the Lasserre semidefinite program without non-negativity constraints, cf. [4].

References

1 Samson Abramsky, Tomáš Jakl, and Thomas Paine. Discrete density comonads and graph parameters. In Helle Hvid Hansen and Fabio Zanasi, editors, *Coalgebraic Methods in Computer Science – 16th IFIP WG 1.3 International Workshop, CMCS 2022, Colocated with ETAPS 2022, Munich, Germany, April 2-3, 2022, Proceedings*, volume 13225 of *Lecture Notes in Computer Science*, pages 23–44. Springer, 2022. doi:10.1007/978-3-031-10736-8_2.

2 Albert Atserias, Phokion G. Kolaitis, and Wei-Lin Wu. On the expressive power of homomorphism counts. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 – July 2, 2021*, pages 1–13. IEEE, 2021. doi:10.1109/LICS52264.2021.9470543.

3 Albert Atserias and Elitza Maneva. Sherali–Adams Relaxations and Indistinguishability in Counting Logics. *SIAM Journal on Computing*, 42(1):112–137, 2013. doi:10.1137/120867834.

4 Albert Atserias and Joanna Ochremiak. Definable Ellipsoid Method, Sums-of-Squares Proofs, and the Isomorphism Problem. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 66–75. ACM, 2018. doi:10.1145/3209108.3209186.

5 Christoph Berkholz and Martin Grohe. Limitations of Algebraic Approaches to Graph Isomorphism Testing. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming*, pages 155–166. Springer Berlin Heidelberg, 2015. doi:10.1007/978-3-662-47672-7_13.

6 Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992. doi:10.1007/BF01305232.

7 Gang Chen and Ilia Ponomarenko. *Lectures on Coherent Configurations*. Central China Normal University Press, Wuhan, 2018. URL: https://www.pdmi.ras.ru/~inp/ccNOTES.pdf.

8 Anuj Dawar, Tomáš Jakl, and Luca Reggio. Lovász-Type Theorems and Game Comonads. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 – July 2, 2021*, pages 1–13. IEEE, 2021. doi:10.1109/LICS52264.2021.9470609.

9 Holger Dell, Martin Grohe, and Gaurav Rattan. Lovász Meets Weisfeiler and Leman. *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, pages 40:1–40:14, 2018. doi:10.4230/LIPICS.ICALP.2018.40.

10 Zdeněk Dvořák. On recognizing graphs by numbers of homomorphisms. *Journal of Graph Theory*, 64(4):330–342, August 2010. doi:10.1002/jgt.20461.

11 Martin Grohe. Counting Bounded Tree Depth Homomorphisms. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '20, pages 507–520, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3373718.3394739.

12 Martin Grohe. word2vec, node2vec, graph2vec, X2vec: Towards a Theory of Vector Embeddings of Structured Data. In Dan Suciu, Yufei Tao, and Zhewei Wei, editors, *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2020, Portland, OR, USA, June 14-19, 2020*, pages 1–16. ACM, 2020. doi:10.1145/3375395.3387641.

13 Martin Grohe and Martin Otto. Pebble Games and Linear Equations. *The Journal of Symbolic Logic*, 80(3):797–844, 2015. doi:10.1017/jsl.2015.28.

14 Martin Grohe, Gaurav Rattan, and Tim Seppelt. Homomorphism Tensors and Linear Equations, 2021. doi:10.48550/arXiv.2111.11313.

15 Martin Grohe, Gaurav Rattan, and Tim Seppelt. Homomorphism Tensors and Linear Equations. In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 70:1–70:20, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2022.70.

16 Martin Grotschel. *Geometric Algorithms and Combinatorial Optimization*. Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-97881-4.

17 Jean B. Lasserre. Global Optimization with Polynomials and the Problem of Moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001. doi:10.1137/S1052623400366802.

18 Monique Laurent. A Comparison of the Sherali–Adams, Lovász–Schrijver, and Lasserre Relaxations for 0-1 Programming. *Mathematics of Operations Research*, 28(3):470–496, 2003. URL: http://www.jstor.org/stable/4126981.

19 László Lovász. Operations with structures. *Acta Mathematica Academiae Scientiarum Hungarica*, 18(3):321–328, September 1967. doi:10.1007/BF02280291.

20 Laura Mančinska and David E. Roberson. Quantum isomorphism is equivalent to equality of homomorphism counts from planar graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 661–672, 2020. doi:10.1109/FOCS46700.2020.00067.

21 Laura Mančinska, David E. Roberson, Robert Samal, Simone Severini, and Antonios Varvitsiotis. Relaxations of Graph Isomorphism. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 76:1–76:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2017.76.

22 Laura Mančinska, David E. Roberson, and Antonios Varvitsiotis. Graph isomorphism: Physical resources, optimization models, and algebraic characterizations, 2020-04-22. doi:10.48550/arXiv.2004.10893.

23 Yoàv Montacute and Nihil Shah. The pebble-relation comonad in finite model theory. In Christel Baier and Dana Fisman, editors, *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 – 5, 2022*, pages 13:1–13:11. ACM, 2022. doi:10.1145/3531130.3533335.

24 Daniel Neuen. Homomorphism-Distinguishing Closedness for Graphs of Bounded Tree-Width, April 2023. doi:10.48550/arXiv.2304.07011.

25 Ryan O'Donnell, John Wright, Chenggang Wu, and Yuan Zhou. Hardness of Robust Graph Isomorphism, Lasserre Gaps, and Asymmetry of Random Graphs. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1659–1677. SIAM, 2014. doi:10.1137/1.9781611973402.120.

26 Gaurav Rattan and Tim Seppelt. Weisfeiler–Leman and Graph Spectra. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2268–2285, 2023. doi:10.1137/1.9781611977554.ch87.

27 David E. Roberson. Oddomorphisms and homomorphism indistinguishability over graphs of bounded degree, June 2022. doi:10.48550/arXiv.2206.10321.

28 David E. Roberson and Tim Seppelt. Lasserre Hierarchy for Graph Isomorphism and Homomorphism Indistinguishability, 2023. doi:10.48550/ARXIV.2302.10538.

29 N. Robertson and P.D. Seymour. Graph Minors XIII. The Disjoint Paths Problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

30 Tim Seppelt. Logical Equivalences, Homomorphism Indistinguishability, and Forbidden Minors, February 2023. doi:10.48550/ARXIV.2302.11290.

31 P.D. Seymour and R. Thomas. Graph Searching and a Min-Max Theorem for Tree-Width. *Journal of Combinatorial Theory, Series B*, 58(1):22–33, 1993. doi:10.1006/jctb.1993.1027.

32 Hanif D. Sherali and Warren P. Adams. A Hierarchy of Relaxations between the Continuous and Convex Hull Representations for Zero-One Programming Problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990. doi:10.1137/0403036.

33 Aaron Snook, Grant Schoenebeck, and Paolo Codenotti. Graph Isomorphism and the Lasserre Hierarchy, 2014. doi:10.48550/arXiv.1401.0758.

34 Maciej M. Sysło. Characterizations of outerplanar graphs. *Discrete Mathematics*, 26(1):47–53, 1979. doi:10.1016/0012-365X(79)90060-8.

35 Edwin R. van Dam and Willem H. Haemers. Which graphs are determined by their spectrum? *Linear Algebra and its Applications*, 373:241–272, 2003. doi:10.1016/S0024-3795(03)00483-X.

36 Boris Weisfeiler. *On Construction and Identification of Graphs*, volume 558 of *Lecture Notes in Mathematics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1976. doi:10.1007/BFb0089374.

Average-Case to (Shifted) Worst-Case Reduction for the Trace Reconstruction Problem

Ittai Rubinstein
Qedma Quantum Computing, Tel Aviv, Israel

Abstract

In the *trace reconstruction problem*, one is given many outputs (called *traces*) of a noise channel applied to the same input message $\mathbf{x}$, and is asked to recover the input message. Common noise channels studied in the context of trace reconstruction include the *deletion channel* which deletes each bit w.p. δ, the *insertion channel* which inserts a G_j i.i.d. uniformly distributed bits before each bit of the input message (where G_j is i.i.d. geometrically distributed with parameter σ) and the symmetry channel which flips each bit of the input message i.i.d. w.p. γ.

De et al. and Nazarov and Peres [12, 20] showed that any string $\mathbf{x}$ can be reconstructed from $\exp(O(n^{1/3}))$ traces. Holden et al. [13] adapted the techniques used to prove this upper bound, to construct an algorithm for average-case trace reconstruction from the insertion-deletion channel with a sample complexity of $\exp(O(\log^{1/3} n))$. However, it is not clear how to apply their techniques more generally and in particular for the recent worst-case upper bound of $\exp(\widetilde{O}(n^{1/5}))$ shown by Chase [7] for the deletion channel.

We prove a general reduction from the average-case to smaller instances of a problem similar to worst-case and extend Chase's upper-bound to this problem and to symmetry and insertion channels as well. Using this reduction and generalization of Chase's bound, we introduce an algorithm for the average-case trace reconstruction from the symmetry-insertion-deletion channel with a sample complexity of $\exp(\widetilde{O}(\log^{1/5} n))$.

2012 ACM Subject Classification Theory of computation → Sample complexity and generalization bounds

Keywords and phrases Trace Reconstruction, Synchronization Channels, Computational Learning Theory, Computational Biology

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.102

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2207.11489`

Acknowledgements We would like to thank Zachary Chase, Roni Con and Aviad Rubinstein for their helpful comments on previous versions of this paper. We would also like to thank Nina Holden, Robin Pemantle, Yuval Peres and Alex Zhai for their help in understanding their paper.

1 Introduction

The *symmetry-insertion-deletion* (SID) channel with bit-flip probability $\gamma \in [0, 1/2)$, insertion probability $\sigma \in [0, 1)$ and deletion probability $\delta \in [0, 1)$, takes as input a binary string $\mathbf{x} \in \{0,1\}^n$. For each j, the jth bit of $\mathbf{x}$ is flipped w.p. γ (we will sometimes think of this portion of the channel as replacing the jth bit of $\mathbf{x}$ with a random bit w.p. 2γ). Then G_j random uniform and independent bits are inserted before the jth bit of $\mathbf{x}$, where the random variables $G_j \geq 0$ are i.i.d. geometrically distributed with parameter σ. Then, each bit of the message is deleted independently with probability δ. The output string $\widetilde{\mathbf{x}}$ is called a *trace*[1].

[1] The trace reconstruction problem was originally defined with only the deletion channel [2] (i.e. with γ and σ fixed to 0). The more general SID channels were first considered in the "open questions" of [18] and were further researched by Andoni et al. [1] and by De et al. [12].

The *trace reconstruction problem* asks the following question: how many traces are necessary to reconstruct an unknown string $\mathbf{x}$?

The main motivation for studying trace reconstruction comes from computational biology, where one often tries to align several DNA sequences to a common ancestor, and it has been extensively researched since the early 2000's [2].

Perhaps the most natural and well-researched version of the trace reconstruction problem, is the *worst-case*, where the input string $\mathbf{x}$ is adversarially chosen. Holenstein et al. [15] established an upper bound of $\exp(\widetilde{O}(n^{1/2}))$ on its sample complexity. This was improved by Nazarov and Peres [20], and De, O'Donnell and Servedio [12] who simultaneously proved upper and lower bounds of $\exp(O(n^{1/3}))$ on the sample complexity of "mean-based" trace reconstruction techniques. Recently, Chase [7] improved on these methods by proving that a "non-linear" method can be used to solve the worst-case deletion-channel trace reconstruction problem with a far lower sample complexity of $\exp(\widetilde{O}(n^{1/5}))$.

De et al. and Nazarov and Peres's results were highly influential and mean-based separators are used as a central component in the analysis of many other versions of the trace reconstruction problem [5, 8, 10, 13, 16, 21]. However, so far, Chase's techniques have not been extended beyond worst-case trace reconstruction from a deletion channel. In particular, we note the coded [5, 10] and the average-case [13, 21] trace reconstruction problems.

The *average-case* trace reconstruction problem was introduced by Batu et al. [2]. In this problem, the input string $\mathbf{x}$ is chosen uniformly at random from $\{0,1\}^n$, and the reconstruction only needs to succeed w.h.p. over the choice of $\mathbf{x}$. McGregor et al. [17] showed that if $H(n)$ traces are necessary for the worst-case trace reconstruction, then at least $H(\log n)$ are needed for the average-case (and under some conditions $H(\log n)\log n$).

Peres and Zhai [21] adapted mean-based separators to the average-case, constructing an efficient algorithm for the average-case deletion-channel trace reconstruction with $\exp(O(\log^{1/2} n))$ samples and low deletion probability ($\delta \leq 1/2$). This was further improved by Holden et al. [14] who reduced the sample complexity to $\exp(O(\log^{1/3} n))$ and generalized the algorithm to work for all insertion-deletion channels.

Motivated by the question of DNA storage, Cheraghchi et al. [10] introduced the *coded* trace reconstruction problem, where one is asked to construct a code $C \subset \{0,1\}^n$ s.t. any codeword $\mathbf{x} \in C$ can be reconstructed w.h.p. given as few independent traces $\widetilde{\mathbf{x}}$ as possible. Brakensiek et al. [5] proved that this problem is essentially equivalent to the average-case trace reconstruction problem.

1.1 Our Contributions

Let $n \in \mathbb{N}$ be arbitrarily large, and let $\gamma \in [0, 1/2)$, $\sigma \in [0, 1)$ and $\delta \in [0, 1)$ be fixed bit-flip, insertion and deletion probabilities, and let $\mathcal{C}$ be the SID channel with these parameters. Let C be a sufficiently large constant[2].

We introduce a new version of the trace reconstruction problem, called the *shifted* trace reconstruction problem (see Definition 1). In this problem, one is asked to reconstruct the first n bits of a much longer string $\mathbf{x}$ from its traces. Moreover, the error channel is also allowed to "shift" the traces by some unknown distance $s \in \mathbb{N}$ (selected i.i.d. from a known and bounded distribution S for each trace).

The shifted trace reconstruction problem often appears as a component in the analysis of other versions of the trace reconstruction problem [14, 8], but so far it has not been formally defined. Moreover, it could be of interest in its own right. Similar to the approximate trace reconstruction problem introduced by Davies et al. [11] and the average-case approximate

[2] C may depend on γ, σ, δ, but not on n.

trace reconstruction problem by Chase and Peres [8] which model the question of using a smaller number of traces to reconstruct some information about the input string $\mathbf{x}$, the shifted trace reconstruction problem asks a similar question, but with the goal of reconstructing the prefix of a long string.

▶ **Definition 1** (Shifted Trace Reconstruction Problem). *In a* shifted trace reconstruction problem *of size $n \in \mathbb{N}$, with* shift inaccuracy $\Delta_S(n)$*, one must reconstruct the $n+1$th bit of any string $\boldsymbol{x} \in \{0,1\}^{\mathbb{N}}$ of length at most 2^n given the value of its first n bits $\boldsymbol{x}_{:n}$, and $H(n) = \exp(h(n))$ i.i.d. traces $\widetilde{\boldsymbol{x}}$ produced by the following process.*

A random shift s is applied to the input string $\overline{\boldsymbol{x}} \stackrel{\text{def}}{=} \boldsymbol{x}_{s:}$, where $s \leftarrow S$ is drawn from a known shift distribution *S, with bounded support $\mathrm{Supp}(S) \subseteq [a, a+\Delta_S]$ for some $0 \le a \le n - \Delta_S$. Then the noise channel $\mathcal{C}$ is applied to the shifted string $\overline{\boldsymbol{x}}$ to obtain a trace $\widetilde{\boldsymbol{x}}$.*

The shifted trace reconstruction problem is clearly at least as hard as the worst-case trace reconstruction problem, but the differences between the two do not seem to affect the leading reconstruction techniques. In particular, we extend Chase's analysis to SID channels and to the shifted trace reconstruction problem, proving that $\exp(\widetilde{O}(n^{1/5}))$ samples suffice for the (shifted) worst-case trace reconstruction problem from an SID channel (Theorem 2).

▶ **Theorem 2.** *For any SID channel $\mathcal{C}$ as defined above, and for any constant $C > 0$, there exists an algorithm A which solves the shifted trace reconstruction problem of size n with shift inaccuracy $\Delta_S(n) = Ch(n)$ and sample complexity $H(n) = \exp(h(n)) = \exp(O(n^{1/5}\log^7 n))$.*

Furthermore, when the deletion probability is sufficiently low ($\delta < 1/2$), the algorithm A runs in time $\exp(O(n^{4/5}\log n))$ and if $q \ge 1/2$, then A runs in time $\exp(O(n))$.

▶ Remark 3. Note that while De et al.'s reconstruction algorithm has a time complexity polynomial in its sample complexity, Chase only proves an upper bound on the sample complexity. A naïve adaptation of Chase's upper bound to an algorithm would yield a time complexity of $\exp(\Theta(n))$.

Holden et al.'s average-case trace reconstruction algorithm works by partially aligning each trace and then using an oracle that solves a version of the shifted trace reconstruction problem to reconstruct each bit of the input message $\mathbf{x}$. However, much of their analysis is specific to their sample complexity of $\exp(\log^{1/3}(n))$.

We transform Holden et al.'s construction into a general reduction from an average-case trace reconstruction of length n to linearly many instances of shifted trace reconstruction problems of length $O(\log(n))$ (Theorem 4). Moreover, our reduction applies to any SID channel.

▶ **Theorem 4** (Average to Shifted Reduction). *Let A be an oracle that solves the shifted trace reconstruction problem with sample complexity $H(n) = \exp(h(n))$ (for $\log(n) \le h(n) \le \sqrt{n}$), shift inaccuracy $\Delta_S = Ch(n)$, and failure probability $< \exp(-n)$.*

Then there exists an algorithm A' that solves the average-case trace reconstruction problem with success probability $1 - o_n(1)$, sample complexity of $\exp(Ch(C\log n))$, and time complexity $t(n) = n^{1+o(1)}$, given n calls to the oracle A.

▶ Remark 5. Note that the assumption that $\log(n) \le h(n) \le \sqrt{n}$ is not very restrictive, since we show an upper bound of $h(n) \le \widetilde{O}(n^{1/5})$ and the lower bound by Chase [6] implies that

$$h(n) \ge \frac{3}{2}\log n\ .$$

It is interesting to compare Theorem 4 to [17, Lemma 10]. Theorem 4 shows that if $H(n)$ traces suffice for shifted trace reconstruction then $\text{poly}(H(\log n))$ traces suffice for average-case trace reconstruction. Compare this to McGregor et al. [17, Lemma 10] who prove that if $H(n)$ traces are required for worst-case trace reconstruction, then $H(\log n)$ traces are required for average-case trace reconstruction. This means that up to the differences between shifted and worst-case trace reconstruction, Theorem 4 is essentially tight.

We also note Brakensiek et al. [5], who proved reductions between the coded and the average-case trace reconstruction problems. When combined with Theorem 4 and [17, Lemma 10], a computational class of trace reconstruction problems begins to emerge (see Figure 1).

An important question to consider in future trace reconstruction research is whether other versions of the trace reconstruction problem can be reduced to one of these classes. For instance, consider the approximate average-case trace reconstruction problem. The best known approximate average-case trace reconstruction technique at the time of writing this paper is due to Chase and Peres [8], whose approach is based on performing calls to a "shifted" average-case trace reconstruction oracle, making it a good candidate for a more general reduction.

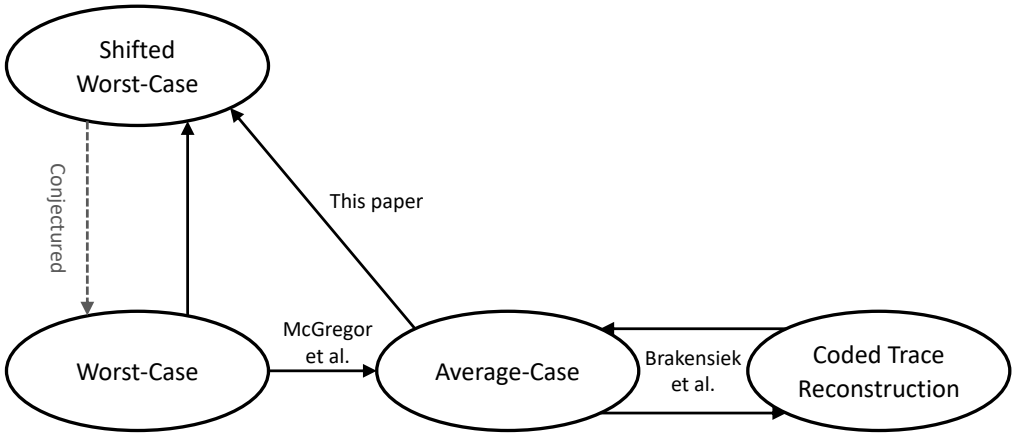


Figure 1 A diagram of several known reductions between trace reconstruction problems. McGregor et al. [17] proved that any solution to the average-case trace reconstruction problem implies a solution to a smaller instance of the worst-case trace reconstruction problem. Brakensiek et al. [5] proved reductions from coded trace reconstruction to average-case trace reconstruction and vice versa. We introduce the shifted trace reconstruction and prove a reduction from the average-case to it. We also show that the current best-known solutions for worst-case trace reconstruction can be extended to shifted trace reconstruction and conjecture that the two are equivalent.

Finally, Theorems 2 and 4 give us an algorithm for the average-case trace reconstruction from SID channels with only $\exp(\widetilde{O}(\log^{1/5} n))$ traces.

▶ **Theorem 6** (Main Result). *For any SID channel $\mathcal{C}$ as defined above, if $\boldsymbol{x} \in \{0,1\}^n$ is a bit-string where the bits are chosen uniformly and independently at random, then we can reconstruct $\boldsymbol{x}$ with probability $1 - o_n(1)$ using $\exp(C \log^{1/5} n \log^7 \log n)$ traces. Moreover, when the deletion probability is sufficiently low ($\delta < 1/2$), this can be done in $n^{1+o(1)}$ time and otherwise, this can be done in polynomial time.*

1.2 An Overview of Previous Constructions

Many trace reconstruction techniques follow a similar high-level pattern [7, 12, 14, 20, 21]. First, a combinatorial analysis allows us to equate some property of the original message $\mathbf{x}$ to a polynomial whose coefficients depend on the traces $\widetilde{\mathbf{x}}$. This polynomial is then analysed on a small sub-arc of the complex disk $\mathbb{D}$ using Borwein and Erdélyi's seminal research on Littlewood polynomials [3] or an extension of it [7], yielding a statistical test on the traces which can be used to reconstruct some property of the original message $\mathbf{x}$.

Our analysis will also follow a similar pattern, and many of its steps will be based on combinations and extensions of components used to prove previous results, so we begin with a short overview of these techniques.

For any $\mathbf{w} \in \{0,1\}^{\mathbb{N}}$, let $I_{\mathbf{w}} : \{0,1\}^{\mathbb{N}} \to \mathbb{R}$ be the function that maps a string $\mathbf{x}$ to 1 if it begins with the prefix $\mathbf{w}$ and otherwise maps it to 0. For any function $f : \{0,1\}^{\mathbb{N}} \to \mathbb{R}$, we define its indicator polynomial on $\mathbf{x}$ to be the polynomial $p_{f,\mathbf{x}}(z) = \sum_j f(\mathbf{x}_{j:})z^j \in \mathbb{R}[z]$.

De et al. and Nazarov and Peres [12, 20] show that the polynomial $p_{\mathbf{x}} \stackrel{\text{def}}{=} p_{I_1,\mathbf{x}}$ whose jth coefficient is the jth bit of the original message $\mathbf{x}_j$ and the polynomial $p_{\widetilde{\mathbf{x}}} \stackrel{\text{def}}{=} \mathbb{E}\left[p_{I_1,\widetilde{\mathbf{x}}}\right]$ whose jth coefficient is the average over the jth bits of the traces $\mathbb{E}\left[\widetilde{\mathbf{x}}_j\right]$ are essentially equivalent up to a parameter change:

$$p_{\widetilde{\mathbf{x}}}(\phi^{-1}(z)) \approx (1-\delta)(1-2\gamma)p_{\mathbf{x}}(z) \tag{1}$$

where $\phi(z) = \frac{(1-\sigma)(\delta+(1-\delta)z)}{1-\sigma z}$ is a Möbius transformation related to the channel parameters[3]. De et al. and Nazarov and Peres then consider points of the form $z = \exp(i\alpha)$ for small $-n^{-1/3} < \alpha < n^{-1/3}$. $p_{\widetilde{\mathbf{x}}}(\phi^{-1}(z))$ can be approximated at such points from a bounded number of traces, because $\left|\phi^{-1}(z)\right| < 1 + O(n^{-2/3})$ and the linear transformation mapping the traces $\widetilde{\mathbf{x}}$ to

$$p_{\widetilde{\mathbf{x}}}(\phi^{-1}(z)) = \sum_{1\le j\le n} \left(\phi^{-1}(z)\right)^j \mathbb{E}\left[\widetilde{\mathbf{x}}_j\right]$$

has bounded coefficients $\left|\phi^{-1}(z)\right|^j \le \exp(O(n^{1/3}))$.

Borwein and Erdélyi [3] showed that for any polynomial $p(z)$ with $\{0, \pm 1\}$ coefficients, there exists some z in this sub-arc $\{\exp(i\alpha) \mid |\alpha| \le n^{-1/3}\}$ for which $|p(z)| \ge \exp(-O(n^{1/3}))$. In the context of trace reconstruction, we take $p(z)$ to be the difference $p_{\mathbf{x}}(z) - p_{\mathbf{y}}(z)$ where $\mathbf{x}$ and $\mathbf{y}$ are two input messages between which we want to differentiate.

This yields a method of differentiating between any two potential input strings $\mathbf{x}, \mathbf{y}$ with $\exp(O(n^{1/3}))$ traces.

Peres and Zhai [21] and Holden et al. [13] use a similar relationship between the original message and the traces, but in their construction the polynomials $p_{\mathbf{x}}$ and $p_{\widetilde{\mathbf{x}}}$ have a much higher degree because they want to reconstruct the first bits of a long string. They overcome this by extending the complex analysis to points of the form $z = \rho\exp(i\alpha)$ for a carefully chosen $\rho = 1 - o(1)$, effectively allowing them to truncate $p_{\mathbf{x}}$ and $p_{\widetilde{\mathbf{x}}}$ to a finite degree.

Holden et al. also use the fact that their input string is random to create partial alignments. The alignments are based on a Boolean test which checks whether or not a substring $\widetilde{\mathbf{w}}$ of a trace $\widetilde{\mathbf{x}}$ was the result of applying the channel to some substring $\mathbf{w}$ of the input message $\mathbf{x}$. This Boolean test is guaranteed to have a low false-positive rate and a non-negligible true-positive rate, when the input string $\mathbf{w}$ is "sufficiently random".

[3] Equation (1) is correct up to minor technical details. Lemma 8 can be used to derive an accurate version of this equation.

Holden et al. use this alignment procedure to reconstruct $\mathbf{x}$ one bit at a time. For each bit, they use this partial alignment to pin the traces to some nearby index and then use a mean-based separator to reconstruct it.

Chen et al. [9] and Narayanan and Ren [19] generalize equation (1) to relate multi-indices where some subsequence $\mathbf{w}$ appears in the input message $\mathbf{x}$ to multi-indices where the same subsequence appears in the traces, but their proof is limited to the deletion channel. As a results, Chase's analysis [7] which relies heavily on this generalized relationship, cannot be directly extended to insertion or symmetry channels.

Chase sets $\mathbf{w}$ to be an "a-periodic" string, in order to ensure that the set of indices where $\mathbf{w}$ appears as a consecutive substring in $\mathbf{x}$ is sparse. Therefore, the polynomial $p_{\mathbf{w},\mathbf{x}}(z) \stackrel{\text{def}}{=} p_{I_{\mathbf{w}},\mathbf{x}}(z)$ has sparse coefficients. Chase uses an extension of Borwein and Erdélyi's methods to prove much stronger bounds on polynomials with sparse coefficients on similar arcs of the unit disk. Balancing out the parameters yields Chase's $\exp(\widetilde{O}(n^{1/5}))$ bound on the worst-case sample complexity.

Much of our paper will be devoted to generalizing and combining the results of Holden et al. [14] and Chase [7]. For the sake of brevity, we will henceforth refer to these papers as the HPPZ and the Chase constructions respectively.

1.3 Sketch of our Proof

We extend these analyses in three key ways.

In the first and most difficult portion of the paper, we extend Chen et al. and Narayanan and Ren's [9, 19] generalization of equation (1) to SID channels. This is non-trivial, as the common method of dealing with insertions and bit-flips is to take a statistic where the unbiased insertions average out to having no effect on the output (this is usually done by looking at the difference between the traces $\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}$ of two potential input strings $\mathbf{x}, \mathbf{y}$). It is not clear how to perform a similar analysis on a multi-bit property, such as an indicator of some magic string $\mathbf{w}$ which is highly non-linear in its input.

Our main observation in dealing with this problem is that the function $\chi_{(-1,\dots,-1)}(\mathbf{x}) \stackrel{\text{def}}{=} (-1)^{\mathbf{x}_1} \cdots (-1)^{\mathbf{x}_k}$, which we call a "full" character, has the property that if any of its input bits $\mathbf{x}_1, \dots, \mathbf{x}_k$ is an inserted bit or was replaced with a random bit by the symmetry channel, then its output is unbiased and does not affect the average over traces. This allows us to prove a similar relationship between $p_{\chi_{(-1,\dots,-1)},\mathbf{x}}$ and $p_{f,\tilde{\mathbf{x}}}$.

Next, we extend this analysis to all characters $f(\mathbf{x}) = \chi_\omega(\mathbf{x}) \stackrel{\text{def}}{=} \prod_j \omega_j^{\mathbf{x}_j}$ for $\omega \in \{\pm 1\}^k$. This extension is more complex and requires several difficult technical lemmas, but it allows us to reconstruct $p_{f,\mathbf{x}}(z)$ for (almost) any function $f : \{0,1\}^k \to \mathbb{R}$ from the traces. We do this by applying the Fourier transformation on Boolean functions to f, allowing us to write $f(\mathbf{x}) = \sum_\omega \chi_\omega(\mathbf{x}) \widehat{f}(\omega)$ as a linear combination of characters, and by extension

$$p_{f,\mathbf{x}}(z) = p_{\sum_\omega \chi_\omega(\cdot)\widehat{f}(\omega),\mathbf{x}}(z) = \sum_\omega \widehat{f}(\omega) p_{\chi_\omega,\mathbf{x}}(z).$$

In the second portion of our analysis, we extend the Borwein and Erdélyi-type bounds proven by Chase [7] to deal with sparse polynomials when evaluated at points within the unit disk. This step is necessary for our extension of Chase's bounds to the shifted trace reconstruction problem.

In the third and final portion of the paper, we generalize Holden et al.'s [13] construction into a reduction from an average-case trace reconstruction problem of size n to linearly many trace reconstruction problems of size $\Theta(\log(n))$. Moreover, we extend Holden et al.'s proofs originally shown for the insertion-deletion channel to SID channels as well.

1.4 Organization of the Paper

Sections 2 and 3 contain the heart of our analysis, where we convert the shifted trace reconstruction problem into a complex analysis one (2) and use complex analysis techniques to solve it (3). We adapt Holden et al.'s techniques to prove a general reduction in Section 4, proving Theorems 4 and 6. Section 5 is reserved for a discussion of our results.

2 Conversion to Complex Analysis

Let $\mathbf{x} \in \{0,1\}^{\mathbb{N}}$ be some input string and let $\widetilde{\mathbf{x}}$ denote its trace from some shifted trace reconstruction problem. The first step of our analysis will be to relate a property of $\mathbf{x}$ to the expectation of some function applied to its traces $\widetilde{\mathbf{x}}$. By bounding this function of the traces in absolute value, we prove that this function can be approximated from a bounded number of traces. Then, in Section 3, we will show that approximating this property of the input string $\mathbf{x}$ allows us to reconstruct $\mathbf{x}$ one bit at a time.

This approach to trace reconstruction is common in recent literature. De et al., Holden et al. and Nazarov and Peres [12, 14, 20] show how "single bit statistics" of the input string $\mathbf{x}$ can be related to its traces through SID channels and shifts. Chase [7], building off of the works of Chen et al. [9] and Narayanan and Ren [19], extended this relationship to multi-bit statistics, in order to prove a stronger bound on the sample complexity of trace reconstruction.

However, Chase's analysis is limited to deletion channels and the main known tools for dealing with insertions and bit-flips are inherently limited to single-bit statistics. Our goal in this section will be to combine these approaches, allowing us to estimate multi-bit properties of the input string $\mathbf{x}$ from traces through an SID channel.

Let $1 \leq \ell \leq 2n^{1/5} + 1$ be some integer. For any function $f : \{0,1\}^{\ell} \to \mathbb{D}$ from the hypercube to the unit disk $\mathbb{D}$ (for our use-case, we will want $f = I_{\mathbf{w}}$ to be the indicator of some marker $\mathbf{w} \in \{0,1\}^{\mathbb{N}}$), we define $q_{f,\mathbf{x}}(z_0, \ldots, z_\ell)$ to be

$$q_{f,\mathbf{x}}(z_0, \ldots, z_\ell) \stackrel{\text{def}}{=} \sum_{k_0 < k_1 < \cdots < k_\ell} (-1)^{\mathbf{x}_0} z_0^{k_0} f(\mathbf{x}_{k_1}, \ldots, \mathbf{x}_{k_\ell}) \prod_{1 \leq j \leq \ell} z_j^{k_j - k_{j-1} - 1}.$$

In essence, $q_{f,\mathbf{x}}(z_0, \ldots, z_\ell)$ is a multivariate polynomial whose coefficients encode the value of f when applied to subsequences of the input string $\mathbf{x}$. Our goal will be to show that the value of this polynomial can be estimated at certain points from a bounded number of traces (see Theorem 7).

▶ **Theorem 7.** *Let z_0 be a point on the arc $\left\{(1 - n^{-4/5}\log^6 n)\exp(i\alpha) \mid \alpha \in [-n^{-2/5}, n^{-2/5}]\right\}$. If $\delta < 1/2$, let $z_1, \ldots, z_\ell = 0$. Otherwise, let $z_1 = \cdots = z_\ell$ be any point in the segment $[1 - c_1, 1 - c_2]$ for sufficiently small constants $c_1 > c_2 > 0$.*

Given $H(n) = \exp(h(n))$ traces with shift inaccuracy $\eta = O(h(n))$, we can estimate $q_{f,\boldsymbol{x}}$ at the point $(z_0, \ldots, z_\ell)$ to within an additive error of order $\pm\exp(-\Omega(h(n)))$ and with success probability $1 - \exp(-\omega(n))$, where $h(n) = n^{1/5}\log^7 n$.

We separate the proof of Theorem 7 into three parts. In the first part of the proof (Lemma 8), we will show that the statement holds for the function $f(\mathbf{x}_1, \ldots, \mathbf{x}_\ell) = \prod_{1 \leq j \leq \ell}(-1)^{\mathbf{x}_j}$. We will call this function a "full character".

We will then extend this proof to any character $f(\mathbf{x}_1, \ldots, \mathbf{x}_\ell) = \chi_\omega(\mathbf{x}_1, \ldots, \mathbf{x}_\ell) = \prod_{1 \leq j \leq \ell} \omega_j^{\mathbf{x}_j}$ (with $\omega \in \{\pm 1\}^{\ell}$) of the Fourier transformation on Boolean functions. Finally, to complete the proof, we will use the fact that any function $f : \{0,1\}^{\ell} \to \mathbb{D}$ can be written as a linear combination of characters $f = \sum_\omega \hat{f}(\omega)\chi_\omega$ (where $\hat{f}$ is the Fourier transformation of f).

▶ **Lemma 8.** *Let S be a shift distribution with bounded support (*$\mathrm{Supp}\,(S) \subseteq \{0,1,\ldots,d\}$*). Let $\boldsymbol{x} \in \{0,1\}^{\mathbb{N}}$ be an input string, and let $\widetilde{\boldsymbol{x}}$ be the trace of $\boldsymbol{x}$, sampled by applying the SID channel with deletion probability δ, insertion probability σ and bit-flip rate γ applied to the randomly shifted string $\boldsymbol{x}_s$: (where $s \leftarrow S$).*

Define $\phi_1(z) \stackrel{\text{def}}{=} (1-\delta)z+\delta$, $\phi_2(z) \stackrel{\text{def}}{=} \frac{(1-\sigma)z}{1-\sigma z}$, $\phi \stackrel{\text{def}}{=} \phi_2 \circ \phi_1$. For all j, we set $\zeta_j = \phi^{-1}(z_j)$. Define $P(z) \stackrel{\text{def}}{=} \sum_{s=0}^{d} \Pr[S=s] z^s$. Then:

$$\begin{aligned} P(z_0^{-1}) &\sum_{k_0<k_1<\cdots<k_\ell} (-1)^{\boldsymbol{x}_0} z_0^{k_0} \prod_{1\le j\le \ell} z_j^{k_j-k_{j-1}-1}(-1)^{\boldsymbol{x}_j-\boldsymbol{x}_{j-1}-1} = \\ &= \left(\prod_{0\le j\le \ell} \frac{\phi_1(\zeta_j)}{(1-\delta)(1-2\gamma)\zeta_j}\right) \mathop{\mathbb{E}}_{\widetilde{\boldsymbol{x}}} \left[\sum_{r_0<\cdots<r_\ell} \zeta_0^{r_0}(-1)^{\widetilde{\boldsymbol{x}}_{r_0}} \left(\prod_{j=1}^{\ell}(-1)^{\widetilde{\boldsymbol{x}}_{r_j}} \zeta_j^{r_j-r_{j-1}-1}\right)\right] \end{aligned} \tag{2}$$

Lemma 8 moves us closer to the goal of proving Theorem 7, because the left-hand-side of equation (2) is essentially equivalent to $q_{f,\mathbf{x}}(z_0,\ldots,z_\ell)$ for the function $f(\mathbf{x}_1,\ldots,\mathbf{x}_\ell) = \prod_{1\le j\le \ell}(-1)^{\mathbf{x}_j}$, while the right-hand-side depends only on the traces.

2.1 Proof of Lemma 8

We begin by proving Lemma 8 for the simpler case, where the shift s and the bit-flip probability are both fixed to 0.

Let D_k denote the event that the kth bit of the input string $\mathbf{x}$ was *not* deleted by the channel. Conditioned on D_k, let R_k be the distribution of the index r_k within the trace $\widetilde{\mathbf{x}}$ to which this bit was mapped. For any distribution V, let $G_V(\zeta) \stackrel{\text{def}}{=} \sum_v \Pr[V=v]\zeta^v$ denote its generating function.

Consider the generating function $G_{R_k}(\zeta)$ of R_k. Each of the first k bits of the input message $\mathbf{x}$ was expanded to an i.i.d. geometrically distributed number of bits, and then each of those was either retained or deleted, resulting in a Bernoulli distribution of bits (except for the last bit which was not deleted, because we conditioned on D_k). Using common identities on products and compositions of generating functions, we derive equation (3).

$$G_{R_k}(\zeta) = \left(G_{\mathrm{Geom}(\sigma)}\left(G_{\mathrm{Bern}(\delta)}(\zeta)\right)\right)^{k-1} G_{\mathrm{Geom}(\sigma)-1}\left(G_{\mathrm{Bern}(\delta)}(\zeta)\right)\zeta = \frac{\zeta}{\phi_1(\zeta)}\phi(\zeta)^k \tag{3}$$

Denote by I_r the event that the rth bit of the trace was an insertion. Conditioned on I_r, the rth bit of $\widetilde{\mathbf{x}}$ is a Bernoulli$\left(\frac{1}{2}\right)$ random variable independent of the rest of the problem. Consider the expectation of $f(\widetilde{\mathbf{x}}_{r_0},\ldots,\widetilde{\mathbf{x}}_{r_\ell}) = \prod_{1\le j\le l}(-1)^{\widetilde{\mathbf{x}}_{r_j}}$ over the traces. Due to our choice of f, if even one of its inputs is an insertion, then its expectation is $\mathbb{E}_{\widetilde{\mathbf{x}}}\left[\prod_{0\le j\le l}(-1)^{\widetilde{\mathbf{x}}_{r_j}} \,\middle|\, I_{r_j}\right] = 0$.

The event that the rth bit of the trace $\widetilde{\mathbf{x}}_r$ was not due to an insertion is exactly equal to the event that some bit $\mathbf{x}_k$ in the input message was not deleted (D_k) and that it was transmitted as the rth bit of the trace ($R_k = r$). Therefore, the expectation of f on the multi-index $r_0,\ldots,r_\ell$ of the trace is equal to

$$\begin{aligned} \mathop{\mathbb{E}}_{\widetilde{\mathbf{x}}}\left[\prod_{0\le j\le l}(-1)^{\widetilde{\mathbf{x}}_{r_j}}\right] &= \sum_{k_0<\cdots<k_\ell}(-1)^{\mathbf{x}_{k_j}} \Pr\left[\bigwedge_{0\le j\le \ell}\left(D_{k_j}\wedge\left(R_{k_j}=r_j\right)\right)\right] \\ &= (1-\delta)^\ell \sum_{k_0<\cdots<k_\ell}(-1)^{\mathbf{x}_{k_j}} \Pr\left[\bigwedge_{0\le j\le \ell}\left(R_{k_j}=r_j\right) \,\middle|\, \bigwedge_{0\le j\le \ell} D_{k_j}\right] \end{aligned} \tag{4}$$

Finally, note that given D_{k_j} and the value of $r_j = R_{k_j}$, the effect of the channel on the next bits is independent of r_j. Therefore, conditioned on D_k and D_{k+1}, we have

$$\Pr_{\tilde{\mathbf{x}}}\left[R_{k_j} = r_j \,\middle|\, R_{k_{j-1}} = r_{j-1}\right] = \Pr_{\tilde{\mathbf{x}}}\left[R_{k_j - k_{j-1}-1} = r_j - r_{j-1} - 1\right]. \tag{5}$$

Combining equations (3), (4) and (5), we see that

$$\begin{aligned}
&(1-\delta)^{-\ell}\mathop{\mathbb{E}}_{\tilde{\mathbf{x}}}\left[\sum_{r_0<\cdots<r_\ell}(-1)^{\tilde{\mathbf{x}}_{r_0}}\zeta_0^{r_0}\prod_{1\le j\le l}(-1)^{\tilde{\mathbf{x}}_{r_j}}\zeta_j^{r_j-r_{j-1}-1}\right] = \\
&\quad = \sum_{r_0<\cdots<r_\ell}\sum_{k_0<\cdots<k_\ell}\Pr\left[\bigwedge_{0\le j\le \ell}(R_{k_j}=r_j)\,\middle|\,\bigwedge_{0\le j\le \ell}D_{k_j}\right] \\
&\qquad\qquad (-1)^{\mathbf{x}_{k_0}}\zeta_0^{r_0}\prod_{1\le j\le l}\zeta_j^{r_j-r_{j-1}-1}(-1)^{\mathbf{x}_{k_j}} = \\
&\quad = \sum_{k_0<\cdots<k_\ell}\frac{\zeta_0}{\phi_1(\zeta_0)}\phi(\zeta_0)^{k_0}(-1)^{\mathbf{x}_{k_0}}\prod_{1\le j\le l}\frac{\zeta_j}{\phi_1(\zeta_j)}\phi(\zeta_j)^{k_j-k_{j-1}-1}(-1)^{\mathbf{x}_{k_j}}
\end{aligned} \tag{6}$$

Some minor manipulations to equation (6), yields equation (2) for the case when s and γ are fixed to 0. Finally, we extend the proof to shifts and bit-flips. Let $\overline{\mathbf{x}}$ denote the output of the shift and symmetry portions of the channel. It is easy to show that

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\overline{\mathbf{x}}}\left[\sum_{k_0<\cdots<k_\ell}z_0^{k_0}(-1)^{\overline{\mathbf{x}}_{k_0}}\prod_{1\le j\le l}z_j^{k_j-k_{j-1}-1}(-1)^{\overline{\mathbf{x}}_{k_j}}\right] = \\
&\quad = (1-2\gamma)^\ell\mathop{\mathbb{E}}_{s\leftarrow S}\left[\sum_{k_0<\cdots<k_\ell}z_0^{k_0}(-1)^{\mathbf{x}_{k_0+s}}\prod_{1\le j\le l}z_j^{k_j-k_{j-1}-1}(-1)^{\mathbf{x}_{k_j+s}}\right] = \\
&\quad = (1-2\gamma)^\ell P\left(\frac{1}{z_0}\right)\sum_{k_0<\cdots<k_\ell}z_0^{k_0}(-1)^{\mathbf{x}_{k_0}}\prod_{1\le j\le l}z_j^{k_j-k_{j-1}-1}(-1)^{\mathbf{x}_{k_j}}
\end{aligned} \tag{7}$$

Combining equations (6) and (7) yields Lemma 8.

2.2 Sketch of the Proof of Theorem 7

Due to space limitations, we reserve the rest of the proof of Theorem 7 to the full version of the paper which can be found on arxiv [23], where we show that Lemma 8 implies Theorem 7. The rest of this section is devoted to giving the high-level idea of this proof.

The first step of the proof is an analysis of the Möbius transformations in Lemma 8. In particular, we show that for the points $z_0, \ldots, z_\ell$ chosen as in Theorem 7, the absolute values of $\zeta_0, \ldots, \zeta_\ell$ are bounded below 1.

This allows us to truncate the RHS of equation (2) to only its low degree terms with a negligible effect on the output. This truncation enables us to evaluate this polynomial at the required points, proving Theorem 7 for the full character $f(\mathbf{x}_1, \ldots, \mathbf{x}_\ell) = \prod_j (-1)^{\mathbf{x}_j}$.

In fact, this allows us to estimate $q_{f,\mathbf{x}}(z_0, \ldots, z_\ell)$ for any choice of $z_1, \ldots, z_\ell$ sufficiently close to those defined Theorem 7 when f is the full character. We use this fact to prove Theorem 7 for general characters. In essence, we show that $q_{\chi_\omega,\mathbf{x}}(z_0, \ldots, z_\ell)$ can be written as a high-order derivative of $q_{f',\mathbf{x}}$ for a full character f' on fewer bits $\ell' < \ell$ and that this derivative can be approximated from a limited number of samples using Lemma 9.

▶ **Lemma 9.** *Let $c, \delta > 0$ be some real parameters and let P be an oracle that computes for a given point $z_1, \ldots, z_l \in [-c, c]^l$ the value of some polynomial p of degree at most n at the given point, up to some additive error $\delta > 0$. Let $\boldsymbol{j} = (j_1, \ldots, j_l)$ be some vector of integers (all smaller than n), define $m_{\boldsymbol{j}} = z_1^{j_1} \cdots z_l^{j_l}$ be the $\boldsymbol{j}$th monomial and $j_{\text{tot}} = \sum_i j_i$.*

Given $\text{poly}(n, 1/c)^{O(l+j_{\text{tot}})}$ queries to P, we can compute the coefficient of $m_{\boldsymbol{j}}$ to within an additive error of $\text{poly}(n, 1/c)^{O(l+j_{\text{tot}})}\delta$ in time $\text{poly}(n, 1/c)^{O(l+j_{\text{tot}})}$.

Finally, we use the fact that $q_{f,\mathbf{x}}$ is linear in our choice of f (i.e. for any f_1, f_2, $q_{f_1+f_2,\mathbf{x}} = q_{f_1,\mathbf{x}} + q_{f_2,\mathbf{x}}$), and the fact that any function f can be written as a linear combination of character functions $f = \sum_{\omega \in \{\pm 1\}^\ell} \hat{f}(\omega)\chi_\omega$ via a Fourier transformation. Combining these observations, we see that

$$q_{f,\mathbf{x}} = \sum_{\omega \in \{\pm 1\}^\ell} \hat{f}(\omega) q_{\chi_\omega,\mathbf{x}}.$$

The RHS of this equation can be estimated from the traces (one element at a time) and the LHS was our original goal, thus proving Theorem 7.

3 Proof of Theorem 2

In Section 2, we showed that for any function f from $\{0,1\}^\ell$ to the unit disk $\mathbb{D}$, we can map it into a polynomial related to the input message which can be approximated to a high degree of accuracy from the traces. In this section, we will construct a function f for which our approximation of $q_{f,\mathbf{x}}$ as promised by Theorem 7 will suffice to reconstruct the $n+1$th bit of the input string $\mathbf{x}$, proving Theorem 2.

A central component of our analysis will be Theorem 10, which is a slight generalisation of [7, Theorem 5]. In this theorem we show that members of a certain class of polynomials have some non-negligible values on a sufficiently small sub-arc of the unit disk $\mathbb{D}$.

The polynomial $p(z)$ in Theorem 10 should be thought of as the difference between two polynomials $q_{f,\mathbf{x}}(z, 0, \ldots, 0) - q_{f,\mathbf{y}}(z, 0, \ldots, 0)$ for two hypotheses $\mathbf{x}$ and $\mathbf{y}$ for the input string. By proving that these polynomials differ at a point where they can be estimated from the traces, we show that this estimation can be used to differentiate between the hypotheses.

▶ **Theorem 10** (Extension of [7, Theorem 5])**.** *Let $\mathcal{P}_n^\mu$ denote the set of polynomials of the form $p(z) = \xi - \eta z^d + \sum_{n^\mu \le j \le n} a_j z^j$ where $\eta \in \{0,1\}$, $\xi \in \partial\mathbb{D}$ and $|a_j| \le 2$.*

For any $\mu \in (0,1)$, there exists some constant $C > 0$, such that for all sufficiently large n, any $p \in \mathcal{P}_n^\mu$, it holds that for every $\rho \in [0,1]$:

$$\max_{|\alpha| \le n^{-2\mu}} \left|p(\rho e^{i\alpha})\right| \ge \exp\left(-Cn^\mu \log^5 n\right)$$

Our proof of Theorem 10 is similar to Chase's proof of [7, Theorem 5], and due to space limitations, we reserve it for the full version of this paper [23]. Throughout the rest of this section, we will prove that Theorem 2 follows from Theorem 10.

In Section 3.1, we will extend Theorem 10 to a wider class of polynomials, proving that the estimation method described in Theorem 7 can be used to distinguish between the traces of any two potential input strings $\mathbf{x}, \mathbf{y}$. In Section 3.2, we will show how this distinguishing oracle can be used to reconstruct a string $\mathbf{x}$ from the shifted trace reconstruction problem, proving Theorem 2.

3.1 Corollaries of Theorem 10

In this section, we will extend Theorem 10 to prove that for any strings $\mathbf{x}, \mathbf{y}$ which agree on their first n bits, the estimation oracle described in Theorem 7 can be used to distinguish between their traces. This proof will follow from two main components.

First, we will show that for any two such strings $\mathbf{x}, \mathbf{y}$, there exists some choice of indicator function $f = I_{\mathbf{w}}$, such that for $p_{f,\mathbf{x}}(z) = q_{f,\mathbf{x}}(z, 0, \ldots, 0)$, the polynomial $p_{\text{diff}}(z) = p_{f,\mathbf{x}}(z) - p_{f,\mathbf{y}}(z)$ (almost) fits the requirements of Theorem 10. Therefore, if $\delta < 1/2$, then there exists some point $(z, 0, \ldots, 0)$ such that we can estimate the evaluation of $q_{f,\mathbf{x}}$ from the traces and that $q_{f,\mathbf{x}}$ and $q_{f,\mathbf{y}}$ differ significantly at this same point. This yields a method of distinguishing between their traces (see Corollary 11).

Then, in Corollary 12, we will extend this analysis to higher deletion probabilities, by showing that a similar distinguishing method can also be used at points of the form $(z, 1-c, \ldots, 1-c)$. For the rest of this section, let $\mu = 1/5$, $\rho = 1 - n^{-4/5} \log^6 n$, $\ell = 2n^{1/5} + 1$, and $\mathcal{A} = \{\rho e^{i\alpha} \mid |\alpha| \leq n^{2/5}\}$.

The following is a corollary of Theorem 10:

▶ **Corollary 11** (Adaptation of Proposition 6.3 from [7]). *Let $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^{\mathbb{N}}$ be binary strings that agree on their first n bits ($\boldsymbol{x}_{:n} = \boldsymbol{y}_{:n}$) and disagree on their $(n+1)$th bit ($\boldsymbol{x}_{n+1} \neq \boldsymbol{y}_{n+1}$). Then there exist some $\boldsymbol{w} \in \{0,1\}^{\ell}$ and $z_0 \in \mathcal{A}$ such that*

$$|q_{I_w,\boldsymbol{x}}(z_0, 0, \ldots, 0) - q_{I_w,\boldsymbol{x}}(z_0, 0, \ldots, 0)| \geq \exp\left(-n^{1/5} \log^6 n\right) \exp\left(-Cn^{1/5} \log^5 n\right)$$

Proof of Corollary 11. Let $\mathbf{x}$ and $\mathbf{y}$ be two hypotheses for the input string to a shifted trace reconstruction problem (that agree on their first n bits and not on their $n+1$th bit).

Let $\mathbf{w}' = \mathbf{x}(n - \ell + 1 : n)$. Lemmas 1 and 2 of [22] imply that at least one of $\mathbf{w}'0$ or $\mathbf{w}'1$ has no period of length $\leq n^{1/5}$ and that for this choice of $\mathbf{w} \in \{\mathbf{w}'0, \mathbf{w}'1\}$, the indices k for which $\mathbf{x}_{k:k+\ell} = \mathbf{w}$ are $n^{1/5}$ separated.

Consider the polynomial

$$\begin{aligned} p_{\mathbf{w}}(z) &\stackrel{\text{def}}{=} z^{\ell-n-1} \left[q_{I_{\mathbf{w}},\mathbf{x}}(z, 0, \ldots, 0) - q_{I_{\mathbf{w}},\mathbf{y}}(z, 0, \ldots, 0)\right] \\ &= \sum_k \left[(-1)^{\mathbf{x}_k} 1_{\mathbf{x}(k+1:k+\ell)=\mathbf{w}} - (-1)^{\mathbf{y}_k} 1_{\mathbf{y}_{k+1:k+\ell}=\mathbf{w}}\right] z^{k+\ell-n-1} \end{aligned}$$

Because $\mathbf{x}$ and $\mathbf{y}$ agree on their first n bits, $p_{\mathbf{w}}(z)$ has no negative powers. By our definition of $\mathbf{w}$ to be either $\mathbf{x}_{n-\ell+1:n+1}$ or $\mathbf{y}_{n-\ell+1:n+1}$, the 0th power of $p_{\mathbf{w}}(z)$ is ± 1. Moreover, all of $p_{\mathbf{w}}(z)$'s coefficients are bounded by 2 in absolute value and its non-zero powers maintain the sparsity condition of Theorem 10.

The only problem with applying Theorem 10 to $p_{\mathbf{w}}(z)$ is that its degree is not bounded by n. We overcome this issue by defining $\widetilde{p_{\mathbf{w}}}(z)$ to be the truncation of $p_{\mathbf{w}}(z)$ to its nth power. Applying Theorem 10 to the polynomial $\widetilde{p_{\mathbf{w}}}$, we see that there exists a point $z_0 \in \mathcal{A}$ for which $|\widetilde{p_{\mathbf{w}}}(z_0)| \geq \exp\left(-C_1 n^{\mu} \log^5 n\right)$

Because we want to evaluate $\widetilde{p_{\mathbf{w}}}(z)$ at points z with absolute value $|z| = \rho$ strictly below 1, we can also bound the effect of this truncation by

$$|\widetilde{p_{\mathbf{w}}}(z) - p_{\mathbf{w}}(z)| \leq \frac{\rho^n}{1-\rho} = \text{poly}(n) \exp(-n^{\mu} \log^6 n) = o\left(|\widetilde{p_{\mathbf{w}}}|\right)$$

From here we apply the triangle inequality to show that $|p_{\mathbf{w}}(z)| \geq \exp\left(-C_2 n^{\mu} \log^5 n\right)$.

Finally, note that $|q_{I_{\mathbf{w}},\mathbf{x}}(z, 0, \ldots, 0) - q_{I_{\mathbf{w}},\mathbf{y}}(z, 0, \ldots, 0)| = |p_{\mathbf{w}}(z)||z|^{n-\ell+1}$, completing the proof of Corollary 11. ◀

Corollary 11 allowed us to use Theorem 7 to distinguish between the traces of any two string $\mathbf{x}$ and $\mathbf{y}$ when the deletion probability of the channel is low ($\delta < 1/2$).

However, this proof relied on our ability to estimate the value of $q_{I_{\mathbf{w}},\mathbf{x}}$ at points where $z_1 = \cdots = z_\ell = 0$, and when the deletion probability is high ($\delta \geq 1/2$), Theorem 7 only allows us to evaluate $q_{I_{\mathbf{w}},\mathbf{x}}$ at points of the form $z_1 = \cdots = z_\ell \in [1-c_1, 1-c_2]$. In order to distinguish between the traces of $\mathbf{x}$ and $\mathbf{y}$ from high deletion probability channels, we extend Theorem 7 to multivariate polynomials sampled at such points. We do this in Corollary 12.

▶ **Corollary 12** (Adaptation of Corollary 6.1 from [7]). *Let $c_1 > c_2 > 0$ be sufficiently small positive constants, and let $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^{\mathbb{N}}$ be as in Corollary 11. There exist some $\boldsymbol{w} \in \{0,1\}^l$, $z_0 \in \mathcal{A}$ and $z_1 = \cdots = z_\ell \in [1-c_1, 1-c_2]$, such that*

$$|q_{I_w,\boldsymbol{x}}(z_0, z_1, \ldots, z_1) - q_{I_w,\boldsymbol{x}}(z_0, z_1, \ldots, z_1)| \geq \exp\left(-n^{1/5}\log^6 n\right)\exp\left(-Cn^{1/5}\log^5 n\right)$$

Proof of Corollary 12. Fix $\mathbf{w}$ and z_0 to be the same as in the proof of Corollary 11. We define Q to be the following polynomial in z_1, for $z_1 \in [0, 1-c_2]$.

$$Q(z_1) \stackrel{\text{def}}{=} (1-\rho)\binom{n}{\ell}^{-1}\left[q_{I_{\mathbf{w}},\mathbf{x}}(z_0, z_1, \ldots, z_1) - q_{I_{\mathbf{w}},\mathbf{y}}(z_0, z_1, \ldots, z_1)\right]$$

Consider the coefficient of the jth power of z_1 in Q. If $j \leq n$, then this coefficient is bounded by 1 in absolute value. This is because our summation over the powers of z_0 can contribute a factor of at most $1/(1-\rho)$, and the number of terms in $q_{I_{\mathbf{w}},\mathbf{x}}$ with total degree j is at most $\binom{n}{\ell}$.

If $j > n$, then the number of monomials of $q_{I_{\mathbf{w}},\mathbf{x}}$ with total degree j is at most $\exp(O(\ell\log(j)))$, but the value of the monomial z_1^j is at most $(1-c_2)^j = \exp(-\Omega(j))$.

Therefore, truncating these higher powers of Q would have a negligible effect on its value. Let $\widetilde{Q}(z_1)$ be the truncation of Q to monomials of degree $\leq n$. $\widetilde{Q}$ is a univariate polynomial in z_1, with coefficients bounded from above by 1, and for any $z_1 \in [0, 1-c_2]$, we have

$$\left|Q(z_1) - \widetilde{Q}(z_1)\right| \leq \exp(-\Omega(n)) \tag{8}$$

In Corollary 11, we showed that $|Q(0)|$ is bounded from below, and this lower bound can be naturally extended to $\left|\widetilde{Q}(0)\right|$. Therefore, $\widetilde{Q}(z_1)$ fits the requirements of Theorem 5.1 of [4], which can be used to show that

$$\max_{z_1 \in [1-c_1, 1-c_2]} \widetilde{Q}(z_1) \geq \exp\left(Cn^\mu \log^6 n\right) \tag{9}$$

Combining equations (8) and (9) yields our claim. ◀

3.2 Completing the Proof

In Section 3.1, we proved that the estimation method promised in Theorem 7 can be used to differentiate between any two potential input strings $\mathbf{x}$ and $\mathbf{y}$ from their traces. In this section, we will show how this distinguishing oracle can be transformed into a reconstruction algorithm, completing the proof of Theorem 2.

The basic idea of this transformation is relatively simple. We enumerate over potential pairs of input strings $\mathbf{y}^0, \mathbf{y}^1$, and use the distinguishing oracle to decide for each pair which is a better candidate for being the input string $\mathbf{x}$.

The main technical difficult we need to overcome is due to the fact that the input string $\mathbf{x}$ may be arbitrarily long, so enumerating over all possible input strings can take an arbitrarily long amount of time. We overcome this, by showing that it suffices to enumerate over the

first $O(n)$ bits of the input string. Moreover, when the deletion probability is below $1/2$, we show that it suffices to enumerate over only a small fraction of the entropy of these $O(n)$ bits, yielding a fast reconstruction algorithm.

Let $\mathbf{x}$ be the input string to the shifted trace reconstruction problem. By our definition of the shifted trace reconstruction problem, the first n bits of $\mathbf{x}$ are known, and our goal is to reconstruct the $n+1$th bit of the input string $\mathbf{x}$.

Let $C > 0$ be a sufficiently large constant. Let $\mathbf{o}^0, \mathbf{o}^1 \in \{0,1\}^{Cn-n-1}$ be two hypotheses for the value of $\mathbf{x}_{n+1:Cn}$. In other words, $\mathbf{y}^0 = \mathbf{x}_{1:n}0\mathbf{o}^0, \mathbf{y}^1 = \mathbf{x}_{1:n}1\mathbf{o}^1$ are our hypotheses for the first Cn bits of $\mathbf{x}$.

If $\delta < 1/2$, let z_0 and $\mathbf{w}$ be as defined in Corollary 11, and let $z_1 = 0$. If $\delta \geq 1/2$, let z_0, z_1 and $\mathbf{w}$ be as defined in Corollary 12.

We use the traces to estimate $p_{I_{\mathbf{w}},\mathbf{x}}(z_0, z_1, \ldots, z_1)$ using the method promised by Theorem 7. This method may have a small failure probability (which would result in a bad estimate), but for the moment we assume that it succeeds. We then compute $p_{I_{\mathbf{w}},\mathbf{y}}(z_0, z_1, \ldots, z_1)$ directly for $\mathbf{y} \in \{\mathbf{y}^0, \mathbf{y}^1\}$.

Consider the case where $\mathbf{y}^b = \mathbf{x}_{:Cn}$ for some $b \in \{0,1\}$. Because we are evaluating $p_{I_{\mathbf{w}},\mathbf{y}}$ at points with coordinates strictly below 1 in absolute value and this polynomial's coefficients are bounded by 1, the contribution of monomials with total degree above Cn is can be bounded. In particular,

$$\begin{aligned}\left|p_{I_{\mathbf{w}},\mathbf{y}^b}(z_0, z_1, \ldots, z_1) - p_{I_{\mathbf{w}},\mathbf{x}}(z_0, z_1, \ldots, z_1)\right| < \exp(-\Omega(Cn^{1/5}\log^6 n)) \ll \\ \ll \left|p_{I_{\mathbf{w}},\mathbf{y}^b}(z_0, z_1, \ldots, z_1) - p_{I_{\mathbf{w}},\mathbf{y}^{1-b}}(z_0, z_1, \ldots, z_1)\right|\end{aligned} \tag{10}$$

Therefore, in this case, our estimate of $p_{I_{\mathbf{w}},\mathbf{x}}(z_0, z_1, \ldots, z_1)$ from the traces will be closer to $p_{I_{\mathbf{w}},\mathbf{y}^b}(z_0, z_1, \ldots, z_1)$ than to $p_{I_{\mathbf{w}},\mathbf{y}^{1-b}}(z_0, z_1, \ldots, z_1)$.

We repeat this process for any such pair $\mathbf{o}^0, \mathbf{o}^1 \in \{0,1\}^{Cn-n-1}$, and for each such pair, we output the value b for which our estimate of $p_{I_{\mathbf{w}},\mathbf{x}}(z_0, z_1, \ldots, z_1)$ from the traces is closest to $p_{I_{\mathbf{w}},\mathbf{y}^b}(z_0, z_1, \ldots, z_1)$.

If $b = \mathbf{x}_{n+1}$, then there exists at least one such $\mathbf{o}^b$ for which the process above always selects b for any $\mathbf{o}^{1-b}$. By enumerating over all pairs, we can find the value of $b = \mathbf{x}_{n+1}$ for which such a string $\mathbf{o}^b$ exists.

This leaves only a few minor technical details in order to prove Theorem 2.

First, we note that the estimation oracle promised in Theorem 7 has a small failure probability. We use the union bound to show that the probability that it will fail even once in the process described above is negligible.

Next we consider the time complexity of our reconstruction. For the high deletion probability regime ($\delta \geq 1/2$), this process can clearly be completed in time $\exp(O(n))$.

For lower deletion probabilities $\delta < 1/2$, we note that $p_{I_{\mathbf{w}},\mathbf{y}}(z_0, 0, \ldots, 0)$ depends only on the indices within $\mathbf{y}$ where the string $\mathbf{w}$ appears as a consecutive substring. By our definition of $\mathbf{w}$ (see the proof of Corollary 11), this set of indices is sparse. By enumerating only over the set of indices where $\mathbf{w}$ appears in $\mathbf{y}$ (and not over the entire Cn bits), we can reduce the time complexity of this reconstruction algorithm to $\exp(o(n))$, thus completing our proof of Theorem 2.

4 Proof of Theorem 4

In Sections 2 and 3, we showed that Chase's worst-case trace reconstruction method can be naturally extended to the shift trace reconstruction problem and to SID channels. In this section, we will construct a general reduction from the average-case trace reconstruction problem to the shifted trace reconstruction problem, proving Theorems 4 and 6.

Our proof will be based based on an adaptation of the HPPZ's methods, and our main contribution is to show that it can be used as a general reduction as well as to extend it to symmetry channels. Due to space limitations, in this version of the paper we will give only a sketch of the proof (for more details, see the full version of this paper [23])

Our reduction will consist of three main ingredients:

- A Boolean test $T(\mathbf{w}, \widetilde{\mathbf{w}})$ on pairs of bit-strings $(\mathbf{w}, \widetilde{\mathbf{w}})$ that returns 1 if $\widetilde{\mathbf{w}}$ is a plausible match for the output of applying the channel $\mathcal{C}$ to $\mathbf{w}$.
- A two-step alignment procedure comprised of a coarse and a fine alignment each of which uses the test T to obtain an estimate τ^k for the positions within some of the traces corresponding to the kth bit of the original message $\mathbf{x}$.
- The reduction target – a bit recovery procedure based on the target of our reduction to produce an estimate of any bit of $\mathbf{x}$ from these aligned traces.

Finally, similar to HPPZ, throughout this section we will perform our analysis when $\delta = \sigma$, but all of these results can be similarly generalized for any values of $\delta, \sigma \in [0, 1)$.

4.1 The Boolean Test

The first component of our reduction is a Boolean test T designed to answer whether a string $\widetilde{\mathbf{w}}$ is likely to have originated from a trace of some string $\mathbf{w}$ or not.

Let $\ell, \lambda < \sqrt{\ell}$ and $c \in (0, 1)$ be parameters of the test. The test $T^c_{\ell,\lambda}$ (when c, ℓ, λ are clear from the context we may omit them) is defined as follows. First, each of the strings $\mathbf{w}$ and $\widetilde{\mathbf{w}}$ is split into $\approx \ell/\lambda$ segments of length λ each. Each segment of each string is assigned a sign $+1$ if most of the bits in this segment are 0s or -1 otherwise. In other words

$$s_i = \operatorname{sign}\left\{ \sum_{i\lambda < j \le (i+1)\lambda} (\mathbf{w}_j - 1/2) \right\} \in \{\pm 1\}.$$

Then, the signs of the segments are compared, and we compute the number of segment pairs whose signs agree. If $\mathbf{w}$ and $\widetilde{\mathbf{w}}$ were two independently distributed random strings, then the number of such pairs would be distributed according to the $\mathrm{Bin}(1/2, \ell/\lambda)$ distribution. If $\mathbf{w}$ and $\widetilde{\mathbf{w}}$ are similar, then we expect these signs to be roughly correlated to one another.

Therefore, we define the test to pass if at least $(1 + c)/2$ fraction of the signs agree.

$$T^c_{\ell,\lambda} = \begin{cases} 1 & \sum_{1 \le i \le \ell/\lambda} s_i \widetilde{s}_i > c \\ 0 & \text{otherwise} \end{cases}$$

We consider this test with two sets of parameters for "coarse" and "fine" alignment procedures. Let $H(n) = \exp(h(n))$ be the sample complexity of the shifted trace reconstruction problem. Then for the coarse and fine alignments we set respectively

$$\ell_c = \Theta\left(\frac{\log^2(n)}{h(\Theta(\log(n)))}\right); \qquad \lambda_c = \Theta\left(\frac{\log(n)}{h(\Theta(\log(n)))}\right)$$
$$\ell_f = \Theta\left(h(\log(n))\right); \qquad \lambda_f = \Theta(1).$$

Ideally, we want this Boolean test to maintain two behaviours:

- If $\widetilde{\mathbf{w}}$ is not a trace of $\mathbf{w}$, the probability that T will return 1 (called a *spurious match*) should be at most $\exp(-\Omega(\ell/\lambda))$.
- If $\widetilde{\mathbf{w}}$ is a trace of $\mathbf{w}$, the probability that T will pass (called a *true match*) will be at least $\exp(-O(\ell/\lambda^2))$.

If these conditions hold, then the probability of a true match may be very small, but when λ is sufficiently large, it will be much higher than the probability of a spurious match. Therefore, when conditioning on a match, it will most likely be a true match. Over the next few paragraphs, we will give a sketch of the proof that these conditions hold for substrings of a random string $\mathbf{x}$.

4.1.1 Spurious Matches are Rare

If $\mathbf{w}$ and $\widetilde{\mathbf{w}}$ are two independently distributed strings chosen uniformly at random, then the signs of their segments s_i as defined above will also be independent and uniformly distributed vectors $s, \widetilde{s} \in \{\pm 1\}^{\ell/\lambda}$. In this case, it can be easily shown from the Chernoff bound that the probability that more than $(1+c)/2$ fraction of their entries agree decays exponentially in their dimension ℓ/λ.

The main difficulty is analysing how this relates to the traces of a random string. Let $\mathbf{w}^0 = \mathbf{x}_{a_0:b_0}$ and $\mathbf{w}^1 = \mathbf{x}_{a_1:b_1}$ be two substrings of the random input string $\mathbf{x}$. If the segments $[a_0, b_0]$ and $[a_1, b_1]$ do not overlap, then (averaging over the random options for the input string $\mathbf{x}$) they are two independent random strings.

Let $\widetilde{\mathbf{w}}^i$ be the trace of $\mathbf{w}^i$. Clearly, when applying the channel $\mathcal{C}$ (which only deletes bits, inserts i.i.d. uniformly distributed bits and replaces some of the bits of $\mathbf{x}$ with i.i.d. uniformly distributed bits) to a random string of length ℓ, the output will also be a random string of length roughly $\frac{1-\delta}{1-\sigma}\ell = \ell$. Therefore, if $\mathbf{w}^0, \mathbf{w}^1$ are non-overlapping substrings of $\mathbf{x}$ as defined above, then $\mathbf{w}^0$ and $\widetilde{\mathbf{w}}^1$ are two independent random strings.

Let us denote by ω the randomness of the channel $\mathcal{C}$. Averaging over both the randomness of the channel and over our selection of the input string $\mathbf{x}$, we have

$$\begin{aligned} \mathop{\mathbb{E}}_{\mathbf{x}}\left[\Pr_{\omega}\left[T(\mathbf{w}^0, \widetilde{\mathbf{w}}^1) = 1\right]\right] &= \Pr_{\omega,\mathbf{x}}\left[T(\mathbf{w}^0, \widetilde{\mathbf{w}}^1) = 1\right] \\ &= \Pr_{\mathbf{w}^0,\mathbf{w}^1 \leftarrow \{0,1\}^\ell}\left[T(\mathbf{w}^0, \mathbf{w}^1)\right] = \exp\left(-\Omega(\ell/\lambda)\right) \end{aligned} \tag{11}$$

We will use equation (11) in the two settings of the alignment procedure. In the coarse alignment, we set $\ell_c/\lambda_c = C\log(n) = \Theta(\log(n))$. Setting C to be sufficiently large, we can ensure that $\Pr_{\omega,\mathbf{x}}\left[T(\mathbf{w}^0, \widetilde{\mathbf{w}}^1) = 1\right] = \exp(-\Omega(C\log(n))) < n^{-10}$ is sufficiently small that a simple union bound on the quasi-linear number of coarse alignment procedures we run will never result in a spurious match.

For the fine alignment procedure, we will have a segment $I = [a, a + C\log(n)]$ of length $\Theta(\log n)$ of the input string $\mathbf{x}$ in which our goal will be to find a subsegment $S = [b, b + \ell_f]$ of length $\ell_f = o(\log(n))$ such that for any non-overlapping subsegment $S' \subset I$ of length ℓ_f, the probability of a spurious match between $\mathbf{w}^0 = \mathbf{x}_S$ and a trace of $\mathbf{w}^1 = \mathbf{x}_{S'}$ is

$$\Pr_{\omega}\left[T(\mathbf{w}^0, \widetilde{\mathbf{w}}^1) = 1\right] = \exp(-\Omega(\ell_f/\lambda_f)).$$

It can be shown from a simple combination of a Markov inequality (used to show that the probability of any such subsegment S to work is $1 - \exp(-\Omega(\ell_f))$) and an enumeration over sufficiently many independent options for S, that at least 1 such subsegment exists w.p. $1 - \exp(-\Omega(C\log(n))) = 1 - n^{-10}$. From here, we can simply apply the union bound over the quasi-linear number of fine alignment procedures in the reduction.

4.1.2 True Matches are Frequent

The next step of our proof will be to show that a string $\mathbf{w}$ and its trace $\widetilde{\mathbf{w}}$ will pass the test $T_{\ell,\lambda}$ with probability at least $\exp\left(-O(\ell/\lambda^2)\right)$. Due to space limitations, we give only a very rough sketch of this proof (for a more detailed proof, see the full version of this paper [23]).

Consider a substring $\mathbf{u} = \mathbf{w}_{i\lambda:(i+1)\lambda}$ of the string $\mathbf{w}$ the matching substring $\widetilde{\mathbf{w}}_{i\lambda:(i+1)\lambda}$ of its trace. The total of the bits in $\mathbf{u}$ is binomially distributed, so there is a non-negligible probability that $\approx 1/2 + \sqrt{1/\lambda}$ fraction of them will be 0 (in which case, its sign will be -1). If this is the case, then with fairly high probability, for any substring $\mathbf{u}' = \mathbf{w}_{i\lambda+d_i,(i+1)\lambda d_{i+1}}$ where $|d_i| < \lambda/100$, at least $\approx 1/2 + \sqrt{1/2\lambda}$ fraction of its bits will be 0.

For now, assume that $\widetilde{\mathbf{w}}_{i\lambda:(i+1)\lambda} = \widetilde{\mathbf{u}'}$ originated from the application of the channel $\mathcal{C}$ to $\mathbf{u}'$. The channel replaced a constant fraction of the bits of $\mathbf{u}'$ with random bits (through the symmetry portion of the channel or the insertion and deletion portions). However, a constant fraction of these bits were retained, so there is some correlation between their total and that of the string $\mathbf{u}'$. It can be shown that this correlation suffices to ensure a probability of at least $1/2 + \Omega(1)$ that the sign of this segment $\widetilde{s}_i$ of the trace will be equal to the sign of the appropriate segment s_i of the input string $\mathbf{w}$.

These correlations suffice to ensure that on average $1/2 + \Omega(1)$ of the segments of the trace $\widetilde{\mathbf{w}}$ of an input string $\mathbf{w}$ will have the same sign as the appropriate segments of the input string $\mathbf{w}$, conditioned on each of the *mismatches* d_i being at most $|d_i| < \lambda/100$ (with probability $1 - \exp(-\Omega(\ell))$ over the choice of $\mathbf{w}$). Therefore, if we properly set the constant c parameter of the test T, under these conditions the probability of a true match will be at least $\Omega(1)$.

The next step of our analysis is to show that the mismatches d_i are sufficiently small with probability at least $\exp(-O(\ell/\lambda^2))$. A formal version of this analysis can be found in the full version of our paper [23].

4.2 Coarse and Fine Alignments

Next, we define our coarse and fine alignment procedures. Let be C a sufficiently large constant. We define the parameters for the test used in our coarse and fine alignment procedures to be:

$$\ell_c = C\frac{\log^2 n}{h(C\log n)}; \qquad \lambda_c = C^{1/2}\frac{\log n}{h(C\log n)}$$
$$\ell_f = C^{2/3}h(C\log n); \qquad \lambda_f = C^{1/12}$$

In the full version of this paper [23], we define a precise condition on the input string $\mathbf{x}$ being "well-behaved" (denoted by $\mathbf{x} \in \Xi_{\text{good}}$), and show that a string $\mathbf{x} \in \{0,1\}^n$ selected uniformly at random is well-behaved with probability $1 - n^{-2}$. We define our alignment procedure for well-behaved strings $\mathbf{x}$.

Let $\mathbf{x} \in \Xi_{\text{good}}$ be a well-behaved string. For any integer $k \in [\ell_c + C\log n, n]$, we set the index $a_1 = k - \ell_c - C\log n$ and select $a_2 \in [k - 2/3C\log n, k - 1/3C]$ through a process defined in the full version of this paper [23].

For any trace $\widetilde{\mathbf{x}}$, we set our *coarse alignment* τ_1^k to be the first integer b for which

$$T_{\ell_c,\lambda_c}(\mathbf{x}([a_1, a_1 + \ell_c]), \widetilde{\mathbf{x}}([b, b + \ell_c])) = 1$$

or ∞ if no such b exists. For any trace $\widetilde{\mathbf{x}}$ with $\tau_1^k < \infty$, we define its *fine alignment* τ_2^k to be the first index $b \in [\tau_1^k - \ell_c, \tau_1^k + 2\ell_c + C\log n]$ such that

$$T_{\ell_f,\lambda_f}(\mathbf{x}([a_2, a_2 + \ell_f]), \widetilde{\mathbf{x}}([b, b + \ell_f])) = 1.$$

We define the *mismatch* $d(k, \tau_i^k)$ of any finite alignment $\tau_i^k < \infty$ as the distance between τ_i^k and the index of the first bit of the trace originating from the kth bit of the input message onwards $\mathbf{x}_{k:}$. The following lemma (which we prove in the full version of this paper [23]) promises that $\tau_i^k < \infty$ with sufficiently high probability and that there is a negligible probability that the mismatch of τ_i^k is large.

▶ **Lemma 13.** *Let $\boldsymbol{x} \in \Xi_{good}$ be a well-behaved string and let $k \in \{\ell_c + C \log n, \ldots, n\}$ be an integer. Then for $a_1, a_2, \tau_1^k, \tau_2^k$ as defined above, the following properties hold:*

- $\Pr\left[\tau_1^k < \infty\right] > \exp(-c_1 C^{1/2} h(C \log n))$
- $\Pr\left[\tau_1^k < \infty \wedge d(k, \tau_1^k) > \ell_c\right] < n^{-2}$
- $\Pr\left[\tau_2^k < \infty \mid \tau_1^k < \infty\right] \geq \exp(-c_2 C^{1/2} h(C \log n))$
- $\Pr\left[\tau_2^k < \infty \wedge d(k, \tau_2^k) > \ell_f \mid \tau_1^k < \infty\right] < \exp(-c_3 C^{7/12} h(C \log n))$

Where the probabilities are taken over the randomness of the channel and $c_1, c_2, c_3, c_4 > 0$ are positive constants that may depend on δ, σ, γ but not on C or n and originate from the $\Omega(\cdot)$s and $O(\cdot)$s of the previous sections.

Moreover, as we prove in the full version of this paper, this alignment can be performed efficiently.

▶ **Lemma 14** (τ_1^k, τ_2^k can be computed efficiently)**.** *There is an algorithm A_{align} such that, for any $\boldsymbol{x} \in \Xi_{good}, k \in \{\ell_c + C \log n, \ldots, n\}$ and any trace $\widetilde{\boldsymbol{x}}$ of $\boldsymbol{x}$ through the channel, given*

$$k, \boldsymbol{x}_{:k}, (\tau_1^1, \ldots, \tau_1^{k-1}), (\tau_2^1, \ldots, \tau_2^k)$$

A_{align} computes τ_1^k, τ_2^k, a_2 in time $n^{o(1)}$, with probability $\geq 1 - n^{-2}$.

4.3 Using the Oracle

In Section 4.1, we introduced the Boolean test which can be used to test whether a substring of a trace $\widetilde{\mathbf{x}}$ originated from a specific substring of the input string $\mathbf{x}$. Then, in Section 4.2, we showed that this test can be used as a central component of an alignment procedure which maps indices of the input string $\mathbf{x}$ to their positions in the traces $\widetilde{\mathbf{x}}$ with high probability. In this section, we will complete the proof of our reduction from the average-case trace reconstruction problem to the shifted trace reconstruction problem.

Proof of Theorem 4. Let $\mathcal{C}$ be an SID channel with parameters γ, σ, δ, and let C to be a sufficiently large constant.

We will prove that given the first $k \geq \ell_c + C \log n$ bits of $\mathbf{x}$, we can reconstruct the rest of its bits one at a time. We can work under this assumption, by adding $\ell_c + C \log n$ virtual 0 bits to the start of $\mathbf{x}$ and adding a trace of 0^k to the beginning of each of the traces $\widetilde{\mathbf{x}}$ before the reconstruction.

Given the first k bits of $\mathbf{x}$, we will show that we can reconstruct the $k+1$th bit of $\mathbf{x}$ and from there, we can continue this process iteratively. Using the alignment algorithm from Lemma 14, we compute τ_1^k and τ_2^k of each of the traces $\widetilde{\mathbf{x}}$.

Given a_2, τ_2^k, we run the shifted trace reconstruction algorithm A with parameters $n', n'-1$, where $n' = k - a_2 \in [1/3C \log n, 2/3C \log n]$, on the set:

$$\mathcal{X} = \left\{ \widetilde{\mathbf{x}}(\tau_2^k :) \middle| \begin{smallmatrix} \widetilde{\mathbf{x}} \text{ is a sample} \\ \tau_2^k(\widetilde{\mathbf{x}}) < \infty \end{smallmatrix} \right\}$$

The first and third claims of Lemma 13, mean that for each of our $N = \exp(Ch(C \log n))$ traces, it will have a finite τ_2^k, with probability at least

$$\exp(-C^{1/2}(c_1 + c_2) h(C \log n)) \geq \exp(-1/3Ch(C \log n)).$$

Therefore, by Hoeffding's inequality, the probability that we will have at least

$$1/2 \exp(2/3Ch(C \log n)) > \exp(1/2Ch(2/3C \log n)) \geq \exp(h(k - a_2)) \log^2(n)$$

traces for which $\tau_2^k < \infty$ is at least

$$1 - \exp(-\Omega(Ch(C \log n))) = 1 - n^{-\omega(1)}$$

Lemma 13 gives us that the probability that any sample for which $\tau_2^k < \infty$ is the result of a spurious match is at most

$$\varepsilon(n) \leq \exp(-(C^{7/12}c_3 - C^{1/2}(c_1 + c_2))h(C \log n)) \leq \exp(-10h(k - a_2))$$

Splitting our samples into $\log^2(n)$ batches of size $\exp(h(k - a_2))$ each, we ensure that

1. From the union bound, for each batch, the probability that even a single sample is due to a spurious match is at most $\exp(-9h(k - a_2)) = o(1)$.
2. For each batch, if this batch contained no spurious matches, then applying the shifted trace reconstruction oracle on this batch separately will yield the correct value of the bit $\mathbf{x}_k$ with probability $1 - o(1)$.
3. The batches are independent of one another.

From here we can use to Chernoff bound to show that the probability that more than $1/3$ of these batches either has at least one spurious match or yielded the wrong output from the shifted trace reconstruction oracle is $\exp(-\Omega(\log^2(n))) = n^{-\omega(1)}$, so taking a majority vote on the applications of the shifted trace reconstruction oracle will yield the correct value of $\mathbf{x}$ with probability $1 - n^{-\omega(1)}$, completing our proof. ◀

5 Conclusions

In this paper we presented two main results. First, we proved a general reduction from the average-case trace reconstruction problem to the shifted trace reconstruction problem, which is similar to the worst-case trace reconstruction problem. Second, we generalised the leading algorithm for the worst-case trace reconstruction problem from deletion channels by Chase [7] to the shifted trace reconstruction problem and to the more general class of symmetry-insertion-deletion channels.

Our reduction is based on the work of Holden et al. [14] who used a similar technique to convert the specific methods of De et al. and Nazarov and Peres [12, 20] from worst-case trace reconstruction to the average-case. Continuing the line of work of Brakensiek et al. [5] who reduced the coded trace reconstruction problem to the average-case trace reconstruction problem, we convert the specific construction of Holden et al. to a reduction. Altogether a computational class of trace reconstruction problems begins to emerge.

Moreover, we note McGregor et al. [17] whose results prove that up to the differences between shifted and worst-case trace reconstruction, our reduction is essentially tight. This leads us to several interesting possibilities for future research on trace reconstruction.

First, many other versions of the trace reconstruction have been introduced over the last few years and analysed with an extension of the methods of De et al. and Nazarov and Peres [12, 20] for worst-case trace. If more of these analyses can be converted to reductions to the worst-case or average-case trace reconstruction problems, this would help to simplify the analysis of the many open questions in this field.

Secondly, it seems that the best known techniques for the worst-case trace reconstruction problem translate nicely to the shifted trace reconstruction problem, leading to the conjecture that the two are equivalent. A reduction between the two would help focus further research on this problem.

Finally, we note our extension of Chase's analysis to symmetry-insertion-deletion channels. This portion of our proof is complicated and would be difficult to extend to other settings. An important question for future research is whether there exists a simpler and more elegant analysis for these channels.

References

1 Alexandr Andoni, Constantinos Daskalakis, Avinatan Hassidim, and Sebastien Roch. Global alignment of molecular sequences via ancestral state reconstruction. *Stochastic Processes and their Applications*, 122(12):3852–3874, 2012.

2 Tugkan Batu, Sampath Kannan, Sanjeev Khanna, and Andrew McGregor. Reconstructing strings from random traces. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 910–918, USA, 2004. Society for Industrial and Applied Mathematics.

3 Peter Borwein and Tamás Erdélyi. Littlewood-type problems on subarcs of the unit circle. *Indiana University mathematics journal*, pages 1323–1346, 1997.

4 Peter Borwein, Tamás Erdélyi, and Géza Kós. Littlewood-type problems on [0, 1]. *Proceedings of the London Mathematical Society*, 79(1):22–46, 1999.

5 Joshua Brakensiek, Ray Li, and Bruce Spang. Coded trace reconstruction in a constant number of traces. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 482–493. IEEE, 2020.

6 Zachary Chase. New lower bounds for trace reconstruction. In *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, pages 627–643. Institut Henri Poincaré, 2021.

7 Zachary Chase. Separating words and trace reconstruction. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 21–31, 2021.

8 Zachary Chase and Yuval Peres. Approximate trace reconstruction of random strings from a constant number of traces. *arXiv preprint*, 2021. `arXiv:2107.06454`.

9 Xi Chen, Anindya De, Chin Ho Lee, Rocco A Servedio, and Sandip Sinha. Polynomial-time trace reconstruction in the smoothed complexity model. *ACM Transactions on Algorithms (TALG)*, 2020.

10 Mahdi Cheraghchi, Ryan Gabrys, Olgica Milenkovic, and Joao Ribeiro. Coded trace reconstruction. *IEEE Transactions on Information Theory*, 66(10):6084–6103, 2020.

11 Sami Davies, Miklós Z Rácz, Benjamin G Schiffer, and Cyrus Rashtchian. Approximate trace reconstruction: Algorithms. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 2525–2530. IEEE, 2021.

12 Anindya De, Ryan O'Donnell, and Rocco A Servedio. Optimal mean-based algorithms for trace reconstruction. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1047–1056, 2017.

13 Nina Holden, Robin Pemantle, Yuval Peres, and Alex Zhai. Subpolynomial trace reconstruction for random strings and arbitrary deletion probability. In *Conference On Learning Theory*, pages 1799–1840. PMLR, 2018.

14 Nina Holden, Robin Pemantle, Yuval Peres, and Alex Zhai. Subpolynomial trace reconstruction for random strings and arbitrary deletion probability. *Mathematical Statistics and Learning*, 2(3):275–309, 2020.

15 Thomas Holenstein, Michael Mitzenmacher, Rina Panigrahy, and Udi Wieder. Trace reconstruction with constant deletion probability and related results. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 389–398, 2008.

16 Akshay Krishnamurthy, Arya Mazumdar, Andrew McGregor, and Soumyabrata Pal. Trace reconstruction: Generalized and parameterized. *IEEE Transactions on Information Theory*, 67(6):3233–3250, 2021.

17 Andrew McGregor, Eric Price, and Sofya Vorotnikova. Trace reconstruction revisited. In *European Symposium on Algorithms*, pages 689–700. Springer, 2014.

18 Michael Mitzenmacher. A survey of results for deletion channels and related synchronization channels. *Probability Surveys*, 6:1–33, 2009. doi:10.1214/08-PS141.

19 Shyam Narayanan and Michael Ren. Circular trace reconstruction. *arXiv preprint*, 2020. arXiv:2009.01346.

20 Fedor Nazarov and Yuval Peres. Trace reconstruction with exp (o (n1/3)) samples. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1042–1046, 2017.

21 Yuval Peres and Alex Zhai. Average-case reconstruction for the deletion channel: subpolynomially many traces suffice. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 228–239. IEEE, 2017.

22 John M Robson. Separating strings with small automata. *Information processing letters*, 30(4):209–214, 1989.

23 Ittai Rubinstein. Average-case to (shifted) worst-case reduction for the trace reconstruction problem. *arXiv preprint*, 2022. arXiv:2207.11489.

The Support of Open Versus Closed Random Walks

Thomas Sauerwald
University of Cambridge, UK

He Sun
University of Edinburgh, UK

Danny Vagnozzi
University of Edinburgh, UK

Abstract

A *closed* random walk of length ℓ on an undirected and connected graph $G = (V, E)$ is a random walk that returns to the start vertex at step ℓ, and its properties have been recently related to problems in different mathematical fields, e.g., geometry and combinatorics (Jiang et al., Annals of Mathematics '21) and spectral graph theory (McKenzie et al., STOC '21). For instance, in the context of analyzing the eigenvalue multiplicity of graph matrices, McKenzie et al. show that, with high probability, the support of a closed random walk of length $\ell \geqslant 1$ is $\Omega(\ell^{1/5})$ on any bounded-degree graph, and leaves as an open problem whether a stronger bound of $\Omega(\ell^{1/2})$ holds for any regular graph.

First, we show that the support of a closed random walk of length ℓ is at least $\Omega(\ell^{1/2}/\sqrt{\log n})$ for any regular or bounded-degree graph on n vertices. Secondly, we prove for every $\ell \geqslant 1$ the existence of a family of bounded-degree graphs, together with a start vertex such that the support is bounded by $O(\ell^{1/2}/\sqrt{\log n})$. Besides addressing the open problem of McKenzie et al., these two results also establish a subtle separation between *closed* random walks and *open* random walks, for which the support on any regular (or bounded-degree) graph is well-known to be $\Omega(\ell^{1/2})$ for all $\ell \geqslant 1$. For irregular graphs, we prove that even if the start vertex is chosen uniformly, the support of a closed random walk may still be $O(\log \ell)$. This rules out a general polynomial lower bound in ℓ for all graphs. Finally, we apply our results on random walks to obtain new bounds on the multiplicity of the second largest eigenvalue of the adjacency matrices of graphs.

2012 ACM Subject Classification Theory of computation → Random walks and Markov chains

Keywords and phrases support of random walks, eigenvalue multiplicity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.103

Category Track A: Algorithms, Complexity and Games

Funding *He Sun*: EPSRC Early Career Fellowship (EP/T00729X/1)
Danny Vagnozzi: EPSRC Early Career Fellowship (EP/T00729X/1)

1 Introduction

A random walk on a graph is a Markov chain in which, starting from some vertex of an undirected graph $G = (V, E)$, the walk moves to one of the neighbors of the current vertex according to the transition matrix of G. As a fundamental stochastic process, random walks have been employed to model numerous mathematical and physical processes. In computer science, random walks have been widely applied in designing randomized and distributed algorithms. Classical examples range from algorithms for satisfiability, deciding connectivity to approximating the volume of convex bodies. The vast majority of research on random walks focuses on "open" random walks, as opposed to *closed* random walks, which are random walks of fixed length ℓ conditioned on being at the start vertex at step ℓ.

Following up on an earlier work by Jiang, Tidor, Yao, Zhang and Zhao [11], McKenzie, Rasmussen and Srivastava [16] develop a more systematic study of closed random walks on finite graphs. In particular, they study the support of closed random walks, which is the number of distinct vertices visited by a closed random walk of some length ℓ. As in [11], they then leverage their proven lower bounds on the support of closed random walks to upper bound the eigenvalue multiplicities via the trace method of graph matrices. One of the main ingredients in [16] are general lower bounds on the support of a random walk for any connected graph. For instance, for any bounded-degree graph, a closed random walk is shown to have support at least $\Omega(\ell^{1/5})$. In the same work, they also ask for sharper bounds:

> **Open Question 3 ([16]):** Let $d > 1$ be a fixed integer. Does there exist an $\alpha > 1/5$ such that for every connected d-regular graph G on n vertices and every vertex u of G, a closed random walk[1] of length $2\ell < n$ rooted at u has support $\Omega(\ell^{\alpha})$ in expectation? Is $\alpha = 1/2$ true? Does such a bound hold for simple random walks in general?

Note that the constant $\alpha = 1/2$ is a natural target, since any open random walk on a regular graph has support $\Omega(\ell^{1/2})$ (cf. [3, 7]), and this is matched by the n-cycle. Furthermore, also for closed random walks on n-cycles as well as the continuous analogue called Brownian bridges, the support can be shown to be $\Theta(\ell^{1/2})$. In fact, McKenzie et al. [16] states that "*we know of no example where the answer is* $o(\ell^{1/2})$".

In this paper, we address the Open Question 3 of McKenzie et al. [16]. Our first result proves a lower bound of almost $\Omega(\ell^{1/2})$, provided the random walk is sufficiently long. Here we use X^t to denote the vertex that a (lazy) random walk visits in time t, $\mathsf{supp}_{\mathbf{P}}(\ell)$ to denote the number of distinct vertices that a lazy random walk of length ℓ visits, and $\mathbf{P}$ to denote the associated transition matrix (see Section 2 for more on notation).

▶ **Theorem 1.1** (informal version of Theorem 3.1)**.** *Consider any connected, n-vertex graph $G = (V, E)$ with minimum degree δ and maximum degree Δ. Then, for a lazy random walk of length $\ell = O((\Delta/\delta) \cdot n^2 \log n)$ and any vertex $u \in V$, it holds that*

$$\mathbf{E}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \;\middle|\; X^{\ell} = X^0 = u\right] = \Omega\left(\frac{\ell^{1/2}}{\sqrt{(\Delta/\delta) \cdot \log n}}\right).$$

This theorem shows that, for any regular or bounded-degree graph, the support of a closed random walk of length ℓ is at least $\Omega\left(\ell^{1/2}/\sqrt{\log n}\right)$; this result improves the lower bound of $\Omega\left(\ell^{1/5}\right)$ from [16, Theorem 1.3] whenever $\ell \geqslant (\log n)^{5/3}$. Apart from the $\ell^{1/2}$-term in our lower bound, one may wonder about the $\sqrt{\log n}$-term, which intuitively does not seem tight. However, we can construct a family of graphs to demonstrate that this $\sqrt{\log n}$-term is needed, establishing that our lower bound is tight up to constant factors. Our upper bound result is summarized as follows:

▶ **Theorem 1.2** (informal version of Theorem 4.1)**.** *The following statements hold:*

- *For any $\ell = \Omega((\log n)^{7/2})$, there exists a family of bounded-degree n-vertex graph $G = (V, E)$ such that a lazy random walk of length ℓ starting at some vertex r satisfies*

$$\mathbf{E}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \;\middle|\; X^{\ell} = X^0 = r\right] = O\left(\frac{\ell^{1/2}}{\sqrt{\log n}}\right).$$

[1] The original formulation in [16] is stated for a randomly chosen closed walk, which has the same distribution as a closed random walk since G is regular.

Table 1 Overview of the lower and upper bounds on the support of closed and open random walk. Results highlighted in green are from this work. The lower bound of $\Omega(\ell^{1/5})$ holds for bounded-degree graphs. Note that, while our lower bounds hold for all such graphs and all ℓ, the upper bounds only hold for a specific graph (family) which depends on ℓ and ℓ may be additionally restricted.

Graph	Closed Random Walk		Open Random Walk	
	Lower Bound	Upper Bound	Lower B.	Upper B.
reg./bound. deg.	$\Omega(\ell^{1/5})$ [16]	$O(\ell^{5/14}), \ell \leqslant (\log n)^{7/2}$	$\Omega(\ell^{1/2})$ [7]	$O(\ell^{1/2})$
	$\Omega(\ell^{1/2}/\sqrt{\log n})$	$O(\ell^{1/2}/\sqrt{\log n}), \ell \geqslant (\log n)^{7/2}$		
arbitrary	–	$O(\log \ell), \ell = \Theta(\log n)$	$\Omega(\ell^{1/3})$ [7]	–

- *For any $\ell = O((\log n)^{7/2})$, there exists a family of bounded-degree n-vertex graph $G = (V, E)$ such that a lazy random walk of length ℓ starting at some vertex r satisfies*

$$\mathbf{E}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \;\middle|\; X^{\ell} = X^{0} = r\right] = O\left(\ell^{5/14}\right).$$

This result shows that the potential lower bound of $\Omega(\ell^{1/2})$ mentioned in [16] (Open Question 3) does not hold in general. In fact, on certain graphs the support of a closed lazy random walk can be as small as $O(\ell^{5/14})$; therefore the "right" exponent must be between $1/5$ and $5/14$, which is a strong separation from the exponent $1/2$ for open random walks. In contrast, when assuming a suitable lower bound on ℓ, the support of closed random walks is $\Theta(\ell^{1/2}/\sqrt{\log n})$, which is nearly the $\Theta(\ell^{1/2})$ bound for open random walks. Table 1 lists the known upper and lower bounds for closed and open random walks; the interplay of these bounds for regular and bounded-degree graphs is further illustrated in Figure 1.

We now proceed to study closed random walks, with a focus on (highly) irregular graphs. McKenzie et al. [16] proves that the support of a randomly chosen closed walk can be as small as $O(\log \ell)$, if starting from a specific vertex (note that a *randomly chosen closed walk* will have a different distribution to a *closed random walk*, unless the graph is regular). Here we provide a similar upper bound for the support of a closed random walk on a family of irregular graphs. Interestingly, this upper bound even holds if we assume that the start vertex is chosen uniformly at random. This lower bound also establishes an "exponential discrepancy" on the support of closed random walks versus open random walks on irregular graphs: while the support of open random walks is known to be at least $\ell^{1/3}$ (cf. [3]) for any graph and any start vertex, one can construct graphs for which the support is only $O(\log \ell)$ for closed random walks.

▶ **Theorem 1.3** (informal version of Theorem 4.5). *There exists a family of connected, n-vertex graphs $G = (V, E)$ such that a lazy random walk of some length $\ell = \Theta(\log n)$ that starts from a vertex chosen uniformly at random satisfies*

$$\mathbf{E}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \mid X^{\ell} = X^{0}\right] = O(\log \ell).$$

Consequently, there is a vertex $r \in V$ such that

$$\mathbf{E}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \mid X^{\ell} = X^{0} = r\right] = O(\log \ell).$$

As side results, we apply our lower bounds on the support of closed random walks to obtain new eigenvalue multiplicity bounds for certain classes of graphs. We state one result below, and an additional result is presented in Section 5.

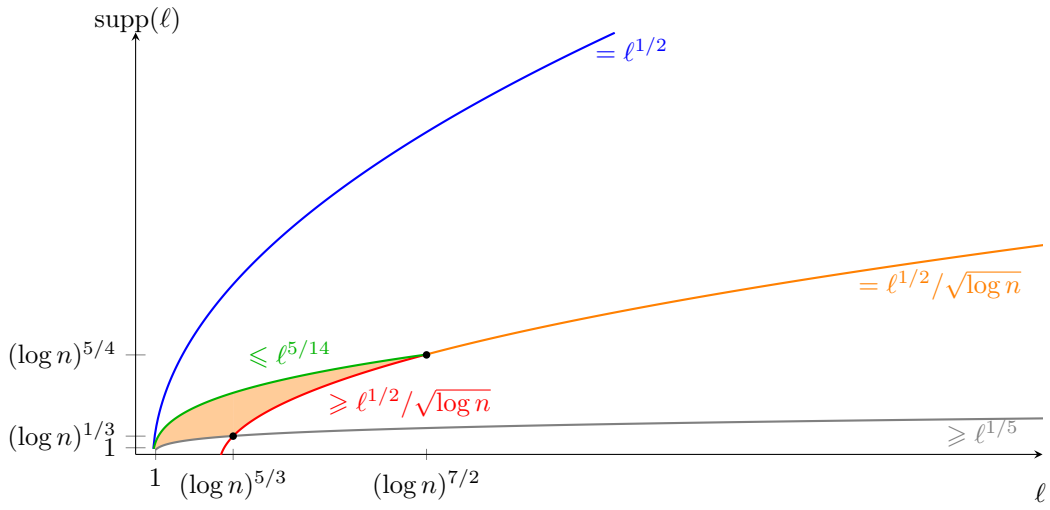


Figure 1 Comparison of the (worst case) support of open random walks to that of closed random walks for bounded-degree graphs. We prove for any $\ell = \Omega((\log n)^{7/2})$ (and $\ell \leqslant n^{1/5}$) that the asymptotic worst case bound for closed random walks equals $\ell^{1/2}/\sqrt{\log n}$, while for $\ell = O((\log n)^{7/2})$ the correct asymptotic bound is confined in the orange area. Since the lower bound $\ell^{1/2}/\sqrt{\log n}$ is tight for any $\ell \geqslant (\log n)^{7/2}$ but becomes trivial for $\ell \leqslant \log n$, we can infer that there must be a phase transition in the interval $[(\log n)^{5/3}, (\log n)^{7/2}]$.

▶ **Theorem 1.4** (informal version of Theorem 5.1). *Consider any connected, n-vertex graph $G = (V, E)$ with minimum degree δ and maximum degree Δ, such that its second largest eigenvalue λ of $\mathbf{P}$ satisfies $|1 - \lambda| = O(\frac{\delta}{\Delta} \cdot \frac{1}{\log^4 n})$. Then, the number of eigenvalues of $\mathbf{P}$ in the range $\left[\left(1 - \frac{\delta}{32c\Delta \cdot \log^5 n}\right) \cdot \lambda, \lambda\right]$ is at most $O(\frac{n}{\log n})$.*

Our eigenvalue multiplicity results are in general incomparable with the ones in [16], and have their own features. For instance, our eigenvalue multiplicity bound above is based on the spectral gap condition of $\mathbf{P}$; as such, this result brings a new connection between the eigenvalue multiplicity and the eigenvalue distribution, the relationship of which is informally established in spectral graph theory through the high-order Cheeger inequalities [12].

1.1 Further Related Work

There is a plethora of works on random walks on graphs, which often revolves around quantities such as mixing times, hitting times and cover times [14]. In particular, properties of short random walks such as their support or return probabilities have found numerous applications in the analysis of randomness amplification [10], space-efficient graph exploration with random walks [2, 7] and the voter model [17]. These concepts have been also successfully applied to algorithmic tasks such as estimating network sizes and densities [4], load balancing [19], information spreading [8], property testing [6] and clustering [20]. In addition to these applications, many of the random walk quantities have close connections to other mathematical areas, such as geometry, group theory, electrical networks and spectral graph theory (see [13, 14] for more details).

More closely related to this work, Benjamini, Izkovsky and Kesten [5] analyze the support of closed random walks on various finite and infinite graphs, with a focus on vertex-transitive and Cayley graphs. In particular, for high-girth expander graphs, they prove that the support

of closed random walks is linear in their length. While there are some studies of closed random walks on finite or infinite graphs with special geometry or symmetries as well as studies of Brownian bridges in continuous space, much less is known about the support of closed random walks on finite graphs (without additional assumptions on their symmetry or geometry).

One specific motivation for studying closed random walks is the relation to the second eigenvalue multiplicity of the normalized adjacency matrix of graphs, as established recently in [11, 15, 16]. By examining the support of closed random walks of short length, it is shown that, for any connected graph G of maximum degree Δ, the second eigenvalue multiplicity of G's normalized adjacency matrix is $\widetilde{O}\left(n \cdot \Delta^{7/5}/\log^{1/5} n\right)$, where the notation $\widetilde{O}(\cdot)$ suppresses $\operatorname{poly}\log\log(n)$ terms. Haiman, Schildkraut, Zhang and Zhao [9] show the existence of infinitely many connected 18-regular graphs G on n vertices with the second largest eigenvalue multiplicity at least $n^{2/5}-1$, and the existence of infinitely many connected n-vertex graphs with maximum degree 4 and second eigenvalue multiplicity at least $\sqrt{n/\log_2 n}$.

1.2 Organization

The remaining part of the paper is organized as follows. Section 2 introduces our notation and provides some basic lemmas used in this work. We derive our lower bound on the support of closed random walks in Section 3, and the proofs of our two upper bound results are presented in Section 4. Finally, we employ our random walk results to analyze the eigenvalue multiplicity problem in Section 5. We summarize our results and point to some open questions in Section 6.

2 Definitions and Preliminaries

All graphs in this paper will be undirected. For any vertex $u \in V$ of a graph $G=(V,E)$, the degree of u is denoted by $\deg(u)$; the maximum and minimum degrees of G are denoted by Δ and δ, respectively. For any $u \in V$ and $\ell \in \mathbb{N}$, let $B_{\leqslant \ell/2}(u) \triangleq \{v \in V\colon \operatorname{dist}(u,v) \leqslant \ell/2\}$. For any integer k, let $[k] \triangleq \{1,\ldots,k\}$.

We use $\mathbf{Q}$ to represent the transition matrix of a non-lazy random walk in G defined by $\mathbf{Q}_{u,v} = \frac{1}{\deg(u)}$ if $\{u,v\} \in E(G)$ and $\mathbf{Q}_{u,v}=0$ otherwise. We use $\mathbf{P}$ to represent the lazy random walk matrix of G, where $\mathbf{P}_{u,u} = \frac{1}{2}$ for all $u \in V$, $\mathbf{P}_{u,v} = \frac{1}{2\deg(u)}$ if $\{u,v\} \in E(G)$, and $\mathbf{P}_{u,v}=0$ otherwise. We use $\mathbf{A}$ to represent the adjacency matrix of G, and $\mathbf{D}$ to represent the diagonal matrix of degrees of G. For any matrix $\Diamond \in \{\mathbf{P},\mathbf{Q}\}$ of size $n \times n$, the eigenvalues of $\Diamond$ are denoted by $\lambda_1(\Diamond) \geqslant \ldots \geqslant \lambda_n(\Diamond)$. We define $M_\Diamond[x,y]$ to be the number of eigenvalues of the matrix $\Diamond$ in the interval $[x,y]$.

For any (non-)lazy random walk that starts from a vertex (or possibly distribution over vertices) $X^0 \in V$, we use X^t to denote the vertex that the random walk reaches at step t for any $t \geqslant 0$, and define

$$p^t_{u,v} \triangleq \mathbf{Pr}\left[X^t = v \;\middle|\; X^0 = u\right]$$

to be the probability that a random walk that starts from u is located at v after t steps; if the start vertex is deterministic and clear from the context, we sometimes omit the conditioning on $X^0=u$. Further, we write $X^0 \sim \mathcal{U}$ if the start vertex of the random walk is chosen uniformly at random from V. We define the support of a random walk of length ℓ to be

$$\mathsf{supp}_\Diamond(\ell) \triangleq \left|\left\{X^i \mid i \leqslant \ell\right\}\right|,$$

where we use $\Diamond \in \{\mathbf{P}, \mathbf{Q}\}$ to distinguish a lazy random walk from a non-lazy one. It is well-known that, for a lazy random walk with loop probability $1/2$ on a connected graph G, it holds for all $u, v \in V$ that $\lim_{t\to\infty} p^t_{u,v} = \pi(v)$, where $\pi \in \mathbb{R}^n_{\geqslant 0}$ is the stationary distribution defined by $\pi(u) \triangleq \frac{\deg(u)}{2\cdot|E(G)|}$ for any $u \in V$.

We list two lemmas used in our analysis. Our first lemma allows us to translate bounds on the support from lazy random walks to non-lazy ones (and vice versa) at the cost of a small constant factor.

▶ **Lemma 2.1.** *For any graph $G = (V, E)$ and any fixed $\ell \geqslant 0$, it holds that $\mathsf{supp}_{\mathbf{Q}}(\ell)$ is stochastically larger than $\mathsf{supp}_{\mathbf{P}}(\ell)$. Moreover, we have for any $x \geqslant 0$ that*

$$\mathbf{Pr}\left[\,\mathsf{supp}_{\mathbf{P}}(4\cdot\ell) \geqslant x\,\right] \geqslant \frac{1}{2}\cdot\mathbf{Pr}\left[\,\mathsf{supp}_{\mathbf{Q}}(\ell) \geqslant x\,\right],$$

and thus

$$\mathbf{E}\left[\,\mathsf{supp}_{\mathbf{P}}(4\cdot\ell)\,\right] \geqslant \frac{1}{2}\cdot\mathbf{E}\left[\,\mathsf{supp}_{\mathbf{Q}}(\ell)\,\right].$$

The next lemma gives a lower bound on the return probability of a random walk. While there are a number of results upper bounding the return probability of a random walk (e.g., [13, 14, 18]),to the best of our knowledge much less is known in terms of lower bounds.

▶ **Lemma 2.2.** *For any connected, n-vertex graph $G = (V, E)$ and a lazy random walk with transition matrix $\mathbf{P}$, it holds for any vertex $u \in V$ and step $t \geqslant 0$ that*

$$p^t_{u,u} \geqslant \pi(u) = \frac{\deg(u)}{2|E|} \geqslant \frac{1}{n^2}.$$

The same also holds for non-lazy random walks with transition matrix $\mathbf{Q}$, if additionally t is even. Furthermore, if G has minimum degree δ and maximum degree Δ, we also have for any $t \geqslant 2$,

$$p^t_{u,u} \geqslant \frac{\delta^t}{\Delta^{t+1}}\cdot\frac{1}{|B_{\leqslant t/2}(u)|},$$

and the same inequality holds for the transition matrix $\mathbf{Q}$ if $t \geqslant 2$ is even.

3 A Lower Bound on the Support of Closed Random Walks

This section provides lower bounds on the support of closed random walks, in particular, we will prove Theorem 1.1. We first present a more detailed formulation of Theorem 1.1, in which all the hidden constants are stated precisely.

▶ **Theorem 3.1.** *Consider any connected, n-vertex graph $G = (V, E)$ with minimum degree δ and maximum degree Δ. Then there is a constant $c \geqslant 1$ independent of n, such that a random walk of length $\ell \leqslant 512\frac{\Delta}{\delta}cn^2\log n$ satisfies for any $u \in V$ that*

$$\mathbf{E}\left[\,\mathsf{supp}_{\Diamond}(\ell)\;\middle|\;X^\ell = X^0 = u\,\right] \geqslant \frac{1}{2}\cdot\left\lfloor\sqrt{\frac{1}{576c}\cdot\frac{\delta}{\Delta}\cdot\frac{\ell}{\log n}}\right\rfloor,$$

where $\Diamond \in \{\mathbf{P}, \mathbf{Q}\}$ (in the case of $\Diamond = \mathbf{Q}$, the length ℓ needs additionally to be even). Furthermore, for any $\mu \in [1, \ell]$ satisfying $\ell \leqslant 32\frac{\Delta}{\delta}cn^2\mu$, it holds that

$$\mathbf{Pr}\left[\,\mathsf{supp}_{\Diamond}(\ell) \leqslant \left\lfloor\sqrt{\frac{1}{64c}\cdot\frac{\delta}{\Delta}\cdot\frac{\ell}{\mu}}\right\rfloor\;\middle|\;X^\ell = X^0 = u\,\right] \leqslant (5/8)^{\mu/2}\cdot\frac{1}{p^\ell_{u,u}},$$

where ℓ is required to be even if $\Diamond = \mathbf{Q}$.

To examine the significance of this result, notice that it is shown in [16, Theorem 1.3] that

$$\mathbf{Pr}\left[\mathsf{supp}_{\mathbf{Q}}(\ell) \leqslant s \;\middle|\; X^{\ell} = u\right] \leqslant \exp\left(-\frac{\ell}{130\Delta^7 s^4}\right),$$

if $s \leqslant \frac{1}{4}\left(\frac{\ell}{2\Delta^7 \log \Delta}\right)^{1/5}$ and ℓ is even. In comparison to their result, our bound is not affected by the density, but only by the degree ratio Δ/δ. Regarding the expected support, it follows that for bounded-degree graphs, the first statement in Theorem 3.1 improves on [16, Theorem 1.3] for moderately longer walks, i.e., $\ell \geqslant (\log n)^{5/3}$, whereas it is worse for $\ell \leqslant (\log n)^{5/3}$.

Next, we present the lemmas needed to prove Theorem 3.1. The first lemma (Lemma 3.3) lower bounds the support of *open* random walks, and relies on the following result by Feige [7] bounding the expected time until a certain number of distinct vertices are visited; similar results are also shown in [3].

▶ **Lemma 3.2** ([7, Theorem 4]). *Consider any connected, n-vertex graph $G = (V, E)$ with minimum degree δ and maximum degree Δ. For any $s \in [1, n]$, let $T(s)$ be the time until a random walk visits s distinct vertices. Then there is a constant $c \geqslant 1$ (independent of s and n), such that for any lazy random walk and any vertex $u \in V$,*

$$\mathbf{E}\left[T(s) \mid X^0 = u\right] \leqslant c \cdot \left(s + \frac{s^2}{\delta} \cdot \min\{s, \Delta\}\right).$$

▶ **Lemma 3.3.** *Consider any connected, n-vertex graph $G = (V, E)$, and a random walk of length ℓ for some $\ell \triangleq 32 \cdot \lceil c \cdot \frac{\Delta}{\delta} \cdot s^2 \rceil$, where $1 \leqslant s \leqslant n$ is any integer. Then, there is some constant $c \geqslant 1$ (independent of n and ℓ), such that it holds for any start vertex $u \in V$ that*

$$\mathbf{Pr}\left[\mathsf{supp}_{\Diamond}(\ell) \geqslant s \mid X^0 = u\right] \geqslant \frac{3}{8},$$

where $\Diamond \in \{\mathbf{P}, \mathbf{Q}\}$.

Proof. First of all, we note that the constant c involved in this result is the constant from Lemma 3.2. With this, we first prove the result for non-lazy random walks, i.e., for $\Diamond = \mathbf{Q}$. Let

$$\widetilde{\ell} \triangleq 8 \cdot \left\lceil c \cdot \frac{\Delta}{\delta} \cdot s^2 \right\rceil.$$

Recall that $T(s)$ is the stopping time until a walk has visited s different vertices. By Lemma 3.2, we have

$$\mathbf{E}\left[T(s) \;\middle|\; X^0 = u\right] \leqslant c \cdot \left(s + \frac{\Delta}{\delta} s^2\right) \leqslant 2 \cdot \left\lceil c \cdot \frac{\Delta}{\delta} s^2 \right\rceil = \frac{1}{4} \cdot \widetilde{\ell}.$$

By Markov's inequality, it holds that

$$\mathbf{Pr}\left[T(s) \geqslant \widetilde{\ell} \;\middle|\; X^0 = u\right] \leqslant \mathbf{Pr}\left[T(s) \geqslant 4 \cdot \mathbf{E}\left[T(s)\right] \;\middle|\; X^0 = u\right] \leqslant \frac{1}{4}.$$

Note that $T(s) \leqslant \widetilde{\ell}$ is equivalent to $\mathsf{supp}(\widetilde{\ell}) \geqslant s$, and this gives us that

$$\mathbf{Pr}\left[\mathsf{supp}(\widetilde{\ell}) \geqslant s \;\middle|\; X^0 = u\right] \geqslant \frac{3}{4} \geqslant \frac{3}{8}, \tag{1}$$

which completes the proof in case of $\Diamond = \mathbf{Q}$. For $\Diamond = \mathbf{P}$, the statement follows immediately from (1), and the second statement of Lemma 2.1, since $\ell = 4 \cdot \widetilde{\ell}$. ◀

We remark that the bound presented in Lemma 3.3 is essentially tight: if one takes a random walk of length ℓ on a path (or cycle), then by the Central Limit Theorem the probability that the walk visits at least $\varepsilon \cdot \sqrt{\ell}$ vertices can be upper bounded by $1 - \delta$ for some $\delta = \delta(\varepsilon) > 0$; in particular, this probability can be bounded independently of ℓ.

Proof of Theorem 3.1. Fix an arbitrary start vertex $u \in V$ as $X^0 = u$. We split the random walk of length ℓ into consecutive sections of length $\ell' \triangleq \lceil \ell/(9 \log n) \rceil$. Without loss of generality, we assume that $\ell \geqslant 576 \frac{\Delta}{\delta} c \log n$, since otherwise the statement holds trivially. Given $\ell \geqslant 576 \frac{\Delta}{\delta} c \log n$ (and $c \geqslant 1$), we have $\ell' \leqslant \ell/(8 \log n)$. Hence, it would take a random walk (at least) $8 \log n$ sections before reaching step ℓ. Next we define the integer

$$\gamma \triangleq \left\lfloor \sqrt{\frac{1}{64c} \cdot \frac{\delta}{\Delta} \cdot \ell'} \right\rfloor,$$

with c being the constant from Lemma 3.2. We make the following observations about the range of γ:

- Since $\ell' \leqslant \ell/(8 \log n)$ and by the precondition $\ell \leqslant 512 \frac{\Delta}{\delta} c n^2 \log n$, we have

$$\gamma \leqslant \sqrt{\frac{1}{64c} \cdot \frac{\delta}{\Delta} \cdot \frac{\ell}{8 \log n}} \leqslant n.$$

- Similarly, since $\ell' \geqslant \ell/(9 \log n)$ and $\ell \geqslant 576 \frac{\Delta}{\delta} c \log n$, we have

$$\gamma \geqslant \left\lfloor \sqrt{\frac{1}{64c} \cdot \frac{\delta}{\Delta} \cdot \frac{\ell}{9 \log n}} \right\rfloor \geqslant 1.$$

In conclusion, γ is an integer between 1 and n. Notice that the definition of γ implies

$$\gamma \leqslant \sqrt{\frac{1}{64c} \cdot \frac{\delta}{\Delta} \cdot \ell'},$$

and thus $\ell' \geqslant 64c \cdot \frac{\Delta}{\delta} \cdot \gamma^2$. Since $\gamma \geqslant 1$, we have $c \cdot \frac{\Delta}{\delta} \cdot \gamma^2 \geqslant 1$ and

$$c \cdot \frac{\Delta}{\delta} \cdot \gamma^2 \geqslant \frac{1}{2} \left\lceil c \cdot \frac{\Delta}{\delta} \cdot \gamma^2 \right\rceil.$$

This implies that

$$\ell' \geqslant 64c \cdot \frac{\Delta}{\delta} \cdot \gamma^2 \geqslant 32 \cdot \left\lceil c \cdot \frac{\Delta}{\delta} \cdot \gamma^2 \right\rceil.$$

We now apply Lemma 3.3 (with $s = \gamma$) and conclude

$$\mathbf{Pr}\left[\mathsf{supp}_{\Diamond}(\ell') \geqslant \gamma \;\middle|\; X^0 = u \right] \geqslant \mathbf{Pr}\left[\mathsf{supp}_{\Diamond}\left(32 \cdot \left\lceil c \cdot \frac{\Delta}{\delta} \cdot \gamma^2 \right\rceil \right) \geqslant \gamma \;\middle|\; X^0 = u \right] \geqslant \frac{3}{8},$$

which holds for any start vertex $u \in V$. Therefore, by considering the at least $8 \log n$ consecutive sections of length ℓ' each, and using the Markov property we have

$$\begin{aligned} \mathbf{Pr}\left[\mathsf{supp}_{\Diamond}(\ell) < \gamma \;\middle|\; X_0 = u \right] &\leqslant \left(\max_{v \in V} \mathbf{Pr}\left[\mathsf{supp}_{\Diamond}(\ell') < \gamma \;\middle|\; X_0 = v \right] \right)^{8 \log n} \leqslant \left(\frac{5}{8} \right)^{8 \log n} \\ &\leqslant n^{-3}, \end{aligned}$$

since $(5/8)^8 \leqslant \mathrm{e}^{-3}$.

By Lemma 2.2, it holds for a lazy random walk (i.e., $\Diamond = \mathbf{P}$) starting with any $u \in V$ and integer $\ell \geqslant 0$ that

$$\mathbf{Pr}\left[X^\ell = u \;\middle|\; X^0 = u \right] \geqslant \pi(u) \geqslant \frac{\delta}{\Delta \cdot n};$$

we also know that the same statement holds for a non-lazy random walk (i.e., $\Diamond = \mathbf{Q}$) and an even value of ℓ. Hence,

$$\begin{aligned}
\mathbf{Pr}\left[\mathsf{supp}_\Diamond(\ell) < \gamma \;\middle|\; X^0 = X^\ell = u \right] &= \frac{\mathbf{Pr}\left[\mathsf{supp}_\Diamond(\ell) < \gamma \cap X^0 = X^\ell = u \right]}{\mathbf{Pr}\left[X^0 = X^\ell = u \right]} \\
&\leqslant \frac{\mathbf{Pr}\left[\mathsf{supp}_\Diamond(\ell) < \gamma \right]}{\mathbf{Pr}\left[X^\ell = u \;\middle|\; X^0 = u \right]} \\
&\leqslant \frac{n^{-3}}{n^{-2}} = n^{-1}. \qquad (2)
\end{aligned}$$

Consequently,

$$\begin{aligned}
\mathbf{E}\left[\mathsf{supp}_\Diamond(\ell) \;\middle|\; X^0 = X^\ell = u \right] &\geqslant \gamma \cdot \mathbf{Pr}\left[\mathsf{supp}_\Diamond(\ell) \geqslant \gamma \;\middle|\; X^0 = X^\ell = u \right] \\
&\geqslant \gamma \cdot \left(1 - n^{-1}\right) \\
&\geqslant \frac{1}{2} \cdot \left\lfloor \sqrt{\frac{1}{64c} \cdot \frac{\delta}{\Delta} \cdot \frac{\ell}{9 \log n}} \right\rfloor = \frac{1}{2} \cdot \left\lfloor \sqrt{\frac{1}{576c} \cdot \frac{\delta}{\Delta} \cdot \frac{\ell}{\log n}} \right\rfloor,
\end{aligned}$$

which proves the first statement.

We proceed to the proof of the second statement, which essentially uses the same argument as before but with different parameters. First note that we may assume $\ell \geqslant 64c \cdot \frac{\Delta}{\delta} \cdot \mu$, since otherwise the statement holds trivially. In particular, this implies $\ell/\mu \geqslant 1$, so if we split the random walk of length ℓ into consecutive sections of length $\ell' \triangleq \lceil \ell/\mu \rceil$, it holds that $\ell' \leqslant 2\ell/\mu$. Hence there are at least $\mu/2$ consecutive sections before reaching step ℓ. We define

$$\gamma \triangleq \left\lfloor \sqrt{\frac{1}{64c} \cdot \frac{\delta}{\Delta} \cdot \ell'} \right\rfloor,$$

and rearranging this implies

$$\ell' \geqslant 64c \cdot \frac{\Delta}{\delta} \cdot \gamma^2 \geqslant 32 \cdot \left\lceil c \cdot \frac{\Delta}{\delta} \cdot \gamma^2 \right\rceil.$$

Again, we examine the range of γ:

- Since $\ell' \leqslant 2\ell/\mu$ and by the precondition $\ell \leqslant 32 \frac{\Delta}{\delta} c n^2 \mu$, we have $\gamma \leqslant n$.
- Since $\ell' \geqslant \ell/\mu$ and $\ell \geqslant 64c \cdot \frac{\Delta}{\delta} \cdot \mu$, we have $\gamma \geqslant 1$.

In conclusion, γ is an integer between 1 and n. By Lemma 3.3 (with $s = \gamma$), it holds that

$$\mathbf{Pr}\left[\mathsf{supp}_\Diamond(\ell') \geqslant \gamma \;\middle|\; X^0 = u \right] \geqslant \mathbf{Pr}\left[\mathsf{supp}_\Diamond\left(32 \left\lceil c \cdot \frac{\Delta}{\delta} \cdot \gamma^2 \right\rceil \right) \geqslant \gamma \;\middle|\; X^0 = u \right] \geqslant \frac{3}{8},$$

and, as in the proof of the first statement,

$$\mathbf{Pr}\left[\mathsf{supp}_\Diamond(\ell) \leqslant \gamma \;\middle|\; X^0 = u \right] \leqslant (5/8)^{\mu/2}.$$

Finally, we apply the same argument as in (2) to conclude that

$$\mathbf{Pr}\left[\mathsf{supp}_\Diamond(\ell) \leqslant \gamma \;\middle|\; X^0 = X^\ell = u \right] \leqslant (5/8)^{\mu/2} \cdot \frac{1}{p^\ell_{u,u}}.$$

◀

4 Upper Bounds on the Support of Closed Random Walks

This section studies upper bounds on the support of closed random walks, by examining certain "worst-case" graphs. The section is structured as follows: we first study a family of bounded-degree graphs, and give the proof of Theorem 1.2 in Section 4.1. We present a more formal statement of Theorem 1.3, and prove the statement in Section 4.2.

4.1 Proof of Theorem 1.2

We first present a more detailed formulation of Theorem 1.2, in which all the hidden constants are stated precisely.

▶ **Theorem 4.1.** *There is a constant $C \geqslant 1$, such that for any pair of integers β being a power of 2 and ℓ with $C \leqslant \ell \leqslant \beta^{1/5}$ the following holds: there is a connected, n-vertex graph G satisfying $n \in [2\beta + 1, 2\beta + \beta^{1/10} - 1]$, $\Delta = 3$, and some vertex $r \in V(G)$, such that it holds for a random walk of length ℓ that starts at r that*

$$\mathbf{E}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \;\middle|\; X^{\ell} = X^{0} = r\right] \leqslant 3 \cdot \ell^{1/2-\varepsilon},$$

where $\varepsilon \triangleq \min\left(1/2 \cdot \log(1/16 \cdot \log_{12} \beta)/\log \ell, 1/7\right)$.

We first construct the family of graphs G used in the proof. Given two integers $p \geqslant 1$ and $D \geqslant 1$, our constructed graph $G = G[p, D]$ is based on the following two graphs:

- let $G_1 = (V_1, E_1)$ be a path graph of length $p + 1$, where $V_1 = \{v_0, v_1, \ldots, v_p\}$ and $E_1 = \{\{v_i, v_{i+1}\} \mid 0 \leqslant i < p\}$;
- let $G_2 = (V_2, E_2)$ be a complete binary tree over $D \geqslant 1$ levels labelled $0, 1, \ldots, D-1$. Hence, G_2 has $2^{D+1} - 1$ vertices.

We set the root of the binary tree to be vertex $z = v_0$, and let G be the union of the graphs G_1 and G_2; see Figure 2 for an illustration of our considered graph. Notice that G has $n = p + (2^{D+1} - 1)$ vertices, as vertex $z = v_0$ appears in both G_1 and G_2. Since all vertices of graph G have degree one, two or three, we have $\Delta = 3$.

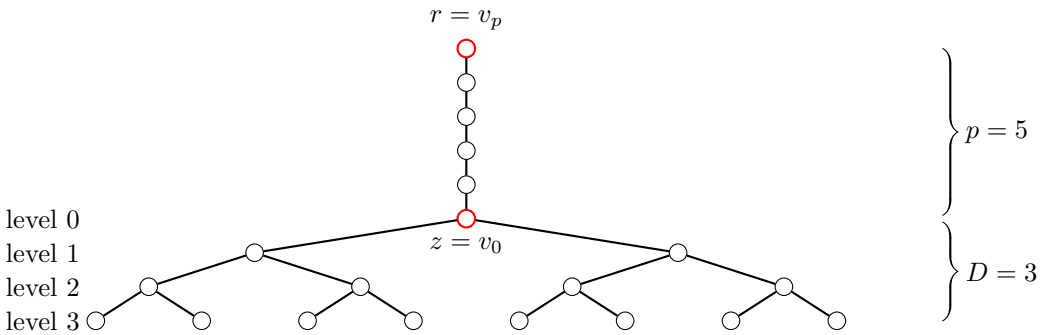


Figure 2 The construction of the graph $G = G[p, D]$ for $p = 5$, $D = 3$. We have $2^4 - 1 = 15$ vertices on the binary tree, and the total number of vertices of G is $n = 5 + 16 - 1 = 20$.

Now we explain the intuition behind our construction. We study a closed random walk from r of length p^{2+c} for some suitably small constant $c > 0$. On one hand, if the closed random walk remains only on the path, then the support of this walk is at most $p \ll (p^{2+c})^{1/2}$. On the other hand, once the random walk leaves the path, the walk is likely to "get lost" in

the binary tree, and the probability for the walk to return to $r = v_p$ is very small. Hence, if we sample from the space of all closed random walks, there is a strong bias towards those walks that never leave the path (and thus have necessarily small support).

Now we turn this intuition into a formal proof. Recall that the *level* of a vertex in the binary tree is the distance to the root vertex z, and recall that the maximum level is equal to the depth D.

▶ **Lemma 4.2.** *Consider a lazy random walk on $G = G[p, D]$ starting at r. For any integer $\ell \geqslant 0$, the following holds with probability at least $1 - \exp(-\ell/288)$: if the random walk makes at least ℓ transitions within the binary tree, it reaches at least once a vertex which is at level $\min(D, \ell/12)$.*

This lemma is quite intuitive, as there is a strong drift on the binary tree to increase the distance to the root, and one can exploit this using Hoeffding's inequality. Next, we present a simple fact of random walks on binary trees.

▶ **Lemma 4.3.** *Consider a lazy random walk in a complete binary tree with levels $0, 1, \ldots, D$ starting at a vertex which has distance $k \in [1, D]$ from the root. Then the probability that the walk reaches the root within ℓ steps is upper bounded by $\ell \cdot 2^{-k}$.*

Lemmas 4.2 and 4.3 together establish the intuitive fact that, once the random walks makes sufficiently many transitions in the binary tree, it is unlikely to return to the root of the tree within a small number of steps.

Next, we consider a lazy random walk on a path with vertices $0, 1, \ldots, p$, starting from p, with the special property that the random walk "gets killed" once it reaches the other endpoint 0. The following lemma lower bounds the probability of the random walk "surviving" after $\gamma \cdot p^2$ steps, i.e., the probability that a random walk does not reach the other endpoint before step $\gamma \cdot p^2$.

▶ **Lemma 4.4.** *Consider a lazy random walk on the integers $\{0, 1, \ldots, p\}$, starting at vertex p, such that the random walk gets killed after reaching vertex 0. More precisely, we define the following $p \times p$ matrix $\mathbf{R}$:* [2]

$$\mathbf{R}_{i,j} = \begin{cases} \frac{1}{2} & \text{if } j = i \in \{1, \ldots, p\}, \\ \frac{1}{2} & \text{if } i = p,\ j = p-1, \\ \frac{1}{4} & \text{if } j = i-1,\ 1 < i < p, \\ \frac{1}{4} & \text{if } j = i+1,\ 1 \leqslant i < p. \end{cases}$$

Let $r^t_{p,\cdot}$ be the probability distribution[3] *of this t-step random walk, when starting at vertex p. Then, it holds for any integer $\gamma \geqslant 1$ that*

$$r^{\gamma \cdot p^2}_{p,p} \geqslant \frac{1}{2p} \cdot 12^{-8 \cdot \gamma}.$$

With the previous lemmas at hand, we are now ready to prove Theorem 4.1.

[2] This matrix is called a "substochastic" matrix [1, Section 3.6.5, page 95]. Note that this is *not* a transition matrix, since from state 1 with probability 1/4 the walk gets killed.

[3] Since the random walk gets killed at vertex 0, $\|r^t_{p,\cdot}\|_1$ may generally not be equal to 1, but it is upper bounded by 1.

Proof of Theorem 4.1. Given the two integers ℓ and β being a power of 2, we will now instantiate a graph $G = G[p, D]$. Firstly, the length of the path is $p \triangleq \ell^\delta$ and $\delta \triangleq 1/2 - \varepsilon$; recall that

$$\varepsilon = \min\left(1/2 \cdot \log(1/16 \cdot \log_{12} \beta)/\log \ell, 1/7\right).$$

Secondly, the depth of the binary tree is $D \triangleq \log_2 \beta$ (so in turn, the binary tree has $2^{\log_2(\beta)+1} - 1$ vertices). Hence, the total number of vertices in G is

$$n = \ell^\delta + 2^{\log_2(\beta)+1} - 1,$$

which is at least $2\beta + 1$ and at most $2\beta + \beta^{1/10}$, as $\ell \leqslant \beta^{1/5}$.

Our objective is to show that the expected support of a closed random walk of length ℓ starting at vertex r is at most $3 \cdot \ell^\delta = 3 \cdot \ell^{1/2-\varepsilon}$. We first upper bound the probability of the event that a random walk (not necessarily closed) visits at least $2\ell^\delta + 1$ vertices in ℓ steps and then returns to r.

In the following, let us define the following events and stopping time:

1. The event $\mathcal{A} := \left\{\mathsf{supp}(\ell) \geqslant 2\ell^\delta + 1\right\}$, meaning the random walk of length ℓ visits at least $2\ell^\delta + 1$ vertices.
2. The event $\mathcal{B} := \left\{X^\ell = r\right\}$, meaning the random walk is at the start vertex at time ℓ.
3. The event $\mathcal{C}$ which occurs if the random walk of length ℓ makes at least ℓ^δ transitions on the binary tree.
4. The stopping time τ, which is the number of transitions on the binary tree until a vertex at level $\min\{D, \ell^\delta/12\}$ is reached for the first time.
5. The event $\mathcal{C}(\tau)$ which occurs if the random walk of length ℓ makes at least τ transitions on the binary tree.

In order to visit at least $2\ell^\delta+1$ vertices, the walk needs to visit at least $2\ell^\delta+1-(\ell^\delta+1) = \ell^\delta$ vertices on the tree (excluding the vertex $z = v_0$), since there are only $\ell^\delta + 1$ vertices on the path. Hence we have

$$\mathcal{A} \subseteq \mathcal{C}. \tag{3}$$

Furthermore, we have by Lemma 4.2 (applied to a random walk on the binary tree with ℓ^δ transitions),

$$\mathbf{Pr}\left[\tau \leqslant \ell^\delta\right] \geqslant 1 - \exp(-\ell^\delta/288). \tag{4}$$

Furthermore, let $T(\tau)$ be the time-step of the random walk on G when the τ-th transition on the binary tree is made; so, $T(\tau) \geqslant \tau$. Then,

$$\mathbf{Pr}\left[X^\ell = r \mid \mathfrak{F}^{T(\tau)}, T(\tau) \leqslant \ell\right] \leqslant \ell \cdot \max\left(2^{-D}, 2^{-\ell^\delta/12}\right), \tag{5}$$

since by Lemma 4.3, the random walk starting from a vertex at level level $\min\{D, \ell^\delta/12\}$ in the binary tree does not even reach $z = v_0$ within ℓ additional steps (and therefore cannot reach the vertex r at step ℓ). By combining the last three inequalities,

$$
\begin{aligned}
&\mathbf{Pr}\left[\mathcal{A} \cap \left\{X^\ell = r\right\}\right] \\
&\leqslant \mathbf{Pr}\left[\mathcal{C} \cap \left\{X^\ell = r\right\}\right] \\
&\leqslant \mathbf{Pr}\left[\mathcal{C} \cap \left\{\tau \leqslant \ell^\delta\right\} \cap \left\{X^\ell = r\right\}\right] + \mathbf{Pr}\left[\tau > \ell^\delta\right] \\
&\leqslant \mathbf{Pr}\left[\mathcal{C}(\tau) \cap \left\{\tau \leqslant \ell^\delta\right\} \cap \left\{X^\ell = r\right\}\right] + \exp\left(-\ell^\delta/288\right) \\
&\leqslant \mathbf{Pr}\left[\tau \leqslant \ell^\delta\right] \cdot \mathbf{Pr}\left[\mathcal{C}(\tau) \mid \tau \leqslant \ell^\delta\right] \cdot \mathbf{Pr}\left[X^\ell = r \mid \mathcal{C}(\tau), \tau \leqslant \ell^\delta\right] + \exp\left(-\ell^\delta\right) \\
&\leqslant \mathbf{Pr}\left[X^\ell = r \mid \mathfrak{F}^{T(\tau)}, T(\tau) \leqslant \ell\right] + \exp\left(-\ell^\delta/288\right) \\
&\leqslant \ell \cdot \max\left(2^{-D}, 2^{-\ell^\delta/12}\right) + \exp\left(-\ell^\delta/288\right) \\
&\leqslant 16\ell \cdot \max\left(\exp(-\ell^\delta/288), 1/\beta\right) =: p_{\mathsf{bad}}.
\end{aligned}
$$

On the other hand, we will now lower bound the probability that a random walk starting at r never leaves the path of length ℓ^δ and is located at vertex r at step ℓ. By Lemma 4.4, we have that this probability is lower bounded by

$$
\mathbf{Pr}\left[\left\{\mathsf{supp}(\ell) \leqslant 2\ell^\delta\right\} \cap \left\{X^\ell = r\right\}\right] \geqslant \frac{1}{2} \cdot \ell^{-1/2+\varepsilon} \cdot 12^{-8\ell^{2\varepsilon}} \geqslant \ell^{-1/2} \cdot 12^{-8\ell^{2\varepsilon}} =: p_{\mathsf{good}}.
$$

Finally, we can now upper bound the expected size of the support of a closed random walk of length ℓ. Using the conditional probabilities and the definitions of p_{bad} and p_{good}, we have that

$$
\begin{aligned}
\frac{\mathbf{Pr}\left[\mathsf{supp}(\ell) \geqslant 2\ell^\delta + 1 \mid X^\ell = r\right]}{\mathbf{Pr}\left[\mathsf{supp}(\ell) \leqslant 2\ell^\delta + 1 \mid X^\ell = r\right]} &= \frac{\mathbf{Pr}\left[\left\{\mathsf{supp}(\ell) \geqslant 2\ell^\delta + 1\right\} \cap \left\{X^\ell = r\right\}\right]}{\mathbf{Pr}\left[\left\{\mathsf{supp}(\ell) \leqslant 2\ell^\delta + 1\right\} \cap \left\{X^\ell = r\right\}\right]} \leqslant \frac{p_{\mathsf{bad}}}{p_{\mathsf{good}}} \\
&\leqslant 16\ell^{1.5} \cdot \max\left(\exp(-\ell^\delta/288), 1/\beta\right) \cdot 12^{8\ell^{2\varepsilon}},
\end{aligned}
$$

which implies that

$$
\mathbf{Pr}\left[\mathsf{supp}(\ell) \geqslant 2\ell^\delta + 1 \mid X^\ell = r\right] \leqslant 16\ell^{1.5} \cdot \max\left(\exp(-\ell^\delta/288), 1/\beta\right) \cdot 12^{8\ell^{2\varepsilon}}.
$$

Therefore, it holds that

$$
\begin{aligned}
&\mathbf{E}\left[\mathsf{supp}(\ell) \mid X^\ell = r\right] \\
&\leqslant \mathbf{Pr}\left[\mathsf{supp}(\ell) < 2\ell^\delta + 1 \mid X^\ell = r\right] \cdot 2\ell^\delta + \mathbf{Pr}\left[\mathsf{supp}(\ell) \geqslant 2\ell^\delta + 1 \mid X^\ell = r\right] \cdot \ell \\
&\leqslant 2\ell^\delta + 1 + 16\ell^{2.5} \cdot \max\left(\exp(-\ell^\delta/288), 1/\beta\right) \cdot 12^{8\ell^{2\varepsilon}}.
\end{aligned}
$$

Since $\varepsilon = \min\left(1/2 \cdot \log(1/16 \cdot \log_{12}\beta)/\log\ell, 1/7\right)$ by definition, we have $\varepsilon \leqslant 1/7$. Together with $\delta = 1/2 - \varepsilon$, this implies $2\varepsilon \leqslant \delta - 1/14$ and therefore

$$
12^{8\ell^{2\varepsilon}} \leqslant 12^{8\ell^{\delta-1/14}} \leqslant \exp\left(\ell^\delta/572\right),
$$

where the last inequality holds if ℓ is lower bounded by a sufficiently large constant $C > 0$. Applying this gives us that

$$
16\ell^{2.5} \cdot \exp(-\ell^\delta/288) \cdot 12^{8\ell^{2\varepsilon}} \leqslant 16\ell^{2.5} \cdot \exp\left(-\ell^\delta/572\right).
$$

Similarly, we have $\varepsilon \leqslant 1/2 \cdot \log(1/16 \cdot \log_{12}\beta)/\log\ell$ by definition, and obtain

$$
16\ell^{2.5} \cdot \frac{1}{\beta} \cdot 12^{8\ell^{2\varepsilon}} \leqslant 16\ell^{2.5} \cdot \frac{1}{\beta} \cdot \sqrt{\beta}.
$$

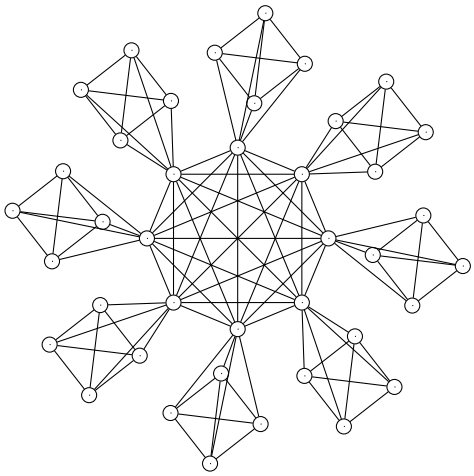


Figure 3 The construction of the graph G, where a large clique of $\beta/\lceil\log\log\beta\rceil$ vertices is connected to $\beta/\lceil\log\log\beta\rceil$ small cliques of size $\lceil\log\log\beta\rceil$ each; recall that our choice of β ensures that $\beta/\lceil\log\log\beta\rceil$ is an integer.

Combining the last two inequalities gives us that

$$\mathbf{E}\left[\,\mathsf{supp}(\ell)\;\middle|\;X^s=r\,\right]\leqslant 2\ell^\delta+1+16\ell^{2.5}\cdot\max\left(\exp\left(-\ell^\delta/572\right),\frac{1}{\sqrt{\beta}}\right)\leqslant 3\ell^\delta,$$

using that $\ell\geqslant C$ for some large constant $C>0$ as well as $\ell\leqslant\beta^{1/5}$. This completes the proof. ◀

4.2 Proof of Theorem 1.3

In this subsection we present a more detailed formulation of Theorem 1.3, and prove the statement afterwards.

▶ **Theorem 4.5** (Formal version of Theorem 1.3). *Let $C\geqslant 1$ be a constant. Then, for every integer $\beta\geqslant C$ such that $\beta/\lceil\log\log\beta\rceil$ is an integer, there is a graph G with $n=\beta+\beta/\lceil\log\log\beta\rceil$ vertices such that a lazy random walk of length $\ell=\lfloor\log\beta\rfloor$ starting from some vertex chosen uniformly at random from $V(G)$ satisfies*

$$\mathbf{E}\left[\,\mathsf{supp}_{\mathbf{P}}(\ell)\;\middle|\;X^\ell=X^0\sim\mathcal{U}\,\right]\leqslant 5\log\ell.$$

In particular, there is a start vertex $r\in V$ such that

$$\mathbf{E}\left[\,\mathsf{supp}_{\mathbf{P}}(\ell)\;\middle|\;X^\ell=X^0=r\,\right]\leqslant 5\log\ell.$$

We first define the graphs used in proving Theorem 4.5. For any given parameter $\beta\in\mathbb{N}$, our graph G is defined as follows:

- Let $G_1=(V_1,E_1)$ consist of $\beta/\lceil\log\log\beta\rceil$ disjoint, "small" cliques of size $\lceil\log\log\beta\rceil$ each.
- Let $G_2=(V_2,E_2)=K_{\beta/\lceil\log\log\beta\rceil}$ be a "big" clique of size $\beta/\lceil\log\log\beta\rceil$.
- Our studied graph G is constructed by taking the union of G_1 and G_2, and additionally connecting each vertex of the smaller cliques to one distinct vertex in the big clique.

See Figure 3 for an illustration of our construction.

Proof of Theorem 4.5. We use the graph G defined above in the proof, and decompose the expected support based on the sampled start vertex according to the uniform distribution over $V(G)$:

$$\begin{aligned}
&\mathbf{E}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \mid X^\ell = X^0 \sim \mathcal{U}\right] \\
&= \sum_{u\in V} \mathbf{E}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \mid X^\ell = X^0 = u\right] \cdot \mathbf{Pr}\left[X^0 = u \mid X^\ell = X^0\right] \\
&= \sum_{u\in V} \mathbf{E}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \mid X^\ell = X^0 = u\right] \cdot \frac{\mathbf{Pr}\left[X^\ell = X^0 = u\right]}{\mathbf{Pr}\left[X^\ell = X^0\right]} \\
&= \sum_{u\in V} \mathbf{E}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \mid X^\ell = X^0 = u\right] \cdot \frac{\mathbf{Pr}\left[X^\ell = u \mid X^0 = u\right] \cdot \frac{1}{n}}{\mathbf{Pr}\left[X^\ell = X^0\right]}
\end{aligned}$$

Splitting the above sum yields

$$\begin{aligned}
&\mathbf{E}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \mid X^\ell = X^0 \sim \mathcal{U}\right] \\
&= \sum_{u\in V(G_1)} \mathbf{E}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \mid X^\ell = X^0 = u\right] \cdot \frac{\mathbf{Pr}\left[X^\ell = u \mid X^0 = u\right] \cdot \frac{1}{n}}{\mathbf{Pr}\left[X^\ell = X^0\right]} \\
&\quad + \sum_{v\in V(G_2)} \mathbf{E}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \mid X^\ell = X^0 = v\right] \cdot \frac{\mathbf{Pr}\left[X^\ell = v \mid X^0 = v\right] \cdot \frac{1}{n}}{\mathbf{Pr}\left[X^\ell = X^0\right]} \\
&= |V(G_1)| \cdot \mathbf{E}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \mid X^\ell = X^0 = u\right] \cdot \frac{\mathbf{Pr}\left[X^\ell = u \mid X^0 = u\right] \cdot \frac{1}{n}}{\mathbf{Pr}\left[X^\ell = X^0\right]} \\
&\quad + |V(G_2)| \cdot \mathbf{E}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \mid X^\ell = X^0 = v\right] \cdot \frac{\mathbf{Pr}\left[X^\ell = v \mid X^0 = v\right] \cdot \frac{1}{n}}{\mathbf{Pr}\left[X^\ell = X^0\right]}, \qquad (6)
\end{aligned}$$

where u is an arbitrary vertex in G_1, v is an arbitrary vertex in G_2, and the last equation holds by symmetry.

Consider now a lazy random walk which starts from some vertex $u \in V_1$ in a small clique. Then the probability that this random walk never leaves the small clique and is at u in step $\ell \triangleq \lfloor \log \beta \rfloor$ is at least

$$\begin{aligned}
&\mathbf{Pr}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \geqslant \lceil \log\log\beta \rceil \cap \{X^\ell = u\} \mid X^0 = u\right] \\
&\geqslant \left(1 - \frac{1}{2\lceil \log\log\beta \rceil}\right)^{\lfloor \log\beta \rfloor - 1} \cdot \min\left(\frac{1}{2}, \frac{1}{2\lceil \log\log\beta \rceil}\right) \\
&\geqslant 8^{-\log\beta/\log\log\beta}. \qquad (7)
\end{aligned}$$

If the random walk leaves a small clique, then the probability of returning to the small clique within ℓ steps is at most $\ell \cdot (\lceil \log\log\beta \rceil)^2/n$; hence, it holds that

$$\begin{aligned}
\mathbf{Pr}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \geqslant \lceil \log\log\beta \rceil \cap \{X^\ell = u\} \mid X^0 = u\right] &\leqslant \ell \cdot \frac{\lceil \log\log\beta \rceil}{\beta/\lceil \log\log\beta \rceil - 1 + \lceil \log\log\beta \rceil} \\
&\leqslant \ell \cdot \frac{(\lceil \log\log\beta \rceil)^2}{\beta}.
\end{aligned}$$

Therefore, as in the proof of Theorem 4.1,

$$\begin{aligned}
&\frac{\mathbf{Pr}\left[\,\mathsf{supp}_{\mathbf{P}}(\ell) \geqslant \lceil \log\log\beta \rceil \;\mid\; X^\ell = X^0 = u\,\right]}{\mathbf{Pr}\left[\,\mathsf{supp}_{\mathbf{P}}(\ell) \leqslant \lceil \log\log\beta \rceil \;\mid\; X^\ell = X^0 = u\,\right]} \\
&= \frac{\mathbf{Pr}\left[\,\mathsf{supp}_{\mathbf{P}}(\ell) \geqslant \lceil \log\log\beta \rceil \cap \left\{X^\ell = u\right\} \;\mid\; X^0 = u\,\right]}{\mathbf{Pr}\left[\,\mathsf{supp}_{\mathbf{P}}(\ell) \leqslant \lceil \log\log\beta \rceil \cap \{X^\ell = u\} \;\mid\; X^0 = u\,\right]} \\
&\leqslant \frac{\ell \cdot \frac{(\lceil \log\log\beta \rceil)^2}{\beta}}{8^{-\log\beta/\log\log\beta}},
\end{aligned}$$

and upper bounding the denominator on the left hand side by 1 yields

$$\mathbf{Pr}\left[\,\mathsf{supp}_{\mathbf{P}}(\ell) \geqslant \lceil \log\log\beta \rceil \;\mid\; X^\ell = u\,\right] \leqslant \frac{\ell \cdot \frac{(\lceil \log\log\beta \rceil)^2}{n}}{8^{-\log\beta/\log\log\beta}}.$$

Therefore, it holds that

$$\begin{aligned}
&\mathbf{E}\left[\,\mathsf{supp}_{\mathbf{P}}(\ell) \;\mid\; X^\ell = X^0 = u\,\right] \\
&\leqslant \mathbf{Pr}\left[\,\mathsf{supp}_{\mathbf{P}}(\ell) \geqslant \lceil \log\log\beta \rceil \;\mid\; X^0 = u\,\right] \cdot \ell \\
&\qquad + \mathbf{Pr}\left[\,\mathsf{supp}_{\mathbf{P}}(\ell) \leqslant \lceil \log\log\beta \rceil \;\mid\; X^0 = u\,\right] \cdot \lceil \log\log\beta \rceil \\
&\leqslant \ell^2 \cdot 8^{\log\beta/\log\log\beta} \cdot \frac{(\lceil \log\log\beta \rceil)^2}{\beta} + 1 \cdot \lceil \log\log\beta \rceil \leqslant 2 \cdot \lceil \log\log\beta \rceil,
\end{aligned}$$

as $\ell = \lfloor \log\beta \rfloor$.

Now we return to (6). By using the three trivial estimates, (i) $|V_1| \leqslant n$, (ii) $|V_2| \leqslant n$ and (iii) $\mathbf{E}\left[\,\mathsf{supp}_{\mathbf{P}}(\ell) \;\mid\; X^\ell = X^0 = v\,\right] \leqslant \ell$, we have

$$\begin{aligned}
&\mathbf{E}\left[\,\mathsf{supp}_{\mathbf{P}}(\ell) \;\mid\; X^\ell = X^0 \sim \mathcal{U}\,\right] \\
&\leqslant n \cdot 2 \cdot \lceil \log\log\beta \rceil \cdot \frac{\mathbf{Pr}\left[\,X^\ell = u \;\mid\; X^0 = u\,\right] \cdot \frac{1}{n}}{\mathbf{Pr}\left[\,X^\ell = X^0\,\right]} + n \cdot \ell \cdot \frac{\mathbf{Pr}\left[\,X^\ell = v \;\mid\; X^0 = v\,\right] \cdot \frac{1}{n}}{\mathbf{Pr}\left[\,X^\ell = X^0\,\right]} \\
&\leqslant 2 \cdot \lceil \log\log\beta \rceil \cdot \frac{\mathbf{Pr}\left[\,X^\ell = u \;\mid\; X^0 = u\,\right]}{\mathbf{Pr}\left[\,X^\ell = X^0\,\right]} + \ell \cdot \frac{\mathbf{Pr}\left[\,X^\ell = v \;\mid\; X^0 = v\,\right]}{\mathbf{Pr}\left[\,X^\ell = X^0\,\right]}. \qquad (8)
\end{aligned}$$

We now proceed to upper bound the expression in Equation (8), by considering the two addends separately. We first upper bound $\frac{\mathbf{Pr}\left[\,X^\ell = u \mid X^0 = u\,\right]}{\mathbf{Pr}\left[\,X^\ell = X^0\,\right]}$. By decomposing and lower bounding the denominator, we have that

$$\begin{aligned}
\mathbf{Pr}\left[\,X^\ell = X^0\,\right] &\geqslant \sum_{u \in V_1} \mathbf{Pr}\left[\,X^0 = u\,\right] \cdot \mathbf{Pr}\left[\,X^\ell = u \;\mid\; X^0 = u\,\right] \\
&\geqslant \frac{1}{2} \cdot \mathbf{Pr}\left[\,X^\ell = u \;\mid\; X^0 = u\,\right] \qquad (9)
\end{aligned}$$

since the probability $\mathbf{Pr}\left[\,X^\ell = X^0 \;\mid\; X^0 = u\,\right]$ is the same for all $u \in V_1$ by symmetry, and by construction of G, at least half of the vertices in G are in V_1. Therefore,

$$\frac{\mathbf{Pr}\left[\,X^\ell = u \;\mid\; X^0 = u\,\right]}{\mathbf{Pr}\left[\,X^\ell = X^0\,\right]} \leqslant 2. \qquad (10)$$

We now turn to the second addend in (8). We first upper bound $\mathbf{Pr}\left[\,X^\ell = v \;\mid\; X^0 = v\,\right] = p^\ell_{v,v}$, where $v \in V_2$. To this end, note that the random walk can only be at v at step ℓ if at least one of the following three cases occurs: (i) the random walk always remains on v by taking ℓ self-loops, (ii) the random walk leaves v, and then returns to v from another vertex

in the big clique, and (iii) the random walk leaves v, and then returns to v from a neighbor in the small clique. Regarding (i), the probability is $2^{-\ell}$. Regarding (ii), the probability of ever using an edge $\{x, v\} \in E$ with $x \in V_2$ during ℓ steps is at most

$$\ell \cdot \frac{1}{2\deg(x)} \leqslant \ell \cdot \frac{1}{2 \cdot \left(\frac{\beta}{\lceil \log\log\beta \rceil} - 1 + \lceil \log\log\beta \rceil \right)} \leqslant \ell \cdot \frac{\lceil \log\log\beta \rceil}{2\beta}.$$

Finally, regarding (iii), the probability that the random walk ever reaches any vertex in V_1 within ℓ steps is upper bounded by

$$\ell \cdot \max_{z \in V_2} \frac{\deg_{V_1}(z)}{2\deg(z)} \leqslant \ell \cdot \frac{\lceil \log\log\beta \rceil}{2 \cdot \left(\frac{\beta}{\lceil \log\log\beta \rceil} - 1 + \lceil \log\log\beta \rceil \right)} \leqslant \ell \cdot \frac{(\lceil \log\log\beta \rceil)^2}{2\beta}.$$

Combining these three cases, we have

$$\mathbf{Pr}\left[X^\ell = v \mid X^0 = v \right] \leqslant 2^{-\ell} + \ell \cdot \frac{(\lceil \log\log\beta \rceil)^2}{\beta}. \tag{11}$$

To lower bound $\mathbf{Pr}\left[X^\ell = X^0 \right]$, we apply (9) and the estimate from (7) to obtain that

$$\mathbf{Pr}\left[X^\ell = X^0 \right] \geqslant \frac{1}{2} \cdot \mathbf{Pr}\left[X^\ell = u \mid X^0 = u \right] \geqslant \frac{1}{2} \cdot 8^{-\log\beta/\log\log\beta}. \tag{12}$$

Finally, combining (10), (11), (12) with (8) gives us that

$$\begin{aligned}
&\mathbf{E}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \mid X^\ell = X^0 \sim \mathcal{U} \right] \\
&\leqslant 4 \cdot \lceil \log\log\beta \rceil + 2 \cdot \ell \cdot 8^{\log\beta/\log\log\beta} \cdot \left(2^{-\ell} + \ell \cdot \frac{(\lceil \log\log\beta \rceil)^2}{\beta} \right) \\
&\leqslant 4 \cdot \lceil \log\log\beta \rceil + 2 \cdot \ell \cdot 8^{\log\beta/\log\log\beta} \cdot \left(2 \cdot \ell \cdot \frac{(\lceil \log\log\beta \rceil)^2}{\beta} \right) \\
&\leqslant 4 \cdot \lceil \log\log\beta \rceil + o(1) \\
&\leqslant 5 \cdot \log\ell,
\end{aligned}$$

where in the second inequality we used the definition $\ell = \lfloor \log\beta \rfloor \geqslant \log\beta - 1$, and similarly the third inequality holds since $\ell = \lfloor \log\beta \rfloor$, and thus the $1/\beta$ term dominates.

For the second statement, note that by conditioning on the vertex sampled for X^0,

$$\begin{aligned}
&\mathbf{E}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \mid X^\ell = X^0 \sim \mathcal{U} \right] \\
&= \sum_{u \in V} \mathbf{E}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \mid X^\ell = X^0 = u \right] \cdot \mathbf{Pr}\left[X^\ell = X^0 = u \mid X^\ell = X^0 \right] \\
&\geqslant \min_{u \in V} \mathbf{E}\left[\mathsf{supp}_{\mathbf{P}}(\ell) \mid X^\ell = X^0 = u \right],
\end{aligned}$$

and thereore the second statement follows immediately from the first statement. ◀

5 Results on Eigenvalue Multiplicity

This section presents some new eigenvalue multiplicity bounds on the transition matrices of random walks. For ease of presentation, we focus on lazy random walks in this section, but our presented method can be employed to analyze non-lazy random walks, too. Our first eigenvalue multiplicity bound is as follows:

▶ **Theorem 5.1** (Formal Statement of Theorem 1.4). *Consider any connected, n-vertex graph $G = (V, E)$ with minimum degree δ and maximum degree Δ. Further assume $\lambda = \lambda_2 \geqslant \gamma$, where $\gamma \triangleq 1 - \frac{\delta}{32c\Delta} \cdot \frac{1}{\log^4 n}$ and $c \geqslant 1$ is the constant from Lemma 3.2. Then, it holds that*

$$M_{\mathbf{P}}\left[\left(1 - \frac{\delta}{32c\Delta \cdot \log^5 n}\right) \cdot \lambda, \lambda\right] = O\left(\frac{n}{\log n}\right). \tag{13}$$

It is shown in [16] that, for the normalized adjacency matrix of any n-vertex graph G with maximum degree Δ, the number of eigenvalues in the range $\left[\left(1 - \frac{\log\log_\Delta}{\log_\Delta n}\right) \cdot \lambda_2, \lambda_2\right]$ is

$$\widetilde{O}\left(n \cdot \frac{\Delta^{7/5}}{\log^{1/5} n}\right). \tag{14}$$

In comparison to their bound, our presented result only holds for graphs with poor expansion. However, Theorem 5.1 does show for such graphs that the number of eigenvalues in our studied range is $O(n/\log n)$, which is significantly smaller than the bound in (14).

The proof of Theorem 5.1 closely follows the approach in [16], by reducing the multiplicity analysis to the support of closed random walks. We consider a lazy random walk of length ℓ on G, and assume that the start vertex X^0 is sampled uniformly at random. We denote by W^ℓ the event $\{X^\ell = X^0\}$ and by $W^{\ell,s}$ the event $\{X^\ell = X^0, \mathsf{supp}(\ell) \leqslant s\}$; that is, the random walk is closed and has support at most s. Abusing notation a bit, let $W^{\ell,\geqslant s}$ be the event where the random walk is closed and has support at least s. We now state the following bound, which is based on the arguments of [16] and the probability bound in Theorem 3.1.

▶ **Lemma 5.2.** *Consider any connected, n-vertex graph, and a lazy random walk of length $\ell \leqslant 32c\mu \cdot \frac{\Delta}{\delta} n^2$ starting from a uniform random vertex. Then with $c > 0$ being the constant from Lemma 3.2, it holds for any $s \leqslant \left\lfloor \sqrt{\frac{1}{64c} \cdot \frac{\delta}{\Delta} \cdot \frac{\ell}{\mu}} \right\rfloor$ that*

$$\mathbf{Pr}\left[W^{\ell,s}\right] \leqslant \frac{\Delta}{\delta} \cdot n \cdot \left(\frac{5}{8}\right)^{\mu/2} \mathbf{Pr}\left[W^\ell\right].$$

Combining Lemma 5.2 with the techniques developed in [16] proves Theorem 5.1.

We further present a different and more elementary approach to bound the multiplicities of the eigenvalue λ_2, and our proof is based on the Random Target Lemma [14, (3.3)].

▶ **Theorem 5.3.** *Consider any connected, n-vertex graph $G = (V, E)$ with average degree d and minimum degree δ. Then there is some constant $C > 0$, such that it holds for any $\varepsilon > 0$ that*

$$M_{\mathbf{P}}[(1-\varepsilon)\lambda_2, \lambda_2] \leqslant C \cdot \frac{d}{\delta} \cdot n \cdot \frac{1 - (1-\varepsilon)\lambda_2}{\sqrt{1-\lambda_2}}.$$

In particular, it holds with $\varepsilon = (1-\lambda_2)/\lambda_2$ that

$$M_{\mathbf{P}}[(1-\varepsilon)\lambda_2, \lambda_2] \leqslant 2C \cdot \frac{d}{\delta} \cdot n \cdot \sqrt{1-\lambda_2}.$$

6 Conclusions

In this work we analyze the support of closed random walks of length ℓ on different graph classes. Contrary to the well-understood worst-case support of *open* random walks, especially on regular and bounded-degree graphs, our results demonstrate that the (worst-case) support

of *closed* random walks is much more complex, and undergoes a delicate phase transition as ℓ varies. While the support is $\Theta(\ell^{1/2}/\sqrt{\log n})$ for $\ell = \Omega((\log n)^{7/2})$, for smaller values of ℓ it is sandwiched between $\Omega(\ell^{1/5})$ and $O(\ell^{5/14})$. This proves a strong separation from the open random walk case, where the support is known to be $\Omega(\ell^{1/2})$ [3, 7], and provides a negative answer to [16, Open Problem 3].

For *(highly) irregular* graphs, we prove that even with a *randomly sampled* start vertex, the support may only be logarithmic in ℓ. This is once more in sharp contrast to open walks, where a lower bound of $\Omega(\ell^{1/3})$ holds for *any* start vertex and any $1 \leqslant \ell \leqslant n^3$ [3].

One interesting open problem is to derive refined bounds on the support of closed random walks. For instance, it is tempting to conjecture that on any bounded-degree expander graph, the lower bound on the support can be improved, possibly even to $\Omega(\ell)$, which would be tight and match the bound for open random walks. To the best of our knowledge, this is only known for the special case where ℓ is upper bounded by the girth of the expander [5].

References

1 David Aldous and James A. Fill. Reversible markov chains and random walks on graphs, 2002. Unpublished. `http://www.stat.berkeley.edu/~aldous/RWG/book.html`.

2 Noga Alon, Chen Avin, Michal Koucký, Gady Kozma, Zvi Lotker, and Mark R. Tuttle. Many random walks are faster than one. *Combinatorics, Probability and Computing*, 20(4):481–502, 2011. `doi:10.1017/S0963548311000125`.

3 Greg Barnes and Uriel Feige. Short random walks on graphs. *SIAM Journal on Discrete Mathematics*, 9(1):19–28, 1996. `doi:10.1137/S0895480194264988`.

4 Anna Ben-Hamou, Roberto I. Oliveira, and Yuval Peres. Estimating graph parameters via random walks with restarts. In *29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'18)*, pages 1702–1714, 2018.

5 Itai Benjamini, Roey Izkovsky, and Harry Kesten. On the Range of the Simple Random Walk Bridge on Groups. *Electronic Journal of Probability*, 12:591–612, 2007. `doi:10.1214/EJP.v12-408`.

6 Artur Czumaj and Christian Sohler. Testing expansion in bounded-degree graphs. *Combinatorics, Probability and Computing*, 19(5-6):693–709, 2010. `doi:10.1017/S096354831000012X`.

7 Uriel Feige. A spectrum of time–space trade-offs for undirected *s*-*t* connectivity. *Journal of Computer and System Sciences*, 54(2):305–316, 1997. `doi:10.1006/jcss.1997.1471`.

8 George Giakkoupis, Frederik Mallmann-Trenn, and Hayk Saribekyan. How to spread a rumor: Call your neighbors or take a walk? In *38th Annual ACM-SIGOPT Principles of Distributed Computing (PODC'19)*, pages 24–33, 2019.

9 Milan Haiman, Carl Schildkraut, Shengtong Zhang, and Yufei Zhao. Graphs with high second eigenvalue multiplicity. *Bulletin of the London Mathematical Society*, pages 1–23, 2022. `doi:10.1112/blms.12647`.

10 Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *BAMS: Bulletin of the American Mathematical Society*, 43, 2006.

11 Zilin Jiang, Jonathan Tidor, Yuan Yao, Shengtong Zhang, and Yufei Zhao. Equiangular lines with a fixed angle. *Annals of Mathematics*, 194(3):729–743, 2021.

12 James R. Lee, Shayan Oveis Gharan, and Luca Trevisan. Multiway spectral partitioning and higher-order cheeger inequalities. *Journal of the ACM*, 61(6):37:1–37:30, 2014.

13 David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2009. With a chapter by James G. Propp and David B. Wilson.

14 László Lovász. Random walks on graphs: A survey. *Combinatorics, Paul Erdős is Eighty*, 2:1–46, 1993.

15 Theo McKenzie. *Random Walks and Delocalization through Graph Eigenvector Structure*. PhD thesis, UC Berkeley, 2022.

16 Theo McKenzie, Peter Michael Reichstein Rasmussen, and Nikhil Srivastava. Support of closed walks and second eigenvalue multiplicity of graphs. In *53rd Annual ACM Symposium on Theory of Computing (STOC'21)*, pages 396–407, 2021.

17 Roberto I. Oliveira. On the coalescence time of reversible random walks. *Transactions of the American Mathematical Society*, 364(4):2109–2128, 2012.

18 Roberto I. Oliveira and Yuval Peres. Random walks on graphs: new bounds on hitting, meeting, coalescing and returning. In *16th Workshop on Analytic Algorithmics and Combinatorics (ANALCO '19)*, pages 119–126, 2019.

19 Thomas Sauerwald and He Sun. Tight bounds for randomized load balancing on arbitrary network topologies. In *53th Annual IEEE Symposium on Foundations of Computer Science (FOCS'12)*, pages 341–350, 2012. doi:10.1109/FOCS.2012.86.

20 Daniel A. Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on Computing*, 42(1):1–26, 2013.

A Auxiliary Tools

This section lists the auxiliary results used in the paper. Our first lemma is a simple upper bound on the lower tails of a sub-multiplicative random variable. We remark that this is a standard result, however we present the proof here for the sake of completeness.

▶ **Lemma A.1.** *Let X be a non-negative integer random variable such that $\mathbf{E}[X] \geqslant b$, and there exists integer $c \geqslant 1$ such that $\mathbf{Pr}[X > kc] \leqslant \mathbf{Pr}[X > c]^k$ for all integers $k \geqslant 0$. Then, it holds for any $a < c$ that*

$$\mathbf{Pr}[X > a] \geqslant \frac{b-a}{b+2c}.$$

Proof. Let $p \triangleq \mathbf{Pr}[X > a] \geqslant \mathbf{Pr}[X > c]$, since $a < c$ and $\mathbf{Pr}[X > x]$ decreasing in x. Hence, it holds that

$$\begin{aligned}
b &\leqslant \mathbf{E}[X] \\
&= \sum_{i=0}^{\infty} \mathbf{Pr}[X > i] \\
&\leqslant a + \sum_{i=a}^{c-1} \mathbf{Pr}[X > i] + c\sum_{k=1}^{\infty} \mathbf{Pr}[X > kc] \\
&\leqslant a + \sum_{i=a}^{c-1} \mathbf{Pr}[X > i] + c\sum_{k=1}^{\infty} \mathbf{Pr}[X > c]^k \\
&\leqslant a + p(c-a) + cp/(1-p).
\end{aligned}$$

This implies $(1-p)b \leqslant a + 2pc$, rearranging gives the result. ◀

▶ **Theorem A.2** (Cauchy's Interlacing Theorem)**.** *Let $\mathbf{A}$ be a real symmetric $n \times n$ matrix, and $\mathbf{B}$ an $m \times m$ principal submatrix of $\mathbf{A}$ (that is, $\mathbf{B}$ is obtained by deleting both the i^{th} row and column for some values of i). Suppose A has eigenvalues $\lambda_1, \ldots, \lambda_n$, and B has eigenvalues $\beta_1, \ldots, \beta_m$. Then, it holds for $1 \leqslant k \leqslant m$ that $\lambda_k \leqslant \beta_k \leqslant \lambda_{k+n-m}$.*

▶ **Theorem A.3** (Hoeffding's Bound). *Let $Y_1, \ldots, Y_\ell$ be independent bounded random variables with $Y_i \in [a, b]$ for each $i \leqslant \ell$, and define $Y = \sum_{i=1}^{\ell} Y_i$. Then, the following hold for all $\lambda \geqslant 0$:*

$$\mathbf{Pr}\left[\, Y_i - \mathbf{E}\left[\, Y \,\right] \geqslant \lambda \,\right] \leqslant \exp\left(-\frac{2\lambda^2}{\ell(b-a)^2}\right)$$

and

$$\mathbf{Pr}\left[\, Y_i - \mathbf{E}\left[\, Y \,\right] \leqslant -\lambda \,\right] \leqslant \exp\left(-\frac{2\lambda^2}{\ell(b-a)^2}\right).$$

Faster Matroid Partition Algorithms

Tatsuya Terao
Research Institute for Mathematical Sciences, Kyoto University, Japan

Abstract

In the matroid partitioning problem, we are given k matroids $\mathcal{M}_1 = (V, \mathcal{I}_1), \ldots, \mathcal{M}_k = (V, \mathcal{I}_k)$ defined over a common ground set V of n elements, and we need to find a partitionable set $S \subseteq V$ of largest possible cardinality, denoted by p. Here, a set $S \subseteq V$ is called partitionable if there exists a partition $(S_1, \ldots, S_k)$ of S with $S_i \in \mathcal{I}_i$ for $i = 1, \ldots, k$. In 1986, Cunningham [7] presented a matroid partition algorithm that uses $O(np^{3/2} + kn)$ independence oracle queries, which was the previously known best algorithm. This query complexity is $O(n^{5/2})$ when $k \leq n$.

Our main result is to present a matroid partition algorithm that uses $\tilde{O}(k^{1/3}np + kn)$ independence oracle queries, which is $\tilde{O}(n^{7/3})$ when $k \leq n$. This improves upon previous Cunningham's algorithm. To obtain this, we present a new approach *edge recycling augmentation*, which can be attained through new ideas: an efficient utilization of the binary search technique by Nguyễn [25] and Chakrabarty-Lee-Sidford-Singla-Wong [5] and a careful analysis of the number of independence oracle queries. Our analysis differs significantly from the one for matroid intersection algorithms, because of the parameter k. We also present a matroid partition algorithm that uses $\tilde{O}((n + k)\sqrt{p})$ rank oracle queries.

2012 ACM Subject Classification Theory of computation → Discrete optimization; Theory of computation → Algorithm design techniques; Mathematics of computing → Matroids and greedoids

Keywords and phrases Matroid Partition, Matroid Union, Combinatorial Optimization

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.104

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2303.05920

Funding This work was partially supported by the joint project of Kyoto University and Toyota Motor Corporation, titled "Advanced Mathematical Science for Mobility Society".

Acknowledgements The author thanks Yusuke Kobayashi for his generous support and helpful comments on the manuscript. The author also thanks the three anonymous reviewers for their valuable comments.

1 Introduction

The *matroid partitioning problem*[1] is one of the most fundamental problem in combinatorial optimization. The problem is sometimes introduced as an important matroid problem along with the *matroid intersection problem*; see [28, Section 41–42] and [21, Section 13.5–6]. In the problem, we are given k matroids $\mathcal{M}_1 = (V, \mathcal{I}_1), \ldots, \mathcal{M}_k = (V, \mathcal{I}_k)$ defined over a common ground set V of n elements, and the objective is to find a partitionable set $S \subseteq V$ of largest possible cardinality, denoted by p. Here, we call a set $S \subseteq V$ partitionable if there exists a partition $(S_1, \ldots, S_k)$ of S with $S_i \in \mathcal{I}_i$ for $i = 1, \ldots, k$. This problem has a number of applications such as matroid base packing, packing and covering of trees and forests, Shannon switching game. There are much more applications; see [28, Section 42].

[1] The *matroid partitioning problem* is sometimes called simply *matroid partition*. Matroid partition is also called *matroid union* or *matroid sum*.

To design an algorithm for arbitrary matroids, it is common to consider an oracle model: an algorithm accesses a matroid through an oracle. The most standard and well-studied oracle is an *independence oracle*, which takes as input a set $S \subseteq V$ and outputs whether $S \in \mathcal{I}$ or not. Some recent studies for fast matroid intersection algorithms also consider a more powerful oracle called *rank oracle*, which takes as input a set $S \subseteq V$ and outputs the size of the maximum cardinality independent subset of S. In the design of efficient algorithms, the goal is to minimize the number of such oracle accesses in a matroid partition algorithm. We consider both independence oracle model and rank oracle model, and present the best query algorithms for both oracle models.

The matroid partitioning problem is closely related to the matroid intersection problem. Actually, the matroid partitioning problem and the matroid intersection problem are polynomially equivalent; see [9, 11].

In the *matroid intersection problem*, we are given two matroids $\mathcal{M}' = (V, \mathcal{I}'), \mathcal{M}'' = (V, \mathcal{I}'')$ defined over a common ground set V of n elements, and the objective is to find a common independent set $S \in \mathcal{I}' \cap \mathcal{I}''$ of largest possible cardinality, denoted by r.

Starting the work of Edmonds [8, 10, 11] in the 1960s, algorithms with polynomial query complexity for the matroid intersection problem have been studied [1–5, 7, 22, 23]. Nguyễn [25] and Chakrabarty-Lee-Sidford-Singla-Wong [5] independently presented a new excellent binary search technique that can find edges in the exchange graph and presented a first combinatorial algorithm that uses $\tilde{O}(nr)$ independence oracle queries[2]. Chakrabarty et al. [5] also presented a $(1-\epsilon)$ approximation matroid intersection algorithm that uses $\tilde{O}(n^{1.5}/\epsilon^{1.5})$ independence oracle queries. Blikstad-van den Brand-Mukhopadhyay-Nanongkai [4] developed a fast algorithm to solve a graph reachability problem, and broke the $\tilde{O}(n^2)$-independence-oracle-query bound by combining this with previous exact and approximation algorithms. Blikstad [2] improved the independence query complexity of the approximation matroid intersection algorithm. This leads to a randomized matroid intersection algorithm that uses $\tilde{O}(nr^{3/4})$ independence oracle queries, which is currently the best algorithm for the matroid intersection problem for the full range of r. This also leads to a deterministic matroid intersection algorithm that uses $\tilde{O}(nr^{5/6})$ independence oracle queries, which is currently the best deterministic algorithm for the matroid intersection problem for the full range of r.

We can solve the matroid partitioning problem by using the reduction to the matroid intersection problem. A well-known reduction reduces the matroid partition to the matroid intersection whose ground set size is kn. Here, one of the input matroids of this matroid intersection is the direct sum of the k matroids. This leads to a matroid partition algorithm using too many independence oracle queries. Even if we use the currently best algorithm for matroid intersection, the naive reduction leads to a matroid partition algorithm that uses $\tilde{O}(k^2np^{3/4})$ independence oracle queries. Since the matroid partition problem itself is an important problem with several applications, it is meaningful to focus on the query-complexity of the matroid partitioning problem.

A direct algorithm for the matroid partitioning problem was first given by Edmonds in 1968 [8]. Algorithms with polynomial query complexity for the matroid partitioning problem have been studied in the literature [3, 7, 12–14, 20, 27].

Cunningham [7] designed a matroid partition algorithm that uses $O(np^{3/2} + kn)$ independence oracle queries. Cunningham uses a *blocking flow* approach, which is similar to Hopcroft-Karp's bipartite matching algorithm or Dinic's maximum flow algorithm. The independence query complexity of Cunningham's algorithm is $O(n^{5/2})$ when $k \leq n$. Note

[2] The $\tilde{O}$ notation omits factors polynomial in $\log n$.

that $p \leq n$ obviously holds. This was the best algorithm for the matroid partitioning problem for nearly four decades. We study faster matroid partition algorithms by using techniques that were recently developed for fast matroid intersection algorithms.

Our first result is the following theorem, which is obtained by combining Cunningham's technique and the binary search technique by Nguyễn [25] and Chakrabarty et al. [5].

▶ **Theorem 1** (Details in Theorem 14). *There is an algorithm that uses $\tilde{O}(kn\sqrt{p})$ independence oracle queries and solves the matroid partitioning problem.*

The independence query complexity of the algorithm given in Theorem 1 improves upon the one of Cunningham's algorithm [7] when k is small. However, when $k = \Theta(n)$, the independence query complexity of the algorithm given in Theorem 1 is $\tilde{O}(n^{5/2})$, and this query complexity is not strictly less than the one in Cunningham's algorithm.

The setting where $k > n$ is unnatural since there must exist a matroid whose independent set is not involved in the optimal partition. Thus, in this paper, we mainly focus on the case where $k \leq n$. Under this assumption, we sometimes bound the number of queries by a function on a single variable n, where we recall that $p \leq n$. This makes it easy to compare the query complexity of different algorithms.

Our second result is to obtain an algorithm that uses $o(kn\sqrt{p})$ independence oracle queries when k is large. It uses $o(n^{5/2})$ independence oracle queries when $k \leq n$.

▶ **Theorem 2** (Details in Theorem 18). *There is an algorithm that uses $\tilde{O}(k^{1/3}np + kn)$ independence oracle queries and solves the matroid partitioning problem.*

This is the main contribution of this paper. The independence query complexity of the algorithm given in Theorem 2 improves the one of the algorithm given in Theorem 1 when $k = \omega(p^{3/4})$. The independence query complexity of the algorithm given in Theorem 2 is $\tilde{O}(n^{7/3})$ when $k \leq n$. This improves the algorithm by Cunningham [7] and our algorithm given in Theorem 1. It should be emphasized here that this is the first improvement since 1986. We note that this algorithm requires $O(k^{2/3}np)$ time complexity other than independence oracle queries.

We also consider the query complexity in the rank oracle model. Note that the rank oracle is at least as powerful as the independence oracle.

▶ **Theorem 3** (Details in the full version of this paper). *There is an algorithm that uses $\tilde{O}((n+k)\sqrt{p})$ rank oracle queries and solves the matroid partitioning problem.*

The rank query complexity of the algorithm given in Theorem 3 is $\tilde{O}(n^{3/2})$ when $k \leq n$.

1.1 Technical Overview

Cunningham's matroid partition algorithm. The auxiliary graph called *exchange graph* plays an important role in almost all combinatorial algorithms for matroid intersection. In matroid intersection algorithms, we begin with an empty set and repeatedly increase the size of the independent set by augmenting along shortest paths in the exchange graph. In the same way, Knuth [20] and Greene-Magnanti [14] give matroid partition algorithms by using the auxiliary graph with $O(np)$ edges, which we call *compressed exchange graph*.

To improve the running time, Cunningham [7] developed *blocking flow* approach for matroid partition and intersection, which is akin to bipartite matching algorithm by Hopcroft-Karp [17]. The blocking flow approach is applied in each phase of the algorithm. In Hopcroft-Karp's bipartite matching algorithm, we find a maximal set of vertex-disjoint shortest paths

and augment along these paths simultaneously. In contrast to this, in a matroid partition algorithm, the augmentations can not be done in parallel, since one augmentation can change the compressed exchange graph. Cunningham showed that we can find multiple augmenting paths of the same length and run all the augmentations in one phase. In Cunningham's matroid partition algorithm, one phase uses only $O(np)$ independence oracle queries (each edge is queried only once in one phase).

Cunningham showed that the number of different lengths of shortest augmenting paths during the algorithm is $O(\sqrt{p})$ and then the number of phases is $O(\sqrt{p})$. Therefore, Cunningham's matroid partition algorithm uses $O(np^{3/2} + kn)$ independence oracle queries in total (enumerating all edges entering sink vertices uses $O(kn)$ independence oracle queries). We note that this query complexity is $O(n^{5/2})$ when $k \leq n$.

Combining blocking flow approach and binary search subroutine. To develop the matroid partition algorithm, given in Theorem 1, that uses $\tilde{O}(kn\sqrt{p})$ independence oracle queries, we combine the blocking flow approach proposed by Cunningham [7] and the binary search procedure proposed by Nguyễn [25] and Chakrabarty et al. [5]. By using the binary search procedure, we obtain an algorithm that uses $\tilde{O}(kn)$ independence oracle queries and performs a breadth first search in the compressed exchange graph. We also obtain an algorithm that uses $\tilde{O}(kn)$ independence oracle queries and runs all the augmentations in a single phase. Since Cunningham showed that the number of phases is $O(\sqrt{p})$, we can easily obtain a matroid partition algorithm that uses $\tilde{O}(kn\sqrt{p})$ independence oracle queries. Our algorithm does not contain technical novelty in a sense that this algorithm is obtained by simply combining Cunningham's technique and the binary search technique by Nguyễn and Chakrabarty et al. Nevertheless, this result is important in a sense that we improve the independence query complexity of a matroid partition algorithm.

Edge Recycling augmentation. In a breadth first search, we need to check, for all vertices v and all indices $i \in [k]$, whether there exists an edge from a vertex v to a vertex $u \in S_i$ in the compressed exchange graph. Then, independence query complexity of a breadth first search of the compressed exchange graph seems to be $\Omega(kn)$, even if we use the binary search procedure. It is not clear whether we can develop a matroid partition algorithm that runs a breadth first search $o(\sqrt{p})$ times, and so, algorithms by the blocking flow approach are now stuck at $\Omega(kn\sqrt{p})$ independence oracle queries. In the setting where $k = \Theta(n)$ and $p = \Theta(n)$, algorithms by the blocking flow approach are stuck at $\Omega(n^{5/2})$ independence oracle queries even if we use the excellent binary search procedure.

In order to break this $O(n^{5/2})$-independence-oracle-query bound, we introduce a new approach *edge recycling augmentation* and develop a matroid partition algorithm whose independence query complexity is sublinear in k. Then we present a matroid partition algorithm that uses $\tilde{O}(n^{7/3})$ independence oracle queries when $k \leq n$.

Our new approach edge recycling augmentation is applied in each phase of the algorithm in the same way as the blocking flow approach. In one phase of edge recycling augmentation, we first compute the edge set E^* in the compressed exchange graph, which uses $O(np)$ independence oracle queries. Then we simply repeat to run a breadth first search and find a shortest path in the compressed exchange graph. This breadth first search is performed by using the information of E^*. The precomputation of E^* may seem too expensive since we have the excellent binary search tool to find edges in the compressed exchange graph. However, we can *recycle* some edges in E^* during the repetition of breadth first searches, which plays an important role in an analysis of our new approach. Note that, edge recycling augmentation runs a breadth first search before every augmentation, while the blocking flow approach runs a breadth first search only once in the beginning of each phase.

Our crucial observation is that all edges entering a vertex in S_i are not changed unless S_i was updated by the augmentation. Then, even after some augmentations, we can use E^* to find edges entering a vertex $u \in S_i$ such that S_i was not updated by the augmentation. This observation is peculiar to the matroid partition. In a breadth first search of the edge recycling augmentation approach, we use the binary search procedure only to find edges entering $u \in S_i$ such that S_i was updated by the augmentation. In one phase, we repeat to run a breadth first search so that the total number of the binary search procedure calls is $O(np)$.

We combine the blocking flow approach algorithm and the edge recycling augmentation approach algorithm. By a careful analysis of independence query complexity, we obtain a matroid partition algorithm that uses $\tilde{O}(k^{1/3}np + kn)$ independence oracle queries.

Note that this edge recycling augmentation approach differs significantly from existing fast matroid intersection algorithms. The key technical contribution of this paper is to introduce this new approach.

1.2 Related Work

Blikstad-Mukhopadhyay-Nanongkai-Tu [3] introduced a new oracle model called dynamic oracle and developed a matroid partitioning algorithm that uses $\tilde{O}((n + r\sqrt{r}) \cdot \mathrm{poly}(k))$ dynamic rank queries, where $r = \max_i \max_{S_i \in \mathcal{I}_i} |S_i|$. Blikstad et al. also obtained an algorithm to solve the k-fold matroid union problem in $\tilde{O}(n\sqrt{r})$ time and dynamic rank queries, which is the special case of the matroid partitioning problem where all matroids $\mathcal{M}_1, \ldots, \mathcal{M}_k$ are identical. Quanrud [27] developed an algorithm that solves the k-fold matroid union problem and uses $\tilde{O}(n^{3/2})$ independence oracle queries for the full range of r and k. Quanrud also considered the k-fold matroid union problem in the more general settings where the elements have integral and real-valued capacities.

For certain special matroids, faster matroid partition algorithms are known. For linear matroids, Cunningham [7] presented an $O(n^3 \log n)$-time algorithm that solves the matroid partitioning problem on $O(n)$ matrices that have n columns and at most n rows. For graphic matroids, the k-forest problem is a special case of the matroid partitioning problem. In the problem, we are given an undirected graph and a positive integer k, and the objective is to find a maximum-size union of k forests. Gabow-Westermann [13] presented an $O(\min\{k^{3/2}\sqrt{nm(m + n\log n)}, k^{1/2}m\sqrt{m + n\log n}, kn^2 \log k, \frac{m^2}{k} \log k\})$-time algorithm to solve the k-forest problem, where n and m denote the number of vertices and edges, respectively. Blikstad et al. [3] and Quanrud [27] independently obtained an $\tilde{O}(m + (kn)^{3/2})$ time algorithm to solve the k-forest problem.

Kawase-Kimura-Makino-Hanna [19] studied matroid partitioning problems for various objective functions.

For the weighted matroid intersection, Huang-Kakimura-Kamiyama [18] developed a technique that transforms any unweighted matroid intersection algorithm into an algorithm that solves the weighted case with an $O(W)$ factor. Huang et al. also presented a $(1 - \epsilon)$ approximation weighted matroid intersection algorithm that uses $\tilde{O}(nr^{3/2}/\epsilon)$ independence oracle queries. Chekuri-Quanrud [6] improved the independence query complexity and presented a $(1 - \epsilon)$ approximation weighted matroid intersection algorithm that uses $O(nr/\epsilon^2)$ independence oracle queries, which can be improved by applying more recent faster approximation unweighted matroid intersection algorithm by Chakrabarty et al. [5] and Blikstad [2]. Tu [29] gave a weighted matroid intersection algorithm that uses $\tilde{O}(nr^{3/4} \log W)$ rank oracle queries, which also uses the binary search procedure proposed by Nguyễn [25] and Chakrabarty et al. [5].

For matroids of rank $n/2$, Harvey [15] showed a lower bound of $(\log_2 3)n - o(n)$ independence oracle queries for matroid intersection. Blikstad-Mukhopadhyay-Nanongkai-Tu [3] showed super-linear $\Omega(n \log n)$ query lower bounds for matroid intersection and partitioning problem in their dynamic-rank-oracle and the independence oracle models.

1.3 Paper Organization

In Section 2, we introduce the notation and the known results for matroid partition and intersection. Next, in Section 3, we present our matroid partition algorithm using blocking flow approach. Then, in Section 4, we present our new approach edge recycling augmentation and our faster matroid partition algorithm for large k. Finally, in Section 5 we conclude by mentioning several open problems relevant to our work.

2 Preliminaries

2.1 Matroids

Notation. For a positive integer a, we denote $[a] = \{1, \ldots, a\}$. For a finite set X, let $\#X$ and $|X|$ denote the cardinality of X, which is also called the *size* of X. We will often write $A+x := A \cup \{x\}$ and $A-x := A \setminus \{x\}$. We will also write $A+B := A \cup B$ and $A-B := A \setminus B$, when no confusion can arise.

Matroid. A pair $\mathcal{M} = (V, \mathcal{I})$ for a finite set V and non-empty $\mathcal{I} \subseteq 2^V$ is called a *matroid* if the following property is satisfied.

(Downward closure) if $S \in \mathcal{I}$ and $S' \subseteq S$, then $S' \in \mathcal{I}$.

(Augmentation property) if $S, S' \in \mathcal{I}$ and $|S'| < |S|$, then there exists $x \in S \setminus S'$ such that $S' + x \in \mathcal{I}$.

A set $S \subseteq V$ is called *independent* if $S \in \mathcal{I}$ and *dependent* otherwise.

Rank. For a matroid $\mathcal{M} = (V, \mathcal{I})$, we define the *rank* of $\mathcal{M}$ as $\mathrm{rank}(\mathcal{M}) = \max\{|S| \mid S \in \mathcal{I}\}$. In addition, for any $S \subseteq V$, we define the *rank* of S as $\mathrm{rank}_{\mathcal{M}}(S) = \max\{|T| \mid T \subseteq S, T \in \mathcal{I}\}$.

Matroid Intersection. Given two matroids $\mathcal{M}' = (V, \mathcal{I}')$, $\mathcal{M}'' = (V, \mathcal{I}'')$, we define their *intersection* by $(V, \mathcal{I}' \cap \mathcal{I}'')$. The *matroid intersection problem* asks us to find the largest common independent set, whose cardinality we denote by r. Note that the intersection of matroids is not a matroid in general and the problem to find a maximum common independent set of more than two matroids is NP-hard.

Matroid Partition (Matroid Union). Given k matroids $\mathcal{M}_1 = (V, \mathcal{I}_1), \ldots, \mathcal{M}_k = (V, \mathcal{I}_k)$, $S \subseteq V$ is called *partitionable* if there exists a partition $(S_1, \ldots, S_k)$ of S such that $S_i \in \mathcal{I}_i$ for $i \in [k]$. The *matroid partitioning problem* asks us to find the largest partitionable set, whose cardinality we denote by p. Let $\tilde{\mathcal{I}}$ be the family of partitionable subset of V. Then, $(V, \tilde{\mathcal{I}})$ is called the *union* or *sum* of k matroids $\mathcal{M}_1 \ldots, \mathcal{M}_k$. Note that Nash-Williams Theorem [24] states that the union $(V, \tilde{\mathcal{I}})$ of the k matroids is also a matroid.

Oracles. Throughout this paper, we assume that we can only access a matroid $\mathcal{M} = (V, \mathcal{I})$ through an oracle. Given a subset $S \subseteq V$, an *independence* oracle outputs whether $S \in \mathcal{I}$ or not. Given a subset $S \subseteq V$, a *rank* oracle outputs $\mathrm{rank}_{\mathcal{M}}(S)$. Since one query of the rank oracle can determine whether a given subset is independent, the rank oracle is more powerful than the independence oracle.

Binary Search Technique. Chakrabarty-Lee-Sidford-Singla-Wong [5] showed that the following procedure can be implemented efficiently by using binary search in the independence oracle model. (This was developed independently by Nguyễn [25].) Given a matroid $\mathcal{M} = (V, \mathcal{I})$, an independent set $S \in \mathcal{I}$, an element $v \in V \setminus S$, and $B \subseteq S$, the objective is to find an element $u \in S$ that is exchangeable with v (that is, $S + v - u \in \mathcal{I}$) or conclude there is no such an element. We skip the proof in this paper; see [5, Section 3] for a proof.

▶ **Lemma 4** (Edge Search via Binary search, Chakrabarty et al. [5], Nguyễn [25]). *There exists an algorithm* `FindOutEdge` *which, given a matroid $\mathcal{M} = (V, \mathcal{I})$, an independent set $S \in \mathcal{I}$, an element $v \in V \setminus S$, and $B \subseteq S$, finds an element $u \in B$ such that $S + v - u \in \mathcal{I}$ or otherwise determine that no such element exists, and uses $O(\log |B|)$ independence queries.*

2.2 Techniques for Matroid Intersection

Here we provide known results about the matroid intersection.

▶ **Definition 5** (Exchange Graph). *Consider a common independent set $S \in \mathcal{I}' \cap \mathcal{I}''$. The* exchange graph *is defined as a directed graph $G(S) = (V \cup \{s, t\}, E)$, with $s, t \notin V$ and $E = E' \cup E'' \cup E_s \cup E_t$, where*

$$\begin{aligned} E' &= \{(u, v) \mid u \in S, v \in V \setminus S, S - u + v \in \mathcal{I}'\}, \\ E'' &= \{(v, u) \mid u \in S, v \in V \setminus S, S - u + v \in \mathcal{I}''\}, \\ E_s &= \{(s, v) \mid v \in V \setminus S, S + v \in \mathcal{I}'\}, \text{ and} \\ E_t &= \{(v, t) \mid v \in V \setminus S, S + v \in \mathcal{I}''\}. \end{aligned}$$

▶ **Lemma 6** (Shortest Augmenting Path). *Let $s, v_1, v_2, \ldots, v_{l-1}, t$ be a shortest (s, t)-path in the exchange graph $G(S)$ relative to a common independent set $S \in \mathcal{I}' \cap \mathcal{I}''$. Then $S' = S + v_1 - v_2 + \cdots - v_{l-2} + v_{l-1} \in \mathcal{I}' \cap \mathcal{I}''$.*

In a matroid intersection algorithm, we begin with an empty set S. Then we repeat to find an augmenting path in the exchange graph $G(S)$ and to update the current set S. If there is no (s, t)-path in the exchange graph $G(S)$, then S is a common independent set of maximum size. If there is an (s, t)-path in the exchange graph $G(S)$, then we pick a shortest path and obtain a common independent set $S' \in \mathcal{I}' \cap \mathcal{I}''$ of $|S| + 1$ elements.

Cunningham's matroid intersection algorithm [7] and recent faster matroid intersection algorithms [2, 4, 5, 25] rely on the following lemma.

▶ **Lemma 7** (Cunningham [7]). *For any two matroids $\mathcal{M}' = (V, \mathcal{I}')$ and $\mathcal{M}'' = (V, \mathcal{I}'')$, if the length of the shortest augmenting path in exchange graph $G(S)$ relative to a common independent set $S \in \mathcal{I}' \cap \mathcal{I}''$ is at least d, then $|S| \geq (1 - \frac{O(1)}{d}) \cdot r$, where r is the size of a largest common independent set.*

Cunningham's [7] matroid intersection algorithm by the blocking flow approach relies on the following monotonicity lemma.

▶ **Lemma 8** (Monotonicity Lemma, [5, 7, 16, 26]). *For any two matroids $\mathcal{M}' = (V, \mathcal{I}')$ and $\mathcal{M}'' = (V, \mathcal{I}'')$, suppose we obtain a common independent set $S' \in \mathcal{I}' \cap \mathcal{I}''$ by augmenting $S \in \mathcal{I}' \cap \mathcal{I}''$ along a shortest augmenting path in $G(S)$. Note that $|S'| > |S|$. Let d denote the distance in $G(S)$ and d' denote the distance in $G(S')$. Then for all $v \in V$,*

- **(i)** *If $d(s, v) < d(s, t)$, then $d'(s, v) \geq d(s, v)$. If $d(v, t) < d(s, t)$, then $d'(v, t) \geq d(v, t)$.*
- **(ii)** *If $d(s, v) \geq d(s, t)$, then $d'(s, v) \geq d(s, t)$. If $d(v, t) \geq d(s, t)$, then $d'(v, t) \geq d(s, t)$.*

2.3 Compressed Exchange Graph for Matroid Partition

The matroid partitioning problem can be solved by a matroid intersection algorithm. Let $\hat{V} = V \times [k]$, and define

$$\hat{\mathcal{I}}' = \{\hat{I} \subseteq \hat{V} \mid \forall v \in V, \#\{i \in [k] \mid (v,i) \in \hat{I}\} \leq 1\},$$
$$\hat{\mathcal{I}}'' = \{\hat{I} \subseteq \hat{V} \mid \forall i \in [k], \{v \in V \mid (v,i) \in \hat{I}\} \in \mathcal{I}_i\}.$$

Then, $\hat{\mathcal{M}}' = (\hat{V}, \hat{\mathcal{I}}')$ is a partition matroid. Since $\hat{\mathcal{M}}'' = (\hat{V}, \hat{\mathcal{I}}'')$ is the direct sum of matroids $(V, \mathcal{I}_i)$ for all $i \in [k]$, it is also a matroid. Then, the family of partitionable subsets of V can be represented as

$$\{S \subseteq V \mid \exists \pi \colon S \to [k], \{(v, \pi(v)) \mid v \in S\} \in \hat{\mathcal{I}}' \cap \hat{\mathcal{I}}''\}.$$

Therefore, we can solve the matroid partitioning problem by computing a common independent set of maximum size in $\mathcal{I}'$ and $\mathcal{I}''$. However, we might use too many independence oracle queries when solving the matroid partitioning problem by using this reduction to the matroid intersection problem. This is due to the following reasons. When solving the matroid intersection problem that was reduced by the matroid partitioning problem, the size of the ground set of that matroid intersection problem is $O(kn)$, and then the number of edges in the exchange graph is $O(knp)$, which depends heavily on k. Furthermore, since we consider the total query complexity of the independence oracle of each matroid $\mathcal{M}_i = (V, \mathcal{I}_i)$ for all $i \in [k]$, the query complexity of the independence query of the matroid $\hat{\mathcal{M}}'' = (\hat{V}, \hat{\mathcal{I}}'')$ also depends heavily on k.

Then, to improve the running time, Knuth [20] and Greene-Magnanti [14] give a matroid partition algorithm that uses the following auxiliary graph with $O(np)$ edges, which we call *compressed exchange graph*.

▶ **Definition 9** (Compressed Exchange Graph [7, 14, 20]). *Consider a partition $(S_1, \ldots, S_k)$ of $S \subseteq V$ such that $S_i \in \mathcal{I}_i$ for all $i \in [k]$. The* compressed exchange graph *is defined as a directed graph $G(S_1, \ldots, S_k) = (V \cup \{s, t_1, \ldots, t_k\}, E)$, with $s, t_1, \ldots, t_k \notin V$ and $E = E' \cup E_s \cup E_t$, where*

$$E' = \{(v,u) \mid \exists i \in [k], u \in S_i, S_i + v \notin \mathcal{I}_i, S_i + v - u \in \mathcal{I}_i\},$$
$$E_s = \{(s,v) \mid v \in V \setminus S\}, \text{ and}$$
$$E_t = \bigcup_{i=1}^{k} \{(v, t_i) \mid v \in V \setminus S_i, S_i + v \in \mathcal{I}_i\}.$$

We set $T = \{t_1, \ldots, t_k\}$.

In the matroid partition algorithm, we begin with an empty set S and initialize $S_i = \emptyset$ for all $i \in [k]$. If there is no vertex in T which is reachable from s in the compressed exchange graph $G(S_1, \ldots, S_k)$, then S is a partitionable set of maximum size. If there is a path from s to T in the compressed exchange graph, then we pick a shortest path $s, v_1, \ldots, v_{l-1}, t_j$. Then we can obtain a partitionable set $S' = S + v_1$ and a partition $(S'_1, \ldots, S'_k)$ of S' such that $S'_i \in \mathcal{I}_i$ for all $i \in [k]$. The validity of the algorithm follows from the following two lemmas, which we use throughout this paper. Cunningham [7] showed these lemmas by using the equivalence of the compressed exchange graph for the matroid partition and the exchange graph for the reduced matroid intersection; see [28, Theorem 42.4] for a direct proof that does not use the reduction to the matroid intersection.

▶ **Lemma 10.** *Given a partition $(S_1, \dots, S_k)$ of S such that $S_i \in \mathcal{I}_i$ for all $i \in [k]$, there exists a partitionable set S' whose size is at least $|S|+1$ if and only if there is a vertex $t_j \in T$ that is reachable from s in the compressed exchange graph $G(S_1, \dots, S_k)$.*

▶ **Lemma 11** (Shortest Augmenting Path)**.** *Let $s, v_1, v_2, \dots, v_{l-1}, t_j$ be a shortest (s,T)-path in the compressed exchange graph $G(S_1, \dots, S_k)$. Then $S' = S + v_1$ is a partitionable set.*

We can construct a partition $(S'_1, \dots, S'_k)$ of S' from a partition $(S_1, \dots, S_k)$ of S and an augmenting path in the compressed exchange graph by using the following procedure `Update` (Algorithm 1).

Algorithm 1 `Update`.

Input: a partition $(S_1, \dots, S_k)$ of S $(\subseteq V)$ such that $S_i \in \mathcal{I}_i$ for all $i \in [k]$. an augmenting path $s, v_1, \dots, v_{l-1}, t_j$.
Output: a partition $(S'_1, \dots, S'_k)$ of S' $(\subseteq V)$ such that $S'_i \in \mathcal{I}_i$ for all $i \in [k]$ and $S' = S + v_1$.

For all $i \in [k]$, set $S'_i \leftarrow S_i$
For all $v \in S$, denote by $\pi(v)$ the index such that $v \in S_{\pi(v)}$
for $i \in [l-2]$ **do**
  $S'_{\pi(v_{i+1})} \leftarrow S'_{\pi(v_{i+1})} + v_i - v_{i+1}$
$S'_j \leftarrow S'_j + v_{l-1}$
return a partition $(S'_1, \dots, S'_k)$ of S'

Cunningham [7] observes that the equivalence between the exchange graph for the matroid intersection of two matroids $\hat{\mathcal{M}}' = (\hat{V}, \hat{\mathcal{I}}')$ and $\hat{\mathcal{M}}'' = (\hat{V}, \hat{\mathcal{I}}'')$ and the compressed exchange graph for the matroid partition of k matroids $(V, \mathcal{I}_1), \dots, (V, \mathcal{I}_k)$ to prove the Lemmas 12 and 13 and to develop an efficient algorithm for matroid partition that employs the blocking flow approach. For a fixed partition $(S_1, \dots, S_k)$ of S and an element $v \in S$, let $\pi(v)$ be the index such that $v \in S_{\pi(v)}$. We also denote by $\hat{S}$ the set $\{(v, \pi(v)) \in \hat{V} \mid v \in S\}$. A path $s, v_1, v_2, \dots, v_{l-1}, t_j$ in the compressed exchange graph for the matroid partition corresponds to a path $s, (v_1, \pi(v_2)), (v_2, \pi(v_2)), (v_2, \pi(v_3)), \dots, (v_{l-1}, \pi(v_{l-1})), (v_{l-1}, j), t$ in the exchange graph for the matroid intersection. Then, for all elements $v \in S$, we have $d_{G(S_1,\dots,S_k)}(s, v) = 1 + \frac{1}{2} d_{G(\hat{S})}(s, (v, \pi(v)))$ and $d_{G(S_1,\dots,S_k)}(v, T) = \frac{1}{2} d_{G(\hat{S})}((v, \pi(v)), t)$. We also have $d_{G(S_1,\dots,S_k)}(s, T) = 1 + \frac{1}{2} d_{G(\hat{S})}(s, t)$.

Cunningham [7] uses the following two lemmas to develop an efficient matroid partition algorithm by using blocking flow approach. These lemmas can be shown from the correspondence between the exchange graph and the compressed exchange graph. We also use these two lemmas in our fast matroid partition algorithms.

▶ **Lemma 12** (Cunningham [7])**.** *Given a partition $(S_1, \dots, S_k)$ of S such that $S_i \in \mathcal{I}_i$ for all $i \in [k]$. If the length of a shortest augmenting path in the compressed exchange graph $G(S_1, \dots, S_k)$ is at least d, then $|S| \geq (1 - \frac{O(1)}{d}) \cdot p$, where p is the size of largest partitionable set.*

▶ **Lemma 13** (Monotonicity Lemma [5, 7, 16, 26])**.** *Suppose we obtain a partition $(S'_1, \dots, S'_k)$ of S' by augmenting a partition $(S_1, \dots, S_k)$ of S along a shortest augmenting path in $G(S_1, \dots, S_k)$. Note that $|S'| > |S|$. let d denote the distance in $G(S_1, \dots, S_k)$ and d' denote the distance in $G(S'_1, \dots, S'_k)$. Then for all $v \in V$,*

(i) *If $d(s,v) < d(s,T)$, then $d'(s,v) \geq d(s,v)$. If $d(v,T) < d(s,T)$, then $d'(v,T) \geq d(v,T)$.*
(ii) *If $d(s,v) \geq d(s,T)$, then $d'(s,v) \geq d(s,T)$. If $d(v,T) \geq d(s,T)$, then $d'(v,T) \geq d(s,T)$.*

As we will see later, we use the binary search technique given in Lemma 4 to find edges in the compressed exchange graph under the independence oracle model. Note that the procedure `FindOutEdge`$(\mathcal{M}_i, S_i, v, B)$ gives us an efficient way to find edges from the vertex v to a vertex $u \in B (\subseteq S_i)$ in the compressed exchange graph.

3 Blocking Flow Algorithm

In this section, we provide our matroid partition algorithms in the independence oracle model, which is obtained by simply combining the *blocking flow* approach proposed by Cunningham [7] and the binary search search procedure proposed by Nguyễn [25] and Chakrabarty-Lee-Sidford-Singla-Wong [5]. In the full version of this paper, we also present a fast matroid partition algorithm using blocking flow approach in the rank oracle model.

3.1 Blocking Flow Algorithm using Independence Oracle

In this subsection we present our matroid partition algorithm using the blocking flow approach in the independence oracle model. We show the following theorem, which implies Theorem 1.

▶ **Theorem 14.** *There is an algorithm that uses $O(kn\sqrt{p}\log p)$ independence oracle queries and solves the matroid partitioning problem.*

This result improves upon the previously known matroid partition algorithm by Cunningham [7] when $k = o(p)$.

For the proof, we first provide the procedure `GetDistanceIndependence` (Algorithm 2) that efficiently finds distances from s to every vertex in the compressed exchange graph. This algorithm simply runs a breadth first search by using the procedure `FindOutEdge`.

Algorithm 2 `GetDistanceIndependence`.

Input: a partition $(S_1, \ldots, S_k)$ of S $(\subseteq V)$ such that $S_i \in \mathcal{I}_i$ for all $i \in [k]$
Output: $d \in \mathbb{R}^{V \cup \{s\} \cup T}$ such that for $v \in V \cup \{s\} \cup T$, d_v is the distance from s to v in $G(S_1, \ldots, S_k)$

$d_s \leftarrow 0$
For all $v \in V \setminus S$ let $d_v \leftarrow 1$
For all $v \in S$ let $d_v \leftarrow \infty$
For all $i \in [k]$ let $d_{t_i} \leftarrow \infty$
$Q \leftarrow \{v \in V \setminus S\}$ // Q : `queue`
For all $i \in [k]$ let $B_i \leftarrow S_i$
while $Q \neq \emptyset$ **do**
  Let v be the element added to Q earliest
  $Q \leftarrow Q - v$
  for $i \in [k]$ *with* $d_{t_i} = \infty$ **do**
    if $v \notin S_i$ *and* $S_i + v \in \mathcal{I}_i$ **then**
      $d_{t_i} \leftarrow d_v + 1$
  for $i \in [k]$ *with* $v \notin S_i$ **do**
    while $u =$ `FindOutEdge`$(\mathcal{M}_i, S_i, v, B_i)$ *satisfies* $u \neq \emptyset$ **do**
      $Q \leftarrow Q + u$
      $d_u \leftarrow d_v + 1$
      $B_i \leftarrow B_i - u$
return d

▶ **Lemma 15** (Breadth First Search using Independence Oracle). *Given a partition $(S_1, \dots, S_k)$ of S $(\subseteq V)$ such that $S_i \in \mathcal{I}_i$ for all $i \in [k]$, the procedure* `GetDistanceIndependence` *(Algorithm 2) outputs $d \in \mathbb{R}^{V \cup \{s\} \cup T}$ such that, for $v \in V \cup \{s\} \cup T$, d_v is the distance from s to v in the compressed exchange graph $G(S_1, \dots, S_k)$. The procedure* `GetDistanceIndependence` *uses $O(kn \log p)$ independence oracle queries.*

Proof. The procedure `GetDistanceIndependence` simply performs a breadth first search in the compressed exchange graph $G(S_1, \dots, S_k)$. Thus, the procedure `GetDistanceIndependence` correctly computes distances from s in $G(S_1, \dots, S_k)$. Note that each vertex $v \in V$ is added to Q at most once and each vertex $v \in S$ is removed from $B_{\pi(v)}$ at most once. Thus, the number of independence oracle queries used in Line 11 is $O(kn)$. The number of `FindOutEdge` calls that do not output $\emptyset$ is $O(p)$, and the number of `FindOutEdge` calls that output $\emptyset$ is $O(kn)$. Hence, by Lemma 4, the number of independence oracle queries used in Line 14 is $O(kn \log p)$, which completes the proof. ◀

Next we provide our augmentation subroutine for our faster matroid partition algorithm. We implement Cunningham's [7] blocking flow approach for matroid partition by using the binary search procedure proposed by Nguyễn [25] and Chakrabarty et al. [5]. This algorithm is similar to Chakrabarty et al.'s matroid intersection algorithm in the rank oracle model [5]. The implementation is described as `BlockFlowIndependence` in the full version of this paper.

In the procedure `BlockFlowIndependence`, given a partition $(S_1, \dots, S_k)$ of S, we first compute the distances from s to every vertex in the compressed exchange graph $G(S_1, \dots, S_k)$ using `GetDistanceIndependence` (Algorithm 2). By using these distances, we divide V into sets $L_1, L_2, \dots$, where each L_i has all vertices v such that the distance from s to v is i in the compressed exchange graph $G(S_1, \dots, S_k)$. Then we search a path $s, a_1, a_2, \dots, a_{d_T-1}, a_{d_T}$ in the compressed exchange graph $G(S_1, \dots, S_k)$, where $a_i \in L_i$ for all $i \in [d_T - 1]$. If we found such a path, we augment a partition $(S_1, \dots, S_k)$ of S and remove a_i from L_i for all $i \in [d_T - 1]$. Then we search a new path again until no (s, T)-path of length d_T can be found. During the search for such a path, if the procedure concludes that some vertex in L_i is not on such a path, then it removes the vertex from L_i. Note that we write $d_T = \min(d_{t_1}, \dots, d_{t_k})$.

▶ **Lemma 16** (Blocking Flow using Independence Oracle). *Given a partition $(S_1, \dots, S_k)$ of S $(\subseteq V)$ such that $S_i \in \mathcal{I}_i$ for all $i \in [k]$, the procedure* `BlockFlowIndependence` *outputs a partition $(S'_1, \dots, S'_k)$ of S' $(\subseteq V)$ such that $S'_i \in \mathcal{I}_i$ for all $i \in [k]$ and $|S'| > |S|$ and $d_{G(S'_1,\dots,S'_k)}(s, T) \geq d_{G(S_1,\dots,S_k)}(s, T) + 1$, or a partition $(S_1, \dots, S_k)$ of S if no such S' exists. The procedure* `BlockFlowIndependence` *uses $O(kn \log p)$ independence oracle queries.*

We provide a proof of Lemma 16 in the full version of this paper.

Now we provide a proof of Theorem 14 by using Lemma 12. In our matroid partition algorithm, we simply apply `BlockFlowIndependence` repeatedly until no (s, T)-path can be found.

Proof of Theorem 14. In our algorithm, we start with $S = \emptyset$ and initialize $S_i = \emptyset$ for all $i \in [k]$. Then we apply `BlockFlowIndependence` repeatedly to augment the current partition $(S_1, \dots, S_k)$ of S until no (s, T)-path can be found in the compressed exchange graph $G(S_1, \dots, S_k)$.

Since each execution of `BlockFlowIndependence` strictly increases $d_{G(S_1,\dots,S_k)}(s, T)$, we have $d_{G(S_1,\dots,S_k)}(s, T) = \Omega(\sqrt{p})$ after $O(\sqrt{p})$ executions of `BlockFlowIndependence`. Lemma 12 implies that, if $d_{G(S_1,\dots,S_k)}(s, T) = \Omega(\sqrt{p})$, then $|S| \geq p - O(\sqrt{p})$. Then the total number of `BlockFlowIndependence` executions is $O(\sqrt{p}) + O(\sqrt{p}) = O(\sqrt{p})$ in the entire matroid partition algorithm. Lemma 16 implies that one execution of `BlockFlowIndependence` uses $O(kn \log p)$ independence oracle queries, which completes the proof. ◀

In the same way as Chakrabarty et al.'s matroid intersection algorithm in the rank oracle model [5], we easily obtain the following theorem.

▶ **Theorem 17.** *For any $\epsilon > 0$, there is an algorithm that uses $O(kn\epsilon^{-1} \log p)$ independence oracle queries and finds a $(1-\epsilon)$ approximation of the largest partitionable set of k matroids.*

Proof. Similar to the proof of Theorem 14, we start with $S = \emptyset$ and initialize $S_i = \emptyset$ for all $i \in [k]$ and apply `BlockFlowIndependence` repeatedly to augment the current partition $(S_1, \ldots, S_k)$ of S. The only difference is that we apply `BlockFlowIndependence` only ϵ^{-1} times, which uses $O(kn\epsilon^{-1} \log p)$ independence oracle queries.

Each execution of `BlockFlowIndependence` strictly increases $d_{G(S_1,\ldots,S_k)}(s,T)$. Thus, after ϵ^{-1} executions of `BlockFlowIndependence`, we have $d_{G(S_1,\ldots,S_k)}(s,T) = \Omega(\epsilon^{-1})$. Lemma 12 implies that, if $d_{G(S_1,\ldots,S_k)}(s,T) = \Omega(\epsilon^{-1})$, then $|S| \geq p - O(p\epsilon)$, which completes the proof. ◀

4 Faster Algorithm for Large k

In this section, we present an algorithm that uses $o(kn\sqrt{p})$ independence oracle queries when k is large. In subsection 3.1, we have presented the algorithm (Algorithm 2), which runs a breadth first search in the compressed exchange graph and uses $O(kn \log p)$ independence oracle queries. In the evaluation of the independence query complexity of the matroid partition algorithm by the *blocking flow* approach given in section 3, a key observation is that the number of different lengths of shortest augmenting paths during the algorithm is $O(\sqrt{p})$. For now, it is not clear whether we can obtain a matroid partition algorithm that runs a breadth first search $o(\sqrt{p})$ times. Then the blocking flow approaches are now stuck at $\Omega(kn\sqrt{p})$ independence oracle queries. To overcome this barrier and improve upon the algorithm that uses $O(kn\sqrt{p} \log p)$ independence oracle queries given in Theorem 14, we introduce a new approach called *edge recycling augmentation*, which can perform breadth first searches with fewer total independence oracle queries. Our new approach can be attained through new ideas: an efficient utilization of the binary search procedure `FindOutEdge` and a careful analysis of the number of independence oracle queries by using Lemma 12. By combining an algorithm by the blocking flow approach and an algorithm by the edge recycling augmentation approach, we obtain the following theorem, which implies Theorem 2.

▶ **Theorem 18.** *There is an algorithm that uses $O(k^{1/3}np \log p + kn)$ independence oracle queries and solves the matroid partitioning problem. When $k \leq n$, the number of queries is $\tilde{O}(n^{7/3})$.*

This theorem implies that we obtain a matroid partition algorithm that uses $o(kn\sqrt{p})$ independence oracle queries when $k = \omega(p^{3/4})$. We note that this algorithm requires $O(k^{2/3}np)$ time complexity other than independence oracle queries.

In Section 4.1, we present our new approach edge recycling augmentation, and in Section 4.2, we present our faster matroid partition algorithm for large k and give a proof of Theorem 18.

4.1 Edge Recycling Augmentation

In order to select appropriate parameters for our algorithm, we have to determine the value of p. However, the size p of a largest partitionable set is unknown before running the algorithm. Instead of using the exact value of p, we use a $\frac{1}{2}$-approximation $\bar{p}$ for p (that is $\bar{p} \leq p \leq 2\bar{p}$), which can be computed using $O(kn)$ independence oracle queries. It is well known that a

$\frac{1}{2}$-approximate solution for the matroid intersection problem can be found by the following simple greedy algorithm; see [21, Proposition 13.26]. We begin with an empty set. For each element in the ground set, we check whether adding it to the set would result in a common independent set. If it does, we add it to the set. Finally, we obtain a maximal common independent set. We convert this algorithm into the following $\frac{1}{2}$-approximation algorithm (Algorithm 3) for the matroid partitioning problem by utilizing the reduction from matroid partition to the intersection of two matroids $\hat{\mathcal{M}}' = (\hat{V}, \hat{\mathcal{I}}')$ and $\hat{\mathcal{M}}'' = (\hat{V}, \hat{\mathcal{I}}'')$ given in subsection 2.3.

Algorithm 3 $\frac{1}{2}$`-ApproximationMatroidPartition`.

```
1 For all i ∈ [k] let S_i ← ∅
2 for i ← 1 to k do
3 |   for v ∈ V \ (⋃_{j=1}^{i-1} S_j) do
4 |   |   if S_i + v ∈ I_i then
5 |   |   |   S_i ← S_i + v
6 return p̄ = |⋃_{i=1}^{k} S_i|
```

Now we present our new approach *Edge Recycling Augmentation*. Our new approach edge recycling augmentation is applied in each phase of the algorithm. One phase of edge recycling augmentation is described as `EdgeRecyclingAugmentation` (Algorithm 5).

In `EdgeRecyclingAugmentation`, we first compute the edges $E^*(\subseteq V \times S)$ in the compressed exchange graph $G(S_1, \ldots, S_k)$, which uses $O(np)$ independence oracle queries. Note that the compressed exchange graph may be changed by augmentations, that is, augmentations may add or delete several edges in the compressed exchange graph, and so, taking one augmenting path may destroy the set E^* of the edges. However, we notice that we can *recycle* some part of the edge set E^* after the augmentations, which is peculiar to the matroid partition.

In `EdgeRecyclingAugmentation`, we simply repeat to run a breadth first search and then to augment the partitionable set. Unlike `GetDistanceIndependence` (Algorithm 2) in Section 3.1, our BFS *recycles* the precomputed edge set E^*. In one phase, we keep a set J of all indices i such that S_i was updated by the augmentations. Our crucial observation is that no edges, in the compressed exchange graph, entering a vertex in S_i are changed by the augmentations unless augmenting paths contain a vertex in $S_i \cup \{t_i\}$. In contrast to `GetDistanceIndependence` that uses the binary search procedure `FindOutEdge` for all indices $i \in [k]$, our new BFS procedure uses `FindOutEdge` only for indices $i \in J$. We can use E^* to search edges entering a vertex in S_i with $i \notin J$. Then, the BFS based on the ideas described above can be implemented as `EdgeRecyclingBFS` (Algorithm 4).

We also provide a new significant analysis of the number of independence oracle queries in entire our matroid partition algorithm. In `EdgeRecyclingAugmentation`, we repeat to run the breadth first search `EdgeRecyclingBFS` so that the total calls of the binary search procedure is $O(np)$. Then, the number of independence oracle queries used by `EdgeRecyclingBFS` in one call of `EdgeRecyclingAugmentation` is almost equal to the one used by the precomputation of E^*. Hence, one call of `EdgeRecyclingAugmentation` uses $\tilde{O}(np)$ independence oracle queries. The number of calls of `EdgeRecyclingBFS` in `EdgeRecyclingAugmentation` depends on how many edges can not be recycled. Thus, to determine the number of calls of `EdgeRecyclingBFS`, we use the value *sum* in the implementation of `EdgeRecyclingAugmentation` (Algorithm 5). In the entire matroid partition

algorithm, we apply `EdgeRecyclingAugmentation` repeatedly. Then, we can obtain a matroid partition algorithm that uses $\tilde{O}(np^{3/2} + kn)$ independence oracle queries. Furthermore, by combining this with the blocking flow approach, the number of total calls of `EdgeRecyclingAugmentation` in the entire matroid partition algorithm can be $O(k^{1/3})$. This leads to obtain a matroid partition algorithm that uses $\tilde{O}(k^{1/3}np + kn)$ independence oracle queries. This analysis differs significantly from that of existing faster matroid intersection algorithms.

For $i \in [k]$, let $F_i (\subseteq V)$ denote the set of vertices adjacent to $t_i \in T$. We first compute the set F_i for all $i \in [k]$ using $O(kn)$ independence oracle queries. Note that, after one augmentation, they can be updated using only $O(n)$ independence oracle queries.

In the following two lemmas, we show the correctness and the independence query complexity of the procedure `EdgeRecyclingAugmentation` (Algorithm 5).

▶ **Lemma 19.** *Given a partition $(S_1, \dots, S_k)$ of $S (\subseteq V)$ such that $S_i \in \mathcal{I}_i$ for all $i \in [k]$, the procedure* `EdgeRecyclingAugmentation` *(Algorithm 5) outputs a partition $(S'_1, \dots, S'_k)$ of $S' (\subseteq V)$ such that $S'_i \in \mathcal{I}_i$ for all $i \in [k]$ and $|S'| \geq |S|$.*

Proof. To prove the correctness of `EdgeRecyclingAugmentation`, we prove the following invariants at the beginning of any iteration of the while loop.

- **(i)** For all $i \in [k] \setminus J$ and all $(v, u) \in V \times S_i$, we have $(v, u) \in E^*$ if and only if $S_i + v - u \in \mathcal{I}_i$ and $S_i + v \notin \mathcal{I}_i$.
- **(ii)** For all $i \in [k]$ and all $v \in V$, we have $v \in F_i$ if and only if $S_i + v \in \mathcal{I}_i$ and $v \notin S_i$.
- **(iii)** For all $i \in [k]$, we have $S_i \in \mathcal{I}_i$.

The invariant is true before the execution of `EdgeRecyclingAugmentation`. Now, assume that the invariants (i)–(iii) hold true at the beginning of an iteration of the while loop. Let a partition $(S_1^{\mathrm{old}}, \dots, S_k^{\mathrm{old}})$ of S^{old} be the partition before the execution of Line 13 and a partition $(S_1^{\mathrm{new}}, \dots, S_k^{\mathrm{new}})$ of S^{new} be the partition after the execution of Line 13. For all $i \in [k] \setminus J$, we have $S_i^{\mathrm{old}} = S_i^{\mathrm{new}}$. Then, invariant (i) remains true. For all $i \in [k] \setminus \{j\}$, we have $|S_i^{\mathrm{old}}| = |S_i^{\mathrm{new}}|$. Hence, for all $i \in [k] \setminus \{j\}$ and all $v \notin S_i^{\mathrm{old}} \cup S_i^{\mathrm{new}}$, we have $S_i^{\mathrm{new}} + v \in \mathcal{I}_i$ if and only if $S_i^{\mathrm{old}} + v \in \mathcal{I}_i$; see [28, Corollary 39.13a] for a proof. Furthermore, for all $i \in [k] \setminus \{j\}$, we have $S_i^{\mathrm{old}} + v \notin \mathcal{I}_i$ for all $v \in S_i^{\mathrm{new}} \setminus S_i^{\mathrm{old}}$ and $S_i^{\mathrm{new}} + v \notin \mathcal{I}_i$ for all $v \in S_i^{\mathrm{old}} \setminus S_i^{\mathrm{new}}$; see [7, Section 5]. Then, invariant (ii) remains true. The procedure `EdgeRecyclingBFS` simply finds a BFS-tree rooted at s by a breadth first search. Thus, if the invariants (i)–(iii) are true, then the procedure `EdgeRecyclingBFS` correctly computes BFS-tree rooted at s. Then, the path P that `EdgeRecyclingBFS` outputs in Line 5 is a shortest augmenting path. Hence, by Lemma 11, invariant (iii) remains true. ◀

▶ **Lemma 20.** *The procedure* `EdgeRecyclingAugmentation` *(Algorithm 5) uses $O(np \log p)$ independence oracle queries.*

Proof of Lemma 20. The number of independence oracle queries used in Line 3 is $O(np)$. Furthermore, the number of independence oracle queries used in Line 14 is $O(np)$, because the number of iterations of the while loop is bounded by p.

Now we show that the number of `FindOutEdge` calls in the entire procedure `EdgeRecyclingAugmentation` is $O(np)$.

In the procedure `EdgeRecyclingBFS`$((S_1, \dots, S_k), E^*, J, \bigcup_{i=1}^k F_i)$, each vertex $v \in V$ is added to Q at most once and each vertex $v \in S$ is removed from $B_{\pi(v)}$ at most once, where $\pi(v)$ is the index such that $v \in S_{\pi(v)}$. This means that the number of `FindOutEdge` calls that do not output $\emptyset$ is bounded by p, and the number of `FindOutEdge` calls that output $\emptyset$ is bounded by $n \cdot |J|$. Then, the number of `FindOutEdge` calls in the procedure `EdgeRecyclingBFS` is $O(p + n \cdot |J|)$.

Algorithm 4 `EdgeRecyclingBFS`.

Input: a partition $(S_1, \ldots, S_k)$ of S $(\subseteq V)$ such that $S_i \in \mathcal{I}_i$ for all $i \in [k]$, a set $E \subseteq V \times S$, a set $J \subseteq [k]$, a set $F = \{v \in V \mid \exists i \in [k], v \notin S_i, S_i + v \in \mathcal{I}_i\}$.
Output: An augmenting (s, T)-path in $G(S_1, \ldots, S_k)$ if one exists.

$Q \leftarrow \{v \in V \setminus S\}$ // Q : `queue`
$B_i \leftarrow S_i$ for all $i \in [k]$
while $Q \neq \emptyset$ **do**
  Let v be the element added to Q earliest
  $Q \leftarrow Q - v$
  if $v \in F$ **then**
    return the shortest augmenting path in the BFS-tree.
  for $i \in J$ **do**
    while $u =$ `FindOutEdge`$(\mathcal{M}_i, S_i, v, B_i)$ *satisfies* $u \neq \emptyset$ **do**
      $Q \leftarrow Q + u$
      $B_i \leftarrow B_i - u$
  for $i \in [k] \setminus J$ **do**
    for $u \in B_i$ *such that* $(v, u) \in E$ **do**
      $Q \leftarrow Q + u$
      $B_i \leftarrow B_i - u$
return NO PATH EXISTS

Algorithm 5 `EdgeRecyclingAugmentation`.

Input: a partition $(S_1, \ldots, S_k)$ of S $(\subseteq V)$ such that $S_i \in \mathcal{I}_i$ for all $i \in [k]$, sets $F_i = \{v \in V \setminus S_i \mid S_i + v \in \mathcal{I}_i\}$ for all $i \in [k]$
Output: a partition $(S'_1, \ldots, S'_k)$ of S' $(\subseteq V)$ such that $S'_i \in \mathcal{I}_i$ for all $i \in [k]$ and $|S'| \geq |S|$.

$sum \leftarrow 0$
$J \leftarrow \emptyset$
$E^* \leftarrow \{(v, u) \in V \times S \mid \exists i \in [k], u \in S_i, S_i + v \notin \mathcal{I}_i, S_i + v - u \in \mathcal{I}_i\}$
while $sum < 2\bar{p}$ **do**
  $P \leftarrow$ `EdgeRecyclingBFS`$((S_1, \ldots, S_k), E^*, J, \bigcup_{i=1}^{k} F_i)$
  if $P =$ NO PATH EXISTS **then**
    break
  For $v \in S$ denote by $\pi(v)$ the index such that $v \in S_{\pi(v)}$
  Denote by $V(P) = \{s, v_1, \ldots, v_{l-1}, t_j\}$ the vertices in the path P
  for $i \leftarrow 2$ **to** $l - 1$ **do**
    $J \leftarrow J + \pi(v_i)$
  $J \leftarrow J + j$
  $(S_1, \ldots, S_k) \leftarrow$ `Update`$((S_1, \ldots, S_k), P)$
  $F_j \leftarrow \{v \in V \mid v \notin S_j$ and $S_j + v \in \mathcal{I}_j\}$
  $sum \leftarrow sum + |J|$
return $(S_1, \ldots, S_k)$

Suppose that the procedure `EdgeRecyclingBFS` is called for $J = J_1, J_2, \ldots, J_c$ in the procedure `EdgeRecyclingAugmentation`. Obivously, $c \leq 2\bar{p} = O(p)$. Furthermore, by the condition of the while loop in `EdgeRecyclingAugmentation`, $\sum_{i=1}^{c} |J_i| = O(p)$. Thus, the number of `FindOutEdge` calls in the entire procedure `EdgeRecyclingAugmentation` is $O\left(\sum_{i=1}^{c}(p + n \cdot |J_i|)\right)$, which is $O(np)$. Hence, by Lemma 4, the number of independence oracle queries by `FindOutEdge` in the entire procedure `EdgeRecyclingAugmentation` is $O(np \log p)$, which completes the proof. ◀

At this point, we can obtain a matroid partition algorithm that uses $O(np^{3/2} \log p + kn)$ independence oracle queries. In the algorithm, we first compute $F_i = \{v \in V \setminus S_i \mid S_i + v \in \mathcal{I}_i\}$ for all $i \in [k]$. Next, we apply `EdgeRecyclingAugmentation` repeatedly to augment the current partition $(S_1, \ldots, S_k)$ of S until no (s, T)-path can be found in the compressed exchange graph $G(S_1, \ldots, S_k)$. As we will show later in Lemma 22, the number of independence oracle queries in this algorithm is $O(np^{3/2} \log p + kn)$. In the next subsection, we improve this by combining the algorithm by the blocking flow approach and the algorithm by the edge recycling augmentation approach.

4.2 Going Faster for Large k by Combining Blocking Flow and Edge Recycling Augmentation

We have already presented two algorithms to solve the matroid partitioning problem in the independence oracle model. We combine the algorithm by the *blocking flow* approach and the one by the *edge recycling augmentation* approach. When the distance from s to T in the compressed exchange graph is small, we use the blocking flow approach. On the other hand, when the distance from s to T in the compressed exchange graph is large, we use the edge recycling augmentation approach. The implementation is described as Algorithm 6. Then we obtain a matroid partitioning algorithm that uses $o(kn\sqrt{p})$ independence oracle queries when $k = \omega(p^{3/4})$. This improves upon the algorithm given in Theorem 14 that uses only the blocking flow approach.

Algorithm 6 Faster Matroid Partition Algorithm for Large k.

1. Compute a $\frac{1}{2}$-approximation $\bar{p}$ for p by running $\frac{1}{2}$-`ApproximationMatroidPartition` (Algorithm 3) and determine the value of d.
2. For all $i \in [k]$ let $S_i \leftarrow \emptyset$
3. Apply `BlockFlowIndependence` repeatedly to augment the current partition $(S_1, \ldots, S_k)$ of S until the distance from s to T in the compressed exchange graph $G(S_1, \ldots, S_k)$ is at least d.
4. For all $i \in [k]$ let $F_i \leftarrow \{v \in V \setminus S_i \mid S_i + v \in \mathcal{I}_i\}$
5. Apply `EdgeRecyclingAugmentation` (Algorithm 5) repeatedly to augment the current partition $(S_1, \ldots, S_k)$ of S and to update F_j with $j \in [k]$ until no (s, T)-path can be found in the compressed exchange graph $G(S_1, \ldots, S_k)$.

The algorithm is parametrized by an integer d which we set in the end. To analyze the independence query complexity of Algorithm 6, we first show that Line 3 uses $\tilde{O}(knd)$ independence oracle queries and Line 5 uses $\tilde{O}\left(\frac{p^{3/2}n}{d^{1/2}}\right)$ independence oracle queries.

▶ **Lemma 21.** *Line 3 of Algorithm 6 uses $O(knd \log p)$ independence oracle queries.*

Proof. Lemma 16 implies that the distance from s to T in the compressed exchange graph increases by at least 1 after the execution of `BlockFlowIndependence`. Consequently, the number of calls of `BlockFlowIndependence` is bounded by d. Furthermore, Lemma 16 implies that the number of independence oracle queries in one call of `BlockFlowIndependence` is $O(kn \log p)$, which completes the proof. ◀

▶ **Lemma 22.** *Line 5 of Algorithm 6 uses* $O\left(\frac{p^{3/2}n}{d^{1/2}} \log p\right)$ *independence oracle queries.*

Proof. Let m denote the number of calls of `EdgeRecyclingAugmentation` in Line 5 of Algorithm 6. By Lemma 20, we only have to show that $m = O\left(\sqrt{\frac{p}{d}}\right)$. For $i \in [m]$, let c_i denote the number of augmenting paths found in the i-th call of `EdgeRecyclingAugmentation`. For $i \in [m-1] \cup \{0\}$, we write $s_i = \sum_{j=i+1}^{m} c_j$.

We first show the following two claims.

▷ Claim 23. There is a positive constant C such that $c_i \geq C\sqrt{s_i}$ for all $i \in [m-1]$.

Proof. Let $i \in [m-1]$. We denote by l_i the length of the augmenting path found in the last `EdgeRecyclingBFS` in the i-th call of `EdgeRecyclingAugmentation`. Lemma 12 implies that $s_i = O\left(\frac{p}{l_i}\right)$. We note that, by Lemma 13, the length of shortest augmenting paths never decreases as the partitionable set size increases.

In the i-th call of `EdgeRecyclingAugmentation`, the sum of the sizes of J is upper bounded by $c_i^2 \cdot l_i$, because the size of J is upper bounded by $c_i \cdot l_i$. Furthermore, by the condition of the while loop in `EdgeRecyclingAugmentation`, the sum of the sizes of J is at least $2\bar{p}(\geq p)$. Thus, we obtain $c_i^2 \cdot l_i \geq p$, and then we have $c_i^2 \geq \frac{p}{l_i}$. Since $s_i = O\left(\frac{p}{l_i}\right)$, we have $\sqrt{s_i} = O(c_i)$, which completes the proof. ◁

▷ Claim 24. For all $i \in [m-1]$, we have $\frac{C}{\sqrt{1+C}} \leq \int_{s_i}^{s_{i-1}} \frac{dx}{\sqrt{x}}$.

Proof. Let $i \in [m-1]$. Since $c_i \geq C\sqrt{s_i}$ by Claim 23, we obtain

$$\begin{aligned}\int_{s_i}^{s_{i-1}} \frac{dx}{\sqrt{x}} = \int_{s_i}^{s_i+c_i} \frac{dx}{\sqrt{x}} &\geq \int_{s_i}^{s_i+C\sqrt{s_i}} \frac{dx}{\sqrt{x}} \geq \int_{s_i}^{s_i+C\sqrt{s_i}} \frac{dx}{\sqrt{s_i+C\sqrt{s_i}}} \\ &= \frac{C\sqrt{s_i}}{\sqrt{s_i+C\sqrt{s_i}}} = \frac{C}{\sqrt{1+C\frac{1}{\sqrt{s_i}}}} \geq \frac{C}{\sqrt{1+C}},\end{aligned}$$

which completes the proof. ◁

By Claim 24, $m-1 = \sum_{i=1}^{m-1} 1 \leq \frac{\sqrt{1+C}}{C} \sum_{i=1}^{m-1} \int_{s_i}^{s_{i-1}} \frac{dx}{\sqrt{x}} = O\left(\int_{s_{m-1}}^{s_0} \frac{dx}{\sqrt{x}}\right) = O\left(\sqrt{s_0}\right)$.

Since Lemma 12 implies that $s_0 = O\left(\frac{p}{d}\right)$, the number of calls of `EdgeRecyclingAugmentation` in Line 5 of Algorithm 6 is $O\left(\sqrt{\frac{p}{d}}\right)$. By Lemma 20, the proof is complete. ◀

In Algorithm 6, we set a parameter d in order to balance the number of independence oracle queries used in Lines 3 and 5. Thus we obtain the following proof.

Proof of Theorem 18. We set $d = \frac{\bar{p}}{k^{2/3}}$ and run Algorithm 6. Then, by Lemmas 21 and 22, the number of independence oracle queries used in Lines 3 and 5 is $O(k^{1/3}np \log p)$. Furthermore, the number of independence oracle queries used in Lines 1 and 4 is $O(kn)$, which completes the proof. ◀

Note that Algorithm 6 requires $O\left(np \cdot \frac{p}{d}\right) = O(k^{2/3}np)$ time complexity other than independence oracle queries. This is because we use the edge set E^* of size np in `EdgeRecyclingBFS` and the number of total `EdgeRecyclingBFS` calls in Algorithm 6 is $O\left(\frac{p}{d}\right)$.

5 Concluding Remarks

By simply combining Cunningham's algorithm [7] and the binary search technique proposed by Nguyễn [25] and Chakrabarty-Lee-Sidford-Singla-Wong [5], we can not break the $O(n^{5/2})$-independence-query bound for the matroid partitioning problem. However, we introduce a new approach *edge recycling augmentation* and break this barrier and obtain an algorithm that $\tilde{O}(n^{7/3})$ independence oracle queries. This result will be a substantial step forward understanding the matroid partitioning problem.

Our key observation is that some edges in the compressed exchange graph will remain the same after an augmentation, and then we need not query again to find them. That is, we can recycle some edges in the compressed exchange graph. This yields a matroid partition algorithm whose independence query complexity is sublinear in k. This idea is quite simple, and we believe that edge recycling augmentation will be useful in the design of algorithms in future.

In a recent breakthrough, Blikstad-van den Brand-Mukhopadhyay-Nanongkai [4] broke the $\tilde{O}(n^2)$-independence-query bound for matroid intersection. Then it is natural to ask whether we can make a similar improvement for the matroid partition algorithm. However, such an improvement is impossible. As one anonymous reviewer pointed out, it is easy to show that the matroid partitioning problem requires $\Omega(kn)$ independence oracle queries, which is $\Omega(n^2)$ when $k = \Theta(n)$.[3] Then, there is a clear difference between these two problems.

We also consider a matroid partition algorithm in the rank oracle model and present a matroid partition algorithm that uses $\tilde{O}(n^{3/2})$ rank oracle queries when $k \leq n$. Blikstad et al. [4] asks whether the tight bounds of the matroid intersection problem are the same under independence oracle model and rank oracle model. The same kind of problem is natural for the matroid partitioning problem. Unlike the matroid intersection problem, we believe there exists a difference between independence oracle and rank oracle in terms of query complexity of the matroid partitioning problem.

References

1 Martin Aigner and Thomas A Dowling. Matching theory for combinatorial geometries. *Transactions of the American Mathematical Society*, 158(1):231–245, 1971.

2 Joakim Blikstad. Breaking $O(nr)$ for matroid intersection. In *the 48th International Colloquium on Automata, Languages, and Programming (ICALP2022)*, pages 31:1–31:17, 2021.

[3] Let $M_1, \ldots, M_k$ be matroids of rank 1 defined over a common ground set V of n elements. Now, we construct a bipartite graph $G = (L \cup R, E)$ with $|L| = n$, $|R| = k$ where $(v, i) \in E$ if and only if $\{v\}$ is independent in M_i. Here, the maximum size of a partitionable set is equal to the maximum size of a matching in G. It can be viewed as having edge-query access to this graph G, since it does not make sense to use an independence query to a set of size at least 2. It requires $\Omega(kn)$ edge queries to find the size of a maximum matching; see [30] for details.

3 Joakim Blikstad, Sagnik Mukhopadhyay, Danupon Nanongkai Nanongkai, and Ta-Wei Tu. Fast algorithms via dynamic-oracle matroids. In *the 55th ACM Symposium on Theory of Computing (STOC2023), to appear*, 2023.

4 Joakim Blikstad, Jan van den Brand, Sagnik Mukhopadhyay, and Danupon Nanongkai. Breaking the quadratic barrier for matroid intersection. In *the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC2021)*, pages 421–432, 2021.

5 Deeparnab Chakrabarty, Yin Tat Lee, Aaron Sidford, Sahil Singla, and Sam Chiu-wai Wong. Faster matroid intersection. In *the 60th Annual Symposium on Foundations of Computer Science (FOCS2019)*, pages 1146–1168. IEEE, 2019.

6 Chandra Chekuri and Kent Quanrud. A fast approximation for maximum weight matroid intersection. In *the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA2016)*, pages 445–457. SIAM, 2016.

7 William H Cunningham. Improved bounds for matroid partition and intersection algorithms. *SIAM Journal on Computing*, 15(4):948–957, 1986.

8 Jack Edmonds. Matroid partition. *Mathematics of the Decision Sciences*, 11:335, 1968.

9 Jack Edmonds. Submodular functions, matroids, and certain polyhedra. *Combinatorial Structures and Their Applications, R. Guy, H. Hanani, N. Sauer, and J. Schonheim, eds. New York*, pages 69–87, 1970.

10 Jack Edmonds. Matroid intersection. In *Annals of Discrete Mathematics*, volume 4, pages 39–49. Elsevier, 1979.

11 Jack Edmonds. Submodular functions, matroids, and certain polyhedra. In *Combinatorial Optimization—Eureka, You Shrink! Papers Dedicated to Jack Edmonds 5th International Workshop Aussois, France, March 5–9, 2001 Revised Papers*, pages 11–26. Springer, 2003.

12 András Frank and Zoltán Miklós. Simple push-relabel algorithms for matroids and submodular flows. *Japan Journal of Industrial and Applied Mathematics*, 29:419–439, 2012.

13 Harold Gabow and Herbert Westermann. Forests, frames, and games: algorithms for matroid sums and applications. In *the Twentieth Annual ACM Symposium on Theory of Computing (STOC1988)*, pages 407–421, 1988.

14 Curtis Greene and Thomas L Magnanti. Some abstract pivot algorithms. *SIAM Journal on Applied Mathematics*, 29(3):530–539, 1975.

15 Nicholas JA Harvey. Matroid intersection, pointer chasing, and young's seminormal representation of s_n. In *the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA2008)*, pages 542–549, 2008.

16 Julian Haselmayr. Schnitt von matroiden theorie und algorithmen. Master's thesis, University of Augsberg, 2008.

17 John E Hopcroft and Richard M Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

18 Chien-Chung Huang, Naonori Kakimura, and Naoyuki Kamiyama. Exact and approximation algorithms for weighted matroid intersection. In *the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA2016)*, pages 430–444. SIAM, 2016.

19 Yasushi Kawase, Kei Kimura, Kazuhisa Makino, and Hanna Sumita. Optimal matroid partitioning problems. *Algorithmica*, 83:1653–1676, 2021.

20 Donald Ervin Knuth. *Matroid partitioning*. Computer Science Department, Stanford University, 1973.

21 Bernhard H Korte and Jens Vygen. *Combinatorial optimization*. Springer, third edition, 2006.

22 Eugene L Lawler. Matroid intersection algorithms. *Mathematical Programming*, 9(1):31–56, 1975.

23 Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *the 56th Annual Symposium on Foundations of Computer Science (FOCS2015)*, pages 1049–1065. IEEE, 2015.

24 Crispin St John Alvah Nash-Williams. An application of matroids to graph theory. In *Theory of Graphs – International Symposium – Théorie des graphes – Journées internationales d'étude Rome*, pages 263–265, 1966.

25 Huy L Nguyen. A note on cunningham's algorithm for matroid intersection, 2019. `arXiv:1904.04129`.

26 Christopher Price. Combinatorial algorithms for submodular function minimization and related problems. Master's thesis, University of Waterloo, 2015.

27 Kent Quanrud. Faster exact and approximation algorithms for packing and covering matroids via push-relabel. *CoRR*, 2023. `arXiv:2303.01478`.

28 Alexander Schrijver et al. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

29 Ta-Wei Tu. Subquadratic weighted matroid intersection under rank oracles. In *the 33rd International Symposium on Algorithms and Computation (ISAAC2022)*, pages 63:1–63:14, 2022.

30 Andrew Chi-Chih Yao. Monotone bipartite graph properties are evasive. *SIAM Journal on Computing*, 17(3):517–520, 1988.

Frameworks for Nonclairvoyant Network Design with Deadlines or Delay

Noam Touitou[1]
Amazon, Tel Aviv, Israel

Abstract

Clairvoyant network design with deadlines or delay has been studied extensively, culminating in an $O(\log n)$-competitive general framework, where n is the number of possible request types (Azar and Touitou, FOCS 2020). In the nonclairvoyant setting, the problem becomes much harder, as $\Omega(\sqrt{n})$ lower bounds are known for certain problems (Azar et al., STOC 2017). However, no frameworks are known for the nonclairvoyant setting, and previous work focuses only on specific problems, e.g., multilevel aggregation (Le et al., SODA 2023).

In this paper, we present the first nonclairvoyant frameworks for network design with deadlines or delay. These frameworks are nearly optimal: their competitive ratio is $\tilde{O}(\sqrt{n})$, which matches known lower bounds up to logarithmic factors.

2012 ACM Subject Classification Theory of computation; Theory of computation → Design and analysis of algorithms; Theory of computation → Online algorithms

Keywords and phrases Online, Deadlines, Delay, Network Design, Nonclairvoyant

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.105

Category Track A: Algorithms, Complexity and Games

1 Introduction

In network design problems with deadlines, online connectivity requests arrive over time, such that every request must be served by its associated deadline. A solution serves pending requests by transmitting sets of items at various times, thus incurring some cost; the online algorithm constructs the solution by deciding, irrevocably, whether to make a transmission at any given time (and which items to transmit). A concrete example is Steiner tree with deadlines, in which a weighted graph with a designated root node is given offline, and each connectivity request names a terminal node to be connected to the root. Each transmission consists of a set of edges, and serves each pending request if its terminal is connected to the root by the transmitted edges; the cost of the transmission is the total cost of edges. In network design with delay, in lieu of a deadline, every request accumulates delay cost while pending, thus motivating quicker service by the algorithm.

A parameter that influences the difficulty of such problems is called *clairvoyance*. In the clairvoyant model, upon the arrival of a request, the algorithm learns its deadline (in the deadline case) or future delay accumulation (in the delay case). However, in the nonclairvoyant deadline model, the algorithm only learns the deadline of a request upon its expiration (and must then immediately serve the request if pending). Similarly, in the nonclairvoyant delay model, the algorithm is only aware of delay accumulated until the current time.

Various network design problems with deadlines or delay were studied in the clairvoyant setting. This includes problems such as multilevel aggregation [6, 11, 3, 5, 27], facility location [3, 7], TCP Acknowledgement [17, 24, 12] and joint replenishment [13, 10, 8, 16].

[1] This work does not relate to the author's position at Amazon.

For general, clairvoyant network design with deadlines or delay, algorithmic frameworks were presented in [4]. These frameworks yield logarithmic competitiveness with respect to multiple parameters: the number of requests, which we denote m; the number of request *types* (e.g., number of possible terminals for Steiner tree), which we denote n; and the number of items in the item set[2].

The nonclairvoyant setting, however, is less thoroughly studied. Some specific problems of the network design with deadlines/delay class gracefully handle nonclairvoyance; for example, for nonclairvoyant set cover with delay, logarithmic competitiveness is known [1]. Other problems, such as joint replenishment, multilevel aggregation, and facility location with deadlines, have $\Omega(\sqrt{n})$ and $\Omega(\sqrt{m})$ lower bounds on competitiveness in the nonclairvoyant setting, even for randomized algorithms [26][3]. (Note that an earlier, deterministic lower bound along the same lines appears in [2] for the service with delay problem.) Recently, Le et al. [26] presented matching and nearly-matching upper bounds for nonclairvoyant joint replenishment and multilevel aggregation with delay, respectively. However, unlike in the clairvoyant case, a general algorithmic framework for nonclairvoyant network design with deadlines or delay is still not known.

1.1 Our Results

We present the first frameworks for general network design problems with deadlines or delay in the nonclairvoyant setting. Specifically, with n the number of possible request types and m the number of requests, we present:

1. A deterministic, $O\left(\min\left\{\sqrt{n \log n}, \sqrt{m \log m}\right\}\right)$-competitive framework for network design with deadlines.
2. A deterministic, $O\left(\min\left\{\sqrt{n \log n}, \sqrt{m \log m}\right\}\right)$-competitive framework for network design with delay.

The competitiveness of our frameworks is nearly optimal, as implied by the lower bounds of $\Omega(\sqrt{n})$ and $\Omega(\sqrt{m})$ on competitiveness for some network design problems, e.g., multilevel aggregation [26].

While our frameworks provide nearly-optimal upper bounds for nonclairvoyant network design, some components require specific properties to be implemented in polynomial time. In Section 5, we show how to implement the frameworks in polynomial time for a class of network design problems; specifically, those problems that admit *Lagrangian prize-collecting* algorithms. In particular, we obtain the following results:

1. A poly-time, deterministic, $O\left(\min\left\{\sqrt{n \log n}, \sqrt{m \log m}\right\}\right)$-competitive algorithm for Steiner tree with deadlines/delay. Note that for Steiner tree, n equals the number of nodes in the graph.
2. A poly-time, deterministic, $O\left(\sqrt{m \log m}\right)$-competitive algorithm for facility location with deadlines/delay[4].

[2] Note that the bound with respect to the number of request types is implicit in [4].

[3] The lower bound in [26] is stated in terms of joint replenishment; however, joint replenishment can be seen as a special case of multilevel aggregation and facility location, and thus the lower bound applies to those problems as well.

[4] A fine point regarding facility location is that the number of request types n does not yield a meaningful bound, and thus we only state the bound with respect to the number of requests m. This is discussed in more detail in Section 5.

3. A poly-time, deterministic, $O\left(\min\left\{\sqrt{n \log n}, \sqrt{m \log m}\right\}\right)$-competitive algorithm for multicut with deadlines/delay on a tree. Note that for multicut, n equals the number of node pairs in the graph (i.e., quadratic in the number of nodes).

For nonclairvoyant facility location with deadlines/delay, our algorithm is the first algorithm. This is also the case for nonclairvoyant multicut with deadlines or delay. For nonclairvoyant Steiner tree, there existed no *explicit* previous algorithm; however, randomly embedding into an HST [18] then using the multilevel aggregation algorithm of [26] would yield a randomized $O(\sqrt{n} \log n)$-competitive algorithm, with no guarantee in m. Our algorithm for Steiner tree improves the power of the logarithm, is deterministic, and has a guarantee in m.

1.2 Other Related Work

A related problem to network design with deadlines/delay is online service, in which one is given a server (or multiple servers) on a metric space, which must then be moved to locations specified in incoming online requests. In fact, Steiner tree (and multilevel aggregation) with delay can be seen as a special case of this problem, in which the input forces the server to rest at the root node of the given graph (or tree) through an infinite stream of urgent requests at the root. Service with delay was first introduced by Azar et al. [2], and has since seen additional work [9, 3, 20, 25, 21].

The problem of set cover with delay was introduced in [14]; in this problem, the algorithm must transmit sets containing requested elements. While set cover is not usually considered a network design problem, set cover with delay falls within the network design with delay category, and thus the framework of [4] yields a logarithmic-competitive clairvoyant algorithm. As mentioned earlier, this problem also admits a logarithmic-competitive *nonclairvoyant* algorithm, as presented in [1] (and later derandomized in [26]). The best lower bound for this problem, given in [28], is nearly tight given the upper bound for network design implied by the framework in [4].

2 Preliminaries

We now introduce definitions and notation related to network design with deadlines/delay; to make these concrete, we also apply these definitions to the special case of Steiner tree with deadlines/delay.

Offline network design. In offline network design, one is given a set of items $\mathcal{E}$ with associated costs c, and a set of requests H. (In Steiner tree, an input graph is given with a designated root node; the items are the edges of the input graph, and every request demands connecting some terminal to the root.) A request can be served by any chosen subset of items, and we assume that serving requests is *upwards-closed*: that is, if request $h \in H$ is served by a subset of items $E \subseteq \mathcal{E}$, then it is also served by any superset of E. (In Steiner tree, a set of edges satisfies a request if it contains a path connecting the request's terminal to the root; note that satisfaction is indeed upwards-closed.) A solution to the offline problem is a subset of items $E \subseteq \mathcal{E}$ which serves all requests in H; the cost of the solution is the total cost of items in E.

Online network design with deadlines. In online network design with deadlines, one is also given a set of items $\mathcal{E}$ with costs. Now, however, the requests Q arrive over time; we denote by $m := |Q|$ the number of requests in the online input. Each request $q \in Q$ has a *type*, and we

denote the set of all possible request types by $\mathcal{H}$, and define $n := |\mathcal{H}|$; in fact, every request type in the online problem corresponds to a possible request in the offline problem. We denote by r_q the release time of request q, and each request q also has a deadline time d_q by which it must be served. At any point in time, the algorithm may transmit any subset of items $E \subseteq \mathcal{E}$; such a transmission is also called a *service*, and incurs a cost of $c(E) := \sum_{e \in E} c(e)$ for the algorithm. Transmitting E serves all pending requests whose request type is served by E in the offline problem. The goal of the algorithm is to serve every request by its deadline, while minimizing the total cost of services, i.e., $\sum_{\text{service } S} \sum_{e \in E(S)} c(e)$ (where $E(S)$ is the set of items transmitted in S).

For concreteness, consider Steiner tree with deadlines, in which requests for connecting terminals to the root are served by transmitting subsets of edges. As transmissions are momentary, it is meaningful for multiple requests (with different release/deadline times) to request connecting the same terminal v; such requests would belong to the same *request type*, as they would be satisfied by exactly the same subsets of items (i.e., those that contain a path from v to the root). In particular, note that for Steiner tree the number of request types n is equal to the number of nodes in the graph.

Online network design with delay. In the (more general) model with delay, each request q has a nondecreasing, continuous delay function d_q, such that for every $t \geq r_q$ the amount $d_q(t)$ is the total delay cost incurred by q until time t (if it is still pending at that time). The goal of the algorithm in this case is to minimize the total cost of services plus the total delay cost. That is, denoting by C_q the time in which request q is served in the algorithm, the goal is to minimize $\sum_{\text{service } S} \sum_{e \in E(S)} c(e) + \sum_{q \in Q} d_q(C_q)$. For ease of notation, for every set of requests Q' and time t, we define $D(Q', t) := \sum_{q \in Q'} d_q(t)$.

Additional notation. For a request q in online network design with deadlines/delay, we define h_q to be the type of request q. For a set of requests Q', we define $H(Q') \subseteq \mathcal{H}$ to be $\{h_q | q \in Q'\}$. For a set of request types H, we use $\mathrm{ND}(H)$ to denote the minimum cost of serving these request types; slightly abusing notation, for a set of requests Q', we define $\mathrm{ND}(Q') := \mathrm{ND}(H(Q'))$.

For a set of request types H, we define $\ell(H) := \lceil \log \mathrm{ND}(\{H\}) \rceil$. When considering a single request type, we sometimes write $\ell(h)$ instead of $\ell(\{h\})$. Finally, for request q and set of requests Q', we define $\ell(q) := \ell(h_q)$ and $\ell(Q') := \ell(H(Q'))$.

3 Framework for Network Design with Deadlines

In this section, we present a framework for nonclairvoyant network design with deadlines. Analyzing this framework, we obtain the following theorem.

▶ **Theorem 1.** *There exists a deterministic, $O(\min\{\sqrt{n \log n}, \sqrt{m \log m}\})$-competitive algorithm for network design with deadlines.*

Algorithm's description. The algorithm assigns levels to each request according to the logarithm of the cost of serving that request. Upon the deadline of a pending request q, the algorithm starts a service λ that serves the request, thus incurring a cost of at most $2^{\ell(q)}$; the level of the service, denoted ℓ_λ, is defined to be the level of the triggering request q. If a service is *large*, it also serves additional requests; it does so by identifying a set of pending request types which can be served, subject to some specific budget for every request type.

Otherwise, a service is *small*, and does not serve additional requests. Whether a service is large or small is determined by the triggering request q, i.e., the request upon whose deadline the service is started; specifically, the service triggered by the deadline of q will be large if and only if the variable b_q is TRUE at that time.

Specifically, suppose a large service λ takes place. The service will identify the *eligible* requests, i.e., the pending requests whose level is at most ℓ_λ. Denoting by H the set of request types of those eligible requests, the service will give a budget of $\tilde{\Theta}(2^{\ell_\lambda}/\sqrt{|H|})$ to each request type, and will attempt to find a subset of requests that can be served without exceeding the budget for those request types. Thus, the cost of such a large service is at most $\tilde{\Theta}(2^{\ell_\lambda} \cdot \sqrt{|H|})$.

The variable b_q, which controls whether q will trigger a large service, is initially TRUE for every request q. However, if a large service of level at least $\ell(q)$ takes place while q is pending, the variable is set to FALSE. Thus, a request that "experienced" a large service for which it was eligible will never trigger a large service upon its deadline. The formal description of the algorithm is given in Algorithm 1.

Algorithm 1 Nonclairvoyant framework for network design with deadlines.

Event Function UPONREQUEST(q)
  $b_q \leftarrow$ TRUE.
Event Function UPONDEADLINE(q)
  start a new service λ, and set $\ell_\lambda = \ell(q)$.
  define E_λ to be the set of pending requests of level at most ℓ_λ, and define $H \leftarrow H(E_\lambda)$.
  transmit ND($\{h_q\}$), serving all requests of type h_q.
  if b_q **then**
    let $x_\lambda \leftarrow 2^{\ell_\lambda} \cdot \sqrt{\frac{\log(1+|H|)}{|H|}}$.
    let $H' \subseteq H$ be a maximal subset such that ND(H') $\leq x_\lambda \cdot |H'|$.
    transmit ND(H'), serving all requests of types in H'.
    // set $b_{q'}$ for eligible requests q' which are still pending.
    let Q_λ be the subset of requests served by λ.
    foreach $q' \in E_\lambda \setminus Q_\lambda$ **do**
      set $b_{q'} \leftarrow$ FALSE.

3.1 Analysis

We now focus on proving Theorem 1.

▶ **Definition 2.** *We define the following terms:*

1. *We denote by Λ, Λ^* the set of services in the algorithm and in the optimal solution, respectively.*
2. *For a service $\lambda \in \Lambda$, we define Q_λ to be the set of requests served by (the transmissions of) λ. For an optimal service $\lambda^* \in \Lambda^*$, we define Q_{λ^*} in a similar way.*
3. *For a service $\lambda \in \Lambda$, we define $\ell_\lambda, E_\lambda, x_\lambda$ to be the values of the variables of those names in* UPONDEADLINE.
4. *We define $H_\lambda = H(E_\lambda)$. As a shorthand, we define $y_\lambda = |H_\lambda|$.*
5. *We define the* triggering request *of λ, denoted $q^\star_\lambda$, to be the request upon whose deadline λ was initiated.*
6. *We define $c(\lambda)$ to be the total transmission cost incurred in service λ.*

7. *We say that an algorithm service $\lambda \in \Lambda$ is* charged *to an optimal service λ^* if $q^\star_\lambda \in Q_{\lambda^*}$. For every $\lambda^* \in \Lambda^*$, we define $\Lambda_{\lambda^*} \subseteq \Lambda$ to be the set of services charged to λ^*.*
8. *We call a service λ a* large *service if upon the start of λ we have $b_{q^\star_\lambda} =$ True. Otherwise, λ is a* small *service.*
9. *For every service λ in the algorithm or in the optimal solution, denote by t_λ the time in which the service takes place.*

We define k to be the maximum size of $H(E_\lambda)$ over all services λ. In particular, note that $k \leq \min\{n, m\}$, which allows us to prove our competitiveness bounds with respect to k. Fix henceforth any optimal service $\lambda^* \in \Lambda^*$.

▶ **Proposition 3.** *For every $\lambda \in \Lambda_{\lambda^*}$, it holds that $t_\lambda \geq t_{\lambda^*}$.*

Proof. Note that λ is triggered by the deadline of request $q^\star_\lambda$. From the definition of Λ_{λ^*}, we have that $q^\star_\lambda \in Q_{\lambda^*}$; thus, $t_{\lambda^*} \leq d_{q^\star_\lambda} = t_\lambda$. ◀

We partition Λ_{λ^*}, into three parts, the costs of which we bound individually:

- $\Lambda^1_{\lambda^*}$: The large services in Λ_{λ^*}.
- $\Lambda^2_{\lambda^*}$: The small services λ such that $b_{q^\star_\lambda}$ was first set to False at time smaller than t_{λ^*}.
- $\Lambda^3_{\lambda^*}$: The small services λ such that $b_{q^\star_\lambda}$ was first set to False at time at least t_{λ^*}.

▶ **Proposition 4.** *For every level ℓ, there exists at most one level-ℓ service in $\Lambda^1_{\lambda^*}$.*

Proof. Assume otherwise that there exist $\lambda_1, \lambda_2 \in \Lambda_{\lambda^*}$ which are both large, and assume without loss of generality that $t_{\lambda_2} > t_{\lambda_1}$. From the previous claim, we have that $t_{\lambda_1} \geq t_{\lambda^*}$, and thus $q^\star_{\lambda_2}$ is pending at t_{λ_1}. Moreover, as $\ell\left(q^\star_{\lambda_2}\right) = \ell_{\lambda_1} = \ell$, we have that $q^\star_{\lambda_2} \in E_{\lambda_1}$. But Line 13 of λ_1 sets $b_{q^\star_{\lambda_2}}$ to be False, and this value is maintained until λ_2. This is in contradiction to λ_2 being a large service. ◀

▶ **Proposition 5.** *For every ℓ, we have $\sum_{\lambda \in \Lambda^1_{\lambda^*} | \ell_\lambda = \ell} c(\lambda) = O(\sqrt{k \log k}) \cdot 2^\ell$.*

Proof. From Proposition 4, it holds that $\Lambda^1_{\lambda^*}$ contains at most one (large) service of level ℓ, and thus the left-hand side of the equation contains at most one summand. The cost of a large service λ of level ℓ consists of two costs: the first cost is the cost of transmitting $\mathrm{ND}(h_{q^\star_\lambda})$, which is at most 2^ℓ, using the fact that $\ell\left(q^\star_\lambda\right) = \ell$; the second cost is the cost of the transmission in Line 10, which is at most

$$y_\lambda \cdot x_\lambda \leq 2^\ell \cdot \sqrt{y_\lambda \log(1 + y_\lambda)} = O(\sqrt{k \log k}) \cdot 2^\ell$$ ◀

▶ **Proposition 6.** *For every ℓ, we have*

$$\sum_{\lambda \in \Lambda^2_{\lambda^*} | \ell_\lambda = \ell} c(\lambda) = O(\sqrt{k / \log k}) \cdot c(\lambda^*).$$

Proof. Fix any ℓ. We define $\Lambda' := \left\{\lambda \in \Lambda^2_{\lambda^*} | \ell_\lambda = \ell\right\}$ and, as a shorthand, define $z := |\Lambda'|$. We define R to be the set of triggering requests of services in Λ'. First, we claim that the request types of triggering requests of services in Λ' are distinct, i.e., that $|H(R)| = |\Lambda'| = z$. To prove this claim, assume for contradiction that there exist two services in $\lambda_1, \lambda_2 \in \Lambda'$ with triggering requests of the same type, and further assume without loss of generality that $t_{\lambda_1} < t_{\lambda_2}$. Since $q^\star_{\lambda_2} \in Q_{\lambda^*}$, it must be pending at t_{λ^*}; moreover, Proposition 3 implies that $t_{\lambda^*} \leq t_{\lambda_1}$, and thus $q^\star_{\lambda_2}$ is pending at t_{λ_1}. But then λ_1 would serve $q^\star_{\lambda_2}$ in Line 6, in contradiction to $q^\star_{\lambda_2}$ triggering λ_2. Thus, the proof of the claim is complete.

Now, consider the first large service $\lambda \in \Lambda$ after which for every triggering request q of a service in Λ' we have $b_q = \textsc{False}$; it must be that $\ell_\lambda \geq \ell$. From the definition of Λ', it holds that $t_\lambda < t_{\lambda^*}$; combining this with Proposition 3, we have that $t_\lambda < t_{\lambda'}$ for every $\lambda' \in \Lambda'$. In particular, all triggering requests of services from Λ' must be pending at t_λ, and since they are of level ℓ, these requests are also in E_λ. However, they all remain pending after λ, which means that $\mathrm{ND}(H(R)) \geq x_\lambda \cdot |H(R)|$. (Otherwise, we would get a contradiction to the maximality of the set H' defined in Line 9, as $\mathrm{ND}(H(R))$ could be added to the solution without exceeding budget.) We thus have

$$c(\lambda^*) \geq \mathrm{ND}(H(R)) \geq x_\lambda \cdot |H(R)| \geq 2^\ell \cdot \sqrt{\frac{\log(1+k)}{k}} \cdot z \tag{1}$$

where the third inequality uses the fact that $y_\lambda \leq k$. Meanwhile, the total cost of services in Λ' is at most $z \cdot 2^\ell$. Combining with Equation (1) completes the proof. ◀

▶ **Proposition 7.** *For every ℓ, we have $\sum_{\lambda \in \Lambda^3_{\lambda^*} | \ell_\lambda = \ell} c(\lambda) = O(\sqrt{k/\log k}) \cdot c(\lambda^*)$.*

Proof. For ease of notation, define $\Lambda' := \left\{\lambda \in \Lambda^3_{\lambda^*} | \ell_\lambda = \ell\right\}$. We also define R to be the set of triggering requests for requests in Λ'. We define $z := |\Lambda'|$; using an identical argument to that in the proof of Proposition 6, it holds that $|H(R)| = z$. Let λ be the first large service of level at least ℓ such that $t_\lambda \geq t_{\lambda^*}$. Note that the following hold:

1. $t_\lambda < t_{\lambda'}$ for every $\lambda' \in \Lambda'$ (stems from the definition of Λ').
2. R are all pending at t_λ and eligible for λ (as $t_\lambda \geq t_{\lambda^*}$).
3. λ changes b_q from $\textsc{True}$ to $\textsc{False}$ for every $q \in R$.

Since R were all eligible for λ but were not served, it must be the case that $\mathrm{ND}(H(R)) \geq x_\lambda \cdot z \geq 2^\ell \cdot \sqrt{z \log(1+z)}$; since λ^* serves R, it thus holds that $c(\lambda^*) \geq 2^\ell \cdot \sqrt{z \log(1+z)}$. We therefore have $\sum_{\lambda \in \Lambda^3_{\lambda^*} | \ell_\lambda = \ell} c(\lambda) \leq 2^\ell \cdot z \leq \sqrt{z/\log(1+z)} \cdot c(\lambda^*)$. We now note that all requests in R were pending during λ; thus, $z \leq k$, which completes the proof. ◀

Proof of Theorem 1. We first observe that $\mathrm{ALG} = \sum_{\lambda \in \Lambda} c(\lambda) = \sum_{\lambda^* \in \Lambda^*} \sum_{\lambda \in \Lambda_{\lambda^*}} c(\lambda)$. We claim that for every $\lambda^* \in \Lambda^*$, we have $\sum_{\lambda \in \Lambda_{\lambda^*}} c(\lambda) \leq O(\sqrt{k \log k}) \cdot c(\lambda^*)$; since $\mathrm{OPT} = \sum_{\lambda^* \in \Lambda^*} c(\lambda^*)$, proving this claim completes the proof of the theorem.

To prove the claim, fix any optimal service $\lambda^* \in \Lambda^*$, and define $\ell := \lceil \log c(\lambda^*) \rceil$. Note that for every service $\lambda \in \Lambda_{\lambda^*}$ we have $\ell_\lambda \leq \ell$; this is since $q^\star_\lambda$ is served by λ^*, which implies $\mathrm{ND}(q^\star_\lambda) \leq c(\lambda^*)$. We partition Λ_{λ^*} into $\Lambda^1_{\lambda^*}, \Lambda^2_{\lambda^*}, \Lambda^3_{\lambda^*}$ as before, and bound each set separately.

First, we bound the cost of $\Lambda^1_{\lambda^*}$ as follows.

$$\begin{aligned} \sum_{\lambda \in \Lambda^1_{\lambda^*}} c(\lambda) &= \sum_{\ell' \leq \ell} \sum_{\lambda \in \Lambda^1_{\lambda^*} | \ell_\lambda = \ell'} c(\lambda) \leq \sum_{\ell' \leq \ell} O(\sqrt{k \log k}) \cdot 2^{\ell'} \\ &\leq O(\sqrt{k \log k}) \cdot 2^\ell \leq O(\sqrt{k \log k}) \cdot c(\lambda^*) \end{aligned} \tag{2}$$

where the first inequality uses Proposition 5, and the second inequality bounds a geometric series.

Defining $\gamma := \lceil \log k \rceil$, we bound the cost of $\Lambda^2_{\lambda^*}$.

$$\begin{aligned} \sum_{\lambda \in \Lambda^2_{\lambda^*}} c(\lambda) &= \sum_{\lambda \in \Lambda^2_{\lambda^*} | \ell_\lambda \leq \ell - \gamma} c(\lambda) + \sum_{\lambda \in \Lambda^2_{\lambda^*} | \ell - \gamma < \ell_\lambda \leq \ell} c(\lambda) \leq \sum_{\ell' \leq \ell - \gamma} k \cdot 2^{\ell'} + \sum_{\lambda \in \Lambda^2_{\lambda^*} | \ell - \gamma < \ell_\lambda \leq \ell} c(\lambda) \\ &\leq \sum_{\ell' \leq \ell - \gamma} k \cdot 2^{\ell'} + \gamma \cdot O(\sqrt{k/\log k}) \cdot c(\lambda^*) \leq 2 \cdot 2^\ell + \gamma \cdot O(\sqrt{k/\log k}) \cdot c(\lambda^*) \\ &\leq O(\sqrt{k \log k}) \cdot c(\lambda^*) \end{aligned} \tag{3}$$

Here, the first inequality is due to the fact that the total cost of level-ℓ' small services in Λ_{λ^*} cannot exceed $k \cdot 2^{\ell'}$. (As seen in the proof of Proposition 6, the requests of level ℓ' triggering small services are of distinct request types, and are all eligible in a single service λ; thus, their number is at most k.) The second inequality is through using Proposition 6, the third inequality is through the definition of γ and through summing a geometric sequence, and the fourth inequality notes that through the definition of ℓ we have $c(\lambda^*) \geq 2^{\ell-1}$.

Replacing Proposition 6 with Proposition 7, an identical argument to the one used for bounding $\Lambda^2_{\lambda^*}$ can be used for bounding $\Lambda^3_{\lambda^*}$, yielding the following:

$$\sum_{\lambda \in \Lambda^3_{\lambda^*}} c(\lambda) = O(\sqrt{k \log k}) \cdot c(\lambda^*) \tag{4}$$

Combining Equations (2) to (4) yields $\sum_{\lambda \in \Lambda_{\lambda^*}} c(\lambda) \leq O(\sqrt{k \log k}) \cdot c(\lambda^*)$, completing the proof. ◀

4 Framework for Network Design with Delay

In this section, we present a framework for nonclairvoyant network design with delay. Using this framework, we prove the following theorem.

▶ **Theorem 8.** *There exists a deterministic, $O\left(\min\left\{\sqrt{n \log n}, \sqrt{m \log m}\right\}\right)$-competitive algorithm for network design with delay.*

4.1 The Algorithm

We now describe the framework for nonclairvoyant network design with delay. For every time t and set of requests Q' which are pending at t, we say that Q' are *critical* at t if $D(Q', t) \geq \mathrm{ND}(Q')$.

Framework's description. The framework for delay contains many analogues to the deadline framework of Section 3. In the deadline case, a service was triggered upon the deadline of a pending request; in the delay case, the framework initiates a service whenever a set of pending requests becomes "critical", which is when its accumulated delay justifies its service. In the deadline case, whether a service triggered by request q was large is determined by the associated variable b_q. In the delay case, we also maintain the variable b_q for every pending request q.

However, the delay case introduces an additional complication: the triggering set contains multiple requests, and thus multiple values for the variables b_q. Where services for deadlines were either "large" or "small", in the delay case this distinction is no longer binary: services identify a budget for expansive service which depends on the large requests in the triggering set.

The service considers the requests q in its triggering set with $b_q = \textsc{True}$, and finds the largest subset of those requests whose delay is at least a constant fraction of its service cost. This subset is considered "mature" enough to justify expansive service, and its service cost is used as a budget for serving pending requests. Thus, where in deadlines the level of the service was simply the level of the triggering requests, for delay the level depends on the cost of the "mature" subset of large requests.

Algorithm 2 Nonclairvoyant framework for network design with delay.

1 **Event Function** UponRequest(q)
2 $\quad b_q \leftarrow$ True.
3 **Event Function** UponCritical(R) // *called when the delay of some set R exceeds $ND(R)$*
4 $\quad$ Start a new service λ; denote the current time by t.
5 $\quad$ Define $R^\top \leftarrow \{q \in R | b_q = \text{True}\}$.
6 $\quad$ Let $R^\star \subseteq R^\top$ be a maximal subset such that $D(R^\star, t) \geq \frac{\text{ND}(H(R^\star))}{2}$.
7 $\quad$ Define $\ell_\lambda \leftarrow \ell(R^\star)$.
8 $\quad$ Define E to be the set of pending requests of level at most ℓ_λ, and define $H \leftarrow H(E)$.
9 $\quad$ Transmit $\text{ND}(H(R))$, serving all requests of types $H(R)$.
10 $\quad$ Let $x_\lambda \leftarrow 2^{\ell_\lambda} \cdot \sqrt{\frac{\log(1+|H|)}{|H|}}$.
11 $\quad$ Let $H' \subseteq H$ be a maximal subset such that $\text{ND}(H') \leq |H'| \cdot x_\lambda$.
12 $\quad$ Transmit $\text{ND}(H')$, serving all requests of types in H'.
$\quad$ // *set $b_{q'}$ for eligible requests q' which are still pending.*
13 $\quad$ let Q_λ be the subset of pending requests served by λ.
14 $\quad$ **foreach** $q' \in E \setminus Q_\lambda$ **do**
15 $\quad\quad$ set $b_{q'} \leftarrow$ False.

4.2 Analysis

Fix any optimal service $\lambda^* \in \Lambda^*$.

▶ **Definition 9.** *For a service $\lambda \in \Lambda$, define $R_\lambda, R^\top_\lambda, R^\star_\lambda, E_\lambda$ to be the values of the variables $R, R^\top, R^\star, E$ in the call to* UponCritical *which started λ. Moreover, we define $R^\perp_\lambda := R_\lambda \setminus R^\top_\lambda$; these are the requests $q \in R_\lambda$ such that $b_q =$ False at t_λ. In addition, define ℓ_λ, Q_λ as they are defined in the call to* UponCritical.

For a service $\lambda \in \Lambda$ and an optimal service $\lambda^* \in \Lambda^*$, for ease of notation, when referring to a set of jobs related to λ we add λ^* to the subscript to intersect this set with Q_{λ^*}. For example, we define $R_{\lambda,\lambda^*} := R_\lambda \cap Q_{\lambda^*}$, $R^\top_{\lambda,\lambda^*} := R^\top_\lambda \cap Q_{\lambda^*}$, $R^\perp_{\lambda,\lambda^*} := R^\perp_\lambda \cap Q_{\lambda^*}$ and $R^\star_{\lambda,\lambda^*} := R^\star_\lambda \cap Q_{\lambda^*}$. We also define $\ell_{\lambda,\lambda^*} := \ell\left(R^\star_{\lambda,\lambda^*}\right)$.

We define the cost of a service λ, denoted by $c(\lambda)$, to be the total transmission cost in λ, plus the total delay cost of requests served in λ. We define $c(\lambda^*)$ identically for an optimal service $\lambda^* \in \Lambda^*$.

▶ **Proposition 10.** *For a service λ, it holds that*

$$c(\lambda) \leq O(1) \cdot \sum_{\lambda^* \in \Lambda^*} D\left(R^\perp_{\lambda,\lambda^*}, t_\lambda\right) + O(\sqrt{k \log k}) \cdot \sum_{\lambda^* \in \Lambda^*} D\left(R^\star_{\lambda,\lambda^*}, t_\lambda\right)$$

Proof. Since a service is triggered whenever a set of requests becomes critical, the algorithm maintains that for every set Q' of pending requests at any time t we have $D(Q', t) \leq \text{ND}(Q')$. In particular, this holds for the set of requests served by λ with respect to the service time t_λ. Thus, the delay cost of the service can be charged to the transmission costs of the service; we hence focus on bounding the transmission costs of λ.

The service λ performs two transmissions, one at Line 9 and one at Line 12. The first transmission costs $\text{ND}(R_\lambda)$; since R_λ is critical at t_λ, it holds that

$$\text{ND}(R_\lambda) = D(R_\lambda, t_\lambda) = D\left(R^\perp_\lambda, t_\lambda\right) + D\left(R^\top_\lambda \setminus R^\star_\lambda, t_\lambda\right) + D\left(R^\star_\lambda, t_\lambda\right).$$

First, let us bound the delay of requests in $R^\top_\lambda \setminus R^\star_\lambda$. From the choice of $R^\star_\lambda$, it must be the case that $D\big(R^\top_\lambda \setminus R^\star_\lambda, t_\lambda\big) < \frac{\mathrm{ND}(R^\top_\lambda \setminus R^\star_\lambda)}{2}$; assuming otherwise, we would have the following:

$$D\big(R^\top_\lambda, t_\lambda\big) = D\big(R^\star_\lambda, t_\lambda\big) + D\big(R^\top_\lambda \setminus R^\star_\lambda, t_\lambda\big) \geq \frac{\mathrm{ND}(R^\star_\lambda)}{2} + \frac{\mathrm{ND}(R^\top_\lambda \setminus R^\star_\lambda)}{2} \geq \frac{\mathrm{ND}(R^\top_\lambda)}{2},$$

in contradiction to the maximality in the definition of $R^\star_\lambda$; thus, $D\big(R^\top_\lambda \setminus R^\star_\lambda, t_\lambda\big) < \frac{\mathrm{ND}(R^\top_\lambda \setminus R^\star_\lambda)}{2} \leq \frac{\mathrm{ND}(R_\lambda)}{2}$. However, we know that $D(R_\lambda, t_\lambda) = \mathrm{ND}(R_\lambda)$, and therefore conclude that

$$\mathrm{ND}(R_\lambda) \leq 2\big(D\big(R^\perp_\lambda, t_\lambda\big) + D\big(R^\star_\lambda, t_\lambda\big)\big)$$

We have thus bounded the cost of the first transmission.

To bound the cost of the second transmission in λ, let H be the set of pending request types in E_λ. We know that the cost of the second transmission is at most $x_\lambda \cdot |H| = 2^{\ell_\lambda} \cdot \sqrt{|H| \log(1+|H|)}$. Additionally, note that $2^{\ell_\lambda} \leq 2 \cdot \mathrm{ND}(R^\star_\lambda) \leq 4 \cdot D\big(R^\star_\lambda, t_\lambda\big)$, where the second inequality is due to the definition of $R^\star_\lambda$. Overall, the cost of the second transmission is at most $4 \cdot D\big(R^\star_\lambda, t_\lambda\big) \cdot \sqrt{k \log(1+k)}$.

To summarize, we proved that

$$\begin{aligned} c(\lambda) &\leq O(1) \cdot D\big(R^\perp_\lambda, t_\lambda\big) + O(\sqrt{k \log k}) \cdot D\big(R^\star_\lambda, t_\lambda\big) \\ &= O(1) \cdot \sum_{\lambda^* \in \Lambda^*} D\Big(R^\perp_{\lambda,\lambda^*}, t_\lambda\Big) + O(\sqrt{k \log k}) \cdot \sum_{\lambda^* \in \Lambda^*} D\Big(R^\star_{\lambda,\lambda^*}, t_\lambda\Big). \end{aligned}$$ ◀

For every $\lambda \in \Lambda, \lambda^* \in \Lambda^*$ we define the *joint cost* of λ and λ^* as the following.

$$c(\lambda, \lambda^*) := D\Big(R^\perp_{\lambda,\lambda^*}, t_\lambda\Big) + \sqrt{k \log(1+k)} \cdot D\Big(R^\star_{\lambda,\lambda^*}, t_\lambda\Big).$$

Through Proposition 10, we have

$$\mathrm{ALG} \leq O(1) \cdot \sum_{\lambda \in \Lambda} \sum_{\lambda^* \in \Lambda^*} c(\lambda, \lambda^*). \tag{5}$$

We henceforth focus on bounding joint costs.

▶ **Proposition 11.** *For every optimal service $\lambda^* \in \Lambda^*$, it holds that*

$$\sum_{\lambda \in \Lambda | t_\lambda \leq t_{\lambda^*}} c(\lambda, \lambda^*) \leq O(\sqrt{k \log k}) \cdot c(\lambda^*).$$

Proof. For every service $\lambda \in \Lambda$ such that $t_\lambda \leq t_{\lambda^*}$, it holds that

$$c(\lambda, \lambda^*) \leq \sqrt{k \log(1+k)} \cdot D\big(R_{\lambda,\lambda^*}, t_\lambda\big) \leq \sqrt{k \log(1+k)} \cdot D\big(R_{\lambda,\lambda^*}, t_{\lambda^*}\big)$$

Note that every request in Q_{λ^*} is critical in at most one service in the algorithm, as critical requests in a service are always served by that service. Thus, summing over different services, we get the following:

$$\begin{aligned} \sum_{\lambda \in \Lambda | t_\lambda \leq t_{\lambda^*}} c(\lambda, \lambda^*) &\leq \sqrt{k \log(1+k)} \cdot \sum_{\lambda \in \Lambda | t_\lambda \leq t_{\lambda^*}} D\big(R_{\lambda,\lambda^*}, t_{\lambda^*}\big) \\ &\leq \sqrt{k \log(1+k)} \cdot D(Q_{\lambda^*}, t_{\lambda^*}) \leq \sqrt{k \log(1+k)} \cdot c(\lambda^*) \end{aligned}$$ ◀

We thus bounded the joint cost of services prior to λ^*. It remains to bound the joint cost of services at time at least t_{λ^*}.

▶ **Proposition 12.** *For every optimal service λ^*, it holds that*

$$\sum_{\lambda\in\Lambda|t_\lambda>t_{\lambda^*}} D\left(R^\star_{\lambda,\lambda^*},t_\lambda\right) \le O(1)\cdot c(\lambda^*).$$

Proof. Let $\Lambda'\subseteq\Lambda$ be the subset of services that occurred after t_{λ^*}. First, we claim that for every integer ℓ, there exists at most one service in Λ' with $\ell_{\lambda,\lambda^*}=\ell$. To prove the claim, assume for contradiction that there are two services $\lambda_1,\lambda_2\in\Lambda'$ such that $\ell_{\lambda_1,\lambda^*}=\ell_{\lambda_2,\lambda^*}=\ell$, and assume without loss of generality that $t_{\lambda_2}>t_{\lambda_1}>t_{\lambda^*}$. As $t_{\lambda_1}>t_{\lambda^*}$, it must be that all requests in $R^\star_{\lambda_2,\lambda^*}$ were pending before and after t_{λ_1}. Moreover, every request $q\in R^\star_{\lambda_2,\lambda^*}$ must have $\ell(q)\le\ell$, as $\ell=\ell\left(R^\star_{\lambda_2,\lambda^*}\right)\ge\ell(q)$. However, $\ell_{\lambda_1}\ge\ell_{\lambda_1,\lambda^*}=\ell$, and thus $R^\star_{\lambda_2,\lambda^*}\subseteq E_{\lambda_1}$. But, Line 15 in λ_1 sets $b_q=\textsc{False}$ for every $q\in R^\star_{\lambda_2,\lambda^*}$, in contradiction to having $b_q=\textsc{True}$ at t_{λ_2}. We conclude that $R^\star_{\lambda_2,\lambda^*}=\emptyset$; however, this implies that $\ell_{\lambda_2,\lambda^*}=-\infty$, in contradiction to $\ell_{\lambda_2,\lambda^*}=\ell$.

Using this claim, for every level ℓ, there exists at most one service $\lambda\in\Lambda'$ such that $\ell_{\lambda,\lambda^*}=\ell$. For such λ, it holds that $D\left(R^\star_{\lambda,\lambda^*},t_\lambda\right)\le\mathrm{ND}(R^\star_{\lambda,\lambda^*})\le 2^\ell$ (note that delay cannot exceed service cost since critical sets trigger a service). Also note that $\max_{\lambda\in\Lambda'}\ell_{\lambda,\lambda^*}\le\ell(Q_{\lambda^*})$. Summing over the possible levels, we get that

$$\sum_{\lambda\in\Lambda|t_\lambda>t_{\lambda^*}} D\left(R^\star_{\lambda,\lambda^*},t_\lambda\right)\le 2\cdot 2^{\max_{\lambda\in\Lambda'}\ell_{\lambda,\lambda^*}}\le 2\cdot 2^{\ell(Q_{\lambda^*})}\le 4\cdot c(\lambda^*).$$ ◀

At this point, the only missing ingredient for the main theorem is the following lemma.

▶ **Lemma 13.** *For every optimal service $\lambda^*\in\Lambda$, it holds that*

$$\sum_{\lambda\in\Lambda|t_\lambda>t_{\lambda^*}} D\left(R^\perp_{\lambda,\lambda^*},t_\lambda\right)\le O(\sqrt{k\log k})\cdot c(\lambda^*)$$

The proof of Lemma 13 appears in Appendix A; assuming Lemma 13 holds, we proceed to prove Theorem 8.

Proof of Theorem 8. Equation (5) implies that it is enough to bound the sum of joint costs. Fix any optimal service λ^*; we have

$$\begin{aligned}\sum_{\lambda\in\Lambda} c(\lambda,\lambda^*) &= \sum_{\lambda\in\Lambda|t_\lambda<t_{\lambda^*}} c(\lambda,\lambda^*)+\sum_{\lambda\in\Lambda|t_\lambda\ge t_{\lambda^*}} c(\lambda,\lambda^*)\\ &\le O(\sqrt{k\log k})\cdot c(\lambda^*)+\sum_{\lambda\in\Lambda|t_\lambda\ge t_{\lambda^*}}\left(D\left(R^\perp_{\lambda,\lambda^*},t_\lambda\right)+\sqrt{k\log(1+k)}\cdot D\left(R^\star_{\lambda,\lambda^*},t_\lambda\right)\right)\\ &\le O(\sqrt{k\log k})\cdot c(\lambda^*)\end{aligned}$$

where the first inequality uses Proposition 11 and the second inequality uses Proposition 12 and Lemma 13. ◀

5 Polynomial Time through Lagrangian Prize Collecting

While the algorithms in this paper yield proper competitiveness bounds, it is not clear how to implement some of their components in polynomial time. In this section, we focus on a subset of network design problems that admit a *Lagrangian prize-collecting* approximation algorithms, and describe a polynomial-time implementation of the framework. For conciseness, we focus on the delay framework of Section 4; the result for deadlines is a special case of the result for delay.

Considering the framework in Algorithm 2 of Section 4, we note the following components which might take super-polynomial time:

1. The framework for the delay model waits until the delay cost of a subset of pending requests exceeds the cost of serving their request types.
2. The framework solves the offline network design problem optimally, i.e., makes calls to ND (e.g., in Lines 7 and 12).
3. The framework finds a subset of requests whose delay exceeds a constant fraction of their service cost (Line 6).
4. The framework includes a component which, given a penalty x and a set of request types H', finds a maximal subset $H' \subseteq H$ such that $\text{ND}(H') \geq |H'| \cdot x$ (Line 11).

The prize-collecting problem. In the offline network design problems we considered thus far, a valid solution must serve all given requests. However, we now consider a more general model of these problems, which is the *prize-collecting* model. In the (offline) prize-collecting model, in addition to the given connectivity requests H, we are also given a penalty function $\pi : H \to \mathbb{R}^+$; a valid solution in this model can now serve only a subset of the input requests, and pay the penalty for the remaining requests. The total cost of the solution is thus the total service cost plus the total penalty cost; for prize-collecting input (H, π), we use $\text{PCND}(H, \pi)$ to refer to the minimum total cost of a feasible solution to the input. Approximation algorithms are known for many such prize-collecting network design problems; given an approximation algorithm $\widetilde{\text{PCND}}$, we again use $\widetilde{\text{PCND}}(H, \pi)$ to refer to the total cost of $\widetilde{\text{PCND}}$ on the input (H, π). We also use the subscripts b and p to refer to the service and penalty costs of an algorithm, respectively (e.g., $\widetilde{\text{PCND}}_{\text{b}}(H, \pi)$).

To give a polynomial-time implementation to the framework, we require an approximation algorithm for the prize-collecting version of the offline network design problem. In fact, we need a slightly stronger notion of approximation, in which the algorithm's penalty cost is more closely bound to the optimal solution than the service cost; we now define this notion, called *Lagrangian* approximation.

▶ **Definition 14.** *We say that an algorithm for the prize-collecting problem is a Lagrangian γ-approximation if for every prize-collecting input (H, π) it holds that*

$$\widetilde{PCND}_b(H, \pi) + \gamma \cdot \widetilde{PCND}_p(H, \pi) \leq \gamma \cdot PCND(H, \pi).$$

In this section, we present an algorithm which proves the following theorem.

▶ **Theorem 15.** *For an online network design problem with deadlines/delay, whose offline network design prize-collecting problem admits a Lagrangian γ approximation, there exists a poly-time algorithm which achieves the competitiveness of Theorem 8 up to a factor polynomial in γ. Specifically, it achieves a competitive ratio of $O(\gamma^3 \cdot \min\left\{\sqrt{n \log n}, \sqrt{m \log m}\right\})$.*

5.1 Applications

To demonstrate the use of Theorem 15, we apply it to some network design problems for which Lagrangian prize-collecting approximation algorithms are known.

Steiner tree. Goemans and Williamson [19] gave a Lagrangian 2-approximation for the prize-collecting Steiner tree problem. Thus, we obtain the following corollary of Theorem 15.

▶ **Corollary 16.** *There exists an $O(\min\left\{\sqrt{|V| \log |V|}, \sqrt{m \log m}\right\})$-competitive nonclairvoyant algorithm for Steiner tree with deadlines/delay on a graph with vertex set V which runs in polynomial time.*

Facility location. First, we explain the way facility location conforms to the network design setting. The set of items consists of two types: an *opening* item for each location, of the cost of opening a facility at that location; and a *connection* item for each (location, request) pair, of the cost of connecting the request to a facility at the given location. To satisfy a request, there must exist a location for which both the opening item, and the connection item to the request, have been bought; note that the upwards-closed property of network design problems holds. Also note that each request requires a separate connection item; thus, no two requests belong to the same type, and thus $n \geq m$. Hence, we do not state competitiveness in terms of n for this problem.

Charikar et al. [15] gave a Lagrangian 3-approximation for the prize-collecting facility location problem. This implies the following corollary of Theorem 15 for facility location.

▶ **Corollary 17.** *There exists an* $O(\min\left\{\sqrt{m \log m}\right\})$*-competitive algorithm for facility location with deadlines/delay which runs in polynomial time.*

Multicut on a tree. Hou et al. [23] gave a Lagrangian 2-approximation for prize-collecting multicut where the underlying graph is a tree. Note that for multicut on a tree, it holds that n is the number of vertex pairs in the tree, i.e., quadratic in the number of vertices. Thus, we obtain the following corollary of Theorem 15 for multicut on a tree.

▶ **Corollary 18.** *There exists an* $O(\min\left\{|V|\sqrt{\log |V|}, \sqrt{m \log m}\right\})$*-competitive algorithm for multicut with deadlines/delay on a tree with vertices* V *which runs in polynomial time.*

5.2 The Algorithm

Consider a problem which admits a Lagrangian γ-approximation, which we denote by $\widetilde{\text{PCND}}$. As a shorthand, we use $\widetilde{\text{ND}}$ to refer to the offline γ-approximation obtained from using $\widetilde{\text{PCND}}$ with the penalties set to ∞.

The procedure PCSOLVE. The algorithm uses $\widetilde{\text{PCND}}$ in the procedure PCSolve (Algorithm 3), which receives a set of request types $\hat{H}$ and penalties π to those requests, and outputs a subset of request types $H' \subseteq \hat{H}$ and a solution S to H'. We prove the following properties of PCSolve:

1. It holds that the cost of solution S to H' is at most $\gamma\pi(H')$.
2. For every subset $H'' \subseteq \hat{H} \setminus H'$, it holds that $\text{ND}(H'') \geq \pi(H'')$.

These two properties make PCSolve a useful primitive, which is used several times in the algorithm.

Algorithm's Description The algorithm is given in Algorithm 4. The algorithm periodically runs the procedure PCSolve on the set of pending requests, where the penalty of every request type is the total current delay of pending requests of that type. Whenever PCSolve returns a non-empty set of request types H' to serve, the procedure starts the service, and the set of pending requests of types H' is called *critical*. This triggers a call to UponCritical.

Inside UponCritical, as in Algorithm 2, the algorithm chooses the subset of critical requests whose variable b_q is True, and looks for a maximal subset of them whose service cost is at most some factor from their delay cost. However, this factor is now linear in the approximation factor γ rather than a constant. To find this request set $R^\star$, the algorithm makes a call to PCSolve.

Now, the service makes its transmissions. First, it transmits the approximate solution previously calculated for the critical requests, thus serving them. Then, it uses PCSolve to find some subset of the request types of eligible requests to serve in the second transmission; it does so by providing a uniform penalty function to PCSolve. The proof of Theorem 15 using Algorithm 4 appears in Appendix B.

Algorithm 3 Prize-collecting procedure.

Function PCSolve *(H, π)*
  Set $\hat{H} \leftarrow H$.
  Set $S \leftarrow \emptyset$.
  while *True* **do**
    Run PCND($\hat{H}, \pi$) to obtain a solution S' which serves some subset $H' \subseteq \hat{H}$ of request types.
    if $H' = \emptyset$ **then break**
    Set $S \leftarrow S \cup S'$, $\hat{H} \leftarrow \hat{H} \setminus H'$
  return $(S, H \setminus \hat{H})$

Algorithm 4 Polynomial Time Framework for Nonclairvoyant Network Design with Delay.

Event Function UponRequest(q)
  $b_q \leftarrow$ True.
Function TestCritical() *// called continuously*
  Let t be the current time, and let Q' be the set of currently-pending requests.
  Define π which maps from request type $h \in H(Q')$ to $\sum_{q \in Q' | h_q = h} d_q(t)$.
  Call PCSolve(Q', π), and obtain the output (H', S).
  if $H' \neq \emptyset$ **then** define $R \leftarrow \{q \in Q' | h_q \in H'\}$.
  call UponCritical(R, S).
Event Function UponCritical(R, S)
  Start a new service λ; denote the current time by t.
  Define $R^\top \leftarrow \{q \in R | b_q = \text{True}\}$.
  Let π_1 map from request type h to $2\gamma \cdot \sum_{q \in R^\star | h_q = h} d_q(t)$.
  Call PCSolve($H(R^\top), \pi_1$), let $H^\star \subseteq H(R^\top)$ be the request types served by the output, and let S^1 be the returned solution for H'.
  Define $R^\star \leftarrow \{q \in R^\top | h_q \in H^\star\}$
  Define $\hat{\ell}_\lambda \leftarrow \left\lceil \log(c(S^1)) \right\rceil$.
  Define E to be the set of pending requests q such that $\check{\ell}(q) \leq \hat{\ell}_\lambda$, and define $H \leftarrow H(E)$.
  Transmit S, serving all requests of types $H(R)$.
  Let $x_\lambda \leftarrow 2^{\hat{\ell}_\lambda} \cdot \sqrt{\frac{\log(1+|H|)}{|H|}}$.
  Let π_2 map from $h \in H$ to x_λ.
  Call PCSolve(H, π_2); let $H' \subseteq H$ and solution S^2 be the output.
  Transmit S^2, serving all requests of types in H'.
  // Set $b_{q'}$ for eligible requests q' which are still pending.
  Let Q_λ be the subset of pending requests served by λ.
  foreach $q' \in E \setminus Q_\lambda$ **do**
    set $b_{q'} \leftarrow$ False.

6 Conclusions and Future Directions

In this paper, we presented frameworks for obtaining $\tilde{O}(\min\{\sqrt{n}, \sqrt{m}\})$-competitive algorithms for network design with deadlines or delay. For some problems, in particular facility location and multilevel aggregation, lower bounds of $\Omega(\sqrt{k})$ and $\Omega(\sqrt{m})$ exist, making these frameworks optimal up to a logarithmic factor. We then discussed running time, and presented a class of problems (namely those that admit Lagrangian prize-collecting approximations) for which these frameworks can be implemented in polynomial time.

An interesting direction for future work would be to implement this framework in polynomial time for additional problems. This could require a different direction from the one in this paper, as not all network design problems seem amenable to Lagrangian prize-collecting approximations. In particular, for Steiner forest, a Lagrangian prize-collecting approximation implies an approximation of similar ratio for k-Steiner forest; however, no subpolynomial approximation for k-Steiner forest is known (see e.g. [22]).

Additionally, we made little attempt to optimize the dependence of the poly-time framework's competitive ratio on the approximation ratio γ of the Lagrangian approximation algorithm. This is since γ is constant for the problems we consider in this paper. However, improving this dependence could be useful for problems which are harder to approximate; we conjecture that a linear dependence is possible.

References

1 Yossi Azar, Ashish Chiplunkar, Shay Kutten, and Noam Touitou. Set cover with delay – clairvoyance is not required. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPIcs*, pages 8:1–8:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ESA.2020.8.

2 Yossi Azar, Arun Ganesh, Rong Ge, and Debmalya Panigrahi. Online service with delay. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 551–563, 2017. doi:10.1145/3055399.3055475.

3 Yossi Azar and Noam Touitou. General framework for metric optimization problems with delay or with deadlines. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 60–71, 2019. doi:10.1109/FOCS.2019.00013.

4 Yossi Azar and Noam Touitou. Beyond tree embeddings – A deterministic framework for network design with deadlines or delay. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1368–1379. IEEE, 2020. doi:10.1109/FOCS46700.2020.00129.

5 Marcin Bienkowski, Martin Böhm, Jaroslaw Byrka, Marek Chrobak, Christoph Dürr, Lukáš Folwarczný, Lukasz Jez, Jirí Sgall, Kim Thang Nguyen, and Pavel Veselý. New results on multi-level aggregation. *Theor. Comput. Sci.*, 861:133–143, 2021. doi:10.1016/j.tcs.2021.02.016.

6 Marcin Bienkowski, Martin Böhm, Jaroslaw Byrka, Marek Chrobak, Christoph Dürr, Lukáš Folwarczný, Lukasz Jez, Jiri Sgall, Nguyen Kim Thang, and Pavel Veselý. Online algorithms for multi-level aggregation. In *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, pages 12:1–12:17, 2016. doi:10.4230/LIPIcs.ESA.2016.12.

7 Marcin Bienkowski, Martin Böhm, Jaroslaw Byrka, and Jan Marcinkowski. Online facility location with linear delay. In Amit Chakrabarti and Chaitanya Swamy, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022, September 19-21, 2022, University of Illinois, Urbana-Champaign, USA (Virtual Conference)*, volume 245 of *LIPIcs*, pages 45:1–45:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.APPROX/RANDOM.2022.45.

8 Marcin Bienkowski, Jaroslaw Byrka, Marek Chrobak, Lukasz Jez, Dorian Nogneng, and Jirí Sgall. Better approximation bounds for the joint replenishment problem. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 42–54, 2014. doi:10.1137/1.9781611973402.4.

9 Marcin Bienkowski, Artur Kraska, and Pawel Schmidt. Online service with delay on a line. In *Structural Information and Communication Complexity – 25th International Colloquium, SIROCCO 2018, Ma'ale HaHamisha, Israel, June 18-21, 2018, Revised Selected Papers*, pages 237–248, 2018. doi:10.1007/978-3-030-01325-7_22.

10 Carlos Fisch Brito, Elias Koutsoupias, and Shailesh Vaya. Competitive analysis of organization networks or multicast acknowledgment: How much to wait? *Algorithmica*, 64(4):584–605, 2012. doi:10.1007/s00453-011-9567-5.

11 Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Ohad Talmon. O(depth)-competitive algorithm for online multi-level aggregation. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1235–1244, 2017. doi:10.1137/1.9781611974782.80.

12 Niv Buchbinder, Kamal Jain, and Joseph Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *Algorithms – ESA 2007, 15th Annual European Symposium, Eilat, Israel, October 8-10, 2007, Proceedings*, pages 253–264, 2007. doi:10.1007/978-3-540-75520-3_24.

13 Niv Buchbinder, Tracy Kimbrel, Retsef Levi, Konstantin Makarychev, and Maxim Sviridenko. Online make-to-order joint replenishment model: primal dual competitive algorithms. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 952–961, 2008. URL: http://dl.acm.org/citation.cfm?id=1347082.1347186.

14 Rodrigo A. Carrasco, Kirk Pruhs, Cliff Stein, and José Verschae. The online set aggregation problem. In *LATIN 2018: Theoretical Informatics – 13th Latin American Symposium, Buenos Aires, Argentina, April 16-19, 2018, Proceedings*, pages 245–259, 2018. doi:10.1007/978-3-319-77404-6_19.

15 Moses Charikar, Samir Khuller, David M. Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In S. Rao Kosaraju, editor, *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA*, pages 642–651. ACM/SIAM, 2001. URL: http://dl.acm.org/citation.cfm?id=365411.365555.

16 Ryder Chen, Jahanvi Khatkar, and Seeun William Umboh. Online weighted cardinality joint replenishment problem with delay. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 40:1–40:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.40.

17 Daniel R. Dooly, Sally A. Goldman, and Stephen D. Scott. TCP dynamic acknowledgment delay: Theory and practice (extended abstract). In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 389–398, 1998. doi:10.1145/276698.276792.

18 Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004. doi:10.1016/j.jcss.2004.04.011.

19 Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '92, pages 307–316, Philadelphia, PA, USA, 1992. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=139404.139468.

20 Anupam Gupta, Amit Kumar, and Debmalya Panigrahi. Caching with time windows. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1125–1138. ACM, 2020. doi:10.1145/3357713.3384277.

21 Anupam Gupta, Amit Kumar, and Debmalya Panigrahi. A hitting set relaxation for k-server and an extension to time-windows. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 504–515. IEEE, 2021. doi:10.1109/FOCS52979.2021.00057.

22 Mohammad Taghi Hajiaghayi and Kamal Jain. The prize-collecting generalized steiner tree problem via a new approach of primal-dual schema. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pages 631–640, Philadelphia, PA, USA, 2006. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=1109557.1109626.

23 Xin Hou, Wen Liu, and Bo Hou. An approximation algorithm for the k-prize-collecting multicut on a tree problem. *Theor. Comput. Sci.*, 844:26–33, 2020. doi:10.1016/j.tcs.2020.07.014.

24 Anna R. Karlin, Claire Kenyon, and Dana Randall. Dynamic TCP acknowledgment and other stories about e/(e-1). *Algorithmica*, 36(3):209–224, 2003.

25 Predrag Krnetic, Darya Melnyk, Yuyi Wang, and Roger Wattenhofer. The k-server problem with delays on the uniform metric space. In Yixin Cao, Siu-Wing Cheng, and Minming Li, editors, *31st International Symposium on Algorithms and Computation, ISAAC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 181 of *LIPIcs*, pages 61:1–61:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ISAAC.2020.61.

26 Ngoc Mai Le, William Umboh, and Ningyuan Xie. The power of clairvoyance for multi-level aggregation and set cover with delay. In *To appear in Symposium on Discrete Algorithms (SODA) 2023*, 2023.

27 Jeremy McMahan. A d-competitive algorithm for the multilevel aggregation problem with deadlines. *CoRR*, abs/2108.04422, 2021. arXiv:2108.04422.

28 Noam Touitou. Nearly-tight lower bounds for set cover and network design with deadlines/delay. In Hee-Kap Ahn and Kunihiko Sadakane, editors, *32nd International Symposium on Algorithms and Computation, ISAAC 2021, December 6-8, 2021, Fukuoka, Japan*, volume 212 of *LIPIcs*, pages 53:1–53:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ISAAC.2021.53.

A Proof of Lemma 13

Henceforth, fix any optimal service $\lambda^* \in \Lambda^*$. For ease of notation, define $R^\perp := \bigcup_{\lambda \in \Lambda | t_\lambda > t_{\lambda^*}} R^\perp_{\lambda,\lambda^*}$. Also define $\ell^* := c(\lambda^*)$. Note that

$$\sum_{\lambda \in \Lambda | t_\lambda > t_{\lambda^*}} D\left(R^\perp_{\lambda,\lambda^*}, t_\lambda\right) \leq \sum_{\lambda \in \Lambda | t_\lambda > t_{\lambda^*}} \sum_{h \in H(R^\perp_{\lambda,\lambda^*})} \mathrm{ND}(h)$$

We claim that no request type appears twice in the summation on the right-hand side of the above equation. That is, we claim that $\sum_{h \in H(R^\perp)} \mathrm{ND}(h) = \sum_{\lambda \in \Lambda | t_\lambda > t_{\lambda^*}} \sum_{h \in H(R^\perp_{\lambda,\lambda^*})} \mathrm{ND}(h)$. Indeed, note that there cannot be two services $\lambda_1, \lambda_2 \in \Lambda$ such that $t_{\lambda^*} \leq t_{\lambda_1} < t_{\lambda_2}$ and requests $q_1 \in R^\perp_{\lambda_1,\lambda^*}, q_2 \in R^\perp_{\lambda_2,\lambda^*}$ such that $h_{q_1} = h_{q_2}$: otherwise, λ_1 would serve q_2. We conclude that

$$\sum_{\lambda \in \Lambda | t_\lambda > t_{\lambda^*}} D\left(R^\perp_{\lambda,\lambda^*}, t_\lambda\right) \leq \sum_{h \in H(R^\perp)} \mathrm{ND}(h) \tag{6}$$

and it is thus enough to bound $\sum_{h \in H(R^\perp)} \mathrm{ND}(h)$. Note that every request $q \in R^\perp$ has had b_q set to False at some point in the algorithm. Define $R^\perp_1 \subseteq R^\perp$ to be the subset of requests q such that b_q was first set to False prior to t_{λ^*}, and define $R^\perp_2 := R^\perp \setminus R^\perp_1$. For every ℓ, further define $R^\perp_{1,\ell} := \left\{q \in R^\perp_1 \middle| \ell(q) = \ell\right\}$; define $R^\perp_{2,\ell}$ analogously.

▶ **Proposition 19.** *For every optimal service λ^*, and for every ℓ, it holds that*

$$\sum_{h \in H(R^{\perp}_{1,\ell})} ND(h) \leq O\left(\sqrt{k/\log k}\right) \cdot c(\lambda^*).$$

Proof. Observe the first service $\lambda \in \Lambda$ after which $b_q = \text{FALSE}$ for every $q \in R^{\perp}_{1,\ell}$. From the definition of $R^{\perp}_{1,\ell}$, we know that $t_\lambda < t_{\lambda^*}$; as $R^{\perp}_{1,\ell}$ are all served after t_{λ^*}, they are all pending both before and after t_λ. As λ set $b_q \leftarrow \text{FALSE}$ for some $q \in R^{\perp}_{1,\ell}$, we have $\ell_\lambda \geq \ell$. This implies $R^{\perp}_{1,\ell} \subseteq E_\lambda$. Define $z := \left|H(R^{\perp}_{1,\ell})\right|$; since no request from $R^{\perp}_{1,\ell}$ was served in λ, we have

$$c(\lambda^*) \geq \text{ND}(R^{\perp}_{1,\ell}) \geq x_\lambda \cdot z \geq 2^{\ell_\lambda} \cdot \sqrt{\frac{\log(1+k)}{k}} \cdot z \geq 2^{\ell} \cdot \sqrt{\frac{\log(1+k)}{k}} \cdot z$$

Noting that $\sum_{h \in H(R^{\perp}_{1,\ell})} \text{ND}(h) \leq z \cdot 2^{\ell}$, this yields $\sum_{h \in H(R^{\perp}_{1,\ell})} \text{ND}(h) \leq \sqrt{\frac{k}{\log(1+k)}} \cdot c(\lambda^*)$. ◀

▶ **Proposition 20.** *For every optimal service λ^*, and for every ℓ, it holds that*

$$\sum_{h \in H(R^{\perp}_{2,\ell})} ND(h) \leq O\left(\sqrt{\frac{k}{\log k}}\right) \cdot c(\lambda^*).$$

Proof. The proof is similar to that of Proposition 19. Consider the first service $\lambda \in \Lambda$ such that $t_\lambda \geq t_{\lambda^*}$ and $\ell_\lambda \geq \ell$. It must be the case that all requests in $R^{\perp}_{2,\ell}$ are pending before and after λ, and moreover, λ sets $b_q \leftarrow \text{FALSE}$ for every request in $R^{\perp}_{2,\ell}$. Define $z := \left|H(R^{\perp}_{2,\ell})\right|$; since no request from $R^{\perp}_{2,\ell}$ was served in λ, we have

$$c(\lambda^*) \geq \text{ND}(R^{\perp}_{2,\ell}) \geq x_\lambda \cdot z \geq 2^{\ell_\lambda} \cdot \sqrt{\frac{\log(1+k)}{k}} \cdot z \geq 2^{\ell} \cdot \sqrt{\frac{\log(1+k)}{k}} \cdot z$$

Noting that $\sum_{h \in H(R^{\perp}_{2,\ell})} \text{ND}(h) \leq z \cdot 2^{\ell}$, the above yields $\sum_{h \in H(R^{\perp}_{2,\ell})} \text{ND}(h) \leq \sqrt{\frac{k}{\log(1+k)}} \cdot c(\lambda^*)$, which completes the proof. ◀

Proof of Lemma 13. Define $\gamma := \lceil \log k \rceil$. The following holds:

$$\begin{aligned}
&\sum_{\lambda \in \Lambda | t_\lambda > t_{\lambda^*}} D\left(R^{\perp}_{\lambda,\lambda^*}, t_\lambda\right) \leq \sum_{h \in H(R^{\perp})} \text{ND}(h) \leq \sum_{\ell=-\infty}^{\infty} \left(\sum_{h \in H(R^{\perp}_{1,\ell})} \text{ND}(h) + \sum_{h \in H(R^{\perp}_{2,\ell})} \text{ND}(h) \right) \\
&= \sum_{\ell=-\infty}^{\ell^*-\gamma} \left(\sum_{h \in H(R^{\perp}_{1,\ell})} \text{ND}(h) + \sum_{h \in H(R^{\perp}_{2,\ell})} \text{ND}(h) \right) + \sum_{\ell=\ell^*-\gamma+1}^{\ell^*} \left(\sum_{h \in H(R^{\perp}_{1,\ell})} \text{ND}(h) + \sum_{h \in H(R^{\perp}_{2,\ell})} \text{ND}(h) \right)
\end{aligned} \tag{7}$$

where the first inequality uses Equation (6), the second inequality partitions $R^{\perp}$ into $\left\{R^{\perp}_{1,\ell}\right\}_\ell$ and $\left\{R^{\perp}_{2,\ell}\right\}_\ell$, and the equality makes use of the fact that $R^{\perp}$ does not contain requests q such that $\ell(q) > \ell^*$ (since $R^{\perp} \subseteq Q_{\lambda^*}$). From the proof of Proposition 19, we know that for every ℓ the requests $R^{\perp}_{1,\ell}$ were all pending during a single service. Thus, we have $\left|H(R^{\perp}_{1,\ell})\right| \leq k$, and therefore $\sum_{h \in H(R^{\perp}_{1,\ell})} \text{ND}(h) \leq 2^{\ell} \cdot k$. Similarly, using the proof of Proposition 20, we have $\left|H(R^{\perp}_{2,\ell})\right| \leq k$ and thus $\sum_{h \in H(R^{\perp}_{2,\ell})} \text{ND}(h) \leq 2^{\ell} \cdot k$. We can therefore conclude that

$$\sum_{\ell=-\infty}^{\ell^*-\gamma} \left(\sum_{h \in H(R^{\perp}_{1,\ell})} \text{ND}(h) + \sum_{h \in H(R^{\perp}_{2,\ell})} \text{ND}(h) \right) \leq 4 \cdot 2^{\ell^*-\gamma} \cdot k \leq 4 \cdot 2^{\ell^*} \leq 8 \cdot c(\lambda^*). \tag{8}$$

Moreover, using Propositions 19 and 20,

$$\sum_{\ell=\ell^*-\gamma+1}^{\ell^*}\left(\sum_{h\in H(R^{\perp}_{1,\ell})}\mathrm{ND}(h)+\sum_{h\in H(R^{\perp}_{2,\ell})}\mathrm{ND}(h)\right)\leq\gamma\cdot O\left(\sqrt{\frac{k}{\log k}}\right)\cdot c(\lambda^*)=O(\sqrt{k\log k})\cdot c(\lambda^*). \quad (9)$$

Combining Equations (7) to (9) yields $\sum_{\lambda\in\Lambda|t_\lambda>t_{\lambda^*}} D\left(R^{\perp}_{\lambda,\lambda^*},t_\lambda\right)\leq O(\sqrt{k\log k})\cdot c(\lambda^*)$. ◀

B Analysis of Lagrangian Approximation Framework

We focus on proving Theorem 15, following the same lines as the proof of Theorem 8. Following the notation of Section 4, we use the subscript λ to refer to the values of variables in the UponCritical call that started service λ. For example, this includes S^1_λ, S^2_λ.

▶ **Proposition 21** (Properties of PCSolve). *Suppose* PCSolve *is called on request types H and penalties π, and outputs H' and solution S. It holds that:*

1. *The cost of solution S to H' is at most $\gamma\pi(H')$.*
2. *For every subset $H''\subseteq H\backslash H'$, it holds that $ND(H'')\geq\pi(H'')$.*

Proof. Let b be the number of iterations of the main loop in PCSolve. We use subscript i to refer to the value of a variable in the i'th iteration of the loop; note that $c(S)\leq\sum_{i\in[b]}c(S_i)$. For every iteration i, through the Lagrangian approximation guarantee, it holds that $c(S_i)+\gamma\cdot\pi(\hat{H}_i\backslash H'_i)\leq\gamma\cdot\pi(\hat{H}_i)$, implying $c(S_i)\leq\gamma\cdot\pi(H'_i)$; thus, we have $c(S)\leq\sum_i\pi(H'_i)=\gamma\pi(H')$, proving the first claim.

To prove the second claim, observe that in the final iteration no request types from $\hat{H}_b$ were served. Through the Lagrangian guarantee, $\gamma\pi(\hat{H}_b)\leq\gamma\cdot(\mathrm{ND}(H'')+\pi(\hat{H}_b\backslash H''))$ for every subset $H''\subseteq\hat{H}_b$, which implies that $\mathrm{ND}(H'')\geq\pi(H'')$. Observing that $\hat{H}_b=H\backslash H'$ completes the proof of the second claim. ◀

▶ **Proposition 22** (Analogue of Proposition 10). *For a service λ, it holds that*

$$c(\lambda)\leq O(\gamma)\cdot\sum_{\lambda^*\in\Lambda^*}D\left(R^{\perp}_{\lambda,\lambda^*},t_\lambda\right)+O(\gamma^3\sqrt{k\log k})\cdot\sum_{\lambda^*\in\Lambda^*}D\left(R^{\star}_{\lambda,\lambda^*},t_\lambda\right)$$

Proof. According to Proposition 21, the cost of the first transmission of solution S_λ (Line 17) is at most $\gamma\cdot D(R_\lambda,t_\lambda)$. However, Proposition 21 also implies that $2\gamma D\left(R^{\top}_\lambda\backslash R^{\star}_\lambda,t_\lambda\right)\leq\mathrm{ND}(R^{\top}_\lambda\backslash R^{\star}_\lambda)$; together with the fact that S_λ serves $R^{\top}_\lambda\backslash R^{\star}_\lambda$ implies that $\gamma D\left(R^{\top}_\lambda\backslash R^{\star}_\lambda,t_\lambda\right)\leq c(S_\lambda)/2$. Combining, we have the following:

$$c(S_\lambda)\leq\gamma D(R_\lambda,t_\lambda)=\gamma(D\left(R^{\perp}_\lambda,t_\lambda\right)+D\left(R^{\star}_\lambda,t_\lambda\right))+c(S_\lambda)/2$$

Simplifying, $c(S_\lambda)\leq 2\gamma(D\left(R^{\perp}_\lambda,t_\lambda\right)+D\left(R^{\star}_\lambda,t_\lambda\right))$, yielding the bound for the first transmission.

For the second transmission, note that $2^{\hat{\ell}_\lambda}\leq 2\cdot c\left(S^1_\lambda\right)\leq 4\gamma^2 D(R^\star,t_\lambda)$, where the second inequality uses Proposition 21 for PCSolve. Applying Proposition 21 again for Line 20, we obtain the following bound for the cost of the solution S^2_λ used for the second transmission:

$$\begin{aligned}c\left(S^2_\lambda\right)&\leq\gamma\cdot|H_\lambda|\cdot x_\lambda=\gamma\cdot|H_\lambda|\cdot 2^{\hat{\ell}_\lambda}\cdot\sqrt{\log(1+|H_\lambda|)/|H_\lambda|}\leq\gamma\sqrt{k\log(1+k)}\cdot 2^{\hat{\ell}_\lambda}\\&\leq 4\gamma^3\cdot\sqrt{k\log(1+k)}\cdot D\left(R^\star,t_\lambda\right)\end{aligned}$$

Combining this with the previous bound for the first transmission completes the proof. ◀

We henceforth define joint costs $c(\lambda, \lambda^*)$ as in Section 4. Note that Proposition 10 holds for Algorithm 4 without modification.

▶ **Proposition 23** (Analogue of Proposition 12). *For every optimal service λ^*, it holds that*

$$\sum_{\lambda \in \Lambda | t_\lambda > t_{\lambda^*}} D\left(R^\star_{\lambda,\lambda^*}, t_\lambda\right) \leq O(1) \cdot c(\lambda^*).$$

Proof. Let $\Lambda' \subseteq \Lambda$ be the subset of services that occurred after t_{λ^*}. First, we claim that for every integer ℓ, there exists at most one service in Λ' with $\ell_{\lambda,\lambda^*} = \ell$. To prove the claim, assume for contradiction that there are two services $\lambda_1, \lambda_2 \in \Lambda'$ such that $\ell_{\lambda_1,\lambda^*} = \ell_{\lambda_2,\lambda^*} = \ell$, and assume without loss of generality that $t_{\lambda_2} > t_{\lambda_1} > t_{\lambda^*}$. As $t_{\lambda_1} > t_{\lambda^*}$, it must be that all requests in $R^\star_{\lambda_2,\lambda^*}$ were pending before and after t_{λ_1}.

Moreover, every request $q \in R^\star_{\lambda_2,\lambda^*}$ must have $\check{\ell}(q) \leq \ell(q) \leq \ell\left(R^\star_{\lambda_2,\lambda^*}\right) = \ell$. However, $\hat{\ell}_{\lambda_1} \geq \ell_{\lambda_1,\lambda^*} = \ell$, and thus $R^\star_{\lambda_2,\lambda^*} \subseteq E_{\lambda_1}$. But, Line 15 in λ_1 sets $b_q = \textsc{False}$ for every $q \in R^\star_{\lambda_2,\lambda^*}$, in contradiction to having $b_q = \textsc{True}$ at t_{λ_2}. We conclude that $R^\star_{\lambda_2,\lambda^*} = \emptyset$; however, this implies that $\ell_{\lambda_2,\lambda^*} = -\infty$, in contradiction to $\ell_{\lambda_2,\lambda^*} = \ell$.

Using this claim, for every level ℓ, there exists at most one service $\lambda \in \Lambda'$ such that $\ell_{\lambda,\lambda^*} = \ell$. For such λ, it holds that $D\left(R^\star_{\lambda,\lambda^*}, t_\lambda\right) \leq \mathrm{ND}(R^\star_{\lambda,\lambda^*}) \leq 2^\ell$. Also note that $\max_{\lambda \in \Lambda'} \ell_{\lambda,\lambda^*} \leq \ell(Q_{\lambda^*})$. Summing over the possible levels, we get that

$$\sum_{\lambda \in \Lambda | t_\lambda > t_{\lambda^*}} D\left(R^\star_{\lambda,\lambda^*}, t_\lambda\right) \leq 2 \cdot 2^{\max_{\lambda \in \Lambda'} \ell_{\lambda,\lambda^*}} \leq 2 \cdot 2^{\ell(Q_{\lambda^*})} \leq 4 \cdot c(\lambda^*). \qquad \blacktriangleleft$$

▶ **Lemma 24** (Analogue of Lemma 13). *For every optimal service $\lambda^* \in \Lambda$, it holds that $\sum_{\lambda \in \Lambda | t_\lambda > t_{\lambda^*}} D\left(R^\perp_{\lambda,\lambda^*}, t_\lambda\right) \leq O(\log\gamma \cdot \sqrt{k \log k}) \cdot c(\lambda^*)$.*

Proof sketch. The proof follows the same main lines as that of Lemma 13. First, define $R^\perp, R^\perp_1, R^\perp_2$ as in the proof of Lemma 13. Now, define $R^\perp_{1,\ell} = \left\{q \in R^\perp_1 | \hat{\ell}_q = \ell\right\}$; define $R^\perp_{2,\ell}$ analogously. Note that the $\hat{\ell}_q$ is used for these definitions, rather than ℓ_q.

We can prove analogues to Proposition 19 and Proposition 20, using identical proofs. Specifically, for every optimal service λ^*, for every ℓ and for every $b \in \{1, 2\}$, it holds that

$$\sum_{h \in H(R^\perp_{b,\ell})} \mathrm{ND}(h) \leq O\left(\sqrt{k/\log k}\right) \cdot c(\lambda^*) \tag{10}$$

Now, following the proof of Lemma 13, define $\delta := \lceil \log k \rceil + \lceil \log \gamma \rceil$.

$$\begin{aligned}\sum_{\lambda \in \Lambda | t_\lambda > t_{\lambda^*}} D\left(R^\perp_{\lambda,\lambda^*}, t_\lambda\right) &\leq \sum_{h \in H(R^\perp)} \mathrm{ND}(h) \leq \sum_{\ell=-\infty}^{\infty} \left(\sum_{h \in H(R^\perp_{1,\ell})} \mathrm{ND}(h) + \sum_{h \in H(R^\perp_{2,\ell})} \mathrm{ND}(h) \right) \\ &= \sum_{\ell=-\infty}^{\ell^*-\delta} \left(\sum_{h \in H(R^\perp_{1,\ell})} \mathrm{ND}(h) + \sum_{h \in H(R^\perp_{2,\ell})} \mathrm{ND}(h) \right) + \sum_{\ell=\ell^*-\delta+1}^{\ell^*} \left(\sum_{h \in H(R^\perp_{1,\ell})} \mathrm{ND}(h) + \sum_{h \in H(R^\perp_{2,\ell})} \mathrm{ND}(h) \right)\end{aligned} \tag{11}$$

Using a similar argument to that in Lemma 13, we note that $\sum_{h \in H(R^\perp_{1,\ell})} \mathrm{ND}(h) \leq k \cdot \gamma \cdot 2^\ell$; a similar bound applies to $\sum_{h \in H(R^\perp_{2,\ell})} \mathrm{ND}(h)$. Combining with the definition of δ, the first term in the RHS of Equation (11) can be bounded by $O(1) \cdot c(\lambda^*)$. Using Equation (10), the second term can be bounded by $O(\delta) \cdot c(\lambda^*)$, which is $O((\log k + \log \gamma) \cdot \sqrt{k/\log k}) \cdot c(\lambda^*)$; this is at most $O(\log\gamma \cdot \sqrt{k \log k}) \cdot c(\lambda^*)$. This completes the proof. ◀

Proof of Theorem 15. Results from combining Propositions 22 and 23 and Lemma 24. ◀

Tight Bounds for Chordal/Interval Vertex Deletion Parameterized by Treewidth

Michał Włodarczyk
Ben-Gurion University, Beer Sheva, Israel

Abstract

In Chordal/Interval Vertex Deletion we ask how many vertices one needs to remove from a graph to make it chordal (respectively: interval). We study these problems under the parameterization by treewidth **tw** of the input graph G. On the one hand, we present an algorithm for Chordal Vertex Deletion with running time $2^{\mathcal{O}(\mathbf{tw})} \cdot |V(G)|$, improving upon the running time $2^{\mathcal{O}(\mathbf{tw}^2)} \cdot |V(G)|^{\mathcal{O}(1)}$ by Jansen, de Kroon, and Włodarczyk (STOC'21). When a tree decomposition of width **tw** is given, then the base of the exponent equals $2^{\omega-1} \cdot 3 + 1$. Our algorithm is based on a novel link between chordal graphs and graphic matroids, which allows us to employ the framework of representative families. On the other hand, we prove that the known $2^{\mathcal{O}(\mathbf{tw} \log \mathbf{tw})} \cdot |V(G)|$-time algorithm for Interval Vertex Deletion cannot be improved assuming Exponential Time Hypothesis.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases fixed-parameter tractability, treewidth, chordal graphs, interval graphs, matroids, representative families

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.106

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2305.03440`

Funding Supported by the European Research Council (ERC) grant titled PARAPATH. Part of the work has been carried out when the author was a postdoctorate student at the Eindhoven University of Technology.

1 Introduction

Treewidth [32, §7] is arguably the most extensively studied width measure in the graph theory. Simply speaking, treewidth measures to what extent a graph is similar to a tree, where trees and forests are exactly the graphs of treewidth 1. It plays a crucial role in Robertson and Seymour's Graph Minors series [62]. The usefulness of treewidth stems from the fact that a broad class of problems can be solved in linear time on graphs of bounded treewidth. The celebrated Courcelle's Theorem [30] states that any graph problem expressible in the Counting Monadic Second Order Logic (CMSO) can be solved in time $f(\mathbf{tw}) \cdot |V(G)|$, where **tw** denotes the treewidth of graph G and f is some computable function. In other words, every such problem is fixed-parameter tractable (FPT) when parameterized by treewidth. Furthermore, bounded-treewidth graphs appear in a wide variety of contexts, which makes treewidth-based algorithms a ubiquitous tool in algorithm design [36, 47, 56, 57, 61]. The function f from Courcelle's Theorem may grow very rapidly and a large body of research has been devoted to optimize the dependency on **tw** for particular problems. In the ideal scenario, we would like the function f to be single-exponential, i.e., $f(\mathbf{tw}) = 2^{\mathcal{O}(\mathbf{tw})}$, while possibly allowing a higher (yet constant) exponent at $|V(G)|$. This is often the best we can hope for because sub-exponential running times usually contradict the Exponential Time Hypothesis[1] (ETH) [42].

[1] The Exponential Time Hypothesis states that there exists a constant $\delta > 0$ so that 3-SAT cannot be solved in time $\mathcal{O}(2^{\delta n})$ on n-variable formulas.

While the standard dynamic programming technique yields single-exponential algorithms for problems with "local constraints", such as Vertex Cover, Dominating Set, or Bipartization, it falls short for problems with "connectivity constraints", such as Feedback Vertex Set, Hamiltonian Cycle, or Connected Vertex Cover, leading to parameter dependency $f(\mathbf{tw}) = 2^{\mathcal{O}(\mathbf{tw}\log\mathbf{tw})}$. On the one hand, this issue was dealt with in the landmark work of Cygan et al. [35], who introduced the Cut & Count technique and obtained randomized single-exponential algorithms for the problems above, among others (see also [59]). In following works, Bodlaender et al. [19] and Fomin et al. [38] presented alternative techniques that allow to circumvent randomization: matrix-based approaches and representative families. On the other hand, Lokshtanov et al. [54] provided a framework for proving "slightly super-exponential" lower bounds under ETH, which paved the way for establishing tight lower bounds for problems that require dependency $f(\mathbf{tw}) = 2^{\mathcal{O}(\mathbf{tw}\log\mathbf{tw})}$. In the same work, they obtained such bounds for Disjoint Paths and Chromatic Number. For problems with a single-exponential dependency $f(\mathbf{tw}) = \mathcal{O}(c^{\mathbf{tw}})$, further research has been devoted to establish the optimal base of the exponent c [31, 33, 35, 53, 67].

Vertex-deletion problems. Many optimization graph problems can be phrased in terms of $\mathcal{H}$-Vertex Deletion: remove the smallest number of vertices from a graph so that the resulting graph belongs to the graph class $\mathcal{H}$. For example, Vertex Cover corresponds to the class $\mathcal{H}$ of edge-less graphs. There is a diverse complexity landscape of ETH-tight running times for various vertex-deletion problems under treewidth parameterization. The classes $\mathcal{H}$ for which tight bounds have been established include: edge-less graphs [53], forests [35] (see also [15]), planar graphs [47, 58], classes defined by a connected forbidden minor [9] (see also [10, 11, 12]), bipartite graphs [53], DAGs [22], even-cycle-free graphs [15, 44], and some classes defined by a forbidden (induced) subgraph [34, 65].

We extend this list by studying the vertex-deletion problems into the classes of chordal and interval graphs. A graph is chordal if it does not contain an induced cycle of length at least 4 (a hole) and a graph is interval if it is an intersection graph of intervals on the real line. Any interval graph is chordal and any chordal graph is perfect. Applications of these two graph classes have been long studied in miscellaneous areas of discrete optimization [8, 14, 25, 50, 60, 63]. On the theoretical side, the treewidth (resp. pathwidth) of a graph G equals the minimum clique number of a chordal (resp. interval) supergraph of G [32, 52]. Moreover, some hard problems become tractable on chordal or interval graphs (or even on graphs with small vertex-deletion distance to chordality) [26, 43, 49].

Our results. The state of the art for Chordal Vertex Deletion (ChVD) is the running time $2^{\mathcal{O}(\mathbf{tw}^2)}n^{\mathcal{O}(1)}$, which follows from a more general result for a hybrid graph measure $\mathcal{H}$-treewidth, where $\mathcal{H} = \texttt{chordal}$ [45]. We improve the dependency on treewidth to single-exponential.

▶ **Theorem 1.1.** *Chordal Vertex Deletion can be solved in deterministic time $\mathcal{O}(c^k k^{\omega+1} n)$ on n-vertex node-weighted graphs when a tree decomposition of width k is provided. The constant c equals $2^{\omega-1}\cdot 3 + 1$.*

Here, $\omega < 2.373$ stands for the matrix multiplication exponent [7]. To prove Theorem 1.1 we establish a new link between chordal graphs and graphic matroids, which allows us to exploit the framework of representative families [37, 38]. ChVD is at least as hard as Feedback Vertex Set, what implies barriers for a significant improvement in the constant c (see Lemma 4.1 and the discussion therein). Thanks to a single-exponential constant-factor FPT approximation for treewidth [20], Theorem 1.1 gives running time $2^{\mathcal{O}(\mathbf{tw})}n$ even when no tree decomposition is provided in the input.

The best known running time for Interval Vertex Deletion is $2^{\mathcal{O}(\mathbf{tw}\log \mathbf{tw})}n$ [64]. (While this algorithm has been described for the edge-deletion variant, we briefly explain in the full version of the article how it can be adapted for vertex deletion.) We show that, unlike the chordal case, this running time is optimal under ETH. This gives a sharp separation between the two studied problems.

▶ **Theorem 1.2.** *Under the assumption of ETH, Interval Vertex Deletion cannot be solved in time* $2^{o(\mathbf{tw}\log \mathbf{tw})}n^{\mathcal{O}(1)}$ *on* n*-vertex unweighted graphs of treewidth* $\mathbf{tw}$*.*

In fact, we show a stronger lower bound that rules out the same running time with respect to a different graph parameter, called treedepth, which is never smaller than treewidth. Our lower bound is obtained via a reduction from $k \times k$ Permutation Clique [54], which produces an instance of size $2^{\mathcal{O}(k)}$ and treedepth $\mathcal{O}(k)$.

Related work. The two considered $\mathcal{H}$-Vertex Deletion problems have been studied in several contexts. Both problems are FPT parameterized by the solution size k, with the best-known running times $\mathcal{O}(8^k(n+m))$ for $\mathcal{H} = \texttt{interval}$ [27] and $2^{\mathcal{O}(k\log k)}n^{\mathcal{O}(1)}$ for $\mathcal{H} = \texttt{chordal}$ [28] (but the problem becomes W[2]-hard for $\mathcal{H} = \texttt{perfect}$ [41]). There are polynomial-time approximation algorithms with approximation factor 8 for $\mathcal{H} = \texttt{interval}$ [27] and $k^{\mathcal{O}(1)}$ for $\mathcal{H} = \texttt{chordal}$ [48]. Observe that, in these two regimes, vertex deletion into chordal graphs seems harder than into interval graphs (although no lower bounds are known to justify such a separation formally); this contrasts our results with respect to the treewidth parameterization.

Both studied problems admit exact exponential algorithms with running times of the form $\mathcal{O}((2-\varepsilon)^n)$ [18] as well as polynomial kernelizations [3, 4, 48]. The obstructions to being chordal (resp. interval) enjoy the Erdős-Pósa property: any graph G either contains k vertex-disjoint subgraphs which are not chordal (resp. not interval) or a vertex set X of size $\mathcal{O}(k^2\log k)$ such that $G-X$ is chordal [51] (resp. interval [2]). Vertex deletion into other subclasses of perfect graphs has been studied as well [1, 5, 6, 68]. For other modification variants, where instead of vertex deletions one considers removals, insertions, or contractions of edges, see, e.g., [17, 26, 27, 28, 39, 55, 70].

The concept of representative families, which plays an important role in our algorithm for ChVD, has found applications outside the context of treewidth as well [66, 71]. Our other tool, boundaried graphs, has revealed fruitful insights for various graph classes [9, 21, 45].

Organization of the paper. We begin by describing our technical contributions informally in Section 2. We provide basic preliminaries in Section 3, while the extended preliminaries including tree decompositions and representative families can be found in the full version of the article. Section 4 is devoted to establishing a connection between chordal graphs and graphic matroids. The description of the dynamic programming algorithm over a tree decomposition follows standard conventions and is provided in the full version. In Section 5 we prove our lower bound for Interval Vertex Deletion. We conclude in Section 6. The proofs of statements indicated with ($\star$) are postponed to the full version. The numbering of statements is adjusted to match in both versions.

2 Techniques

Chordal Vertex Deletion. The standard approach to design algorithms over a bounded-width tree decomposition is to assign a data structure to each node t in the decomposition, which stores information about partial solutions for the subgraph associated with the subtree of t. Suppose that $X \subseteq V(G)$ is a bag of t, $A \subseteq V(G)\setminus X$ denote the set of vertices appearing

in the bags of the descendants of t (but not in X), and $B \subseteq V(G)$ is the set of remaining vertices. We say that a subset $S \subseteq V(G)$ is a *solution* if $G[S]$ is chordal; we want to *maximize* the size of S. Next, a pair $(S_A \subseteq A, S_X \subseteq X)$ is a *partial solution* if $G[S_A \cup S_X]$ is chordal. A set $S_B \subseteq B$ is an *extension* of a partial solution (S_A, S_X) if $S_A \cup S_X \cup S_B$ is a solution. Since X separates S_A from S_B, the graph $G[S_A \cup S_X \cup S_B]$ can be regarded as a result of *gluing* $G[S_A \cup S_X]$ with $G[S_B \cup S_X]$ alongside the *boundary* S_X. For a node t and $S_X \subseteq X$, we want to store a family of partial solutions $\mathcal{G}_{t,S_X}$ so that for every possible $S_B \subseteq B$: if S_B is an extension for some partial solution (S_A, S_X), then there exists a partial solution $(S'_A, S_X) \in \mathcal{G}_{t,S_X}$ for which (a) S_B is still a valid extension, and (b) S'_A is at least as large as S_A. We say that such a family satisfies the correctness invariant for (t, S_X).

Jansen et al. [45] showed that any chordal graph H with a boundary of size k can be *condensed* to a graph H' on $\mathcal{O}(k)$ vertices that exhibits the same behavior in terms of gluing. More precisely, the gluing product of H with any graph J is chordal if and only if the gluing product of H' with J is chordal. Since there are $2^{\mathcal{O}(\mathbf{tw}^2)}$ graphs on $\mathcal{O}(\mathbf{tw})$ vertices and $2^{\mathcal{O}(\mathbf{tw})}$ choices for the boundary S_X, it suffices to store only $2^{\mathcal{O}(\mathbf{tw}^2)}$ partial solutions.

We take this idea one step further and show that it is actually sufficient to store only $2^{\mathcal{O}(\mathbf{tw})}$ partial solutions. To this end, we investigate the properties of the class of chordal graphs with respect to the gluing operation and prove a homomorphism theorem relating it to graphic matroids. A *graphic matroid* of a graph J is a set system $\mathcal{I}$ over $E(J)$ where a subset $S \subseteq E(J)$ belongs to $\mathcal{I}$ (and is called *independent*) when S contains no cycles. A *rank* of a matroid is the largest size of an independent set; here this coincides with the size of any spanning forest in J. In the following statement, $\mathcal{G}_{X,B}$ is a family of graphs H that satisfy (a) $V(H) \supseteq X$ and (b) $H[X] = B$. For graphs $H_1, H_2 \in \mathcal{G}_{X,B}$ we assume that $V(H_1) \cap V(H_2) = X$ and define their gluing product as $H_3 = (H_1, X) \oplus (H_2, X)$ where $V(H_3) = V(H_1) \cup V(H_2)$ and $E(H_3) = E(H_1) \cup E(H_2)$.

▶ **Theorem 2.1.** *Consider a family of graphs $\mathcal{G}_{X,B}$ for some pair (X, B). There exists a graphic matroid $M = (E, \mathcal{I})$ of rank at most $|X|-1$ and a polynomial-time computable mapping $\sigma : \mathcal{G}_{X,B} \to 2^E$ such that $(H_1, X) \oplus (H_2, X)$ is chordal if and only if $\sigma(H_1) \cap \sigma(H_2) = \emptyset$ and $\sigma(H_1) \cup \sigma(H_2) \in \mathcal{I}$.*

With this criterion at hand, we can employ the machinery of representative families to truncate the number of partial solutions to be stored for a node of a tree decomposition. Technical details aside, for a family $\mathcal{S}$ of independent sets in a matroid $M = (E, \mathcal{I})$, a subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ is called *representative* for $\mathcal{S}$ if for every independent set Y in M: if there exists $X \in \mathcal{S}$ so that $X \cap Y = \emptyset$ and $X \cup Y \in \mathcal{I}$, then there exists $\widehat{X} \in \widehat{\mathcal{S}}$ so that $\widehat{X} \cap Y = \emptyset$ and $\widehat{X} \cup Y \in \mathcal{I}$. Fomin et al. [38] showed that for any family $\mathcal{S}$ in a graphic matroid (more generally, in a linear matroid) of rank k there exists a representative family of size at most 2^k and it can be constructed in time $2^{\mathcal{O}(k)}$. We use Theorem 2.1 to translate this result into the language of chordal graphs and gluing. When $\mathcal{G}_{t,S_X}$ is a family of partial solutions that satisfies the correctness invariant for (t, S_X), a representative family for $\sigma(\mathcal{G}_{t,S_X})$ in the related graphic matroid M corresponds to a subfamily $\widehat{\mathcal{G}}_{t,S_X} \subseteq \mathcal{G}_{t,S_X}$ that satisfies condition (a) of the correctness invariant and $|\widehat{\mathcal{G}}_{t,S_X}| \leq 2^{\mathbf{tw}}$. In order to satisfy condition (b), we need to assign weights to the elements of the matroid M, encoding the size of the largest partial solution mapped to each element. We can then utilize the weighted variant of representative families, which preserves the largest-weight elements [38]. By storing only the condensed forms of the partial solutions (having $\mathcal{O}(\mathbf{tw})$ vertices), we also achieve a linear dependency on $|V(G)|$.

In order to prove Theorem 2.1, we give a novel criterion for testing chordality of a gluing product. When G originates from gluing two chordal graphs G_1, G_2 alongside boundary X, then any hole in G must visit both $V(G_1) \setminus X$ and $V(G_2) \setminus X$, so it must traverse X multiple times. We show that if a hole H intersects at least two connected components of

$G[X]$, then it corresponds to a cycle in the graph obtained from G by contracting each of the connected components of $G[X]$, $G_1 - X$, $G_2 - X$ into single vertices. Otherwise, let C be the unique connected component of $G[X]$ that is intersected by the hole. We prove that there exists a vertex set $S \subseteq V(C)$ that is disjoint from $V(H)$ and $C - S$ has two connected components C_1, C_2 satisfying $N_C(C_1) = N_C(C_2) = S$ (below we refer to such components as *relevant*) and having non-empty intersections with $V(H)$. Moreover, every vertex from $V(H) \cap C$ belongs to some relevant component. Consider a graph $\texttt{Aux}(G, X, S)$ obtained from G by (1) removing the connected components of $G[X]$ different than C, (2) contracting relevant components of $C - S$ into single vertices while removing the irrelevant ones, and (3) contracting the components of $G_1 - X$, $G_2 - X$ into single vertices. A detailed construction is given in Definition 4.10; see also Figure 1 on page 9. Then the hole H corresponds to a cycle in $\texttt{Aux}(G, X, S)$. The first scenario can be analyzed with this approach as well, by taking $S = \emptyset$. We prove that considering all minimal vertex separators S in $G[X]$ and checking acyclity of each auxiliary graph $\texttt{Aux}(G, X, S)$ yields a necessary and sufficient condition for G to be chordal.

This criterion allows us to construct a graphic matroid encoding all the information about each of the graphs G_1, G_2 necessary to reconstruct the graphs $\texttt{Aux}(G, X, S)$ and to determine whether G is chordal. In order to bound the rank of this matroid, we investigate the structure of minimal vertex separators in a chordal graph and bound the size of a spanning forest in a certain graph obtained from the union of $\texttt{Aux}(G, X, S)$. A criterion of a similar kind is known for testing planarity of a gluing product of planar graphs when the boundary has a Hamiltonian cycle; then the corresponding auxiliary graph (defined in a different way) should be bipartite [13]. Our criterion can be also compared to the one used by Bonnet et al. [23] for analyzing gluing products with respect to certain subclasses of chordal graphs. We elaborate more on their approach in the full version of the paper.

Interval Vertex Deletion. In order to prove Theorem 1.2 we present a parameterized reduction from $k \times k$ Permutation Clique. Here, the input is a graph G on vertex set $[k] \times [k]$, and we ask whether there exists a permutation $\pi\colon [k] \to [k]$ such that $(1, \pi(1)), (2, \pi(2)), \ldots, (k, \pi(k))$ forms a clique in G. Lokshtanov et al. [54] proved that $k \times k$ Permutation Clique cannot be solved in time $2^{o(k \log k)}$ under ETH. So we seek a reduction from $k \times k$ Permutation Clique to Interval Vertex Deletion that produces a graph of treewidth $\mathcal{O}(k)$.

Imagine an interval model of a complete graph Y on vertex set $[k]$ in which all the right endpoints of the intervals coincide and all the left endpoints are distinct. Choosing the order of the left endpoints encodes some permutation $\pi\colon [k] \to [k]$ (see Figure 2 on page 13). We can extend this interval model by inserting a new vertex v only if $N(v)$ corresponds to a set of intervals intersecting at a single point. This is possible only when $N(v) = \pi([\ell])$ for some $\ell \in [k]$. Furthermore, inserting to Y independent vertices $v_1, v_2, \ldots, v_k$, such that $|N(v_i)| = i$ and $N(v_i) \subset N(v_{i+1})$, enforces the choice of permutation π. We can thus encode a permutation π by an ascending family of sets $N_1 \subset N_2 \subset \cdots \subset N_k = [k]$, satisfying $N_i = \pi([i])$, which correspond to the neighborhoods of $v_1, v_2, \ldots, v_k$ in Y. On the other hand, any ascending family of sets for which the construction above gives an interval graph, must encode some permutation. On an intuitive level, a partial interval model of a size-k separator can encode one of $k!$ permutations.

We need a mechanism to verify that a chosen permutation π encodes a clique, i.e., that it satisfies $\binom{k}{2}$ constraints of the form $(i, \pi(i))(j, \pi(j)) \in E(G)$. To implement a single constraint, we construct a *choice gadget*, inspired by the reduction to Planar Vertex Deletion [58]. Such a gadget $C_{i,j}$ is defined as a path-like structure, divided into blocks, so that each block has some special vertices adjacent to Y (see Figure 3 on page 14). We show that

any minimum-size interval deletion set in $C_{i,j}$ must "choose" one block and leave its special vertices untouched while it can remove the remaining special vertices. We use this gadget to check if a permutation π encoded by an ascending family of sets $N_1 \subset N_2 \subset \cdots \subset N_k$ satisfies the constraint $(i, \pi(i))(j, \pi(j)) \in E(G)$. As $\pi(i)$ is the only element in $N_i \setminus N_{i-1}$, this information can be extracted from the tuple $(N_{i-1}, N_i, N_{j-1}, N_j)$. We create a single block in $C_{i,j}$ for each valid tuple. Since the number of such tuples is $2^{\mathcal{O}(k)}$, we need a choice gadget of exponential length, unlike the mentioned reduction which works in polynomial time. However, producing an instance of size $2^{\mathcal{O}(k)}$ and treewidth $\mathcal{O}(k)$ is still sufficient to achieve the claimed lower bound.

3 Preliminaries

We write $[k] = \{1, 2, \ldots, k\}$ and assume that $[0] = \emptyset$. We abbreviate $X \setminus v = X \setminus \{v\}$. For a function $w\colon X \to \mathbb{N}$ and $S \subseteq X$ we use shorthand $w(S) = \sum_{x \in S} w(x)$. We follow the standard notational conventions for graphs, which are omitted from this extended abstract.

Separators. For vertices $u, v \in V(G)$ a vertex set $S \subseteq V(G) \setminus \{u, v\}$ is called a (u, v)-separator if u, v belong to different connected components of $G - S$. A (u, v)-separator is minimal when no proper subset of it is a (u, v)-separator. A vertex set S is called a minimal vertex separator if S is a minimal (u, v)-separator for some $u, v \in V(G)$.

▶ **Lemma 3.1** (⋆). *Let u, v be vertices in a graph G and S be a (u, v)-separator in G. Denote by C_u, C_v the connected components of $G - S$ that contain respectively u and v. Then S is minimal if and only if $N_G(C_u) = N_G(C_v) = S$.*

A vertex (or a vertex set) is called *simplicial* if its open neighborhood is a clique.

▶ **Lemma 3.2** (⋆). *Let S be a minimal vertex separator in a graph G. Then S does not contain any simplicial vertices.*

Chordal and interval graphs. An interval graph is an intersection graph of intervals on the real line. In an interval model $\mathcal{I}_G = \{I(v) \mid v \in V(G)\}$ of a graph G, each vertex $v \in V(G)$ corresponds to a closed interval $I(v)$; there is an edge between vertices u and v if and only if $I(v) \cap I(u) \neq \emptyset$.

A *hole* in a graph is an induced (i.e., chordless) cycle of length at least four. A graph is chordal when it does not contain any hole. An equivalent definition states that a chordal graph is an intersection graph of a family of subtrees in a tree [40]. This implies that any interval graph is chordal. For more background on these graph classes see surveys [16, 24].

The characterization of the two classes as intersection graphs of intervals/subtrees leads to the following observation.

▶ **Observation 3.3.** *The classes of chordal and interval graphs are closed under vertex deletions and edge contractions.*

An *asteroidal triple* (AT) is a triple of vertices such that for any two of them there exists a path between them avoiding the closed neighborhood of the third. Interval graphs cannot contain ATs, which is a consequence of a linear ordering of any interval model. It turns out that this is the only property that separates the two graph classes.

▶ **Lemma 3.4** ([24]). *A graph is interval if and only if it is chordal and does not contain an AT.*

We collect two more useful facts about chordal graphs.

▶ **Lemma 3.5** ([24]). *Every non-empty chordal graph contains a simplicial vertex.*

When a chordal graph contains a cycle then it also contains a triangle. As a bipartite graph does not have any triangles, we obtain the following.

▶ **Observation 3.6.** *If a graph is chordal and bipartite, then it is a forest.*

A vertex set S in graph G is called a *chordal deletion set* (resp. *interval deletion set*) if $G - S$ is chordal (resp. interval). The Chordal/Interval Vertex Deletion problem is defined as follows. We are given a graph G, a non-negative weight function $w\colon V(G) \to \mathbb{N}$, an integer p, and we ask whether there exists a chordal (resp. interval) deletion set S in G such that $w(S) \leq p$.

Boundaried graphs. For a set X and a graph B on vertex set X, we define a family $\mathcal{G}_{X,B}$ of graphs G that satisfy (a) $V(G) \supseteq X$, (b) $G[X] = B$. For graphs $G_1, G_2 \in \mathcal{G}_{X,B}$ we define their gluing product $(G_1, X) \oplus (G_2, X)$ by taking a disjoint union of G_1 and G_2 and identifying vertices from X. Note that two vertices from X are adjacent in G_1 if and only if they are adjacent in G_2.

For $X \subseteq V(G)$ a pair (G, X) is called a boundaried graph. We say that two boundaried graphs $(G_1, X), (G_2, X)$ are compatible if $G_1, G_2 \in \mathcal{G}_{X,B}$ for some B. We remark that it is common in the literature to define a boundaried graph as a triple (G, X, λ) where $\lambda\colon X \to [|X|]$ is a labeling (cf. [9, 21]). Since we do not need to perform gluing of abstract boundaried graphs, but only ones originating from subgraphs of a fixed graph, this simpler definition is sufficient.

As an example, consider a graph G and $X \subseteq V(G)$. Then for any $A \subseteq V(G) \setminus X$ the graph $G[A \cup X]$ belongs to $\mathcal{G}_{X,G[X]}$. When $A, B \subseteq V(G) \setminus X$ are disjoint and non-adjacent then $G[A \cup B \cup X]$ is isomorphic to $(G[A \cup X], X) \oplus (G[B \cup X], X)$.

4 Chordal Deletion

We begin with a simple treewidth-preserving reduction from Feedback Vertex Set.

▶ **Lemma 4.1** ($\star$)**.** *Let G be a graph and $\ell \in \mathbb{N}$. Let G' be obtained from G by subdividing each edge. Then $\boldsymbol{tw}(G') = \boldsymbol{tw}(G)$ and G has a feedback vertex set (FVS) of size ℓ if and only if G' has a chordal deletion set of size ℓ.*

As a consequence, the base of the exponent c in Theorem 1.1 must be at least 3 under Strong Exponential Time Hypothesis [35] and c must be at least $2^\omega + 1$ if the current-best deterministic algorithm for Feedback Vertex Set parameterized by treewidth is optimal [69]. While we have no evidence that the mentioned algorithm should be optimal for deterministic time, we provide this comparison to indicate that breaching this gap for ChVD would imply the same for a more heavily studied problem.

Minimal vertex separators. We set the stage for the proof of Theorem 2.1. First we need to develop some theory about minimal vertex separators in chordal graphs.

▶ **Definition 4.2.** *Let $\texttt{MinSep}(G)$ denote the set of minimal vertex separators in a graph G. For a graph G and a (possibly empty) set $S \subseteq V(G)$, we define $\texttt{Comp}(G, S)$ to be the set of connected components C_i of $G - S$ for which it holds that $N_G(C_i) = S$.*

Note that whenever G is disconnected then $\emptyset \in \texttt{MinSep}(G)$ and $\texttt{Comp}(G, \emptyset)$ is just the set of connected components of G. According to Lemma 3.1, the set S is a minimal (u, v)-separator if and only if u, v belong to some (distinct) components from $\texttt{Comp}(G, S)$. For later use, we establish a relation between sets $\texttt{MinSep}(G)$, $\texttt{Comp}(G, S)$ in G and a graph obtained by a removal of a simplicial vertex.

▶ **Lemma 4.3** (⋆). *Let v be a simplicial vertex in G and $S \in \mathit{MinSep}(G)$. If $S \neq N_G(v)$ then $S \in \mathit{MinSep}(G - v)$ and $|\mathit{Comp}(G, S)| = |\mathit{Comp}(G - v, S)|$.*

We need a simple technical lemma about minimal vertex separators.

▶ **Lemma 4.4** (⋆). *Let G be a connected graph and $V_1, \ldots, V_k \subseteq V(G)$, $k \geq 2$, be disjoint sets so that $G[V_i]$ is connected, for $i \in [k]$, and $E_G(V_i, V_j) = \emptyset$, for $i \neq j$. Then there exists a minimal vertex separator $S \subseteq V(G) \setminus (V_1 \cup \cdots \cup V_k)$ in G which is a (V_i, V_j)-separator for some $i \neq j$ and each set V_i is contained in some component $C \in \mathit{Comp}(G, S)$.*

We will use the following concept which appears in the current-best algorithm for ChVD by Jansen et al [45]. In the full version, we also provide several properties of this operation, used to process partial solutions in a treewidth DP.

▶ **Definition 4.5** ([46, Def. 5.55]). *For a graph G and a vertex set $X \subseteq V(G)$ let the graph $\mathit{Condense}(G, X)$ be obtained from G by contracting the connected connected components of $G - X$ into single vertices and then removing those of them which are simplicial.*

In this section we will exploit the following property of condensation.

▶ **Lemma 4.9** (⋆). *Consider a graph G with a vertex set X so that $G[X]$ is chordal. Then G is chordal if and only if the following conditions hold:*
1. *for each connected component C of $G - X$ the graph $G[X \cup C]$ is chordal,*
2. *the graph $\mathit{Condense}(G, X)$ is chordal.*

In order to turn Lemma 4.9 into a more convenient criterion, we will compress information about a graph G with a vertex subset X into multiple auxiliary graphs, one for each minimal vertex separator in $G[X]$.

▶ **Definition 4.10.** *Consider a graph G with a vertex set X so that $G[X]$ is chordal. For a set $S \in \mathit{MinSep}(G[X])$ we construct the graph $\mathit{Aux}(G, X, S)$ as follows:*
1. *contract each $C \in \mathit{Comp}(G[X], S)$ into a vertex and remove the remaining vertices of X (including all of S),*
2. *contract each connected component of $G - X$ into a vertex.*

Note that $\mathtt{Aux}(G, X, \emptyset)$ is obtained by just contracting each connected component of $G[X]$ and each connected component of $G - X$. Moreover, observe that $\mathtt{Aux}(G, X, S)$ is always a bipartite graph because there can be no edges between two components from $\mathtt{Comp}(G[X], S)$ nor between two components of $G - X$. See Figure 1 for an example of this construction.

To make a connection between holes in G and cycles in $\mathtt{Aux}(G, X, S)$, we need a criterion to derive existence of a cycle from a closed walk with certain properties. In the following lemma we consider a cyclic order on a sequence of length k. We define the successor operator as $s(i) = i + 1$, for $i \in [k - 1]$, and $s(k) = 1$.

▶ **Lemma 4.11** (⋆). *Let G be a bipartite graph with vertex partition $V(G) = A \cup B$. Suppose there exists a sequence of vertices $(v_1, \ldots, v_k)$ in G such that:*
1. *for $i \in [k]$ it holds $v_i = v_{s(i)}$ or $v_i v_{s(i)} \in E(G)$,*
2. *the multiset $\{v_1, \ldots, v_k\}$ contains at most one occurrence of each vertex from A,*
3. *the set $\{v_1, \ldots, v_k\}$ contains at least two vertices from B.*

Then G contains a cycle.

We are ready to prove a proposition creating a link between chordality and acyclicity.

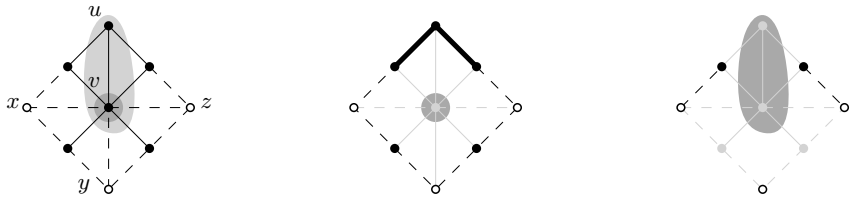


Figure 1 On the left: graph G and set $X \subseteq V(G)$ represented by black disks. The graph $G[X]$ is drawn with solid edges. There are two minimal vertex separators in $G[X]$: $S_1 = \{v\}$ and $S_2 = \{u, v\}$, sketched in gray. In the middle: the graph $\texttt{Aux}(G, X, S_1)$ with thick edges indicating a component that gets contracted into a single vertex; the gray vertices and edges are removed. On the right: the graph $\texttt{Aux}(G, X, S_2)$; note that $|\texttt{Comp}(G[X], S_2)| = 2$ because the lower vertices of X are not adjacent to every vertex in S_2. The graph $\texttt{Aux}(G, X, S_1)$ contains a cycle and this witnesses that G is not chordal. However, removing from G any single vertex among x, y, z results in a chordal graph.

▶ **Proposition 4.12.** *Consider a graph G with a vertex subset $X \subseteq V(G)$ so that for each connected component C of $G - X$ the graph $G[X \cup C]$ is chordal. Then G is chordal if and only if for each $S \in \texttt{MinSep}(G[X])$ the graph $\texttt{Aux}(G, X, S)$ is acyclic.*

Proof. First we argue that if G is chordal then all graphs $\texttt{Aux}(G, X, S)$ are acyclic. Because the class of chordal graphs is closed under vertex deletions and edge contractions, the graphs $\texttt{Aux}(G, X, S)$ are chordal as well. Since each graph $\texttt{Aux}(G, X, S)$ is also bipartite, by Observation 3.6 we obtain that $\texttt{Aux}(G, X, S)$ is acyclic.

Now suppose that G is not chordal. Let $G' = \texttt{Condense}(G, X)$ (recall Definition 4.5). By Lemma 4.9, the graph G' is not chordal as well but for each vertex $v \in V(G') \setminus X$ the graph $G'[X \cup \{v\}]$ is chordal (because contraction preserves chordality). Note that $\texttt{Aux}(G', X, S)$ is an induced subgraph of $\texttt{Aux}(G, X, S)$ for each $S \in \texttt{MinSep}(G[X])$ (they may differ only due to removal of simplicial vertices), so it suffices to show that one of the graphs $\texttt{Aux}(G', X, S)$ has a cycle.

As G' is not chordal, it contains a hole $H = (u_1, \ldots, u_k)$. We consider two cases: either $V(H)$ intersects at least two connected components of $G'[X]$ or only one. In the first case, let $\phi_0 \colon V(G') \to V(\texttt{Aux}(G', X, \emptyset))$ be the mapping given by the contractions from Definition 4.10. Recall that $V(G') \setminus X$ is an independent set in G' so ϕ_0 is an identity on this set. The sequence $(\phi_0(u_1), \ldots, \phi_0(u_k))$ meets the preconditions of Lemma 4.11 for $A = V(G') \setminus X$ and $B = \phi_0(X)$ so $\texttt{Aux}(G', X, \emptyset)$ has a cycle. As $G'[X] = G[X]$ is disconnected, we have $\emptyset \in \texttt{MinSep}(G[X])$.

In the second case, let $Y \subseteq X$ induce the only connected component of $G'[X]$ that intersects $V(H)$. Let $V_1, \ldots, V_\ell \subseteq Y$ be the vertex sets of maximal subpaths of H within Y. By the definition of a hole, we have $E_{G'}(V_i, V_j) = \emptyset$ for distinct $i, j \in [\ell]$. It must be $\ell \geq 2$ because for each $v \in V(G') \setminus X$ the graph $G'[X \cup \{v\}]$ is chordal and the hole H must visit at least two vertices from the independent set $V(G') \setminus X$. By Lemma 4.4, there exists a minimal vertex separator $S \subseteq Y \setminus V(H)$ in $G'[Y]$ such that every set V_i is contained in some component from $\texttt{Comp}(G'[Y], S)$ and at least two components from $\texttt{Comp}(G'[Y], S)$ intersect $V(H)$. Note that $S \in \texttt{MinSep}(G[X])$. Let C_S be the union of the components from $\texttt{Comp}(G'[Y], S)$; note that $V(H) \subseteq V(C_S) \cup (V(G') \setminus X)$.

Let $\phi_S \colon V(C_S) \cup (V(G') \setminus X) \to V(\texttt{Aux}(G', X, S))$ be the mapping given by the contractions from Definition 4.10 which turn each component from $\texttt{Comp}(G'[Y], S)$ into a single vertex. Again, the sequence $(\phi_S(u_1), \ldots, \phi_S(u_k))$ meets the preconditions of Lemma 4.11 for $A = V(G') \setminus X$ and $B = \phi_S(V(C_S))$ so $\texttt{Aux}(G', X, S)$ has a cycle. See Figure 1 for an illustration. ◀

Signatures of boundaried graphs. The next step is to construct a graphic matroid M_B for a chordal graph B so that for any two graphs $G_1, G_2 \in \mathcal{G}_{X,B}$ the information about chordality of $(G_1, X) \oplus (G_2, X)$ could be read from M_B. Proposition 4.12 already relates chordality to acyclicity but the corresponding graphic matroids for G_1, G_2 are disparate. To circumvent this, we will further compress the information about cycles.

▶ **Definition 4.13.** *Consider a graph B. For $S \in \mathit{MinSep}(B)$, let $\mathtt{Base}(B,S)$ be the complete graph on vertex set $\mathtt{Comp}(B,S)$. The graph $\mathtt{Base}(B)$ is a disjoint union of all the graphs $\mathtt{Base}(B,S)$ for $S \in \mathit{MinSep}(B)$.*

That is, we treat the components from $\mathtt{Comp}(B,S)$ as abstract vertices of a new graph which is a union of cliques.

The following transformation is similar to the one used in the algorithm for Steiner Tree based on representative families [38]. For the sake of disambiguation, in the definition below we assume an implicit linear order on the vertices of B; this order may be arbitrary. Since vertices of $\mathtt{Base}(B)$ correspond to distinct subsets of $V(B)$, which can ordered lexicographically, fixing the order on $V(B)$ yields an order on $V(\mathtt{Base}(B))$. We can thus assume that also the vertices of $V(\mathtt{Base}(B))$ are linearly ordered.

▶ **Definition 4.14.** *Consider a chordal graph B and $Y \subseteq V(B)$. We define the* spanning signature *$\mathit{Span}(B,Y) \subseteq E(\mathtt{Base}(B))$ as follows. For each $S \in \mathit{MinSep}(B)$ let $C_{S,Y} \subseteq V(\mathtt{Base}(B,S))$ be given by components from $\mathtt{Comp}(B,S)$ with a non-empty intersection with Y. Let $P_{S,Y} \subseteq E(\mathtt{Base}(B,S))$ be the path connecting the vertices of $C_{S,Y}$ in the increasing order. Then $\mathit{Span}(B,Y) = \bigcup_{S \in \mathit{MinSep}(B)} P_{S,Y}$.*

In other words, $\mathtt{Span}(B,Y)$ is a disjoint union of paths in the graph $\mathtt{Base}(B)$, where each path encodes the relation between Y and a respective minimal vertex separator in B.

The next lemma states that under certain conditions replacing a vertex v with a tree over $N(v)$ (in particular: a path) does not affect acyclicity of the graph. Note that due to the precondition $|N(u) \cap N(v)| \leq 1$ we never attempt to insert an edge that is already present.

▶ **Lemma 4.15 ($\star$).** *Let G be a bipartite graph with a vertex partition $V(G) = A \cup B$ so that for each distinct $u, v \in A$ it holds that $|N_G(u) \cap N_G(v)| \leq 1$. Consider a graph G' obtained from G by replacing each vertex $v \in A$ by an arbitrary tree on vertex set $N_G(v)$. Then G is acyclic if and only if G' is acyclic.*

This allows us to translate the criterion from Proposition 4.12 into a more convenient one, in which the vertex set of the auxiliary graph depends only on $G[X]$ rather than G.

▶ **Lemma 4.16.** *Consider a graph G with a vertex subset $X \subseteq V(G)$. Let $\mathcal{C}$ denote the family of connected components of $G - X$. Suppose that for each $C \in \mathcal{C}$ the graph $G[X \cup C]$ is chordal. Then G is chordal if and only if:*

1. *the sets $\mathit{Span}(G[X], N_G(C))$, for different $C \in \mathcal{C}$, are pairwise disjoint,*
2. *the union of sets $\mathit{Span}(G[X], N_G(C))$, over $C \in \mathcal{C}$, forms an acyclic edge set in $E(\mathtt{Base}(G[X]))$.*

Proof. From Proposition 4.12 we know that G is chordal if and only if for each $S \in \mathtt{MinSep}(G[X])$ the graph $\mathtt{Aux}(G,X,S)$ is acyclic. We consider two cases.

First, suppose that for some $S \in \mathtt{MinSep}(G[X])$ there are two vertices representing distinct components $C_1, C_2 \in \mathcal{C}$ that share two common neighbors x, y in $\mathtt{Aux}(G,X,S)$. In other words, there are two components from $\mathtt{Comp}(G[X],S)$ that intersect both $N_G(C_1)$ and $N_G(C_2)$. Then $\mathtt{Aux}(G,X,S)$ contains a cycle of length 4, so G is not chordal. If $\mathtt{Span}(G[X], N_G(C_1))$

and $\texttt{Span}(G[X], N_G(C_2))$ share an edge, then condition (1) fails, so suppose this is not the case. But then the paths $P_{S,N(C_1)}$ and $P_{S,N(C_2)}$ (recall Definition 4.14) are edge-disjoint and they both visit x and y. As a consequence, x, y lie on a cycle contained in the edge set $\texttt{Span}(G[X], N_G(C_1)) \cup \texttt{Span}(G[X], N_G(C_2))$ so condition (2) fails. In summary, both G is not chordal and one of conditions (1, 2) does not hold.

Next, suppose that for each $S \in \texttt{MinSep}(G[X])$ and any two vertices representing distinct components $C_1, C_2 \in \mathcal{C}$ the intersection of their neighborhoods in $\texttt{Aux}(G, X, S)$ contains at most one element. This implies condition (1). Consider a graph H given by a disjoint union of all graphs $\texttt{Aux}(G, X, S)$ over $S \in \texttt{MinSep}(G[X])$. This graph meets the preconditions of Lemma 4.15. Replacing each $\mathcal{C}$-component-vertex in $\texttt{Aux}(G, X, S)$ by the path $P_{S,N(C)}$ transforms H into a subgraph of $\texttt{Base}(G[X])$ with the edge set $\bigcup_{C \in \mathcal{C}} \texttt{Span}(G[X], N_G(C))$. By Lemma 4.15, this graph is acyclic if and only if the graph H is. By Proposition 4.12, this condition is equivalent to G being chordal. The lemma follows. ◀

We are ready to define the graphic matroid encoding all the necessary information about where a hole can appear after gluing two chordal graphs. Recall that a graphic matroid of a graph G is a set system over $E(G)$ where a subset $S \subseteq E(G)$ is called independent when S contains no cycles. More information about matroids can be found in the preliminaries of the full version of the article.

▶ **Definition 4.17.** *For a graph B on vertex set X we define matroid M_B as the graphic matroid of the graph $\texttt{Base}(B)$. For a graph $G \in \mathcal{G}_{X,B}$ the signature $\texttt{Sign}(G, X) \subseteq E(\texttt{Base}(B))$ is defined as a union of $\texttt{Span}(B, N_G(C))$ over all connected components C of $G - X$.*

It follows from Lemma 4.16 that whenever G is chordal then $\texttt{Sign}(G, X)$ is acyclic and so it forms an independent set in the matroid $M_{G[X]}$. We can now give the existential part of Theorem 2.1. The mapping $\sigma \colon \mathcal{G}_{X,B} \to 2^{E(M_B)}$ therein is given here as $\sigma(G) = \texttt{Sign}(G, X)$.

▶ **Lemma 4.18** ($\star$)**.** *Let (G_1, X) and (G_2, X) be compatible boundaried chordal graphs. Then $G = (G_1, X) \oplus (G_2, X)$ is chordal if and only if the sets $\texttt{Sign}(G_1, X)$, $\texttt{Sign}(G_2, X) \subseteq E(\texttt{Base}(G[X]))$ are disjoint and $\texttt{Sign}(G_1, X) \cup \texttt{Sign}(G_2, X)$ is acyclic.*

Furthermore, $\texttt{Sign}(G, X) = \texttt{Sign}(G_1, X) \cup \texttt{Sign}(G_2, X)$.

The following lemma is the main ingredient in the running time analysis. As the bound on the representative family's size is exponential in the rank of a matroid[2], it is necessary to bound the rank of M_B. It is known that the number of minimal vertex separators in a chordal graph is bounded by the number of vertices but we need a strengthening of this fact.

▶ **Lemma 4.19.** *For a non-empty chordal graph B, the rank of M_B is at most $|V(B)| - 1$.*

Proof. Let $k = |V(B)|$. The rank of M_B equals the size of a spanning forest in $\texttt{Base}(B)$. The vertex sets of connected components of $\texttt{Base}(B)$ are the sets $\texttt{Comp}(B, S)$ for $S \in \texttt{MinSep}(B)$. Therefore it suffices to estimate

$$\sum_{S \in \texttt{MinSep}(B)} (|\texttt{Comp}(B, S)| - 1) \leq k - 1.$$

We first prove the inequality for connected chordal graphs by induction on k. For $k = 1$ the sum is zero. Consider $k > 1$. By Lemma 3.5, B contains a simplicial vertex. Let v be a simplicial vertex in B and suppose that the claim holds for the graph $B - v$ (which is

[2] We remark that Fomin et al. [38] also considered a case when the rank might be large and the exponential term is governed by a different parameter but it is not applicable in our case.

connected). Let S be a minimal vertex separator in B. By Lemma 4.3 when $S \neq N_B(v)$ then $S \in \texttt{MinSep}(B - v)$ and $|\texttt{Comp}(B, S)| = |\texttt{Comp}(B - v, S)|$. In that case the summand coming from S is the same for B and $B - v$.

It remains to handle the case $S = N_B(v)$. Clearly, $\{v\} \in \texttt{Comp}(B, S)$. If $|\texttt{Comp}(B, S)| = 1$ then $S \notin \texttt{MinSep}(B)$ (Lemma 3.1). If $|\texttt{Comp}(B, S)| = 2$ then $S \in \texttt{MinSep}(B) \setminus \texttt{MinSep}(B - v)$ and the sum grows by one. If $|\texttt{Comp}(B, S)| \geq 3$ then $S \in \texttt{MinSep}(B) \cap \texttt{MinSep}(B - v)$ and $|\texttt{Comp}(B, S)| = |\texttt{Comp}(B - v, S)| + 1$ so the sum again grows by one. This concludes the proof of the inequality for connected chordal graphs.

When B is disconnected, let $B_1, B_2, \ldots, B_t$ denote its connected components and let $k_i = |V(B_i)|$. We have $|\texttt{Comp}(B, \emptyset)| - 1 = t - 1$. Together with the sums for $B_1, B_2, \ldots, B_t$ the total sum is at most $\sum_{i=1}^{t} k_i - t + t - 1 = k - 1$. ◀

The last thing to be checked is whether we can compute the signatures efficiently. To this end, we enumerate minimal vertex separators using Lemma 4.3.

▶ **Lemma 4.20** (⋆). *There is a polynomial-time algorithm that, given a graph G with a vertex subset $X \subseteq V(G)$ such that $G[X]$ is chordal, computes $\texttt{Sign}(G, X)$.*

Lemmas 4.18, 4.19, and 4.20 entail Theorem 2.1 but instead of working with that abstract statement we use these three lemmas directly when describing the final algorithm. The results of this section allow us to employ the framework of representative families in order to truncate the number of partial solutions stored at a node of a tree decomposition to $2^{\mathcal{O}(\mathbf{tw})}$. The dynamic programming algorithm follows the lines of proofs in [37] and is described in detail in the full version. The main technical hurdle comes from the necessity to store only the condensed counterparts of the partial solutions. The condensed graphs have only $\mathcal{O}(\mathbf{tw})$ vertices each, what is the key to obtain a linear dependency on $|V(G)|$.

5 Interval Deletion

We switch our attention to Interval Vertex Deletion and show that in this case it is unlikely to achieve any speed-up over the existing $2^{\mathcal{O}(\mathbf{tw} \log \mathbf{tw})} \cdot n$-time algorithm. We prove Theorem 1.2 via a parameterized reduction from $k \times k$ Permutation Clique, which is defined as follows.

$k \times k$ Permutation Clique
Input: Graph G over the vertex set $[k] \times [k]$.
Question: Is there a permutation $\pi\colon [k] \to [k]$ so that $(1, \pi(1)), (2, \pi(2)), \ldots, (k, \pi(k))$ forms a clique in G?

Permutation gadget. We will encode a permutation $\pi\colon [k] \to [k]$ as a family of sets $N_1, N_2, \ldots, N_k$ so that $N_i = \pi([i])$ (i.e., N_i is the set of i numbers appearing first in π). First, we need a gadget to verify that such a family represents some permutation.

▶ **Definition 5.1.** *For an integer k, let Y_k be a graph on a vertex set $\{y_1, y_2, \ldots, y_{k+2}\}$ so that $\{y_1, y_2, \ldots, y_{k+1}\}$ induces a clique and y_{k+2} is adjacent only to y_{k+1}.*

We shall enforce a linear order on $N_1, \ldots, N_k$ by demanding that a particular supergraph of Y_k is interval. The corresponding interval model is depicted on Figure 2.

▶ **Lemma 5.3** (⋆). *Let $N_1, \ldots, N_\ell \subseteq [k]$. Consider a graph G obtained from Y_k by inserting an independent set of vertices $x_1, \ldots, x_\ell$ so that $N_G(x_i) = \{y_j \mid j \in N_i\}$. Then G is interval if and only if there exists a permutation $\pi\colon [k] \to [k]$ so that for each $i \in \ell$ it holds that $N_i = \pi([n_i])$ where $n_i = |N_i|$.*

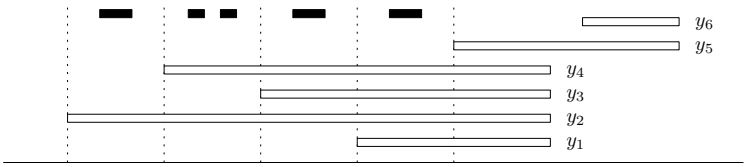


Figure 2 Illustration for Lemma 5.3. The intervals for vertices of Y_4 are blank, ordered from bottom to top. They encode permutation $(2, 4, 3, 1)$. The black intervals represent vertices x_1, x_2, x_3, x_4, x_5 with neighborhoods encoding sets $\{2\}$, $\{2, 4\}$ (twice), $\{2, 4, 3\}$, and $\{2, 4, 3, 1\}$.

Choice gadget. We need to verify that $(i, \pi(i))(j, \pi(j)) \in E(G)$ for each $1 \leq i < j \leq k$. As $\pi(i)$ is the only element in $N_i \setminus N_{i-1}$, the information whether $(i, \pi(i)), (j, \pi(j)) \in E(G)$ can be extracted from the tuple $(N_{i-1}, N_i, N_{j-1}, N_j)$. We construct a gadget that enforces a solution to select one such valid tuple.

We use a following convention to describe the gadgets. When P is a graph with a distinguished vertex named v and a graph H is constructed using explicit vertex-disjoint copies of the graph P, referred to as $P_1, P_2, \ldots, P_\ell$, we refer to the copy of v within the subgraph P_i as $P_i[v]$. We construct the choice gadget as a path-like structure consisting of blocks, each equipped with four special vertices. These are the only vertices that later get connected to the permutation gadget. On the intuitive level, a solution should choose one block, leave its special vertices untouched, and remove the remaining special vertices. See Figure 3 for an illustration.

▶ **Definition 5.4.** *The graph P is obtained from a path $(u_1, u_2, \ldots, u_9)$ by appending to u_2 two subdivided edges, one subdivided edge to u_7, and inserting edge u_4u_8.*

The choice gadget of order s is a graph constructed as follows. We begin with a vertex set $\bigcup_{i=1}^{s}\{v_i^1, v_i^2, v_i^3\} \cup \{v_{left}, v_{right}\}$. For each pair (x, y) of the form $(v_i^1, v_i^2), (v_i^2, v_i^3), (v_i^3, v_i^1), (v_i^3, v_{i+1}^1)$ as well as for (v_{left}, v_1^1), (v_s^3, v_{right}) we create two subdivided edges between x and y. We refer to the subgraph given by the two subdivided edges between x, y as $\langle x, y\rangle$. We refer to the union of $\langle v_i^1, v_i^2\rangle, \langle v_i^2, v_i^3\rangle, \langle v_i^3, v_i^1\rangle$ as Q_i.

Next, for each $i \in [s]$ we create four copies of the graph P, denoted $P_i^1, P_i^2, P_i^3, P_i^4$. We insert edges between v_i^2 and $P_i^1[u_1], P_i^2[u_1], P_i^3[u_1], P_i^4[u_1]$. We refer to vertices $P_i^\alpha[u_8]$, $P_i^\alpha[u_9]$, $\alpha \in [4]$, as respectively h_i^α, g_i^α.

The choice gadget is designed to enforce a special structure of minimum-size interval deletion sets. We exploit the fact that P contains two vertex-disjoint subgraphs with asteroidal triples (see Figure 3) so any interval deletion set in a choice gadget must contain at least two vertices from each copy of P.

We prove several properties of the choice gadget which are analogous to the properties of the gadget used by Pilipczuk in the lower bound for Planar Vertex Deletion [58]. However, in that construction every block has only one special vertex with edges leaving the gadget, while in our case there are four special vertices. We also need to ensure that when the special vertices in some block are not being removed then a solution can remove their neighbors in the gadget. (Inserting a planar graph attached to a single vertex of G does not affect planarity of G but the analogous property does not hold for the class of interval graphs.) The special structure of the graph P allows us to resolve these two issues.

▶ **Lemma 5.6** ($\star$). *Let H_s be the choice gadget of order s.*

1. *The minimal size of an interval deletion set in H_s is $10s$.*
2. *For every $i \in [s]$ there exists a minimum-size interval deletion set X in H_s such that $\{h_i^1, h_i^2, h_i^3, h_i^4\} \subseteq X$ and $\{g_j^1, g_j^2, g_j^3, g_j^4\} \subseteq X$ for each $j \neq i$.*

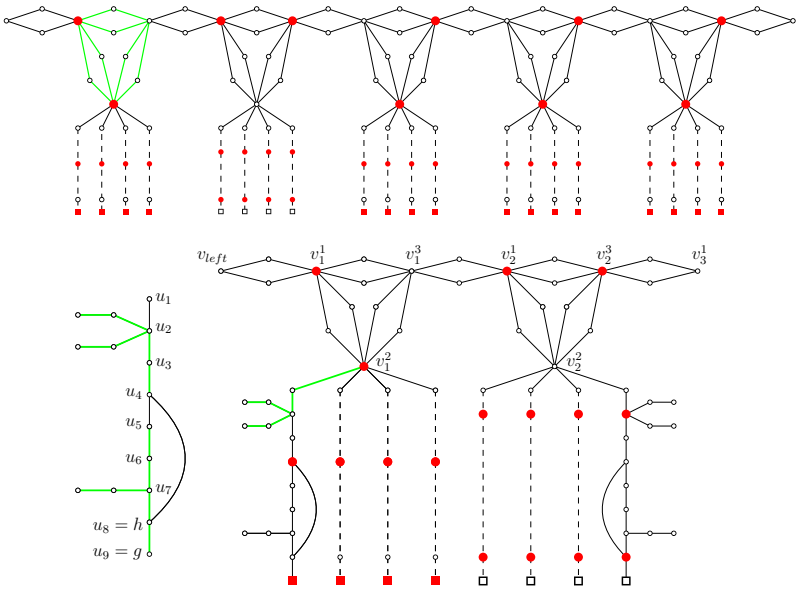


Figure 3 Top: the choice gadget H_5 with the subgraph Q_1 highlighted in green. The copies of P are sketched symbolically with dashed lines and the squares represent vertices g_i^α. The red disks and squares represent a solution constructed in Lemma 5.6(2). This solution "chooses" $i = 2$, leaves untouched the four vertices g_2^α, and removes h_2^α as well as g_i^α for $i \neq 2$. Bottom left: the graph P and vertices named h, g. Two vertex-disjoint non-interval subgraphs of P have green edges. Bottom right: a closer look at the first two blocks of H_5 with two copies of P drawn in detail. The subgraph highlighted in green witnesses that if a minimum-size solution removes g_i^α for at least one $\alpha \in [4]$ then it must also remove v_i^2, what is exploited in Lemma 5.6(3).

3. *For every minimum-size interval deletion set X in H_s there is $i \in [s]$ such that $\{g_i^1, g_i^2, g_i^3, g_i^4\} \cap X = \emptyset$.*
4. *If $s \leq 2^k$ then $\boldsymbol{td}(H_s) \leq \boldsymbol{td}(H_1) + k$, where $\boldsymbol{td}(G)$ stands for the treedepth of G.*

Lokshtanov et al. [54] proved that $k \times k$ Permutation Clique cannot be solved in time $2^{o(k \log k)}$ assuming ETH. According to the reduction below, this also rules out running time of the form $2^{o(\mathbf{td} \log \mathbf{td})} \cdot n^{\mathcal{O}(1)}$ for Interval Vertex Deletion, where $\mathbf{td}$ is the treedepth of the input graph. As $\mathbf{tw}(G) \leq \mathbf{td}(G)$, this entails the same hardness for treewidth, what proves Theorem 1.2.

▶ **Proposition 5.7.** *There is an algorithm that, given an instance (G, k) of $k \times k$ Permutation Clique, runs in time $2^{\mathcal{O}(k)}$ and returns an equivalent unweighted instance (H, p) of Interval Vertex Deletion such that $|V(H)| = 2^{\mathcal{O}(k)}$ and $\boldsymbol{td}(H) = \mathcal{O}(k)$.*

Proof. For $1 \leq i < j \leq k$ and $x \neq y \in [k]$ let $\mathcal{S}_{i,x,j,y}$ be the family of tuples (S_1, S_2, S_3, S_4) of subsets of $[k]$ satisfying:

- $S_1 \subset S_2 \subseteq S_3 \subset S_4$,
- $|S_1| = i - 1$,
- $S_2 \setminus S_1 = \{x\}$,
- $|S_3| = j - 1$,
- $S_4 \setminus S_3 = \{y\}$.

Furthermore, for $1 \leq i < j \leq k$, let $\mathcal{S}_{i,j}$ be the union of $\mathcal{S}_{i,x,j,y}$ over all pairs $x \neq y \in [k]$ such that $(i, x)(j, y) \in E(G)$. Let $s_{i,j} = |\mathcal{S}_{i,j}|$ and $\rho_{i,j} \colon [s_{i,j}] \to \mathcal{S}_{i,j}$ be an arbitrary bijection. Clearly $s_{i,j} \leq 4^k k^2$.

The graph H consists of a permutation gadget Y_k and, for each $1 \leq i < j \leq k$, a choice gadget $C_{i,j}$ of order $s_{i,j}$. For $S \subseteq [k]$ we use shorthand $Y_k[S] = \{y_i \mid i \in S\}$. For $\ell \in [s_{i,j}]$ and $(S_1, S_2, S_3, S_4) = \rho_{i,j}(\ell)$ the vertices $C_{i,j}[g^1_\ell]$, $C_{i,j}[g^2_\ell]$, $C_{i,j}[g^3_\ell]$, $C_{i,j}[g^4_\ell]$ get connected to vertex sets $Y_k[S_1], Y_k[S_2], Y_k[S_3], Y_k[S_4]$, respectively. This finishes the construction of H. The number of vertices in H is clearly $2^{\mathcal{O}(k)}$ and the construction can be performed in time polynomial in the size of H. We set $p = 10 \cdot \sum_{1 \leq i < j \leq k} s_{i,j}$.

▷ Claim 5.8. If (G, k) admits a solution, then H has an interval deletion set of size p.

Proof. Let $\pi\colon [k] \to [k]$ be a permutation encoding a clique in G. By the construction, for each $1 \leq i < j \leq k$ we have $(\pi([i-1]), \pi([i]), \pi([j-1]), \pi([j]) \in \mathcal{S}_{i,j}$. Let $\ell \in [s_{i,j}]$ be the index mapped to this tuple by $\rho_{i,j}$. By Lemma 5.6(2) the choice gadget $C_{i,j}$ has an interval deletion set $X_{i,j} \subseteq V(C_{i,j})$ of size $10s_{i,j}$ such that $\{C_{i,j}[h^1_\ell], C_{i,j}[h^2_\ell], C_{i,j}[h^3_\ell], C_{i,j}[h^4_\ell]\} \subseteq X_{i,j}$ and $\{C_{i,j}[g^1_r], C_{i,j}[g^2_r], C_{i,j}[g^3_r], C_{i,j}[g^4_r]\} \subseteq X_{i,j}$ for each $r \neq \ell$. In other words, $X_{i,j}$ contains all vertices in $C_{i,j}$ which are adjacent to Y_k except for the $C_{i,j}$-copies of $g^1_\ell, g^2_\ell, g^3_\ell, g^4_\ell$ and $X_{i,j}$ also contains the neighbors of $C_{i,j}[g^1_\ell], C_{i,j}[g^2_\ell], C_{i,j}[g^3_\ell], C_{i,j}[g^4_\ell]$ in $C_{i,j}$.

We set $X = \bigcup_{1 \leq i < j \leq k} X_{i,j}$. Then the only connected component of $H - X$ which is not a connected component of any $C_{i,j} - X_{i,j}$ is given by Y_k together with an independent set of the vertices described above. The neighborhood of each such vertex in Y_k is of the form $Y_k[\pi([k'])]$ for some $0 \leq k' \leq k$. By Lemma 5.3 this component is an interval graph. This shows that X is indeed an interval deletion set. ◁

▷ Claim 5.9. If H has an interval deletion set of size at most p, then (G, k) admits a solution.

Proof. Let X be an interval deletion set in H. By Lemma 5.6(1) a minimum-size interval deletion set in $C_{i,j}$ has size $10s_{i,j}$. As the choice gadgets are vertex-disjoint subgraphs of H, the set X must contain exactly $10s_{i,j}$ vertices from $V(C_{i,j})$. This also implies that $V(Y_k) \cap X = \emptyset$.

Let $X_{i,j} = V(C_{i,j}) \cap X$. By Lemma 5.6(3) there exists $\ell \in [s_{i,j}]$ such that $\{C_{i,j}[g^1_\ell], C_{i,j}[g^2_\ell], C_{i,j}[g^3_\ell], C_{i,j}[g^4_\ell]\} \cap X_{i,j} = \emptyset$. Therefore for each pair (i,j) there is a tuple $(S^1_{i,j}, S^2_{i,j}, S^3_{i,j}, S^4_{i,j}) \in \mathcal{S}_{i,j}$ so that vertices from $C_{i,j}$ with neighborhoods $Y_k[S^1_{i,j}], Y_k[S^2_{i,j}], Y_k[S^3_{i,j}], Y_k[S^4_{i,j}]$ are present in $H - X$. By Lemma 5.3 there exists a single permutation $\pi\colon [k] \to [k]$ so that each set $S^\alpha_{i,j}$ is of the form $\pi([|S^\alpha_{i,j}|])$. By the definition of family $\mathcal{S}_{i,j}$ this implies that $(i, \pi(i))(j, \pi(j)) \in E(G)$ for each pair (i,j). Hence there is a k-clique in G. ◁

▷ Claim 5.10. The treedepth of H is $\mathcal{O}(k)$.

Proof. The treedepth of H is at most $|Y_k| = k + 2$ plus $\mathbf{td}(H - Y_k)$, which equals the maximum of $\mathbf{td}(C_{i,j})$ over all employed choice gadgets $C_{i,j}$. As $s_{i,j} \leq 4^k k^2$, Lemma 5.6(4) implies that $\mathbf{td}(C_{i,j}) \leq 2k + 2\log_2 k + \mathcal{O}(1)$. ◁

This concludes the proof of the proposition. ◀

6 Conclusion and open problems

We have obtained ETH-tight bounds for vertex-deletion problems into the classes of chordal and interval graphs, under the treewidth parameterization. The status of the corresponding edge-deletion problems remains unclear (see [64]). The related problem, Feedback Vertex Set, can be solved using representative families within the same running time as our algorithm for ChVD [37]. However, it admits a faster deterministic algorithm based on the determinant approach [69] and an even faster randomized algorithm based on the Cut & Count technique [35]. Could ChVD also be amenable to one of those techniques?

Our algorithm for ChVD is based on a novel connection between chordal graphs and graphic matroids, which might come in useful in other settings. In particular, we ask whether this insight can be leveraged to improve the running time for ChVD parameterized by the solution size k, where the current-best algorithm runs in time $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$ [29]. A direct avenue for a potential improvement would be to reduce the problem in time $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ to the case with treewidth $\mathcal{O}(k)$ and then apply Theorem 1.1. Such a strategy has been employed in the state-of-the-art algorithm for Planar Vertex Deletion parameterized by the solution size [47].

References

1 Akanksha Agrawal, Sudeshna Kolay, Daniel Lokshtanov, and Saket Saurabh. A faster FPT algorithm and a smaller kernel for block graph vertex deletion. In Evangelos Kranakis, Gonzalo Navarro, and Edgar Chávez, editors, *LATIN 2016: Theoretical Informatics – 12th Latin American Symposium, Ensenada, Mexico, April 11-15, 2016, Proceedings*, volume 9644 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2016. doi:10.1007/978-3-662-49529-2_1.

2 Akanksha Agrawal, Daniel Lokshtanov, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. Erdös-Pósa property of obstructions to interval graphs. In Rolf Niedermeier and Brigitte Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, volume 96 of *LIPIcs*, pages 7:1–7:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.STACS.2018.7.

3 Akanksha Agrawal, Daniel Lokshtanov, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. Feedback vertex set inspired kernel for chordal vertex deletion. *ACM Trans. Algorithms*, 15(1):11:1–11:28, 2019. doi:10.1145/3284356.

4 Akanksha Agrawal, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. Interval vertex deletion admits a polynomial kernel. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 1711–1730, USA, 2019. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611975482.103.

5 Jungho Ahn, Eduard Eiben, O joung Kwon, and Sang il Oum. A Polynomial Kernel for 3-Leaf Power Deletion. In Javier Esparza and Daniel Kráľ, editors, *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*, volume 170 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:14, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2020.5.

6 Jungho Ahn, Eun Jung Kim, and Euiwoong Lee. Towards constant-factor approximation for chordal/distance-hereditary vertex deletion. *Algorithmica*, 84(7):2106–2133, 2022. doi:10.1007/s00453-022-00963-7.

7 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '21, pages 522–539, USA, 2021. Society for Industrial and Applied Mathematics. doi:10.5555/3458064.3458096.

8 Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph (Seffi) Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. *J. ACM*, 48(5):1069–1090, September 2001. doi:10.1145/502102.502107.

9 Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos. A complexity dichotomy for hitting connected minors on bounded treewidth graphs: the chair and the banner draw the boundary. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 951–970. SIAM, 2020. doi:10.1137/1.9781611975994.57.

10 Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos. Hitting minors on bounded treewidth graphs. I. General upper bounds. *SIAM J. Discret. Math.*, 34(3):1623–1648, 2020. doi:10.1137/19M1287146.

11 Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos. Hitting minors on bounded treewidth graphs. II. Single-exponential algorithms. *Theor. Comput. Sci.*, 814:135–152, 2020. doi:10.1016/j.tcs.2020.01.026.

12 Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos. Hitting minors on bounded treewidth graphs. III. Lower bounds. *J. Comput. Syst. Sci.*, 109:56–77, 2020. doi:10.1016/j.jcss.2019.11.002.

13 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTR, USA, 1st edition, 1998. URL: https://dl.acm.org/doi/10.5555/551884.

14 Seymour Benzer. On the topology of the genetic fine structure. *Proceedings of the National Academy of Sciences of the United States of America*, 45(11):1607–1620, 1959. URL: http://www.jstor.org/stable/90127.

15 Benjamin Bergougnoux, Édouard Bonnet, Nick Brettell, and O joung Kwon. Close Relatives of Feedback Vertex Set Without Single-Exponential Algorithms Parameterized by Treewidth. In Yixin Cao and Marcin Pilipczuk, editors, *15th International Symposium on Parameterized and Exact Computation (IPEC 2020)*, volume 180 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:17, 2020. doi:10.4230/LIPIcs.IPEC.2020.3.

16 Jean R. S. Blair and Barry Peyton. An introduction to chordal graphs and clique trees. In Alan George, John R. Gilbert, and Joseph W. H. Liu, editors, *Graph Theory and Sparse Matrix Computation*, pages 1–29, New York, NY, 1993. Springer New York.

17 Ivan Bliznets, Marek Cygan, Pawel Komosa, Michal Pilipczuk, and Lukás Mach. Lower bounds for the parameterized complexity of minimum fill-in and other completion problems. *ACM Trans. Algorithms*, 16(2):25:1–25:31, 2020. doi:10.1145/3381426.

18 Ivan Bliznets, Fedor V Fomin, Michał Pilipczuk, and Yngve Villanger. Largest chordal and interval subgraphs faster than 2^n. *Algorithmica*, 76(2):569–594, 2016.

19 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.

20 Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michał Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM Journal on Computing*, 45(2):317–378, 2016. doi:10.1137/130947374.

21 Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (Meta) Kernelization. *J. ACM*, 63(5):44:1–44:69, 2016. doi:10.1145/2973749.

22 Marthe Bonamy, Lukasz Kowalik, Jesper Nederlof, Michal Pilipczuk, Arkadiusz Socala, and Marcin Wrochna. On directed feedback vertex set parameterized by treewidth. In Andreas Brandstädt, Ekkehard Köhler, and Klaus Meer, editors, *Graph-Theoretic Concepts in Computer Science – 44th International Workshop, WG 2018, Cottbus, Germany, June 27-29, 2018, Proceedings*, volume 11159 of *Lecture Notes in Computer Science*, pages 65–78. Springer, 2018. doi:10.1007/978-3-030-00256-5_6.

23 Édouard Bonnet, Nick Brettell, O joung Kwon, and Dániel Marx. Generalized Feedback Vertex Set Problems on Bounded-Treewidth Graphs: Chordality Is the Key to Single-Exponential Parameterized Algorithms. In Daniel Lokshtanov and Naomi Nishimura, editors, *12th International Symposium on Parameterized and Exact Computation (IPEC 2017)*, volume 89 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:13, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.IPEC.2017.7.

24 Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph classes: a survey*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999. doi:10.1137/1.9780898719796.

25 Peter Buneman. A characterisation of rigid circuit graphs. *Discrete Math.*, 9(3):205–212, September 1974. doi:10.1016/0012-365X(74)90002-8.

26 Leizhen Cai. Parameterized complexity of vertex colouring. *Discrete Applied Mathematics*, 127(3):415–429, 2003. doi:10.1016/S0166-218X(02)00242-1.

27 Yixin Cao. Linear recognition of almost interval graphs. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1096–1115. SIAM, 2016. doi:10.1137/1.9781611974331.ch77.

28 Yixin Cao and Dániel Marx. Interval deletion is fixed-parameter tractable. *ACM Trans. Algorithms*, 11(3), January 2015. doi:10.1145/2629595.

29 Yixin Cao and Dániel Marx. Chordal editing is fixed-parameter tractable. *Algorithmica*, 75(1):118–137, May 2016. doi:10.1007/s00453-015-0014-x.

30 Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic – A Language-Theoretic Approach*, volume 138 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2012. doi:10.1017/CBO9780511977619.

31 Radu Curticapea, Nathan Lindzey, and Jesper Nederlof. A tight lower bound for counting hamiltonian cycles via matrix rank. In *Proceedings of the 2018 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1080–1099, 2018. doi:10.1137/1.9781611975031.70.

32 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

33 Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast hamiltonicity checking via bases of perfect matchings. *J. ACM*, 65(3):12:1–12:46, 2018. doi:10.1145/3148227.

34 Marek Cygan, Dániel Marx, Marcin Pilipczuk, and Michał Pilipczuk. Hitting forbidden subgraphs in graphs of bounded treewidth. *Information and Computation*, 256:62–82, 2017. doi:10.1016/j.ic.2017.04.009.

35 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan M. M. Van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. *ACM Trans. Algorithms*, 18(2), March 2022. doi:10.1145/3506707.

36 Erik D. Demaine and MohammadTaghi Hajiaghayi. The bidimensionality theory and its algorithmic applications. *The Computer Journal*, 51(3):292–302, 2008. doi:10.1093/comjnl/bxm033.

37 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Representative sets of product families. In Andreas S. Schulz and Dorothea Wagner, editors, *Algorithms – ESA 2014*, pages 443–454, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. doi:10.1007/978-3-662-44777-2_37.

38 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016. doi:10.1145/2886094.

39 Fedor V. Fomin and Yngve Villanger. Subexponential parameterized algorithm for minimum fill-in. *SIAM Journal on Computing*, 42(6):2197–2216, 2013. doi:10.1137/11085390X.

40 Fănică Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974. doi:10.1016/0095-8956(74)90094-X.

41 Pinar Heggernes, Pim van 't Hof, Bart M.P. Jansen, Stefan Kratsch, and Yngve Villanger. Parameterized complexity of vertex deletion into perfect graph classes. *Theoretical Computer Science*, 511:172–180, 2013. Exact and Parameterized Computation. doi:10.1016/j.tcs.2012.03.013.

42 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

43 Ashwin Jacob, Fahad Panolan, Venkatesh Raman, and Vibha Sahlot. Structural parameterizations with modulator oblivion. In Yixin Cao and Marcin Pilipczuk, editors, *15th International Symposium on Parameterized and Exact Computation, IPEC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 180 of *LIPIcs*, pages 19:1–19:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.IPEC.2020.19.

44 Hugo Jacob, Thomas Bellitto, Oscar Defrain, and Marcin Pilipczuk. Close Relatives (Of Feedback Vertex Set), Revisited. In Petr A. Golovach and Meirav Zehavi, editors, *16th International Symposium on Parameterized and Exact Computation (IPEC 2021)*, volume 214 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:15, Dagstuhl, Germany, 2021. doi:10.4230/LIPIcs.IPEC.2021.21.

45 Bart M. P. Jansen, Jari J. H. de Kroon, and Michał Włodarczyk. Vertex deletion parameterized by elimination distance and even less. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 1757–1769, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451068.

46 Bart M. P. Jansen, Jari J. H. de Kroon, and Michal Wlodarczyk. Vertex deletion parameterized by elimination distance and even less. *CoRR*, abs/2103.09715, 2021. arXiv:2103.09715v4.

47 Bart M. P. Jansen, Daniel Lokshtanov, and Saket Saurabh. A near-optimal planarization algorithm. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1802–1811. SIAM, 2014. doi:10.1137/1.9781611973402.130.

48 Bart M. P. Jansen and Marcin Pilipczuk. Approximation and kernelization for chordal vertex deletion. *SIAM J. Discret. Math.*, 32(3):2258–2301, 2018. doi:10.1137/17M112035X.

49 J.Mark Keil. Finding hamiltonian circuits in interval graphs. *Information Processing Letters*, 20(4):201–206, 1985. doi:10.1016/0020-0190(85)90050-X.

50 David Kendall. Incidence matrices, interval graphs and seriation in archeology. *Pacific Journal of mathematics*, 28(3):565–570, 1969.

51 Eun Jung Kim and O-joung Kwon. Erdős-Pósa property of chordless cycles and its applications. *J. Comb. Theory, Ser. B*, 145:65–112, 2020. doi:10.1016/j.jctb.2020.05.002.

52 Lefteris M. Kirousis and Christos H. Papadimitriou. Interval graphs and searching. *Discrete Mathematics*, 55(2):181–184, 1985. doi:10.1016/0012-365X(85)90046-9.

53 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. *ACM Trans. Algorithms*, 14(2), April 2018. doi:10.1145/3170442.

54 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. *SIAM J. Comput.*, 47(3):675–702, 2018. doi:10.1137/16M1104834.

55 Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. On the hardness of eliminating small induced subgraphs by contracting edges. In Gregory Z. Gutin and Stefan Szeider, editors, *Parameterized and Exact Computation – 8th International Symposium, IPEC 2013, Sophia Antipolis, France, September 4-6, 2013, Revised Selected Papers*, volume 8246 of *Lecture Notes in Computer Science*, pages 243–254. Springer, 2013. doi:10.1007/978-3-319-03898-8_21.

56 Dániel Marx. Four shorts stories on surprising algorithmic uses of treewidth. In Fedor V. Fomin, Stefan Kratsch, and Erik Jan van Leeuwen, editors, *Treewidth, Kernels, and Algorithms – Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday*, volume 12160 of *Lecture Notes in Computer Science*, pages 129–144. Springer, 2020. doi:10.1007/978-3-030-42071-0_10.

57 Dániel Marx, Barry O'Sullivan, and Igor Razgon. Finding small separators in linear time via treewidth reduction. *ACM Trans. Algorithms*, 9(4):30:1–30:35, 2013. doi:10.1145/2500119.

58 Marcin Pilipczuk. A tight lower bound for vertex planarization on graphs of bounded treewidth. *Discret. Appl. Math.*, 231:211–216, 2017. doi:10.1016/j.dam.2016.05.019.

59 Michał Pilipczuk. Problems parameterized by treewidth tractable in single exponential time: A logical approach. In Filip Murlak and Piotr Sankowski, editors, *Mathematical Foundations of Computer Science 2011*, pages 520–531, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. doi:10.1007/978-3-642-22993-0_47.

60 R. C. Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, 1957. doi:10.1002/j.1538-7305.1957.tb01515.x.

61 N. Robertson and P.D. Seymour. Graph minors XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

62 Neil Robertson and P.D Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986. doi:10.1016/0196-6774(86)90023-4.

63 Donald J Rose. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. In *Graph theory and computing*, pages 183–217. Elsevier, 1972.

64 Toshiki Saitoh, Ryo Yoshinaka, and Hans L. Bodlaender. Fixed-treewidth-efficient algorithms for edge-deletion to interval graph classes. In Ryuhei Uehara, Seok-Hee Hong, and Subhas C. Nandy, editors, *WALCOM: Algorithms and Computation – 15th International Conference and Workshops, 2021, Yangon, Myanmar*, volume 12635 of *Lecture Notes in Computer Science*, pages 142–153. Springer, 2021. doi:10.1007/978-3-030-68211-8_12.

65 Ignasi Sau and Uéverton dos Santos Souza. Hitting Forbidden Induced Subgraphs on Bounded Treewidth Graphs. In Javier Esparza and Daniel Kráľ, editors, *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*, volume 170 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 82:1–82:15, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2020.82.

66 Hadas Shachnai and Meirav Zehavi. Representative families: A unified tradeoff-based approach. *Journal of Computer and System Sciences*, 82(3):488–502, 2016. doi:10.1016/j.jcss.2015.11.008.

67 Johan MM Van Rooij, Hans L Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In *European Symposium on Algorithms*, pages 566–577. Springer, 2009. doi:10.1007/978-3-642-04128-0_51.

68 Pim van 't Hof and Yngve Villanger. Proper interval vertex deletion. *Algorithmica*, 65(4):845–867, 2013. doi:10.1007/s00453-012-9661-3.

69 Michał Włodarczyk. Clifford algebras meet tree decompositions. *Algorithmica*, 81(2):497–518, 2019. doi:10.1007/s00453-018-0489-3.

70 Mihalis Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic Discrete Methods*, 2(1):77–79, 1981. doi:10.1137/0602010.

71 Meirav Zehavi. Mixing color coding-related techniques. In *Algorithms-ESA 2015*, pages 1037–1049. Springer, 2015. doi:10.1007/978-3-662-48350-3_86.

The Wrong Direction of Jensen's Inequality Is Algorithmically Right

Or Zamir ✉
Princeton University, NJ, USA

Abstract

Let $\mathcal{A}$ be an algorithm with expected running time e^X, conditioned on the value of some random variable X. We construct an algorithm $\mathcal{A}'$ with expected running time $O\left(e^{\mathsf{E}[X]}\right)$, that fully executes $\mathcal{A}$. In particular, an algorithm whose running time is a random variable T can be converted to one with expected running time $O\left(e^{\mathsf{E}[\ln T]}\right)$, which is never worse than $O(\mathsf{E}[T])$. No information about the distribution of X is required for the construction of $\mathcal{A}'$.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Algorithm design techniques; Theory of computation → Computational complexity and cryptography

Keywords and phrases algorithms, complexity, Jensen's inequality

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.107

Category Track A: Algorithms, Complexity and Games

Acknowledgements The author would like to thank Avi Wigderson for pointing out important references.

1 Introduction

Let $\mathcal{A}$ be a *Las Vegas*[1] randomized algorithm. Assume that conditioned on the value of some random variable X, the expected running time of $\mathcal{A}$ is e^X. By Jensen's inequality, $\mathsf{E}[e^X] \geq e^{\mathsf{E}[X]}$, and in fact $\mathcal{A}$'s expected running time might be much larger than $e^{\mathsf{E}[X]}$: Consider for example X that gets the value $\frac{1}{p}\mathsf{E}[X]$ with probability p and 0 otherwise, for any choice of $p > 0$; While the expectation of X is always $\mathsf{E}[X]$, the expectation of e^X is $p \cdot e^{\frac{1}{p}\mathsf{E}[X]}$ which can be arbitrarily large. We show that, surprisingly, any such $\mathcal{A}$ can be converted to a different Las-Vegas randomized algorithm $\mathcal{A}'$ that gives the same answer yet runs in expected time $O\left(e^{\mathsf{E}[X]}\right)$. Transforming $\mathcal{A}$ to $\mathcal{A}'$ does not require any assumption or knowledge about the distribution of X.

▶ **Theorem 1.** *There exists an algorithm T that receives as an input a randomized* Las Vegas *algorithm $\mathcal{A}$, and fully executes it. If the expected running time of $\mathcal{A}$ is e^X when conditioned on the value of some random variable X, then the expected running time of $T(A)$ is $O\left(e^{\mathsf{E}[X]}\right)$.*

As a corollary, any algorithm whose running time is a random variable T can be converted to one with expected running time $O\left(e^{\mathsf{E}[\ln T]}\right)$, which is never worse than $O(\mathsf{E}[T])$.

Recently we used the following simple version of Theorem 1 in a late revision of [10] to substantially simplify the analysis in the paper. The paper improves the running time of exact exponential-time algorithms for general Constraint Satisfaction Problems.

[1] A randomized algorithm is called *Las Vegas* if it always returns the correct answer, but its running time is a random variable.

▶ **Lemma 2** (from an up-to-date version of [10]). *Let $\mathcal{A}$ be an algorithm with expected running time 2^X conditioned on the value of a random variable X. There exists an algorithm $\mathcal{A}'$ that fully executes $\mathcal{A}$ and has an expected running time of $O\left(2^{\mathsf{E}[X]} \cdot \mathsf{E}[X]\right)$. Transforming $\mathcal{A}$ to $\mathcal{A}'$ requires knowing $\mathsf{E}[X]$.*

In this paper we focus on Theorem 1 itself, obtaining an optimal version of it.

We transform an algorithm $\mathcal{A}$ by using a sequence of *truncated evaluations*. A *truncated evaluation* of an algorithm $\mathcal{A}$ for t steps is the process of running algorithm $\mathcal{A}$ and aborting its run if it did not fully execute in the first t computational steps of its run. Each of the algorithms we present is thus a sequence of values $t_1, t_2, \ldots, t_i, \ldots$ which we use as thresholds for truncated evaluations of $\mathcal{A}$. We stop at the first time $\mathcal{A}$ is fully executed. These thresholds can be defined deterministically or be random variables. In the simpler algorithms we present, the thresholds depend on $\mathsf{E}[X]$ or even on the entire distribution X. For the proof of Theorem 1 the thresholds are completely independent of X and $\mathcal{A}$.

Truncated evaluations are frequently used in complexity theory (for example, see the proof of the time and space hierarchies in [2]). The first algorithmic use of such a sequence of truncated evaluations that we are aware of, is by Alt, Guibas, Mehlhorn, Karp and Wigderson [1]. They used it to convert *Las Vegas* randomized algorithms to *Monte Carlo* randomized algorithms, with success probability larger than what Markov's inequality gives. Luby, Sinclair and Zuckerman [5] then introduced a universal strategy for truncated evaluations. That is a sequence that is guaranteed to run in time $O(s \log s)$ if there exists any sequence that runs in time $O(s)$ for the same algorithm. Our contribution thus is two-fold: first, we prove the existence of good strategies in terms of $\mathsf{E}[X]$, and second, we show that these strategies can be explicitly constructed (i.e., without paying additional logarithmic factors). Not paying additional factors guarantees, due to Jensen's inequality, that our transformed algorithm is never worse than the original algorithm.

A natural use for such theorems is the regime of exponential-time algorithms. Consider the following toy example. As part of the classic algorithm of Schöning [9] it was shown that given an assignment α and a 3-SAT formula φ it is possible to test in $O\left(2^r\right)$ expected time whether φ has a satisfying assignment α_0 with $HAM\left(\alpha_0, \alpha\right) \leq r$, where $HAM(\cdot,\cdot)$ is the standard Hamming distance. This claim was also derandomized later [6]. We can use this primitive naively to obtain a non-trivial 3-SAT algorithm: Pick a random assignment α, and then run the above procedure of Schöning. Let X be the Hamming distance between α and a satisfying assignment α_0 of the input formula φ, this is a random variable. Conditioned on X, the expected running time of our algorithm is 2^X. The expected running time of our algorithm is thus $\mathsf{E}\left[2^X\right] = \sum_{r=0}^{n} \binom{n}{r} 2^{-n} 2^r = 2^{-n}(1+2)^n = \left(\frac{3}{2}\right)^n$. Using this paper's main Theorem, on the other hand, we can notice that $E[X] = \frac{n}{2}$ and thus we can convert the above algorithm in a black-box manner into one with expected running time $2^{\mathsf{E}[X]} = \left(\sqrt{2}\right)^n < \left(\frac{3}{2}\right)^n$. We note that Schöning already presented an algorithm using this procedure that is faster than both of the algorithms above.

Another similar example is the famous PPSZ algorithm for solving k-SAT, including its recent improvements, and generalizations for CSPs [7, 4, 3, 10]. In these algorithms, a randomly chosen permutation determines the number of input variables that we need to *guess* the values of. The expectation of this number of variables is then analyzed. The success probability or running time is exponential in this number. In the original PPSZ algorithm the analyzed quantity is the success probability and thus Jensen's inequality is applicable to bound this probability from below. In other variations (including [10]), the analyzed quantity is the running time and then Jensen's inequality is no longer applicable and either a more complicated analysis or the statement of this paper is necessary. Further discussion on possible applications and in particular possible implications for SAT algorithms appears in Section 3.

1.1 Preliminaries

We use standard notation throughout the paper. The notation $\ln x$ is used for the natural logarithm, and $\log x$ is used for the base two logarithm.

▶ **Definition 3** (Iterated functions). *Let $f : \mathbb{R} \to \mathbb{R}$ be a function. We define the iterated functions $f^{(k)} : \mathbb{R} \to \mathbb{R}$ recursively as follows. $f^{(0)}(x) := x$, and for any $k > 0$ we let $f^{(k)}(x) := f\left(f^{(k-1)}(x)\right)$.*

▶ **Definition 4** (Star functions). *Let $f : \mathbb{R} \to \mathbb{R}$ be a function. Assume f is strictly increasing and strictly shrinking[2] for all $x \geq x_0$. The star function of f, defined with respect to x_0 for every $x \geq x_0$, is*

$$f^{\star}(x) = \min\{k \mid f^{(k)}(x) \leq x_0\}.$$

The (general) Tower function $Tower_b(n,\, x) : \mathbb{N} \times \mathbb{R} \to \mathbb{R}$ is defined as $f^{(n)}(x)$ where $f(x) = b^x$. The standard Tower function $Tower : \mathbb{N} \to \mathbb{N}$ is defined as $Tower(n) = Tower_2(n, 1)$. The discrete inverse of the Tower function is $\log^{\star}$, defined with respect to $x_0 = 2$. That is, $\log^{\star} n$ is the smallest integer such that $Tower(\log^{\star} n) \geq n$.

2 Proof of Theorem 1

We begin by presenting a simple proof of Lemma 2.

Let $\mathcal{A}$ be an algorithm whose expected running time is e^X when we condition on the value of some non-negative random variable X. We observe, by Markov's inequality, that

$$\Pr\left(X > \mathsf{E}[X] + 1\right) \leq \frac{\mathsf{E}[X]}{\mathsf{E}[X]+1} = 1 - \frac{1}{\mathsf{E}[X]+1}.$$

Hence, consider the following algorithm.

Algorithm 1 Simple repetition.

Input: $\mathcal{A}$, $\mathsf{E}[X]$.
1: **repeat**
2: Run $\mathcal{A}$ for $2e^{\mathsf{E}[X]+1}$ computational steps.
3: **until** $\mathcal{A}$ completed a run.

▶ **Lemma 5.** *Algorithm 1 is expected to terminate in $O\left(e^{\mathsf{E}[X]} \cdot \mathsf{E}\left[X\right]\right)$ time.*

Proof. If $X \leq \mathsf{E}[X]+1$ then the expected running time of $\mathcal{A}$ is at most $e^{\mathsf{E}[X]+1}$, and hence by Markov's inequality a truncated evaluation of $\mathcal{A}$ for $2e^{\mathsf{E}[X]+1}$ steps concludes with probability at least $\frac{1}{2}$. By another application of Markov's inequality we got $\Pr\left(X \leq \mathsf{E}[X]+1\right) > \frac{1}{\mathsf{E}[X]+1}$. Hence, the expected number of iterations until the truncated evaluations concludes is at most $2\left(\mathsf{E}\left[X\right]+1\right)$. Each iteration takes $O\left(e^{\mathsf{E}[X]}\right)$ time. ◀

[2] That is, $f(x) < x$.

2.1 Optimal bound when the distribution of X is known

The bound given by Markov's inequality in

$$\Pr\left(X \geq \mathsf{E}[X]+1\right) \leq \frac{\mathsf{E}[X]}{\mathsf{E}[X]+1} = 1 - \frac{1}{\mathsf{E}[X]+1}$$

is attained *only* by the following distribution of X:

$$\Pr\left(X=k\right) := \begin{cases} \frac{1}{\mathsf{E}[X]+1} & k=0 \\ 1-\frac{1}{\mathsf{E}[X]+1} & k=\mathsf{E}[X]+1 \end{cases}.$$

In this distribution, on the other hand, the value of X is very small with a relatively high probability. In particular, in the case where $X < \mathsf{E}[X]+1$ we need to run $\mathcal{A}$ for much less than $e^{\mathsf{E}[X]+1}$ computational steps. Hence, it is sensible to hope that every distribution X has some threshold other than $\mathsf{E}[X]+1$ for which an algorithm similar to Algorithm 1 results in a better bound. Consider the following algorithm, which is a generalization of Algorithm 1 in which the threshold can be arbitrary.

Algorithm 2 Simple repetition with variable threshold.

Input: $\mathcal{A}$, t.
1: **repeat**
2: Run $\mathcal{A}$ for $2e^t$ computational steps.
3: **until** $\mathcal{A}$ completed a run.

▶ **Lemma 6.** *Let X be a non-negative random variable. There exists $t \in [0, \mathsf{E}[X]+1]$ such that $\frac{e^t}{\Pr(X<t)} \leq e^{\mathsf{E}[X]+1}$.*

Proof. Assume by contradiction that $\frac{e^t}{\Pr(X<t)} > e^{\mathsf{E}[X]+1}$ for every $t \in [0, \mathsf{E}[X]+1]$. Equivalently,

$$\Pr\left(X \geq t\right) = 1 - \Pr\left(X < t\right) > 1 - e^{t-(\mathsf{E}[X]+1)}.$$

Therefore,

$$\begin{aligned} \mathsf{E}[X] &= \int_0^\infty \Pr(X \geq t)\,\mathrm{d}t \geq \int_0^{\mathsf{E}[X]+1} \Pr(X \geq t)\,\mathrm{d}t \\ &> \int_0^{\mathsf{E}[X]+1} \left(1 - e^{t-(\mathsf{E}[X]+1)}\right)\mathrm{d}t \\ &= (\mathsf{E}[X]+1) - \left(1 - e^{-(\mathsf{E}[X]+1)}\right) = \mathsf{E}[X] + e^{-(\mathsf{E}[X]+1)} \\ &> \mathsf{E}[X], \end{aligned}$$

which is a contradiction. ◀

Lemma 6 implies the following.

▶ **Corollary 7.** *For every distribution X there exists a value of $t = t(X)$ for which Algorithm 2 runs in $O\left(e^{\mathsf{E}[X]}\right)$ time.*

We note that the additive constant $+1$ in the exponent in Lemma 6 is necessary. For a parameter E, consider the random variable X supported on $[0, E+1+\ln\left(1+e^{-(E+1)}\right)]$ and distributed with density $f(x) := e^{x-(E+1)}$; Its expectation is

$$\begin{aligned}\mathsf{E}[X] &= \int_0^{E+1+\ln\left(1+e^{-(E+1)}\right)} x f(x)\,\mathrm{d}x \\ &= \left((x-1)\,e^{x-(E+1)}\right)\Big|_0^{E+1+\ln\left(1+e^{-(E+1)}\right)} \\ &= \left(E+\ln\left(1+e^{-(E+1)}\right)\right)\cdot\left(1+e^{-(E+1)}\right)+e^{-(E+1)} \\ &= E+O\left(e^{-(E+1)}\right) = E+o(1),\end{aligned}$$

where the $o(1)$ term is vanishing when $E \to \infty$. On the other hand, for any $t \geq 0$ we have

$$\frac{e^t}{\Pr\left(X<t\right)} = \frac{e^t}{\min\left(1,\ e^{t-(E+1)}-e^{-(E+1)}\right)} > \frac{e^t}{e^{t-(E+1)}} = e^{E+1}.$$

2.2 Optimal algorithm when the distribution of X is unknown

If the only thing known about the distribution of X is its expectation $\mathsf{E}[X]$, then there is no fixed value of t for which Algorithm 2 is better than Algorithm 1. Fix a value of $\mathsf{E}[X]$ and a choice of t. If $t < \mathsf{E}[X]$ then with the constant distribution $X \equiv \mathsf{E}[X]$ Algorithm 2 never terminates. Otherwise, $t \geq \mathsf{E}[X]$ and we consider the following distribution X:

$$\Pr\left(X=k\right) := \begin{cases} 1-\frac{\mathsf{E}[X]}{t+1} & k=0 \\ \frac{\mathsf{E}[X]}{t+1} & k=t+1 \end{cases}.$$

For this distribution, the expected running time of Algorithm 2 is

$$\frac{e^t}{1-\frac{\mathsf{E}[X]}{t+1}} = e^{\mathsf{E}[X]}\cdot\frac{e^s}{\left(\frac{s+1}{\mathsf{E}[X]+s+1}\right)} = e^{\mathsf{E}[X]}\left(1+\frac{\mathsf{E}[X]}{s+1}\right)e^s \geq e^{\mathsf{E}[X]}\mathsf{E}[X]\cdot\frac{e^s}{s+1} \geq e^{\mathsf{E}[X]}\mathsf{E}[X],$$

where $s := t-\mathsf{E}[X] \geq 0$ and the last inequality follows as $e^s \geq s+1$ for any s.

To improve Algorithm 1 then, we need to consider *several thresholds*. We demonstrate this idea with the following Lemma.

▶ **Lemma 8.** *Let X be a non-negative random variable. It holds that either $\Pr\left(X \leq \mathsf{E}\left[X\right]-\ln\mathsf{E}\left[X\right]\right) > \frac{1}{\mathsf{E}[X]+1}$ or $\Pr\left(X \leq \mathsf{E}\left[X\right]+2\right) > \frac{1}{\ln\mathsf{E}[X]+2}$.*

Proof. Assume that $p := \Pr\left(X \leq \mathsf{E}\left[X\right]-\ln\mathsf{E}\left[X\right]\right) \leq \frac{1}{\mathsf{E}[X]+1}$. We observe that

$$\begin{aligned}\mathsf{E}\left[X\right] &= p\,\mathsf{E}\left[X \mid X \leq \mathsf{E}\left[X\right]-\ln\mathsf{E}\left[X\right]\right] + (1-p)\,\mathsf{E}\left[X \mid X > \mathsf{E}\left[X\right]-\ln\mathsf{E}\left[X\right]\right] \\ &\geq (1-p)\,\mathsf{E}\left[X \mid X > \mathsf{E}\left[X\right]-\ln\mathsf{E}\left[X\right]\right],\end{aligned}$$

and hence

$$\begin{aligned}\mathsf{E}\left[X \mid X > \mathsf{E}\left[X\right]-\ln\mathsf{E}\left[X\right]\right] &\leq \frac{1}{1-p}\mathsf{E}[X] \\ &\leq \frac{1}{1-\frac{1}{\mathsf{E}[X]+1}}\mathsf{E}[X] \\ &= \mathsf{E}[X]+1.\end{aligned}$$

Denote by $Y := X - (\mathsf{E}[X] - \ln \mathsf{E}[X])$. The above can now be rephrased as $\mathsf{E}[Y \mid Y > 0] \leq \ln \mathsf{E}[X] + 1$. Applying Markov's inequality to Y conditioned on $Y > 0$, we get

$$\Pr(Y > \ln \mathsf{E}[X] + 2 \mid Y > 0) \leq \frac{\ln \mathsf{E}[X] + 1}{\ln \mathsf{E}[X] + 2} = 1 - \frac{1}{\ln \mathsf{E}[X] + 2}.$$

We conclude by noting that

$$\Pr(X > \mathsf{E}[X] + 2) = \Pr(Y > \ln \mathsf{E}[X] + 2) \leq \Pr(Y > \ln \mathsf{E}[X] + 2 \mid Y > 0). \quad \blacktriangleleft$$

Consider the following Algorithm.

Algorithm 3 Two thresholds algorithm.

Input: $\mathcal{A}$, $\mathsf{E}[X]$.
1: **repeat**
2: **for** $\lceil \mathsf{E}[X] + 1 \rceil$ **times do**
3: Run $\mathcal{A}$ for $2e^{\mathsf{E}[X] - \ln \mathsf{E}[X]}$ computational steps.
4: **for** $\lceil \ln \mathsf{E}[X] + 2 \rceil$ **times do**
5: Run $\mathcal{A}$ for $2e^{\mathsf{E}[X]+2}$ computational steps.
6: **until** $\mathcal{A}$ completed a run.

Due to Lemma 8, each iteration of the outermost loop of Algorithm 3 succeeds to fully execute $\mathcal{A}$ with probability larger than $1 - e^{-1}$. Thus, in expectation we run this loop for a constant number of iterations. The first **for** loop takes $O\left(\mathsf{E}[X] \cdot e^{\mathsf{E}[X] - \ln \mathsf{E}[X]}\right) = O\left(e^{\mathsf{E}[X]}\right)$ expected time, and the second takes $O\left(e^{\mathsf{E}[X]} \ln \mathsf{E}[X]\right)$. We conclude the following.

▶ **Corollary 9.** *Algorithm 3 runs in expected time* $O\left(e^{\mathsf{E}[X]} \ln \mathsf{E}[X]\right)$.

Intuitively, the *proof* of Lemma 8 can be viewed as a reduction from the variable X to the variable $Y \mid (Y > 0)$, that has a much lower expectation: $\mathsf{E}[Y \mid Y > 0] \leq \ln \mathsf{E}[X] + 1$. We can thus hope that iterating the proof for $\ln^\star \mathsf{E}[X]$ times would result in reducing X to a variable with constant expectation. A natural implementation of this idea would result in an algorithm that runs in expected time $O\left(e^{\mathsf{E}[X]} \ln^\star \mathsf{E}[X]\right)$. We next formalize this intuition, and do so in a more careful manner to avoid the $\ln^\star \mathsf{E}[X]$ factor.

▶ **Definition 10.** *Let* $\lambda(x) := 3\ln(x)$ *and note it is strictly increasing and shrinking for all* $x \geq 5$. *We define* $\lambda^\star(x)$, *for* $x \geq 5$, *to be the smallest* $k \in \mathbb{N}$ *such that* $\lambda^{(k)}(x) \leq 5$.

▷ Claim 11. The following hold for all $x \geq 5$:
1. $\lambda^\star(x) = \Theta(\log^\star x)$.
2. $\lambda^{(\lambda^\star(x))}(x) > 4$.
3. $\sum_{i=0}^{\lambda^\star(x)} \frac{1}{\lambda^{(i)}(x)} < 2$.

Proof. (1) We have that $\lambda^{(2)}(x) \leq \log x \leq \lambda(x)$ for all $x \geq 410$. In particular, $\log^\star x \leq \lambda^\star(x) \leq 2\log^\star x + \lambda^\star(410)$.

(2) $\lambda^{(\lambda^\star(x)-1)}(x) > 5$ and hence $\lambda^{(\lambda^\star(x))}(x) > \lambda(5) > 4.82$.

(3) For all $x \geq 17$ it holds that $\lambda(x) \leq \frac{x}{2}$. Let k' be the smallest integer such that $\lambda^{(k')}(x) < 17$. We thus have

$$\sum_{i=0}^{k'-1} \frac{1}{\lambda^{(i)}(x)} < \frac{1}{17} \sum_{i=0}^{\infty} 2^{-i} = \frac{2}{17}.$$

On the other hand, there are at most $\lambda^\star(17)$ summands that are strictly larger than $\frac{1}{17}$, thus by (2) we have

$$\sum_{i=k'}^{\lambda^\star(x)} \frac{1}{\lambda^{(i)}(x)} < \frac{\lambda^\star(17)}{4} = \frac{5}{4}. \qquad \lhd$$

We are now ready to prove a generalized version of Lemma 8, that is going to be the core of our final algorithm.

▶ **Lemma 12.** *Let X be a non-negative distribution and $E \geq \max(\mathsf{E}[X], 5)$ be an upper bound on its expectation. There either exists $1 \leq k \leq \lambda^\star(E)$ such that $\Pr\left(X < E - \lambda^{(k)}(E)\right) \geq \left(\left(\lambda^{(k-1)}(E)+2\right)^2+1\right)^{-1}$, or it holds that $\Pr(X < E+10) \geq \frac{1}{2}$.*

Proof. We recursively denote by $E_0 := E$ and by $E_k := \lambda^{(k)}(E) + \sum_{i=0}^{k-1} \frac{1}{E_i}$ for $1 \leq k \leq \lambda^\star(E)$. Note that $E_k \geq \lambda^{(k)}(E)$ and hence also

$$E_k = \lambda^{(k)}(E) + \sum_{i=0}^{k-1} \frac{1}{E_i} \leq \lambda^{(k)}(E) + \sum_{i=0}^{k-1} \frac{1}{\lambda^{(i)}(E)} < \lambda^{(k)}(E) + 2,$$

where the last inequality follows from Claim 11. In particular, $\lambda^{(k)}(E) \leq E_k < \lambda^{(k)}(E) + 2$.

Assume that $\Pr\left(X < E - \lambda^{(k)}(E)\right) < \left(\left(\lambda^{(k-1)}(E)+2\right)^2+1\right)^{-1} < \frac{1}{(E_{k-1})^2+1}$ for every $1 \leq k \leq \lambda^\star(E)$.

Denote by $Y_k := X - \left(E - \lambda^{(k)}(E)\right)$ for $k \geq 0$. We prove by induction on k that $\mathsf{E}[Y_k \mid Y_k \geq 0] \leq E_k$. For $k = 0$ the claim is straightforward as $Y_0 = X$ and $E_0 = E$. For the inductive step, we assume the hypothesis holds for $k-1$ and show it holds for k. We note that $Y_{k-1} > Y_k$ and hence if $Y_k \geq 0$ then $Y_{k-1} \geq 0$ as well. Hence,

$$\begin{aligned}\mathsf{E}[Y_{k-1} \mid Y_{k-1} \geq 0] &\geq \Pr(Y_k \geq 0 \mid Y_{k-1} \geq 0)\, \mathsf{E}[Y_{k-1} \mid Y_k \geq 0] \\ &\geq \Pr(Y_k \geq 0)\, \mathsf{E}[Y_{k-1} \mid Y_k \geq 0].\end{aligned}$$

Thus, by the induction hypothesis we have

$$\begin{aligned}\mathsf{E}[Y_{k-1} \mid Y_k \geq 0] &\leq \frac{\mathsf{E}[Y_{k-1} \mid Y_{k-1} \geq 0]}{\Pr(Y_k \geq 0)} \\ &\leq \frac{E_{k-1}}{1 - \frac{1}{(E_{k-1})^2+1}} \\ &= E_{k-1} + \frac{1}{E_{k-1}}.\end{aligned}$$

Therefore,

$$\begin{aligned}\mathsf{E}[Y_k \mid Y_k \geq 0] &= \mathsf{E}[Y_{k-1} \mid Y_k \geq 0] + \lambda^{(k)}(E) - \lambda^{(k-1)}(E) \\ &\leq E_{k-1} + \frac{1}{E_{k-1}} + \lambda^{(k)}(E) - \lambda^{(k-1)}(E) \\ &= E_k.\end{aligned}$$

In particular, we have that $\mathsf{E}\left[Y_{\lambda^\star(E)} \mid Y_{\lambda^\star(E)} \geq 0\right] \leq E_{\lambda^\star(E)} < \lambda^{(\lambda^\star(E))}(E) + 2 \leq 7$.

Therefore,

$$\begin{aligned}\Pr\left(X \geq E+10\right) &\leq \Pr\left(X \geq E+10 \;\middle|\; X \geq E-\lambda^{(\lambda^\star(E))}(E)\right)\\ &= \Pr\left(Y_{\lambda^\star(E)} \geq \lambda^{(\lambda^\star(E))}(E)+10 \;\middle|\; Y_{\lambda^\star(E)} \geq 0\right)\\ &\leq \Pr\left(Y_{\lambda^\star(E)} \geq 14 \;\middle|\; Y_{\lambda^\star(E)} \geq 0\right) < \frac{7}{14} = \frac{1}{2}.\end{aligned}$$

◀

Consider the following algorithm.

Algorithm 4 Multiple thresholds algorithm.

Input: $\mathcal{A}$, E.

1: **repeat**
2: **for** $k = 1$ to $\lambda^\star(E)$ **do**
3: **for** $2\lceil\left(\lambda^{(k-1)}(E)+2\right)^2+1\rceil$ **times do**
4: Run $\mathcal{A}$ for $2e^{E-\lambda^{(k)}(E)}$ computational steps.
5: **for** 2 **times do**
6: Run $\mathcal{A}$ for $2e^{E+10}$ computational steps.
7: **until** $\mathcal{A}$ completed a run.

▶ **Corollary 13** (of Lemma 12). *Each* ***repeat*** *loop of Algorithm 4 fully executes $\mathcal{A}$ with probability at least $\frac{3}{4}$.*

▶ **Lemma 14.** *Let $E \geq \max\left(\mathsf{E}[X],\ 5\right)$, Algorithm 4 runs in $O\left(e^E\right)$ expected time.*

Proof. By Corollary 13 we enter the **repeat** loop a constant number of times in expectation. We thus analyze the computational cost of a single **repeat** loop. The evaluations in Lines $5-6$ take $O\left(e^E\right)$ time. The evaluations in Lines $2-4$ take

$$\sum_{k=1}^{\lambda^\star(E)} 2\lceil\left(\lambda^{(k-1)}(E)+2\right)^2+1\rceil \cdot 2e^{E-\lambda^{(k)}(E)} = O\left(e^E \cdot \sum_{k=1}^{\lambda^\star(E)} \left(\lambda^{(k-1)}(E)\right)^2 e^{-\lambda^{(k)}(E)}\right)$$

time. By the definition of $\lambda(x)$, we have $e^{-\lambda^{(k)}(x)} = e^{-3\ln\left(\lambda^{(k-1)}(x)\right)} = \left(\lambda^{(k-1)}(x)\right)^{-3}$. In particular,

$$\sum_{k=1}^{\lambda^\star(E)} \left(\lambda^{(k-1)}(E)\right)^2 e^{-\lambda^{(k)}(E)} = \sum_{k=1}^{\lambda^\star(E)} \left(\lambda^{(k-1)}(E)\right)^{-1} < 2,$$

where the last inequality follows from Claim 11. ◀

Finally, we also get rid of the necessity to provide the algorithm with E or $\mathsf{E}[X]$.

▶ **Theorem 15.** *Algorithm 5 runs in expected time $O\left(e^{\mathsf{E}[X]}\right)$.*

Proof. By Lemma 14 the iteration of the outermost **for** loop corresponding to E takes at most $C \cdot e^E$ time, for some global constant C. All iterations in which $E < \mathsf{E}[X]$ thus take $O\left(e^{\mathsf{E}[X]}\right)$ time. By Corollary 13, each subsequent iteration succeeds with probability at least $\frac{3}{4}$. Thus the expected running time is bounded by

$$Ce^{\mathsf{E}[X]} \cdot \sum_{t=0}^{\infty} e^t \left(\frac{1}{4}\right)^t = O\left(e^{\mathsf{E}[X]}\right).$$

◀

Algorithm 5 Final algorithm.

Input: $\mathcal{A}$.

for $E = 5$ to ∞ **do**
  for $k = 1$ to $\lambda^\star(E)$ **do**
    for $2\lceil\left(\lambda^{(k-1)}(E)+2\right)^2+1\rceil$ **times do**
      Run $\mathcal{A}$ for $2e^{E-\lambda^{(k)}(E)}$ computational steps.
  for 2 **times do**
    Run $\mathcal{A}$ for $2e^{E+10}$ computational steps.
  return if $\mathcal{A}$ completed a run.

3 Conclusions and Open Problems

We showed that a Las-Vegas algorithm with expected running e^X conditioned on the value of some random variable X, can always be converted to a Las-Vegas algorithm with expected running time $O\left(e^{\mathsf{E}[X]}\right)$. In particular, an algorithm whose running time is a random variable T can be converted to one with expected running time $O\left(e^{\mathsf{E}[\ln T]}\right)$, which is never worse than $O(\mathsf{E}[T])$.

We demonstrated a use of this theorem to simplify a proof in the regime of exponential time algorithms [10]. It is interesting to try applying it to other exponential and non-exponential time algorithms and see if it can simplify or even improve the analysis.

3.1 Considering the variance

In terms of $\mathsf{E}[X]$ only, we can not get any better than $O\left(e^{\mathsf{E}[X]}\right)$ as the distribution of X might be constant. In that case though, the variance of X is zero. Can we get a better bound just by assuming that the variance of X is large? Unfortunately, with the standard definition of variance this is not the case. For any choice of E and $V \geq 2E^2e^{-E}$ consider the following distribution:

$$\Pr(X=k) := \begin{cases} e^{-E} & k=0 \\ 1-\frac{Ve^{-E}}{V-E^2e^{-E}} & k=E \\ \frac{\left(Ee^{-E}\right)^2}{V-E^2e^{-E}} & k=\frac{V}{Ee^{-E}} \end{cases}.$$

Its expectation is E, its variance is V, which can be arbitrarily large, and nevertheless $\Pr(X < E) = e^{-E}$ so no strategy can beat $O\left(e^E\right)$.

On the other hand, the wishful thinking above is true with some other notions of deviation. For example, if we consider mean absolute deviation instead of standard deviation (i.e., $\mathsf{E}\left[\,|X-\mathsf{E}[X]|\,\right]$), then *it is* true that if the deviation is large then we can get a better running time. It is intriguing to find useful notion of deviation for which such a statement is true, with the goal of improving the running time of algorithms by analyzing the deviation of X.

In particular, consider the PPSZ algorithm for solving k-SAT [7] [4]. The algorithm uses randomization in two ways: first, a random permutation of the variables in the input formulas is drawn; Then, the chosen permutation determines the number of variables we need to *guess* the value of. In a recent improvement of the PPSZ analysis, Scheder [8] showed that in some large subset of permutations the number of guessed variables is smaller than what we expect when taking a uniformly random permutation. In particular, this implies that there is some non-negligible variance in the original algorithm's running time. Can we get better SAT algorithms by analyzing this variance?

References

1. Helmut Alt, Leonidas Guibas, Kurt Mehlhorn, Richard Karp, and Avi Wigderson. A method for obtaining randomized algorithms with small tail probabilities. *Algorithmica*, 16(4):543–547, 1996.
2. Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.
3. Thomas Dueholm Hansen, Haim Kaplan, Or Zamir, and Uri Zwick. Faster k-sat algorithms using biased-ppsz. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 578–589, 2019.
4. Timon Hertli. 3-sat faster and simpler—unique-sat bounds for ppsz hold in general. *SIAM Journal on Computing*, 43(2):718–729, 2014.
5. Michael Luby, Alistair Sinclair, and David Zuckerman. Optimal speedup of las vegas algorithms. *Information Processing Letters*, 47(4):173–180, 1993.
6. Robin A Moser and Dominik Scheder. A full derandomization of schöning's k-sat algorithm. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 245–252, 2011.
7. Ramamohan Paturi, Pavel Pudlák, Michael E Saks, and Francis Zane. An improved exponential-time algorithm for k-sat. *Journal of the ACM (JACM)*, 52(3):337–364, 2005.
8. Dominik Scheder. Ppsz is better than you think. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 205–216. IEEE, 2022.
9. T Schoning. A probabilistic algorithm for k-sat and constraint satisfaction problems. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 410–414. IEEE, 1999.
10. Or Zamir. Faster algorithm for unique $(k, 2)$-csp. *ESA*, 2022.

A Hyperbolic Extension of Kadison-Singer Type Results

Ruizhe Zhang
The University of Texas at Austin, TX, USA

Xinzhi Zhang
University of Washington, Seattle, WA, USA

Abstract

In 2013, Marcus, Spielman, and Srivastava resolved the famous Kadison-Singer conjecture. It states that for n independent random vectors $v_1, \cdots, v_n$ that have expected squared norm bounded by ϵ and are in the isotropic position in expectation, there is a positive probability that the determinant polynomial $\det(xI - \sum_{i=1}^n v_i v_i^\top)$ has roots bounded by $(1+\sqrt{\epsilon})^2$. An interpretation of the Kadison-Singer theorem is that we can always find a partition of the vectors $v_1, \cdots, v_n$ into two sets with a low discrepancy in terms of the spectral norm (in other words, rely on the determinant polynomial).

In this paper, we provide two results for a broader class of polynomials, the hyperbolic polynomials. Furthermore, our results are in two generalized settings:

- The first one shows that the Kadison-Singer result requires a weaker assumption that the vectors have a bounded sum of hyperbolic norms.
- The second one relaxes the Kadison-Singer result's distribution assumption to the Strongly Rayleigh distribution.

To the best of our knowledge, the previous results only support determinant polynomials [Anari and Oveis Gharan'14, Kyng, Luh and Song'20]. It is unclear whether they can be generalized to a broader class of polynomials. In addition, we also provide a sub-exponential time algorithm for constructing our results.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures

Keywords and phrases Kadison-Singer conjecture, Hyperbolic polynomials, Strongly-Rayleigh distributions, Interlacing polynomials

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.108

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: http://arxiv.org/abs/2305.02566

Funding *Ruizhe Zhang*: Supported by the University Graduate Continuing Fellowship from UT Austin.
Xinzhi Zhang: Research supported by NSF grant CCF-1813135, NSF CAREER Award and Packard Fellowships.

1 Introduction

Introduced by [30], the Kadison-Singer problem was a long-standing open problem in mathematics. It was resolved by Marcus, Spielman, and Srivastrava in their seminal work [43]: For any set of independent random vectors $u_1, \cdots, u_n$ such that each u_i has finite support, and $u_1, \cdots, u_n$ are in isotropic positions in expectation, there is positive probability that $\sum_{i=1}^n u_i u_i^*$ has spectral norm bounded by $1 + O(\max_{i\in[n]} \|u_i\|)$. The main result of [43] is as follows:

▶ **Theorem 1** (Main result of [43]). *Let $\epsilon > 0$ and let $v_1, \cdots, v_n \in \mathbb{C}^m$ be n independent random vectors with finite support, such that $\mathbb{E}[\sum_{i=1}^n v_i v_i^*] = I$, and $\mathbb{E}[\|v_i\|^2] \leq \epsilon$, $\forall i \in [n]$. Then*

$$\Pr\left[\left\|\sum_{i\in[n]} v_i v_i^*\right\| \leq (1+\sqrt{\epsilon})^2\right] > 0.$$

The Kadison-Singer problem is closely related to discrepancy theory, which is an essential area in mathematics and theoretical computer science. A classical discrepancy problem is as follows: given n sets over n elements, can we color each element in red or blue such that each set has roughly the same number of elements in each color? More formally, for vectors $a_1, \ldots, a_n \in \mathbb{R}^n$ with $\|a_i\|_\infty \leq 1$ and a coloring $s \in \{\pm 1\}^n$, the discrepancy is defined by $\mathrm{Disc}(a_1, \ldots, a_n; s) := \|\sum_{i\in[n]} s_i a_i\|_\infty$. The famous Spencer's Six Standard Deviations Suffice Theorem [57] shows that there exists a coloring with discrepancy at most $6\sqrt{n}$, which beats the standard Chernoff bound showing that a random coloring has discrepancy $\sqrt{n \log n}$. More generally, we can consider the "matrix version" of discrepancy: for matrices $A_1, \ldots, A_n \in \mathbb{R}^{d\times d}$ and a coloring $s \in \{\pm 1\}^n$,

$$\mathrm{Disc}(A_1, \ldots, A_n; s) := \left\|\sum_{i\in[n]} s_i A_i\right\|.$$

Theorem 1 is equivalent to the following discrepancy result for rank-1 matrices:

▶ **Theorem 2** ([43]). *Let $u_1, \ldots, u_n \in \mathbb{C}^m$ and suppose $\max_{i\in[n]} \|u_i u_i^*\| \leq \epsilon$ and $\sum_{i=1}^n u_i u_i^* = I$. Then,*

$$\min_{s\in\{\pm1\}^n} \mathrm{Disc}(u_1 u_1^*, \ldots, u_n u_n^*; s) \leq O(\sqrt{\epsilon}).$$

In other words, the minimum discrepancy of rank-1 isotropic matrices is bounded by $O(\sqrt{\epsilon})$, where ϵ is the maximum spectral norm. This result also beats the matrix Chernoff bound [60], which shows that a random coloring for matrices has discrepancy $O(\sqrt{\epsilon \log d})$. The main techniques in [43] are the method of interlacing polynomials and the barrier methods developed in [42].

Several generalizations of the Kadison-Singer-type results, which have interesting applications in theoretical computer science, have been established using the same technical framework as described in [43]. In particular, Kyng, Luh, and Song [36] provided a "four derivations suffice" version of Kadison-Singer conjecture: Instead of assuming every independent random vector has a bounded norm, the main result in [36] only requires that the sum of the squared spectral norm is bounded by σ^2, and showed a discrepancy bound of 4σ:

▶ **Theorem 3** ([36]). *Let $u_1, \ldots, u_n \in \mathbb{C}^m$ and $\sigma^2 = \|\sum_{i=1}^n (u_i u_i^*)^2\|$. Then, we have*

$$\Pr_{\xi\sim\{\pm1\}^n}\left[\left\|\sum_{i=1}^n \xi_i u_i u_i^*\right\| \leq 4\sigma\right] > 0.$$

This result was recently applied by [38] to approximate solutions of generalized network design problems.

Moreover, Anari and Oveis-Gharan [6] generalized the Kadison-Singer conjecture into the setting of real-stable polynomials. Instead of assuming the random vectors are independent, [6] assumes that the vectors are sampled from any homogeneous strongly Rayleigh distribution with bounded marginal probability, have bounded norm, and are in an isotropic position:

▶ **Theorem 4** ([6]). *Let μ be a homogeneous strongly Rayleigh probability distribution on $[n]$ such that the marginal probability of each element is at most ϵ_1, and let $u_1, \cdots, u_n \in \mathbb{R}^m$ be vectors in an isotropic position, $\sum_{i=1}^n u_i u_i^* = I$, such that $\max_{i\in[n]} \|u_i\|^2 \le \epsilon_2$. Then*

$$\Pr_{S\sim\mu}\left[\left\|\sum_{i\in S} u_i u_i^*\right\| \le 4(\epsilon_1+\epsilon_2)+2(\epsilon_1+\epsilon_2)^2\right] > 0.$$

Theorem 4 has a direct analog in spectral graph theory: Given any (weighted) connected graph $G = (V, E)$ with Laplacian L_G. For any edge $e = (u, v) \in E$, define the vector corresponding to e as $v_e = L_G^{\dagger/2}(\mathbf{1}_u - \mathbf{1}_v)$ (here $L_G^\dagger$ is the Moore-Penrose inverse). Then the set of $\{v_e : e \in E\}$ are in isotropic position, and $\|v_e\|^2$ equals to the graph effective resistance with respect to e. Also, any spanning tree distribution of the edges in E is homogeneous strongly Rayleigh. It follows from Theorem 4 that any graph with bounded maximum effective resistance has a spectrally-thin spanning tree [6]. Moreover, [7] provided an exciting application to the asymmetric traveling salesman problem and obtained an $O(\log\log n)$-approximation.

Another perspective of generalizing the Kadison-Singer theorem is to study the discrepancy with respect to a more general norm than the spectral norm, which is the largest root of a determinant polynomial. A recent work by Bränden [19] proved a high-rank version of Theorem 2 for *hyperbolic* polynomial, which is a larger class of polynomials including the determinant polynomial. Moreover, the hyperbolic norm on vectors is a natural generalization of the matrix spectral norm. We will introduce hyperbolic polynomials in the full version of our paper. From this perspective, it is very natural to ask:

Can we also extend Theorem 3 and Theorem 4 to a more general class of polynomials, e.g., hyperbolic polynomials?

1.1 Our results

In this work, we provide an affirmative answer by generalizing both Theorem 3 and Theorem 4 into the setting of hyperbolic polynomials. Before stating our main results, we first introduce some basic notation of hyperbolic polynomials below.

Hyperbolic polynomials form a broader class of polynomials that encompasses determinant polynomials and homogeneous real-stable polynomials. An m-variate, degree-d homogeneous polynomial $h \in \mathbb{R}[x_1, \cdots, x_m]$ is *hyperbolic* with respect to a direction $e \in \mathbb{R}^m$ if the univariate polynomial $t \mapsto h(te - x)$ has only real roots for all $x \in \mathbb{R}^m$. The set of $x \in \mathbb{R}^m$ such that all roots of $h(te - x)$ are non-negative (or strictly positive) is referred to as the hyperbolicity cone $\Gamma_+^h(e)$ (or $\Gamma_{++}^h(e)$). It is a widely recognized result [16] that any vector x in the open hyperbolicity cone $\Gamma_{++}^h(e)$ is itself hyperbolic with respect to the polynomial h and have the same hyperbolicity cone as e, meaning that $\Gamma_{++}^h(e) = \Gamma_{++}^h(x)$. Therefore, the unique hyperbolicity cone of h can simply be expressed as Γ_+^h.

The hyperbolic polynomials have similarities to determinant polynomials of matrices, as they both can be used to define trace, norm, and eigenvalues. Given a hyperbolic polynomial $h \in \mathbb{R}[x_1, \cdots, x_m]$ and any vector $e \in \Gamma_{++}^h$, we can define a norm with respect to $h(x)$ and e as follows: for any $x \in \mathbb{R}^m$, its *hyperbolic norm* $\|x\|_h$ is equal to the largest root (in absolute value) of the linear restriction polynomial $h(te - x) \in \mathbb{R}[t]$. Similar to the eigenvalues of matrices, we define the *hyperbolic eigenvalues* of x to be the d roots of $h(te - x)$, denoted by $\lambda_1(x) \ge \cdots \ge \lambda_d(x)$. We can also define the *hyperbolic trace* and the *hyperbolic rank*:

$$\mathrm{tr}_h[x] := \sum_{i=1}^d \lambda_i(x), \quad \text{and} \quad \mathrm{rank}(x)_h := |\{i \in [d] : \lambda_i(x) \neq 0\}|.$$

Recall that both Theorem 3 and Theorem 4 upper-bound the spectral norm of the sum $\|\sum_{i=1}^n \xi_i v_i v_i^\top\|$. In the setting of hyperbolic polynomials, we should upper bound the hyperbolic norm $\|\sum_{i=1}^n \xi_i v_i\|_h$ for vectors $v_1, \ldots, v_n$ in the hyperbolicity cone, which is the set of vectors with all non-negative hyperbolic eigenvalues.

Our main results are as follows:

▶ **Theorem 5** (Main Result I, informal hyperbolic version of Theorem 1.4, [36]). *Let $h \in \mathbb{R}[x_1, \ldots, x_m]$ denote a hyperbolic polynomial in direction $e \in \mathbb{R}^m$. Let $v_1, \ldots, v_n \in \Gamma_+^h$ be n vectors in the closed hyperbolicity cone. Let $\xi_1, \ldots, \xi_n$ be n independent random variables with finite supports and $\mathbb{E}[\xi_i] = \mu_i$ and $\mathbf{Var}[\xi_i] = \tau_i^2$. Suppose $\sigma := \|\sum_{i=1}^n \tau_i^2 \operatorname{tr}_h[v_i] v_i\|_h$. Then there exists an assignment $(s_1, \ldots, s_n)$ with s_i in the support of ξ_i for all $i \in [n]$, such that*

$$\left\| \sum_{i=1}^n (s_i - \mu_i) v_i \right\|_h \leq 4\sigma.$$

We remark that Theorem 5 does not require the isotropic position condition of $v_1, \cdots, v_n$ as in [19]. In addition, we only need the sum of $\operatorname{tr}_h[v_i]v_i$'s hyperbolic norm to be bounded, while [19]'s result requires each vector's trace to be bounded individually.

We would also like to note that the class of hyperbolic polynomials is much broader than that of determinant polynomials, which were used in the original Kadison-Singer-type theorems. Lax conjectured in [39] that every 3-variate hyperbolic/real-stable polynomial could be represented as a determinant polynomial, this was later resolved in [28, 40]. However, the Lax conjecture is false when the number of variables exceeds 3, as demonstrated in [17, 20] with counterexamples of hyperbolic/real-stable polynomials $h(x)$ for which even $(h(x))^k$ cannot be represented by determinant polynomials for any $k > 0$.

Our second main result considers the setting where the random vectors are not independent, but instead, sampled from a strongly Rayleigh distribution. We say a distribution μ over the subsets of $[n]$ is *strongly Rayleigh* if its generating polynomial $g_\mu(z) := \sum_{S \subseteq [n]} \mu(S) z^S \in \mathbb{R}[z_1, \ldots, z_n]$ is a *real-stable polynomial*, which means $g_\mu(z)$ does not have any root in the upper-half of the complex plane, i.e., $g_\mu(z) \neq 0$ for any $z \in \mathbb{C}^n$ with $\Re(z) \succ 0$.

▶ **Theorem 6** (Main Result II, informal hyperbolic version of Theorem 1.2, [6]). *Let $h \in \mathbb{R}[x_1, \ldots, x_m]$ denote hyperbolic polynomial in direction $e \in \mathbb{R}^m$. Let μ be a homogeneous strongly Rayleigh probability distribution on $[n]$ such that the marginal probability of each element is at most ϵ_1.*

Suppose $v_1, \cdots, v_n \in \Gamma_+^h$ are in the hyperbolicity cone of h such that $\sum_{i=1}^n v_i = e$, and for all $i \in [n]$, $\|v_i\|_h \leq \epsilon_2$. Then there exists $S \subseteq [n]$ in the support of μ, such that

$$\left\| \sum_{i \in S} v_i \right\|_h \leq 4(\epsilon_1 + \epsilon_2) + 2(\epsilon_1 + \epsilon_2)^2.$$

It is worth mentioning that the previous paper [36, 6] focused on the determinant polynomial, leaving the question of whether their techniques could be extended to the hyperbolic/real-stable setting unresolved. In our paper, we address this gap by developing new techniques specifically tailored to hyperbolic polynomials.

In addition, we follow the results from [11] and give an algorithm that can find the approximate solutions of both Theorem 5 and Theorem 6 in time sub-exponential to m:

▶ **Proposition 7** (Sub-exponential algorithm for Theorem 5, informal). *Let $h \in \mathbb{R}[x_1, \ldots, x_m]$ denote a hyperbolic polynomial with direction $e \in \mathbb{R}^m$. Let $v_1, \ldots, v_n \in \Gamma_+^h$ be n vectors in the hyperbolicity cone Γ_+^h of h. Suppose $\sigma = \|\sum_{i=1}^n \operatorname{tr}_h[v_i] v_i\|_h$.*

Let $\mathcal{P}$ be the interlacing family used in the proof of Theorem 6. Then there exists an sub-exponential time algorithm KadisonSinger$(\delta, \mathcal{P})$*, such that for any $\delta > 0$, it returns a sign assignment $(s_1, \cdots, s_n) \in \{\pm 1\}^n$ satisfying*

$$\left\| \sum_{i=1}^{n} s_i u_i \right\|_h \leq 4(1+\delta)\sigma.$$

▶ **Proposition 8** (Sub-Exponential algorithm for Theorem 6, informal)**.** *Let $h \in \mathbb{R}[x_1, \ldots, x_m]$ denote a hyperbolic polynomial in direction $e \in \mathbb{R}^m$. Let μ be a homogeneous strongly Rayleigh probability distribution on $[n]$ such that the marginal probability of each element is at most ϵ_1, and let $v_1, \cdots, v_n \in \Gamma_+^h$ be n vectors such that $\sum_{i=1}^n v_i = e$, and for all $i \in [n]$, $\|v_i\|_h \leq \epsilon_2$.*

Let $\mathcal{Q}$ be the interlacing family used in the proof of Theorem 6. Then there exists an sub-exponential time algorithm KadisonSinger$(\delta, \mathcal{Q})$*, such that for any $\delta > 0$, it returns a set S in the support of μ satisfying*

$$\left\| \sum_{i \in S} u_i \right\|_h \leq (1+\delta) \cdot \left(4(\epsilon_1 + \epsilon_2) + 2(\epsilon_1 + \epsilon_2)^2\right).$$

2 Related work

Real-Stable Polynomials

Real-stability is an important property for multivariate polynomials. In [13], the authors used the real-stability to give a unified framework for Lee-Yang type problems in statistical mechanics and combinatorics. Real-stable polynomials are also related to the permanent. Gurvits [25] proved the Van der Waerden conjecture, which conjectures that the permanent of n-by-n doubly stochastic matrices are lower-bounded by $n!/n^n$, via the capacity of real-stable polynomials. Recently, [26] improved the capacity lower bound for real-stable polynomials, which has applications in matrix scaling and metric TSP. In addition, real-stable polynomials are an important tool in solving many counting and sampling problems [46, 9, 8, 58, 10, 5, 12, 3, 4].

Hyperbolic Polynomials

Hyperbolic polynomial was originally defined to study the stability of partial differential equations [23, 29, 34]. In theoretical computer science, Güler [24] first introduced hyperbolic polynomial for optimization (hyperbolic programming), which is a generalization of LP and SDP. Later, a few algorithms [50, 44, 53, 51, 45, 52] were designed for hyperbolic programming. On the other hand, a significant effort has been put into the equivalence between hyperbolic programming and SDP, which is closely related to the "Generalized Lax Conjecture" (which conjectures that every hyperbolicity cone is spectrahedral) and its variants [28, 40, 18, 35, 54, 2, 48].

Strongly Rayleigh Distribution

The strongly Rayleigh distribution was introduced by [14]. The authors also proved numerous basic properties of strongly Rayleigh distributions, including negative association, and closure property under operations such as conditioning, product, and restriction to a subset. [47] proved a concentration result for Lipschitz functions of strongly Rayleigh variables. [37] showed a matrix concentration for strongly Rayleigh random variables, which implies that adding a small number of uniformly random spanning trees gives a graph spectral sparsifier.

Strongly Rayleigh distribution also has many algorithmic applications. [9] exploited the negative dependence property of homogeneous strongly Rayleigh distributions, and designed efficient algorithms for generating approximate samples from Determinantal Point Process using Monte Carlo Markov Chain. The strongly Rayleigh property of spanning tree distribution is a key component for improving the approximation ratios of TSP [31, 32] and k-edge connected graph problem [33].

Other generalizations of the Kadison-Singer-type results

The upper bound of the rank-one Kadison-Singer theorem was improved by [15, 49]. [1] further extended [49]'s result to prove a real-stable version of Anderson's paving conjecture. However, they used a different norm for real-stable polynomials, and hence their results and ours are incomparable. In the high-rank case, [21] also proved a Kadison-Singer result for high-rank matrices. [56] relaxed [19]'s result to the vectors in sub-isotropic position. In addition, they proved a hyperbolic Spencer theorem for constant-rank vectors.

Another direction of generalizing the Kadison-Singer-type result is to relax the $\{+1,-1\}$-coloring to $\{0,1\}$-coloring, which is called the one-sided version of Kadison-Singer problem in [61]. More specifically, given n isotropic vectors $v_1,\ldots,v_n \in \mathbb{R}^m$ with norm $\frac{1}{\sqrt{N}}$, the goal is to find a subset $S \subset [n]$ of size k such that $\|\sum_{i\in S} v_i v_i^\top\| \leq \frac{k}{n} + O(1/\sqrt{N})$. Unlike the original Kadison-Singer problem, Weaver [61] showed that this problem can be solved in polynomial time. Very recently, Song, Xu and Zhang [55] improved the time complexity of the algorithm via an efficient inner product search data structure.

Applications of Kadison-Singer Problem

There are many interesting results developed from the Kadison-Singer theorem. In spectral graph theory, [27] exploited the same proof technique of interlacing families to show a sufficient condition of the spectrally thin tree conjecture. [6] used the strongly-Rayleigh extension of Kadison-Singer theorem to show a weaker sufficient condition. Based on this result, [7] showed that any k-edge-connected graph has an $O(\frac{\log\log(n)}{k})$-thin tree, and gave a $\mathrm{poly}(\log\log(n))$-integrality gap of the asymmetric TSP. [41, 22] used the Kadison-Singer theorem to construct bipartite Ramanujan graphs of all sizes and degrees. In the network design problem, [38] exploited the result in [36], and built a spectral rounding algorithm for the general network design convex program, which has applications in weighted experimental design, spectral network design, and additive spectral sparsifier.

3 Proof Overview

3.1 Hyperbolic Deviations

In this section, we will sketch the proof of our hyperbolic generalization of the Kadison-Singer theorem (Theorem 5). Details of the proof are deferred to the full version of the paper. We will use the same strategy as the original Kadison-Singer theorem (Theorem 1) in [42, 43], following three main technical steps.

For simplicity, we assume that the random variables $\xi_1,\ldots,\xi_n \in \{\pm 1\}$ are independent Rademacher random variables, i.e., $\Pr[\xi_i = 1] = \frac{1}{2}$ and $\Pr[\xi_i = -1] = \frac{1}{2}$ for all $i \in [n]$.

To generalize the Kadison-Singer statement into the hyperbolic norm, one main obstacle is to define the variance of the hyperbolic norm of the sum of random vectors $\sum_{i=1}^n \xi_i v_i$. In the determinant polynomial case, each v_i corresponds to a rank-1 matrix $u_i u_i^*$, and it is easy

to see that the variance of the spectral norm is $\|\sum_{i=1}^n (u_i u_i^*)^2\|$. However, there is no analog of "matrix square" in the setting of hyperbolic/real-stable polynomials. Instead, we define the *hyperbolic variance*:

$$\left\|\sum_{i=1}^n \mathrm{tr}_h[v_i]v_i\right\|_h$$

in terms of the hyperbolic trace, and show that *four hyperbolic deviations suffice*.

Defining interlacing family of characteristic polynomials

In the first step, we construct a family of *characteristic polynomials* $\{p_s : s \in \{\pm 1\}^t, t \in \{0, \cdots, n\}\}$ as follows: For each $\mathbf{s} \in \{\pm 1\}^n$, define the leaf-node-polynomial:

$$p_{\mathbf{s}}(x) := \left(\prod_{i=1}^n p_{i,s_i}\right) \cdot h\left(xe + \sum_{i=1}^n s_i v_i\right) \cdot h\left(xe - \sum_{i=1}^n s_i v_i\right),$$

and for all $\ell \in \{0, \ldots, n-1\}$, $\mathbf{s}' \in \{\pm 1\}^\ell$, we construct an inner node with a polynomial that corresponds to the bit-string $\mathbf{s}'$:

$$p_{\mathbf{s}'}(x) := \sum_{\mathbf{t} \in \{\pm 1\}^{n-\ell}} p_{(\mathbf{s}',\mathbf{t})}(x).$$

where $(\mathbf{s}', \mathbf{t}) \in \{\pm 1\}^n$ is the bit-string concatenated by $\mathbf{s}'$ and $\mathbf{t}$.

We will then show that the above family of characteristic polynomials forms an *interlacing family*. By basic properties of interlacing family, we can always find a leaf-root-polynomial p_s (where $s \in \{\pm 1\}^n$) whose largest root is upper bounded by the largest root of the top-most polynomial.

$$p_\emptyset(x) = \mathop{\mathbb{E}}_{\xi_1,\cdots,\xi_n}\left[h\Big(xe + \sum_{i=1}^n \xi_i v_i\Big) \cdot h\Big(xe - \sum_{i=1}^n \xi_i v_i\Big)\right].$$

(we call $p_\emptyset$ to be the *mixed characteristic polynomial*). Notice that by rewriting the largest root of p_s to be the expected hyperbolic norm of $\sum_{i=1}^n s_i v_i$, we get that

$$\lambda_{\max}(p_\emptyset) = \left\|\sum_{i=1}^n s_i v_i\right\|_h. \tag{1}$$

Also, we will take $s \in \{\pm 1\}^n$ as the corresponding sign assignment in the main theorem (Theorem 5) It then suffices to upper-bound the largest root of the mixed characteristic polynomial.

From mixed characteristic polynomial to multivariate polynomial

In the second step, we will show that the mixed characteristic polynomial that takes the average on n random variables

$$p_\emptyset(x) = \mathop{\mathbb{E}}_{\xi_1,\ldots,\xi_n}\left[h\Big(xe + \sum_{i=1}^n \xi_i v_i\Big) \cdot h\Big(xe - \sum_{i=1}^n \xi_i v_i\Big)\right]$$

is equivalent to a polynomial with n extra variables $z_1, \cdots, z_n$:

$$\prod_{i=1}^n \Big(1 - \frac{1}{2}\frac{\partial^2}{\partial z_i^2}\Big)\Big|_{z=0} \left(h\Big(xe + \sum_{i=1}^n z_i v_i\Big)\right)^2. \tag{2}$$

Thus, we can reduce the upper bound of $\chi_{\max}(p_\emptyset)$ to an upper bound of the largest root in (2). The latter turns out to be easier to estimate with the help of a barrier argument [43].

To show such equivalence holds, we use induction on the random variables $\xi_1, \ldots, \xi_n$. More specifically, we start from ξ_1 and are conditioned on any fixed choice of $\xi_2, \ldots, \xi_n$. We prove that taking expectation over ξ_1 is equivalent to applying the operator $(1 - \frac{\partial^2}{\partial z_1^2})$ to the polynomial

$$\left(h(xe + z_1 v_1 + \sum_{i=2}^{n} \xi_i v_i)\right)^2$$

and setting $z_1 = 0$. Here we use the relation between expectation and the second derivatives: for any Rademacher random variable ξ,

$$\mathop{\mathbb{E}}_{\xi}[h(x_1 - \xi v) \cdot h(x_2 + \xi v)] = \left(1 - \frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}t^2}\right)\Big|_{t=0} h(x_1 + tv)h(x_2 + tv).$$

Repeating this process and removing one random variable at a time. After n iterations, we obtain the desired multivariate polynomial.

We also need to prove the real-rootedness of the multivariate polynomial (Eqn. (2)). We first consider an easy case where h itself is a real-stable polynomial, as in the determinant polynomial case. Then the real-rootedness easily follows from the closure properties of the real-stable polynomial. More specifically, we can show that $(h(xe + \sum_{i=1}^n z_i v_i))^2$ is also a real-stable polynomial. Furthermore, applying the operators $(1 - \frac{1}{2}\frac{\partial^2}{\partial z_i^2})$ and restricting $z = 0$ preserve the real-stability. Therefore, the multivariate polynomial is a univariate real-stable polynomial, which is equivalent to being real-rooted.

Next, we show that when h is a hyperbolic polynomial, the multivariate polynomial (Eqn. (2)) is also real-rooted. our approach is to show that the linear restriction of h: $h(xe + \sum_{i=1}^n z_i v_i)$ is a real-stable polynomial in $\mathbb{R}[x, z_1, \ldots, z_n]$. A well-known test for real-stability is that if for any $a \in \mathbb{R}^{n+1}_{>0}, b \in \mathbb{R}^{n+1}$, the one-dimensional restriction $p(at+b) \in \mathbb{R}[t]$ is non-zero and real-rooted, then $p(x)$ is real-stable. We test $h(xe + \sum_{i=1}^n z_i v_i)$ by restricting to $at + b$, and get the following polynomial:

$$h\Big((a_1 e + \sum_{i=1}^{n} a_{i+1} v_i)t + y\Big) \in \mathbb{R}[t],$$

where y is a fixed vector depending on b. Since $a_i > 0$ for all $i \in [n+1]$ and $e, v_1, \ldots, v_n$ are vectors in the hyperbolicity cone, it implies that the vector $a_1 e + \sum_{i=1}^n a_{i+1} v_i$ is also in the hyperbolicity cone. Then, by the definition of hyperbolic polynomial, we immediately see that $h((a_1 e + \sum_{i=1}^n a_{i+1} v_i)t + y)$ is real-rooted for any $a \in \mathbb{R}^{n+1}_{>0}$ and $b \in \mathbb{R}^{n+1}$. Hence, we can conclude that the restricted hyperbolic polynomial $h(xe + \sum_{i=1}^n z_i v_i)$ is real-stable and the remaining proof is the same as the real-stable case.

Applying barrier argument

Finally, we use *barrier argument* to find an "upper barrier vector" whose components lie above any roots of multivariate polynomial can take. In particular, we consider the multivariate polynomial $P(x, z) = (h(xe + \sum_{i=1}^n z_i v_i))^2$. Define the *barrier function* of any variable $i \in [n]$ as the following:

$$\Phi_P^i(\alpha(t), -\delta) = \frac{\partial_{z_i} P(x, z)}{P(x, z)}\Big|_{x=\alpha(t), z=-\delta},$$

where $\delta \in \mathbb{R}^n$ where $\delta_i = t \operatorname{tr}_h[v_i]$ for $i \in [n]$ and $\alpha(t) > t$ is a parameter that depends on t.

As a warm-up, consider the case when $\sigma = 1$ and assuming $\|\sum_{i=1}^n \mathrm{tr}_h[v_i]v_i\|_h \le 1$. It is easy to show that $(\alpha(t), -\delta)$ is an upper barrier of P, from the linearity of the hyperbolic eigenvalues and the assumption. Next, we upper-bound the barrier function's value at $(\alpha(t), -\delta)$. When h is a determinant polynomial, this step is easy because the derivative of $\log\det$ is the trace of the matrix. For a general hyperbolic polynomial, we will rewrite the partial derivative ∂_{z_i} as a directional derivative D_{v_i} and get

$$\Phi_P^i(\alpha(t), -\delta) = 2 \cdot \frac{(D_{v_i}h)(\alpha e - te + t(e - \sum_{j=1}^n \mathrm{tr}_h[v_j]v_j))}{h(\alpha e - te + t(e - \sum_{j=1}^n \mathrm{tr}_h[v_j]v_j))}.$$

We observe that our assumption $\|\sum_{i=1}^n \mathrm{tr}_h[v_i]v_i\|_h \le 1$ implies that $e - \sum_{j=1}^n \mathrm{tr}_h[v_j]v_j \in \Gamma_+^h$. By the concavity of the function $\frac{h(x)}{D_{v_i}h(x)}$ in the hyperbolicity cone, we can prove that

$$\Phi_P^i(\alpha(t), -\delta) \le \frac{2\,\mathrm{tr}_h[v_i]}{\alpha(t) - t}.$$

Now, we can apply the barrier update lemma in [36] with $\alpha(t) = 2t = 4$ to show that

$$\Phi^j_{(1-\frac{1}{2}\partial^2_{z_i})P}(4, -\delta + \delta_i \mathbf{1}_i) \le \Phi_P^j(4, -\delta).$$

In other words, the partial differential operator $(1 - \frac{1}{2}\partial_{z_i}^2)$ shifts the upper-barrier by $(0, \cdots, 0, \delta_i, 0, \cdots, 0)$. Using induction for the variables $\delta_1, \cdots, \delta_n$, we can finally finally get an upper-barrier of

$$(4, -\delta + \sum_{i=1}^n \delta_i \mathbf{1}_i) = (4, 0, \ldots, 0),$$

which implies that $(4, 0, \ldots, 0)$ is above the roots of

$$\prod_{i=1}^n \Big(1 - \frac{1}{2}\frac{\partial^2}{\partial z_i^2}\Big) \left(h\Big(xe + \sum_{i=1}^n z_i\tau_i v_i\Big)\right)^2 \tag{3}$$

A challenge in this process is ensuring that the barrier function remains nonnegative. To achieve this, we use the multidimensional convexity of the hyperbolic barrier function as established in [59]. For cases where $\sigma \neq 1$, this requirement is satisfied through a simple scaling argument.

Combining the above three steps together, we can prove that $\Pr_{\xi_1,\cdots,\xi_n}[\|\sum_{i=1}^n \xi_i v_i\|_h \le 4\sigma] > 0$ for vectors $v_1, \ldots, v_n$ in the hyperbolicity cone with $\|\sum_{i=1}^n \mathrm{tr}_h[v_i]v_i\|_h = \sigma^2$.

3.2 Generalization to Strongly Rayleigh Distributions

Our main technical contribution to Theorem 6 is a more universal and structured method to characterize the mixed characteristic polynomial. Define the mixed characteristic polynomial as

$$q_S(x) = \mu(S) \cdot h\left(xe - \sum_{i\in S} v_i\right). \tag{4}$$

we want to show that it is equivalent to the restricted multivariate polynomial:

$$\prod_{i=1}^n (1 - \frac{1}{2}\frac{\partial^2}{\partial z_i^2})\Big(h(xe + \sum_{i=1}^n z_i v_i) g_\mu(x\mathbf{1} + z)\Big)\Big|_{z=0} \in \mathbb{R}[x, z_1, \cdots, z_n]. \tag{5}$$

Although Eqn. (4) and Eqn. (5) are the hyperbolic generalization of [6], we are unable to apply the previous techniques. This is because [6] computes the mixed characteristic polynomial explicitly, which heavily relies on the fact that the characteristic polynomial is a determinant. It is unclear how to generalize this method to hyperbolic/real-stable characteristic polynomials.

The key step in [6] is to show the following equality between mixed characteristic polynomial and multivariate polynomial:

$$\begin{aligned}&x^{d_\mu-d}\cdot \mathop{\mathbb{E}}_{S\sim\mu}\left[\det\left(x^2I-\sum_{i\in S}2v_iv_i^\top\right)\right]\\&=\prod_{i=1}^n(1-\partial_{z_i}^2)\left(g_\mu(x\mathbf{1}+z)\cdot\det(xI+\sum_{i=1}^n z_iv_iv_i^\top)\right)\Big|_{z=\mathbf{0}}\end{aligned}$$

where d_μ is the degree of the homogeneous strongly-Rayleigh distribution μ (i.e. the degree of g_μ), and m is the dimension of v_i.

Then they expand the right-hand side to get:

$$\begin{aligned}\text{RHS}&=\sum_{k=0}^m(-1)^kx^{d_\mu+m-2k}\sum_{S\in\binom{[n]}{k}}\Pr_{T\sim\mu}[S\subseteq T]\cdot\sigma_k(\sum_{i\in S}2v_iv_i^\top)\\&=x^{d_\mu-m}\cdot\mathop{\mathbb{E}}_{S\sim\mu}\left[\det\left(x^2I-\sum_{i\in S}2v_iv_i^\top\right)\right]=\text{LHS},\end{aligned}$$

where $\sigma_k(M)$ equals to the sum of all $k\times k$ principal minors of $M\in\mathbb{R}^{m\times m}$. The first step comes from expanding the product $\prod_{i=1}^n(1-\partial^2z_i)$, and the second step comes from that

$$\det(x^2I-\sum_{i=1}^n v_iv_i^\top)=\sum_{k=0}^m(-1)^{2k}x^{2m-2k}\sum_{S\in\binom{[n]}{k}}\sigma_k(\sum_{i\in S}v_iv_i^\top).$$

The naive generalization of a technique to hyperbolic/real-stable polynomial h faces challenges. One such challenge is the absence of an explicit form for h, unlike in the case of $h=\det$ where the determinant can be expressed as a combination of minors. This lack of a well-defined minor presents difficulty in rewriting the hyperbolic/real-stable polynomial. To tackle this issue, we devised a new and structured proof that relies on induction, offering a novel solution to this problem.

Inductive step

We first rewrite the expectation over the Strongly-Rayleigh distribution $T\sim\mu$ as follows:

$$\begin{aligned}x^{d_\mu}\cdot2^{-n}\cdot\mathop{\mathbb{E}}_{T\sim\mu}[h(xe-\sum_{i\in T}v_i)]=\frac{1}{2}\mathop{\mathbb{E}}_{\xi_2,\cdots,\xi_n\sim\{0,1\}^{n-1}}\Big[&(1-\partial_{z_1})h(x_2+z_1v_1)x\partial_{z_1}g_2(x+z_1)\\&+h(x_2)(1-x\partial_{z_1})g_2(x+z_1)\Big|_{z_1=0}\Big]\end{aligned}$$

where g_2 is defined as

$$\begin{aligned}g_2(t):=x^{\sum_{i=2}^n\xi_i}&\cdot\\&\prod_{i=2}^n\Big(\xi_i\partial_{z_i}+(1-\xi_i)(1-x\partial_{z_i})\Big)g_\mu(t,x+z_2,x+z_3,\cdots,x+z_n)\Big|_{z_2,\ldots,z_n=0}\end{aligned}$$

and $x_2 = x^2 e - \sum_{i=2}^{n} \xi_i v_i$. The main observation is that the marginals of a homogeneous Strongly-Rayleigh distribution can be computed from the derivatives of its generating polynomial.

Then, we can expand the term inside the expectation as

$$(1 - \frac{x}{2}\partial_{z_1}^2)\Big(h(x_2 + z_1 v_1) g_2(x + z_1)\Big)\Big|_{z_1=0},$$

using the fact that $\mathrm{rank}(v_1)_h \le 1$ and the degree of $g_2(t)$ is at most 1.

Hence, we obtain our inductive step as

$$\begin{aligned} & x^{d_\mu} \cdot 2^{-n} \cdot \mathop{\mathbb{E}}_{\xi\sim\mu}\left[h(xe - \sum_{i=1}^{n} \xi_i v_i)\right] \\ = & \frac{1}{2}(1 - \frac{x}{2}\partial_{z_1}^2)\left(\mathop{\mathbb{E}}_{\xi_2,\cdots,\xi_n}\left[h(xe - \sum_{i=2}^{n} \xi_i v_i + z_1 v_1) \cdot g_2(x + z_1)\right]\right)\Big|_{z_1=0}. \end{aligned}$$

Applying the step inductively

Repeating the above process for n times, we finally get

$$x^{d_\mu} \cdot \mathop{\mathbb{E}}_{\xi\sim\mu}\left[h(x^2 e - (\sum_{i=1}^{n} \xi_i v_i))\right] = \sum_{T\subseteq[n]} (-\frac{x}{2})^{|T|} \partial_{z^T}^2 \Big(h(x^2 e + \sum_{i=1}^{n} z_i v_i) g_\mu(x\mathbf{1} + z)\Big)\Big|_{z=0}.$$

Then, we rewrite the partial derivatives as directional derivatives. For any subset $T \subseteq [n]$ of size k, we have

$$\begin{aligned} & (-\frac{x}{2})^k \partial_{z^T}^2 \Big(h(x^2 e + \sum_{i=1}^{n} z_i v_i) g_\mu(x\mathbf{1} + z)\Big)\Big|_{z=0} \\ = & (-\frac{x}{2})^k \cdot 2^k \cdot \left(\prod_{i\in T} D_{v_i}\right) h(x^2 e) \cdot g_\mu^{(T)}(x\mathbf{1}), \end{aligned}$$

where $g_\mu^{(T)}(x\mathbf{1}) = \prod_{i\in T} \partial_{z_i} g_\mu(x\mathbf{1} + z)\Big|_{z=0}$. And by the homogeneity of h, it further equals to

$$x^d \cdot (-\frac{1}{2})^k \partial_{z^T}^2 \left(h(xe + \sum_{i=1}^{n} z_i v_i) g_\mu(x\mathbf{1} + z)\right)\Big|_{z=0}.$$

Therefore, we prove the following formula that relates the characteristic polynomial under SR distribution to the multivariate polynomial:

$$x^{d_\mu} \cdot \mathop{\mathbb{E}}_{\xi\sim\mu}\left[h(x^2 e - (\sum_{i=1}^{n} \xi_i v_i))\right] = x^d \cdot \prod_{i=1}^{n} (1 - \frac{1}{2}\partial_{z_i}^2)\Big(h(xe + \sum_{i=1}^{n} z_i v_i) g_\mu(x\mathbf{1} + z)\Big)\Big|_{z=0}.$$

References

1. Kasra Alishahi and Milad Barzegar. Paving property for real stable polynomials and strongly rayleigh processes. *arXiv preprint*, 2020. `arXiv:2006.13923`.
2. Nima Amini. Spectrahedrality of hyperbolicity cones of multivariate matching polynomials. *Journal of Algebraic Combinatorics*, 50(2):165–190, 2019.
3. Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials ii: high-dimensional walks and an fpras for counting bases of a matroid. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1–12, 2019.

4 Nima Anari, Kuikui Liu, Shayan Oveis Gharan, Cynthia Vinzant, and Thuy-Duong Vuong. Log-concave polynomials iv: approximate exchange, tight mixing times, and near-optimal sampling of forests. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 408–420, 2021.

5 Nima Anari, Tung Mai, Shayan Oveis Gharan, and Vijay V Vazirani. Nash social welfare for indivisible items under separable, piecewise-linear concave utilities. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2274–2290. SIAM, 2018.

6 Nima Anari and Shayan Oveis Gharan. The kadison-singer problem for strongly rayleigh measures and applications to asymmetric tsp. In *arXiv preprint*. `https://arxiv.org/pdf/1412.1143.pdf`, 2014.

7 Nima Anari and Shayan Oveis Gharan. Effective-resistance-reducing flows, spectrally thin trees, and asymmetric tsp. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 20–39. IEEE, 2015.

8 Nima Anari and Shayan Oveis Gharan. A generalization of permanent inequalities and applications in counting and optimization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 384–396, 2017.

9 Nima Anari, Shayan Oveis Gharan, and Alireza Rezaei. Monte carlo markov chain algorithms for sampling strongly rayleigh distributions and determinantal point processes. In *Conference on Learning Theory*, pages 103–115. PMLR, 2016.

10 Nima Anari, Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. Nash social welfare, matrix permanent, and stable polynomials. *arXiv preprint*, 2016. `arXiv:1609.07056`.

11 Nima Anari, Shayan Oveis Gharan, Amin Saberi, and Nikhil Srivastava. Approximating the largest root and applications to interlacing families. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1015–1028. SIAM, 2018.

12 Nima Anari, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials, entropy, and a deterministic approximation algorithm for counting bases of matroids. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 35–46. IEEE, 2018.

13 Julius Borcea and Petter Brändén. The lee-yang and pólya-schur programs. ii. theory of stable polynomials and applications. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 62(12):1595–1631, 2009.

14 Julius Borcea, Petter Brändén, and Thomas Liggett. Negative dependence and the geometry of polynomials. *Journal of the American Mathematical Society*, 22(2):521–567, 2009.

15 Marcin Bownik, Pete Casazza, Adam W Marcus, and Darrin Speegle. Improved bounds in weaver and feichtinger conjectures. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 2019(749):267–293, 2019.

16 Petter Brändén. Notes on hyperbolicity cones. *Verfügbar unter https://math. berkeley. edu/˜ bernd/branden. pdf*, 2010.

17 Petter Brändén. Obstructions to determinantal representability. *Advances in Mathematics*, 226(2):1202–1212, 2011.

18 Petter Brändén. Hyperbolicity cones of elementary symmetric polynomials are spectrahedral. *Optimization Letters*, 8(5):1773–1782, 2014.

19 Petter Brändén. Hyperbolic polynomials and the kadison-singer problem. *arXiv preprint*, 2018. `arXiv:1809.03255`.

20 Sam Burton, Cynthia Vinzant, and Yewon Youm. A real stable extension of the vamos matroid polynomial. *arXiv preprint*, 2014. `arXiv:1411.2038`.

21 Michael Cohen. Improved spectral sparsification and Kadison-Singer for sums of higher-rank matrices. In *Banff International Research Station for Mathematical Innovation and Discovery*. `https://open.library.ubc.ca/cIRcle/collections/48630/items/1.0340957`, 2016.

22 Michael B Cohen. Ramanujan graphs in polynomial time. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 276–281. IEEE, 2016.

23 Lars Gårding. Linear hyperbolic partial differential equations with constant coefficients. *Acta Mathematica*, 85:1–62, 1951.

24 Osman Güler. Hyperbolic polynomials and interior point methods for convex programming. *Mathematics of Operations Research*, 22(2):350–377, 1997.

25 Leonid Gurvits. Van der waerden/schrijver-valiant like conjectures and stable (aka hyperbolic) homogeneous polynomials: one theorem for all. *arXiv preprint*, 2007. `arXiv:0711.3496`.

26 Leonid Gurvits and Jonathan Leake. Capacity lower bounds via productization. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 847–858, 2021.

27 Nicholas JA Harvey and Neil Olver. Pipage rounding, pessimistic estimators and matrix concentration. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 926–945. SIAM, 2014.

28 J William Helton and Victor Vinnikov. Linear matrix inequality representation of sets. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 60(5):654–674, 2007.

29 L Hormander. The analysis of linear partial differential operators ii. *Grundlehren*, 257, 1983.

30 Richard V Kadison and Isadore M Singer. Extensions of pure states. *American journal of mathematics*, 81(2):383–400, 1959.

31 Anna R Karlin, Nathan Klein, and Shayan Oveis Gharan. An improved approximation algorithm for tsp in the half integral case. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 28–39, 2020.

32 Anna R Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved approximation algorithm for metric tsp. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 32–45, 2021.

33 Anna R Karlin, Nathan Klein, Shayan Oveis Gharan, and Xinzhi Zhang. An improved approximation algorithm for the minimum k-edge connected multi-subgraph problem. *arXiv preprint*, 2021. `arXiv:2101.05921`.

34 N.V. Krylov. On the general notion of fully nonlinear second-order elliptic equations. *Transactions of the American Mathematical Society*, 347(3):857–895, 1995.

35 Mario Kummer, Daniel Plaumann, and Cynthia Vinzant. Hyperbolic polynomials, interlacers, and sums of squares. *Mathematical Programming*, 153(1):223–245, 2015.

36 Rasmus Kyng, Kyle Luh, and Zhao Song. Four deviations suffice for rank 1 matrices. In *Advances in Mathematics*. arXiv preprint arXiv:1901.06731, 2020.

37 Rasmus Kyng and Zhao Song. A matrix chernoff bound for strongly rayleigh distributions and spectral sparsifiers from a few random spanning trees. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 373–384. IEEE, 2018.

38 Lap Chi Lau and Hong Zhou. A spectral approach to network design. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 826–839, 2020.

39 Peter D Lax. Differential equations, difference equations and matrix theory. Technical report, New York Univ., New York. Atomic Energy Commission Computing and Applied Mathematics Center, 1957.

40 Adrian Lewis, Pablo Parrilo, and Motakuri Ramana. The lax conjecture is true. *Proceedings of the American Mathematical Society*, 133(9):2495–2499, 2005.

41 A. Marcus, D. Spielman, and N. Srivastava. Interlacing families iv: Bipartite ramanujan graphs of all sizes. *SIAM Journal on Computing*, 47(6):2488–2509, 2018. `doi:10.1137/16M106176X`.

42 Adam W. Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing families I: Bipartite Ramanujan graphs of all degrees. *Ann. of Math. (2)*, 182(1):307–325, 2015.

43 Adam W. Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing families II: Mixed characteristic polynomials and the Kadison-Singer problem. *Ann. of Math. (2)*, 182(1):327–350, 2015.

44 Tor Myklebust and Levent Tunçel. Interior-point algorithms for convex optimization based on primal-dual metrics. *arXiv preprint*, 2014. `arXiv:1411.2129`.

45 Simone Naldi and Daniel Plaumann. Symbolic computation in hyperbolic programming. *Journal of Algebra and Its Applications*, 17(10):1850192, 2018.

46 Aleksandar Nikolov and Mohit Singh. Maximizing determinants under partition constraints. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 192–201, 2016.

47 Robin Pemantle and Yuval Peres. Concentration of lipschitz functionals of determinantal and other strong rayleigh measures. *Combinatorics, Probability and Computing*, 23(1):140–160, 2014.

48 Prasad Raghavendra, Nick Ryder, Nikhil Srivastava, and Benjamin Weitz. Exponential lower bounds on spectrahedral representations of hyperbolicity cones. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2322–2332. SIAM, 2019.

49 Mohan Ravichandran and Jonathan Leake. Mixed determinants and the kadison–singer problem. *Mathematische Annalen*, 377(1):511–541, 2020.

50 James Renegar. Hyperbolic programs, and their derivative relaxations. *Foundations of Computational Mathematics*, 6(1):59–79, 2006.

51 James Renegar. "Efficient" subgradient methods for general convex optimization. *SIAM Journal on Optimization*, 26(4):2649–2676, 2016.

52 James Renegar. Accelerated first-order methods for hyperbolic programming. *Mathematical Programming*, 173(1-2):1–35, 2019.

53 James Renegar and Mutiara Sondjaja. A polynomial-time affine-scaling method for semidefinite and hyperbolic programming. *arXiv preprint*, 2014. `arXiv:1410.6734`.

54 James Saunderson. A spectrahedral representation of the first derivative relaxation of the positive semidefinite cone. *Optimization Letters*, 12(7):1475–1486, 2018.

55 Zhao Song, Zhaozhuo Xu, and Lichen Zhang. Speeding up sparsification using inner product search data structures, 2022. `arXiv:2204.03209`.

56 Zhao Song and Ruizhe Zhang. Hyperbolic concentration, anti-concentration, and discrepancy. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

57 Joel Spencer. Six standard deviations suffice. *Transactions of the American mathematical society*, 289(2):679–706, 1985.

58 Damian Straszak and Nisheeth K Vishnoi. Real stable polynomials and matroids: Optimization and counting. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 370–383, 2017.

59 Terence Tao. Real stable polynomials and the kadison-singer problem. URL: `https://terrytao.wordpress.com/2013/11/04/real-stable-polynomials-and-the-kadison-singer-problem/`, 2013.

60 Joel A. Tropp. An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, 8(1-2):1–230, 2015. `doi:10.1561/2200000048`.

61 Nik Weaver. The Kadison-Singer problem in discrepancy theory. *Discrete Math.*, 278(1-3):227–239, 2004.

On Semantically-Deterministic Automata

Bader Abu Radi
School of Computer Science and Engineering, Hebrew University, Jerusalem, Israel

Orna Kupferman
School of Computer Science and Engineering, Hebrew University, Jerusalem, Israel

Abstract

Nondeterminism is a fundamental notion in Theoretical Computer Science. A nondeterministic automaton is *semantically deterministic* (SD) if different nondeterministic choices in the automaton lead to equivalent states. Semantic determinism is interesting as it is a natural relaxation of determinism, and as some applications of automata in formal methods require deterministic automata, yet in fact can use automata with some level of nondeterminism, tightly related to semantic determinism.

In the context of finite words, semantic determinism coincides with determinism, in the sense that every pruning of an SD automaton to a deterministic one results in an equivalent automaton. We study SD automata on infinite words, focusing on Büchi, co-Büchi, and weak automata. We show that there, while semantic determinism does not increase the expressive power, the combinatorial and computational properties of SD automata are very different from these of deterministic automata. In particular, SD Büchi and co-Büchi automata are exponentially more succinct than deterministic ones (in fact, also exponentially more succinct than history-deterministic automata), their complementation involves an exponential blow up, and decision procedures for them like universality and minimization are PSPACE-complete. For weak automata, we show that while an SD weak automaton need not be pruned to an equivalent deterministic one, it can be determinized to an equivalent deterministic weak automaton with the same state space, implying also efficient complementation and decision procedures for SD weak automata.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Automata on infinite words, Nondeterminism, Succinctness, Decision procedures

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.109

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `https://arxiv.org/abs/2305.15489` [2]

Funding This research is supported by the Israel Science Foundation, Grant 2357/19, the European Research Council, Advanced Grant ADVANSYNT, and the Neubauer Foundation of Ph.D Fellowship.

1 Introduction

Automata are among the most studied computation models in theoretical computer science. Their simple structure has made them a basic formalism for the study of fundamental notions, such as *determinism* and *nondeterminism* [35]. While a deterministic computing machine examines a single action at each step of its computation, nondeterministic machines are allowed to examine several possible actions simultaneously. Understanding the power of nondeterminism is at the core of open fundamental questions in theoretical computer science (most notably, the P vs. NP problem).

A prime application of *automata on infinite words* is specification, verification, and synthesis of nonterminating systems. The automata-theoretic approach reduces questions about systems and their specifications to questions about automata [28, 41], and is at the heart of many algorithms and tools. A run of an automaton on infinite words is an infinite

sequence of states, and acceptance is determined with respect to the set of states that the run visits infinitely often. For example, in *Büchi* automata, some of the states are designated as accepting states, and a run is accepting iff it visits states from the set α of accepting states infinitely often [9]. Dually, in *co-Büchi* automata, a run is accepting if it visits the set α only finitely often. Then, *weak* automata are a special case of both Büchi and co-Büchi automata in which every strongly connected component in the graph induced by the automaton is either contained in α or is disjoint from α. We use DBW and NBW to denote determinisitic and nondeterministic Büchi word automata, respectively, and similarly for D/NCW, D/NWW, and D/NFW, for co-Büchi, weak, and automata on finite words, respectively.

For automata on infinite words, nondeterminism not only leads to exponential succinctness, but may also increase the expressive power. This is the case, for example, in Büchi and weak automata, thus NBWs are strictly more expressive than DBWs [29], and NWWs are strictly more expressive than DWWs [6]. On the other hand, NCWs are as expressive as DCWs [32], and in fact, also as NWWs [27]. In some applications of the automata-theoretic approach, such as model checking, algorithms can be based on nondeterministic automata, whereas in other applications, such as synthesis and reasoning about probabilistic systems, they cannot. There, the advantages of nondeterminism are lost, and algorithms involve a complicated determinization construction [36] or acrobatics for circumventing determinization [26, 21].

In a deterministic automaton, the transition function maps each state and letter to a single successor state. In recent years there is growing research on weaker types of determinism. This includes, for example, *unambiguous automata*, which may have many runs on each word, yet only one accepting run [10, 12], automata that are *deterministic in the limit*, where each accepting run should eventually reach a deterministic sub-automaton [40], and automata that are *determinizable by pruning* (DBP), thus embody an equivalent deterministic automaton [4].

In terms of applications, some weaker types of determinism have been defined and studied with specific applications in mind. Most notable are *history-deterministic* automata (HD), which can resolve their nondeterministic choices based on the history of the run [18, 8, 23][1], and can therefore replace deterministic automata in algorithms for synthesis and control, and *good-for-MDPs* automata (GFM), whose product with Markov decision processes maintains the probability of acceptance, and can therefore replace deterministic automata when reasoning about stochastic behaviors [16, 39].

The different levels of determinism induce classes of automata that differ in their succinctness and in the complexity of operations and decision problems on them. Also, some classes are subclasses of others. For example, it follows quite easily from the definitions that every automaton that is deterministic in the limit is GFM, and every automaton that is DBP is HD.

In this paper we study the class of *semantically deterministic* automata (SD). An automaton $\mathcal{A}$ is SD if its nondeterministic choices lead to equivalent states. Formally, if $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, with a transition function $\delta : Q \times \Sigma \to 2^Q$, then $\mathcal{A}$ is SD if for every state $q \in Q$, letter $\sigma \in \Sigma$, and states $q_1, q_2 \in \delta(q, \sigma)$, the set of words accepted from $\mathcal{A}$ with initial state q_1 is equal to the set of words accepted from $\mathcal{A}$ with initial state q_2. Since all nondeterministic choices lead to equivalent states, one may be tempted to think that SD automata are DBP or at least have similar properties to deterministic automata. This is

[1] The notion used in [18] is *good for games* (GFG) automata, as they address the difficulty of playing games on top of a nondeterministic automaton. As it turns out, the property of being good for games varies in different settings and HD is good for applications beyond games. Therefore, we use the term *history determinism*, introduced by Colcombet in the setting of quantitative automata with cost functions [11].

indeed the case for SD-NFWs, namely when one considers automata on finite words. There, it is not hard to prove that any pruning of an SD-NFW to a DFW results in an equivalent DFW. Thus, SD-NFWs are not more succinct than DFWs, and operations on them are not more complex than operations on DFWs.

Once, however, one considers automata on infinite words, the simplicity of SD automata is lost. In order to understand the picture in the setting of infinite words, let us elaborate some more on HD automata, which are strongly related to SD automata. Formally, a nondeterministic automaton $\mathcal{A}$ is HD if there is a strategy g that maps each finite word $u \in \Sigma^*$ to a transition to be taken after u is read, and following g results in accepting all the words in the language of $\mathcal{A}$. Obviously, every DBP automaton is HD – the strategy g can suggest the same transition in all its visits in a state. On the other hand, while HD-NWWs are always DBP [25, 33], this is not the case for HD-NBWs and HD-NCWs [7]. There, the HD strategy may need to suggest different transitions in visits (with different histories) to the same state.

It is easy to see that a strategy g as above cannot choose a transition to states whose language is strictly contained in the language of states that are reachable by other transitions. Thus, all HD automata can be pruned in polynomial time to SD automata [20, 5]. The other direction, however, does not hold: it is shown in [3] that an SD-NWW need not be HD (hence, SD-NBWs and SD-NCWs need not be HD too). Moreover, while all all HD-NWWs are DBP, this is not the case for all SD-NWWs.

In this work we study the succinctness of SD automata with respect to deterministic ones, as well as the complexity of operations on them and decision problems about them. Our goal is to understand the difference between determinism and semantic determinism, and to understand how this difference varies among different acceptance conditions. Our study is further motivated by the applications of automata with different levels of nondeterminism in algorithms for synthesis and for reasoning in a stochastic setting. In particular, beyond the connection to HD automata discussed above, as runs of an SD-NBW on words in the language are accepting with probability 1, all SD-NBWs are GFM [3].

We study semantic determinism for Büchi, co–Büchi, and weak automata. We consider automata with both *state-based* acceptance conditions, as defined above, and *transition-based* acceptance conditions. In the latter, the acceptance condition is given by a subset α of transitions, and a run is required to traverse transitions in α infinitely often (in Büchi automata, termed tD/tNBW) or finitely often (in co-Büchi automata, termed tD/tNCW). As it turns out, especially in the context of HD automata, automata with transition-based acceptance conditions may differ in their properties from automata with traditional state-based acceptance conditions. For example, while HD-tNCWs can be minimized in PTIME [1], minimization of HD-NCWs is NP-complete [38]. In addition, there is recently growing use of transition-based automata in practical applications, with evidences they offer a simpler translation of LTL formulas to automata and enable simpler constructions and decision procedures [14, 15, 13, 40, 30]. Our results for all types of acceptance conditions are summarized in Table 1 below, where we also compare them with known results about deterministic and HD automata.

Let us highlight the results we find the most interesting and surprising. While all three types of SD automata are not DBP, we are able to determinize SD-NWWs in polynomial time, and end up with a DWW whose state space is a subset of the state space of the original SD-NWW. Essentially, rather than pruning transitions, the construction redirects transitions to equivalent states in deep strongly connected components of the SD-NWW, which we prove

Table 1 Succinctness (determinization blow-up), complementation (blow-up in going from an automaton to a complementing one), universality (deciding whether an automaton accepts all words), and minimization (deciding whether an equivalent automaton of a given size exists, and the described results apply also for the case the given automaton is deterministic). All blow-ups are tights, except for HD-NBW determinization, where the quadratic bound has no matching lower bound; all NL, NP, and PSPACE bounds are complete.

	Deterministic	HD	SD
Succinctness	n (B,C,W)	n^2 (B), 2^n (C), n (W) [20, 25, 33]	2^n (B,C), n (W) Theorems 5, 14, 22
Complementation	n (B,C,W) [22]	n (B), 2^n (C), n (W) [20]	2^n (B,C), n (W) Theorems 7, 16, 23
Universality	NL (B,C,W) [22]	P (B,C,W) [17, 20]	PSPACE (B,C), P (W) Theorems 9, 17, 24
Minimization (state based)	NP (B,C), P (W) [37, 31]	NP (B,C), P (W) [38, 25, 31]	PSPACE (B,C), P (W) Theorems 10, 18, 24
Minimization (transition based)	open (B,C), P (W) [31]	open (B), P (C,W) [1, 25, 31]	PSPACE (B,C), P (W) Theorems 10, 18, 24

to result in an equivalent deterministic DWW[2]. This suggests that despite the "not DBP anomaly" of SD-NWWs, they are very similar in their properties to DWWs. On the other hand, except for their expressive power, SD-NBWs and SD-NCWs are not at all similar to DBWs and DCWs: while HD-NBWs are only quadratically more succinct than DBWs, succinctness jumps to exponential for SD-NBWs. This also shows that SD-NBWs may be exponentially more succinct than HD-NBWs, and we show this succinctness gap also for co-Büchi automata, where exponential succinctness with respect to DCWs holds already in the HD level.

The succinctness results are carried over to the blow-up needed for complementation, and to the complexity of decision procedures. Note that this is not the case for HD automata. There, for example, complementation of HD-NBWs results in an automaton with the same number of states [20]. Moreover, even though HD-NCWs are exponentially more succinct than DCWs, language-containment for HD-NCWs can be solved in PTIME [17, 20], and HD-tNCWs can be minimized in PTIME [1]. For SD automata, we show that complementation involves an exponential blow-up. Also, universality and minimization of SD Büchi and co-Büchi automata, with either state-based or transition-based acceptance conditions, is PSPACE-complete, as it is for NBWs and NCWs. We also study the *D-to-SD minimization problem*, where we are given a deterministic automaton $\mathcal{A}$ and a bound $k \geq 1$, and need to decide whether $\mathcal{A}$ has an equivalent SD automaton with at most k states. Thus, the given automaton is deterministic, rather than SD. By [19], the D-to-N minimization problem for automata on finite words (that is, given a DFW, minimize it to an NFW) is PSPACE-complete. It is easy to see that the D-to-SD minimization problem for for automata on finite words can be solved in PTIME. We show that while this is the case also for weak automata, D-to-SD minimization for Büchi and co-Büchi automata is PSPACE-complete.

[2] Our result implies that checking language-equivalence between states in an SD-NWW can be done in PTIME, as the check can be performed on the equivalent DWWs. We cannot, however, use this complexity result in our algorithm, as this involves a circular dependency. Consequently, our construction of the equivalent DWWs involves a language-approximation argument.

Our results show that in terms of combinatorial and computational properties, semantic determinism is very similar to determinism in weak automata, whereas for Büchi and co-Büchi automata, semantic determinism is very similar to nondeterminism.

Due to the lack of space, some proofs are missing, and can be found in the full version [2].

2 Preliminaries

2.1 Languages and Automata

For a finite nonempty alphabet Σ, an infinite *word* $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$ is an infinite sequence of letters from Σ. A *language* $L \subseteq \Sigma^\omega$ is a set of words. For $1 \leq i \leq j$, we use $w[i,j]$ to denote the infix $\sigma_i \cdot \sigma_{i+1} \cdots \sigma_j$ of w, use $w[i]$ to denote the letter σ_i, and use $w[i,\infty]$ to denote the infinite suffix $\sigma_i \cdot \sigma_{i+1} \cdots$ of w. We also consider languages $R \subseteq \Sigma^*$ of finite words, denote the empty word by ϵ, and denote the set of nonempty words over Σ by Σ^+; thus $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$. For a set S, we denote its complement by $\overline{S}$. In particular, for languages $R \subseteq \Sigma^*$ and $L \subseteq \Sigma^\omega$, we have $\overline{R} = \Sigma^* \setminus R$ and $\overline{L} = \Sigma^\omega \setminus L$.

A *nondeterministic automaton* is a tuple $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$, where Σ is an alphabet, Q is a finite set of *states*, Q_0 is a set of *initial states*, $\delta : Q \times \Sigma \to 2^Q \setminus \emptyset$ is a *transition function*, and α is an *acceptance condition*, to be defined below. For states q and s and a letter $\sigma \in \Sigma$, we say that s is a *σ-successor* of q if $s \in \delta(q, \sigma)$. Note that the transition function of $\mathcal{A}$ is total, thus for all states $q \in Q$ and letters $\sigma \in \Sigma$, q has at least one σ-successor. If $|Q_0| = 1$ and every state q has a single σ-successor, for all letters σ, then $\mathcal{A}$ is *deterministic*. The transition function δ can be viewed as a transition relation $\Delta \subseteq Q \times \Sigma \times Q$, where for every two states $q, s \in Q$ and letter $\sigma \in \Sigma$, we have that $\langle q, \sigma, s \rangle \in \Delta$ iff $s \in \delta(q, \sigma)$. We define the *size* of $\mathcal{A}$, denoted $|\mathcal{A}|$, as its number of states, thus, $|\mathcal{A}| = |Q|$.

Given an input word $w = \sigma_1 \cdot \sigma_2 \cdots$, a *run* of $\mathcal{A}$ on w is a sequence of states $r = r_0, r_1, r_2, \ldots$, such that $r_0 \in Q_0$, and for all $i \geq 0$, we have that $r_{i+1} \in \delta(r_i, \sigma_{i+1})$, i.e, the run starts in some initial state and proceeds according to the transition function. If the word in the input is infinite, then so is the run. We sometimes view the run $r = r_0, r_1, r_2, \ldots$ on $w = \sigma_1 \cdot \sigma_2 \cdots$ as a sequence of successive transitions $\langle r_0, \sigma_1, r_1 \rangle, \langle r_1, \sigma_2, r_2 \rangle, \ldots$. Note that a deterministic automaton has a single run on an input w. We sometimes extend δ to sets of states and finite words. Then, $\delta : 2^Q \times \Sigma^* \to 2^Q$ is such that for every $S \in 2^Q$, finite word $u \in \Sigma^*$, and letter $\sigma \in \Sigma$, we have that $\delta(S, \epsilon) = S$, $\delta(S, \sigma) = \bigcup_{s \in S} \delta(s, \sigma)$, and $\delta(S, u \cdot \sigma) = \delta(\delta(S, u), \sigma)$. Thus, $\delta(S, u)$ is the set of states that $\mathcal{A}$ may reach when it reads u from some state in S.

The acceptance condition α determines which runs are "good". For automata on finite words, $\alpha \subseteq Q$, and a run is *accepting* if it ends in a state in α. For automata on infinite words, we consider *state-based* and *transition-based* acceptance conditions. Let us start with *state-based* conditions. Here, $\alpha \subseteq Q$, and we use the terms *α-states* and *$\overline{\alpha}$-states* to refer to states in α and in $Q \setminus \alpha$, respectively. For a run $r \in Q^\omega$, let $\mathit{sinf}(r) \subseteq Q$ be the set of states that r visits infinitely often. Thus, $\mathit{sinf}(r) = \{q : q = r_i \text{ for infinitely many } i\text{'s}\}$. In *Büchi* automata, r is *accepting* iff $\mathit{sinf}(r) \cap \alpha \neq \emptyset$, thus if r visits states in α infinitely often. Dually, in *co-Büchi* automata, r is *accepting* iff $\mathit{sinf}(r) \cap \alpha = \emptyset$, thus if r visits states in α only finitely often.

We proceed to *transition-based* conditions. There, $\alpha \subseteq \Delta$ and acceptance depends on the set of transitions that are traversed infinitely often during the run. We use the terms *α-transitions* and *$\bar{\alpha}$-transitions* to refer to transitions in α and in $\Delta \setminus \alpha$, respectively. For a run $r \in \Delta^\omega$, we define $\mathit{tinf}(r) = \{\langle q, \sigma, s \rangle \in \Delta : q = r_i, \sigma = \sigma_{i+1}, \text{ and } s = r_{i+1}, \text{ for infinitely many } i\text{'s}\}$. As expected, in transition-based Büchi automata, r is accepting iff $\mathit{tinf}(r) \cap \alpha \neq \emptyset$, and in transition-based co-Büchi automata, r is accepting iff $\mathit{tinf}(r) \cap \alpha = \emptyset$.

Consider an automaton $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$. In all automata classes, a run of $\mathcal{A}$ that is not accepting is *rejecting*. A word w is accepted by an automaton $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on w. The language of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts.

Consider a directed graph $G = \langle V, E \rangle$. A *strongly connected set* in G (SCS, for short) is a set $C \subseteq V$ such that for every two vertices $v, v' \in C$, there is a path from v to v'. An SCS is *maximal* if for every non-empty set $C' \subseteq V \setminus C$, it holds that $C \cup C'$ is not an SCS. The *maximal strongly connected sets* are also termed *strongly connected components* (SCCs, for short). An automaton $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$ induces a directed graph $G_{\mathcal{A}} = \langle Q, E \rangle$, where $\langle q, q' \rangle \in E$ iff there is a letter $\sigma \in \Sigma$ such that $\langle q, \sigma, q' \rangle \in \Delta$. The SCSs and SCCs of $\mathcal{A}$ are those of $G_{\mathcal{A}}$.

An automaton $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$ with a state-based acceptance condition $\alpha \subseteq Q$ is *weak* [34] if for each SCC C of $\mathcal{A}$, either $C \subseteq \alpha$ in which case C is *accepting*, or $C \cap \alpha = \emptyset$ in which case C is *rejecting*. We view $\mathcal{A}$ as a Büchi automaton, yet note that a weak automaton can be viewed as both a Büchi and a co-Büchi automaton. Indeed, a run of $\mathcal{A}$ visits α infinitely often iff it gets trapped in an SCC that is contained in α iff it visits states in $Q \setminus \alpha$ only finitely often. Note also that when $\mathcal{A}$ uses a transition-based acceptance condition, we can ignore the membership in α of transitions between SCCs (indeed, such transitions are traversed only finitely often), and say that $\mathcal{A}$ is weak if the transitions in each SCC are all in α or all disjoint from α.

Consider two automata $\mathcal{A}_1$ and $\mathcal{A}_2$. We say that $\mathcal{A}_1$ is *contained in* $\mathcal{A}_2$ if $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$. Then, $\mathcal{A}_1$ and $\mathcal{A}_2$ are *equivalent* if $L(\mathcal{A}_1) = L(\mathcal{A}_2)$, and $\mathcal{A}_1$ is *universal* if $L(\mathcal{A}_1) = \Sigma^\omega$ (or $L(\mathcal{A}_1) = \Sigma^*$, in case it runs on finite words). Finally, $\mathcal{A}_1$ is *minimal* (with respect to a class of automata, say state-based deterministic Büchi automata) if for all automata $\mathcal{A}_2$ equivalent to $\mathcal{A}_1$, we have that $|\mathcal{A}_1| \leq |\mathcal{A}_2|$.

2.2 SD and HD Automata

Consider an automaton $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$. For a state $q \in Q$ of $\mathcal{A}$, we define $\mathcal{A}^q = \langle \Sigma, Q, \{q\}, \delta, \alpha \rangle$, as the automaton obtained from $\mathcal{A}$ by setting the set of initial states to be $\{q\}$. We say that two states $q, s \in Q$ are *equivalent*, denoted $q \sim_{\mathcal{A}} s$, if $L(\mathcal{A}^q) = L(\mathcal{A}^s)$. We say that q *is reachable* if there is a finite word $x \in \Sigma^*$ with $q \in \delta(Q_0, x)$, and say that q *is reachable from* s if q is reachable in $\mathcal{A}^s$. We say that $\mathcal{A}$ is *semantically deterministic* (SD, for short) if different nondeterministic choices lead to equivalent states. Thus, all initial states are equivalent, and for every state $q \in Q$ and letter $\sigma \in \Sigma$, all the σ-successors of q are equivalent. Formally, if $s, s' \in \delta(q, \sigma)$, then $s \sim_{\mathcal{A}} s'$. The following proposition, termed the *SDness property*, follows immediately from the definitions and implies that in SD automata, for all finite words x, all the states in $\delta(Q_0, x)$ are equivalent. Intuitively, it means that a run of an SD automaton can take also bad nondeterministic choices, as long as it does so only finitely many times.

▶ **Proposition 1.** *Consider an SD automaton $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$, and states $q, s \in Q$. If $q \sim_{\mathcal{A}} s$, then for every $\sigma \in \Sigma$, $q' \in \delta(q, \sigma)$, and $s' \in \delta(s, \sigma)$, we have that $q' \sim_{\mathcal{A}} s'$.*

An automaton $\mathcal{A}$ is *history deterministic* (*HD*, for short) if its nondeterminism can be resolved based on the past, thus on the prefix of the input word read so far. Formally, $\mathcal{A}$ is *HD* if there exists a *strategy* $f : \Sigma^* \to Q$ such that the following hold:

1. The strategy f is consistent with the transition function. That is, $f(\epsilon) \in Q_0$, and for every finite word $u \in \Sigma^*$ and letter $\sigma \in \Sigma$, we have that $f(u \cdot \sigma) \in \delta(f(u), \sigma)$.
2. Following f causes $\mathcal{A}$ to accept all the words in its language. That is, for every word $w = \sigma_1 \cdot \sigma_2 \cdots$, if $w \in L(\mathcal{A})$, then the run $f(\epsilon), f(w[1,1]), f(w[1,2]), \ldots$ is an accepting run of $\mathcal{A}$ on w.

We say that the strategy f *witnesses* $\mathcal{A}$'s HDness. Note that, by definition, we can assume that every SD and HD automaton $\mathcal{A}$ has a single initial state. Thus, we sometimes abuse notation and write $\mathcal{A}$ as $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, where q_0 is the single initial state of the SD (or HD) automaton $\mathcal{A}$.

For an automaton $\mathcal{A}$, we say that a state q of $\mathcal{A}$ is *HD*, if $\mathcal{A}^q$ is HD. Note that every deterministic automaton is HD. Also, while not all HD automata can be pruned to deterministic ones [7], removing of transitions that are not used by a strategy f that witnesses $\mathcal{A}$'s HDness does not reduce the language of $\mathcal{A}$ and results in an SD automaton. Moreover, since every state that is used by f is HD, the removal of non-HD states does not affect $\mathcal{A}$'s language nor its HDness. Accordingly, we have the following [20, 5].

▶ **Proposition 2.** *Every HD automaton $\mathcal{A}$ can be pruned to an equivalent SD automaton all whose states are HD.*

We use three-letter acronyms in $\{\mathrm{D}, \mathrm{N}\} \times \{\mathrm{B}, \mathrm{C}, \mathrm{W}, \mathrm{F}\} \times \{\mathrm{W}\}$ to denote the different automata classes. The first letter stands for the branching mode of the automaton (deterministic or nondeterministic); the second for the acceptance condition type (Büchi, co-Büchi, weak, or an automaton that runs over finite inputs); and the third indicates that we consider automata on words. For transition-based automata, we start the acronyms with the letter "t", and for HD or SD automata, we add an HD or SD prefix, respectively. For example, an HD-tNBW is a transition-based HD nondeterministic Büchi automaton, a DFW is a state-based deterministic automaton on finite words, and an SD-NWW is a state-based weak SD automaton.

3 Semantically Deterministic Büchi Automata

In this section we examine SD-tNBWs and SD-NBWs. Our results use the following definitions and constructions: For a language $R \subseteq \Sigma^*$ of finite words, we use ∞R to denote the language of infinite words that contain infinitely many disjoint infixes in R. Thus, $w \in \infty R$ iff $\epsilon \in R$ or there are infinitely many indices $i_1 \leq i'_1 < i_2 \leq i'_2 < \cdots$ such that $w[i_j, i'_j] \in R$, for all $j \geq 1$. For example, taking $\Sigma = \{a, b\}$, we have that $\infty\{ab\}$ is the language of words with infinitely many ab infixes, namely all words with infinitely many a's and infinitely many b's. We say that a finite word $x \in \Sigma^*$ is a *good prefix* for a language $R \subseteq \Sigma^*$ if for all finite words $y \in \Sigma^*$, we have that $x \cdot y \in R$. For example, while the language $(a + b)^* \cdot a$ does not have a good prefix, the word a is a good prefix for the language $a \cdot (a + b)^*$.

Theorem 3 below suggests that one can encode an NFW-recognizable language R in an SD-tNBW $\mathcal{A}$ for ∞R, and Theorem 4 suggests that one can decode an NFW for R from an SD-tNBW for $\infty(\$ \cdot R \cdot \$)$, where $\$ \notin \Sigma$. The blow-up in the sizes of the automata is constant, and both theorems play a major rule in the rest of this section.

▶ **Theorem 3.** *Given an NFW $\mathcal{N}$, one can obtain, in polynomial time, an SD-tNBW $\mathcal{A}$ such that $L(\mathcal{A}) = \infty L(\mathcal{N})$ and $|\mathcal{A}| = |\mathcal{N}|$.*

Proof. Let $\mathcal{N} = \langle \Sigma, Q, Q_0, \delta, F \rangle$. Then, $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta', \alpha \rangle$ is obtained from $\mathcal{N}$ by adding transitions to Q_0 from all states with all letters. A new σ-transition is in α if $Q_0 \cap F \neq \emptyset$ or when $\mathcal{N}$ could transit with σ to a state in F. Formally, for all $s \in Q$ and $\sigma \in \Sigma$, we have that $\delta'(s, \sigma) = \delta(s, \sigma) \cup Q_0$, and $\alpha = \{\langle s, \sigma, q \rangle : q \in Q_0 \text{ and } (Q_0 \cap F \neq \emptyset \text{ or } \delta(s, \sigma) \cap F \neq \emptyset)\}$.

It is easy to see that $|\mathcal{A}| = |\mathcal{N}|$. In order to prove that $\mathcal{A}$ is SD and $L(\mathcal{A}) = \infty L(\mathcal{N})$, we prove in the full version that for every state $q \in Q$, it holds that $L(\mathcal{A}^q) = \infty L(\mathcal{N})$. ◀

▶ **Theorem 4.** *Consider a language $R \subseteq \Sigma^*$ and a letter $\$ \notin \Sigma$. For every SD-tNBW $\mathcal{A}$ such that $L(\mathcal{A}) = \infty(\$ \cdot R \cdot \$)$, there exists an NFW $\mathcal{N}$ such that $L(\mathcal{N}) = R$ and $|\mathcal{N}| \leq |\mathcal{A}| + 1$. In addition, if R has no good prefixes, then $|\mathcal{N}| \leq |\mathcal{A}|$.*

Proof. If R is trivial, then one can choose $\mathcal{N}$ to be a one-state NFW. Assume that R is nontrivial. Let $\mathcal{A} = \langle \Sigma \cup \{\$\}, Q, q_0, \delta, \alpha \rangle$ be an SD-tNBW for $\infty(\$ \cdot R \cdot \$)$, W.l.o.g we assume that all the states of $\mathcal{A}$ are reachable. For a nonempty set of states $S \in 2^Q \setminus \emptyset$, we define the *universal $\overline{\alpha}$ language* of S as

$$L_{u\overline{\alpha}}(S) = \{w \in (\Sigma \cup \{\$\})^\omega : \text{for all } q \in S, \text{ all the runs of } \mathcal{A}^q \text{ on } w \text{ do not traverse } \alpha\}.$$

We say that S is *hopeful* when $(\$ \cdot \overline{R})^\omega \subseteq L_{u\overline{\alpha}}$. Note that S is hopeful iff for every state $q \in S$, it holds that $\{q\}$ is hopeful. Also, if S is hopeful, $x \in \Sigma^*$ is a finite word, and there is a run from S on $\$ \cdot x$ that traverses α, then $x \in R$. Then, we say that S is *good* when for all words $x \in \Sigma^*$, it holds that $x \in \overline{R}$ iff all the runs from S on $\$ \cdot x$ do not traverse α, and the set $\delta(S, \$ \cdot x)$ is hopeful. Note that as R is nontrivial, there exists a word x in $\overline{R}$, and thus by definition, all the \$-labled transitions going out from a good set S are in $\overline{\alpha}$.

Good sets in $\mathcal{A}$ characterize the language R, and as we argue below, their existence induces an NFW for R. In the full version, we prove that a good set exists. We show now that a good set in $\mathcal{A}$ induces an NFW $\mathcal{N}$ for R with the required properties. Let $S \in 2^Q \setminus \emptyset$ be a good set. We define the NFW $\mathcal{N} = \langle \Sigma, Q \cup \{q_{acc}\}, Q_0^S, \delta_S, F_S \rangle$, where $Q_0^S = \delta(S, \$)$, $F_S = \{q_{acc}\} \cup \{q \in Q : \text{the set } \{q\} \text{ is not hopeful}\}$, and the transition function δ_S is defined as follows. For every two states $q, s \in Q$ and letter $\sigma \in \Sigma$, it holds that $s \in \delta_S(q, \sigma)$ iff $\langle q, \sigma, s \rangle \in \overline{\alpha}$. Also, $q_{acc} \in \delta_S(q, \sigma)$ iff there is a σ-labeled α-transition going out from q in $\mathcal{A}$. Also, for all letters $\sigma \in \Sigma$, it holds that $\delta(q_{acc}, \sigma) = \{q_{acc}\}$; that is, q_{acc} is an accepting sink. Thus, $\mathcal{N}$ behaves as the states in $\delta(S, \$)$ as long as it reads $\overline{\alpha}$ transitions of $\mathcal{A}$, moves to the accepting sink q_{acc} whenever an α-transition is encountered, and accepts also whenever it reaches a state in Q that is not hopeful.

In the full version, we prove that $L(\mathcal{N}) = R$. Essentially, this follows from the fact that if we consider a word $x \in \Sigma^*$ such that all the runs from Q_0^S on x in $\mathcal{A}$ do not traverse α, then S being a good set implies that $x \in R$ iff $\delta(S, \$ \cdot x) \cap F_S \neq \emptyset$.

Since the state space of $\mathcal{N}$ is $Q \cup \{q_{acc}\}$, then $|\mathcal{N}| = |\mathcal{A}| + 1$. Moreover, as q_{acc} is an accepting sink, a word $x \in L(\mathcal{N})$ that has a run that ends in q_{acc} is a good prefix for $L(\mathcal{N})$. Hence, as $L(\mathcal{N}) = R$, if R has no good prefixes, then q_{acc} is not reachable in $\mathcal{N}$ and thus can be removed without affecting $\mathcal{N}$'s language. Thus, in this case, we get an NFW $\mathcal{N}$ for R whose size is at most $|\mathcal{A}|$. ◀

3.1 Succinctness and Complementation

In this section we study the succinctness of SD Büchi automata with respect to deterministic ones, and the blow-up involved in their complementation. We show that SD-tNBWs are exponentially more succinct than tDBWs, matching the known upper bound [3], and in fact, also from HD-tNBWs. We also prove an exponential lower bound for complementation. Similar results for SD-NBWs follow, as the transition between the two types of acceptance conditions is linear.

▶ **Theorem 5.** *There is a family $L_1, L_2, L_3, \ldots$ of languages such that for every $n \geq 1$, there is an SD-tNBW with $3n + 3$ states that recognizes L_n, yet every tDBW or HD-tNBW that recognizes L_n needs at least 2^n states.*

Proof. For $n \geq 1$, let $[n] = \{1, \ldots, n\}$, and let $\Sigma_n = \{1, \ldots, n, \$, \#\}$. We say that a word $z \in \Sigma_n^*$ is *good* if $z = \$ \cdot x \cdot \# \cdot i$, where $x \in [n]^+$ and i appears in x. Let $R_n \subseteq \Sigma_n^*$ be the language of all good words. We define $L_n = \infty R_n$. First, it is not hard to see that R_n can be recognized by an NFW $\mathcal{N}_n$ with $3n + 3$ states. Essentially, $\mathcal{N}_n$ guesses the last letter i in the input word and then checks that the guess is correct (see sketch in Figure 1). By Theorem 3, there is an SD-tNBW for L_n with $3n + 3$ states.

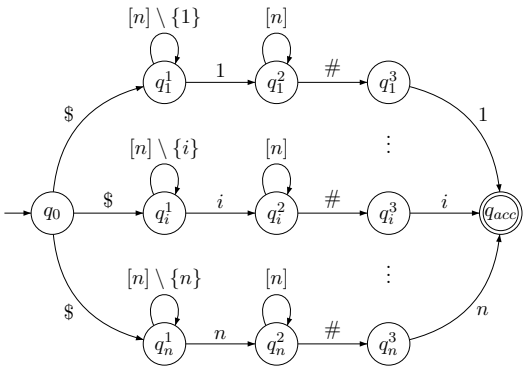


Figure 1 The NFW $\mathcal{N}_n$. Missing transitions lead to a rejecting-sink.

Before we prove that every tDBW or HD-tNBW that recognizes L_n needs at least 2^n states, let us note that it is already known that going from a DFW for a language $R \subseteq \Sigma^*$ to a tDBW for ∞R, may involve a blow-up of $2^{n-2-log_2(n)}$[24]. Thus, Theorem 3 implies an exponential gap between SD-tNBWs and tDBWs. Moreover, as HD-tNBWs are at most quadratically more succinct than tDBWs [20], the above can be extended to a $2^{\frac{n-2-log_2(n)}{2}}$ lower bound for the succinctness of SD-tNBWs with respect to HD-tNBWs. Our example here is tighter.

It is left to prove that every HD-tNBW that recognizes L_n needs at least 2^n states. Assume towards contradiction that $\mathcal{A} = \langle \Sigma_n, Q, q_0, \delta, \alpha \rangle$ is an HD-tNBW for L_n with $|Q| < 2^n$ states. In the full version, we iteratively define infinite sequences of finite words $x_1, x_2, x_3, \ldots$ and states $q_0, q_1, \ldots$ such that for all $k \geq 1$, the word x_k starts with \$, has no good infixes, and there is a run of the form $r_k = q_{k-1} \xrightarrow{x_k} q_k$ in $\mathcal{A}$ on x_k that traverses α. The challenging part in the construction is to make it valid also for HD (and not only deterministic) automata. For this, the definition of the words in the sequence is defined with respect to a strategy that attempts to witness the HDness of $\mathcal{A}$. To see why such sequences imply a contradiction, note that the concatenation of the runs $r_1, r_2, \ldots$ is an accepting run of $\mathcal{A}$ on the word $x = x_1 \cdot x_2 \cdots$. As $\mathcal{A}$ recognizes $L_n = \infty R_n$, it follows that $x \in \infty R_n$. On the other hand, x has no good infixes, and so $x \notin \infty R_n$. Indeed, if there is a good infix in x, then it must contain letters from different x_k's; in particular, it must contain at least two \$'s. ◀

▶ Remark 6. In order to get a slightly tighter bound, one can show that a minimal tDBW for L_n needs at least 2^{n+1} states and that the language L_n is not *HD-helpful*. That is, a minimal HD-tNBW for L_n is not smaller than a minimal tDBW for L_n, and so the 2^{n+1} bound holds also for a minimal HD-tNBW. The proof starts with a tDBW for L_n that has 2^{n+1} states, considers its complement tDCW, and shows that the application of the polynomial minimization algorithm of [1] on it, namely the algorithm that returns an equivalent minimal HD-tNCW, does result in a smaller automaton. The result then follows from the fact that a minimal HD-tNCW for a language is smaller than a minimal HD-tNBW for its complement [20]. The proof involves many notions and observations from [1] about minimal HD-tNCWs.

In the full version, we still describe a tDBW with 2^{n+1} states for L_n, and readers familiar with [1] can observe that the application of the HD-tNCW minimization algorithm on its dual tDCW does not make it smaller.

▶ **Theorem 7.** *There is a family $L_1, L_2, L_3, \ldots$ of languages such that for every $n \geq 1$, there is an SD-tNBW with $O(n)$ states that recognizes L_n, yet every SD-tNCW that recognizes $\overline{L_n}$ needs at least $2^{O(n)}$ states.*

Proof. Let $\Sigma = \{0, 1\}$. For $n \geq 1$, let $R_n = \{w : w \in 0 \cdot (0+1)^{n-1} \cdot 1 + 1 \cdot (0+1)^{n-1} \cdot 0\}$. It is easy to see that R_n can be recognized by an NFW $\mathcal{N}_n$ with $O(n)$ states. We define $L_n = \infty R_n$. First, by Theorem 3, there is an SD-tNBW with $O(n)$ states for L_n. In order to prove that an SD-tNCW for $\overline{L_n}$ needs at least $2^{O(n)}$ states, we prove that in fact every tNCW for $\overline{L_n}$ needs that many states. For this, note that $\overline{L_n}$ consists of all words w for which there is $u \in (0+1)^n$ such that $w \in (0+1)^* \cdot u^\omega$. Indeed, for such words w, the suffix u^ω contains no infix in R_n. Also, if a word contain only finitely many infixes in R_n, then it must have a suffix with no such infixes, namely a suffix of the form u^ω for some $u \in (0+1)^n$. Then, the proof that a tNCW for $\overline{L_n}$ needs exponentially many states is similar to the proof that an NFW for $\{u \cdot u : u \in (0+1)^n\}$ needs exponentially many states. Indeed, it has to remember the last n letters read. ◀

3.2 Decision Problems

We continue to decision problems about SD-tNBWs and SD-NBWs, and show that the exponential succinctness comes with a price: the complexity of all the problems we study coincides with the one known for tNBWs and NBWs. Accordingly, we only prove lower bounds for SD-tNBWs. Matching upper bounds follow from the known complexity for tNBWs, and same bounds for SD-NBWs follow from linear translations between SD-tNBWs and SD-NBWs. The problems we study are *language containment:* given two SD-tNBWs $\mathcal{A}_1$ and $\mathcal{A}_2$, decide whether $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$, *universality:* given an SD-tNBW $\mathcal{A}_1$, decide whether $L(\mathcal{A}_1) = \Sigma^\omega$, and *minimization:* given an SD-tNBW $\mathcal{A}_1$ and an integer $k \geq 1$, decide whether there is an SD-tNBW $\mathcal{A}_2$ such that $L(\mathcal{A}_1) = L(\mathcal{A}_2)$ and $|\mathcal{A}_2| \leq k$.

The exponential succinctness of SD automata motivates also the study of the *D-to-SD minimization problem*. Here, we are given a tDBW $\mathcal{A}_1$ and an integer $k \geq 1$, and we need to decide whether there is an SD-tNBW $\mathcal{A}_2$ such that $L(\mathcal{A}_1) = L(\mathcal{A}_2)$ and $|\mathcal{A}_2| \leq k$. For automata on finite words, the D-to-N minimization problem is known to be PSPACE-complete [19].

Note that a lower bound for universality implies a lower bound also for language containment. We still start with language containment, as it is much simpler.

▶ **Theorem 8.** *The language-containment problem for SD-tNBWs is PSPACE-hard.*

Proof. We describe a reduction from the universality problem for NFWs. Given an NFW $\mathcal{N}$ over Σ, let $\mathcal{N}'$ be an NFW over $\Sigma \cup \{\$\}$ such that $L(\mathcal{N}') = \$ \cdot L(\mathcal{N}) \cdot \$$. Now, let $\mathcal{A}_1$ be a 1-state tDBW over $\Sigma \cup \{\$\}$ such that $L(\mathcal{A}_1) = \infty\$$, and let $\mathcal{A}_2$ be the SD-tNBW obtained by applying the operation from Theorem 3 on $\mathcal{N}'$. Note that $L(\mathcal{A}_2) = \infty(\$ \cdot L(\mathcal{N}) \cdot \$)$ and $|\mathcal{A}_2| = |\mathcal{N}| + 3$.

We claim that $\mathcal{N}$ is universal iff $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$. First, if $L(\mathcal{N}) = \Sigma^*$, then $L(\mathcal{A}_2) = \infty(\$ \cdot \Sigma^* \cdot \$) = \infty\$$ and so $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$. Conversely, if there is a word $x \in \Sigma^* \setminus L(\mathcal{N})$, then the word $w = (\$ \cdot x)^\omega$ is in $L(\mathcal{A}_1) \setminus L(\mathcal{A}_2)$. Indeed, w has infinitely many \$'s, yet for every infix $\$ \cdot y \cdot \$$ of w, we have that $y \notin L(\mathcal{N})$, and so $w \notin L(\mathcal{A}_2)$. ◀

The proof in Theorem 8 uses $\infty\$$ as the "contained language". For the universality problem, where we cannot relay on hints from words in the contained language, we have to work much harder and generate such hints from runs of Turing machines. Specifically, we prove PSPACE hardness by a generic reduction from polynomial space Turing machines. Such reductions associate with a Turing machine T an automaton $\mathcal{A}$ that recognizes the language R of words that do not encode legal rejecting computations of T, and so $R = \Sigma^*$ iff the machine has no rejecting computations. The automaton $\mathcal{A}$ is nondeterministic, as it has to guess violations of attempts to encode legal accepting computations. In order to replace $\mathcal{A}$ by an SD automaton, we manipulate the Turing machine so that the language of the generated automaton is of the form ∞R, for which we can construct an SD-tNBW.

▶ **Theorem 9.** *The universality problem for SD-tNBWs is PSPACE-hard.*

Proof. We do a reduction from polynomial-space Turing machines. Given a Turing machine T with space complexity $s : \mathbb{N} \to \mathbb{N}$, we construct in time polynomial in $|T|$ and $s(0)$, an SD-tNBW $\mathcal{A}$ of size polynomial in T and $s(0)$, such that $\mathcal{A}$ is universal iff T accepts the empty tape[3]. Let $n_0 = s(0)$. Thus, each configuration in the computation of T on the empty tape uses at most n_0 cells. We assume that T halts from all configurations (that is, not just from these reachable from an initial configuration of T); Indeed, by adding a polynomial-space counter to T, one can transform a polynomial-space Turing machine that need not halt from all configurations to one that does halt. We also assume, without loss of generality, that once T reaches a final (accepting or rejecting) state, it erases the tape, moves with its reading head to the leftmost cell, and moves to the initial state. Thus, all computations of T are infinite and after visiting a final configuration for the first time, they eventually consists of repeating the same finite computation on the empty tape that uses at most n_0 tape cells.

We define $\mathcal{A}$ so that it accepts a word w iff (C1) no suffix of w is an encoding of a legal computation of T that uses at most n_0 tape cells, or (C2) w has infinitely many infixes that encode the accepting state of T.

It is not hard to see that T accepts the empty tape iff $\mathcal{A}$ is universal. Indeed, if T accepts the empty tape, and there is a word w that does not satisfy (C1), thus w has a suffix that is an encoding of a legal computation of T that uses at most n_0 cells, then the encoded computation eventually reaches a final configuration, from which it eventually repeats the accepting computation of T on the empty tape infinitely many times, and so w satisfies (C2). Conversely, If T rejects the empty tape, then the word w that encodes the computation of T on the empty tape does not satisfy (C1) nor (C2), and so $\mathcal{A}$ does not accept w.

Finally, the fact that T is a polynomial-space Turing machine enables us to define $\mathcal{A}$ with polynomially many states, as we detail in the full version. ◀

We continue to the minimization problem. Note that here, a PSPACE upper bound does not follow immediately from the known PSPACE upper bound for tNBWs, as the candidate automata need to be SD. Still, as SDness can be checked in PSPACE [3], a PSPACE upper bound follows. Also note that here, the case of SD-NBWs is easy, as a non-empty SD-NBW

[3] This is sufficient, as one can define a generic reduction from every language L in PSPACE as follows. Let T_L be a Turing machine that decides L in polynomial space $f(n)$. On input w for the reduction, the reduction considers the machine T_w that on every input, first erases the tape, writes w on its tape, and then runs as T_L on w. Then, the reduction outputs an automaton $\mathcal{A}$, such that T_w accepts the empty tape iff $\mathcal{A}$ is SD. Note that the space complexity of T_w is $s(n) = \max(n, f(|w|))$, and that w is in L iff T_w accepts the empty tape. Since $\mathcal{A}$ is constructed in time polynomial in $s(0) = f(|w|)$ and $|T_w| = \text{poly}(|w|)$, it follows that the reduction is polynomial in $|w|$.

is universal iff it has an equivalent SD-NBW with one state. For transition-based acceptance, the language of a single-state SD-tNBW need not be trivial, and so we have to examine the specific language used for the universality PSPACE-hardness proof (see proof in the full version):

▶ **Theorem 10.** *The minimization problem for SD-tNBWs is PSPACE-hard.*

As we show below, the minimization problem stays hard even when we start from a deterministic automaton:

▶ **Theorem 11.** *The D-to-SD minimization problem for Büchi automata is PSPACE-hard.*

Proof. We start with Büchi automata with transition-based acceptance, and describe a reduction from the D-to-N minimization problem for automata on finite words: given a DFW $\mathcal{A}_1$, and an integer $k \geq 1$, decide whether there is an NFW $\mathcal{A}_2$ such that $L(\mathcal{A}_1) = L(\mathcal{A}_2)$ and $|\mathcal{A}_2| \leq k$. In [19], the authors prove that the problem is PSPACE-hard, in fact PSPACE-hard already for DFWs that recognize a language that has no good prefixes.

Consider a language $R \subseteq \Sigma^*$. The reduction is based on a construction that turns an NFW for R into an SD-tNBW for $\infty(\$ \cdot R \cdot \$)$, for a letter $\$ \notin \Sigma$. Note that by applying the construction from Theorem 3 on the language $\$ \cdot R \cdot \$$, we can get an SD-tNBW for $\infty(\$ \cdot R \cdot \$)$. The construction there, however, does not preserve determinism. Therefore, we need a modified polynomial construction, which takes advantage of the \$'s. We describe the modified construction below.

Given an NFW $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, F \rangle$, and a letter $\$ \notin \Sigma$, we construct the tNBW $\mathcal{A}' = \langle \Sigma \cup \{\$\}, Q, Q_0, \delta', \alpha \rangle$, where for all states $q \in Q$ and letters $\sigma \in \Sigma$, we have that $\delta'(q, \sigma) = \delta(q, \sigma)$. Also, $\delta'(q, \$) = Q_0$, and $\alpha = \{\langle s, \sigma, q \rangle : q \in Q_0 \text{ and } s \in F\}$. Thus, $\mathcal{A}'$ is obtained from $\mathcal{A}$ by adding \$-transitions from all states to Q_0. A new transition is in α iff its source is an accepting state of $\mathcal{A}$. In the full version, we prove that $\mathcal{A}'$ is an SD-tNBW and $L(\mathcal{A}') = \infty(\$ \cdot L(\mathcal{A}) \cdot \$)$. Also, as we only added transitions labeled \$ to Q_0, it is easy to see that if $\mathcal{A}$ is deterministic, then so is $\mathcal{A}'$.

We now describe the reduction. Given a DFW $\mathcal{A}$ over Σ such that $L(\mathcal{A})$ has no good prefixes, and given an integer k, the reduction returns the polynomial tDBW $\mathcal{A}'$ and the integer k. We prove next that the reduction is correct. Thus, the DFW $\mathcal{A}$ has an equivalent NFW with at most k states iff the tDBW $\mathcal{A}'$ has an equivalent SD-tNBW with at most k states. For the first direction, if $\mathcal{B}$ is an NFW equivalent to $\mathcal{A}$ whose size is at most k, then by the above construction, it holds that $\mathcal{B}'$ is an SD-tNBW whose size is at most k, and $L(\mathcal{B}') = \infty(\$ \cdot L(\mathcal{B}) \cdot \$) = \infty(\$ \cdot L(\mathcal{A}) \cdot \$) = L(\mathcal{A}')$.

Conversely, if $\mathcal{B}'$ is an SD-tNBW for $\infty(\$ \cdot L(\mathcal{A}) \cdot \$)$ whose size is at most k, then as $L(\mathcal{A})$ has no good prefixes, we get by Theorem 4 that there is an NFW $\mathcal{B}$ for $L(\mathcal{A})$ whose size is at most k, and we are done.

Finally, since the transitions between automata with transition-based and state-based acceptance may involve a linear blow-up, here we need to be careful in extending the result to the state-based setting. In the full version, we prove that the arguments in Theorem 11 can be adapted to automata with state-based acceptance, thus PSPACE-completeness holds also for Büchi automata with state-based acceptance. ◀

4 Semantically Deterministic co-Büchi Automata

In this section we study SD co-Büchi automata. Here too, our results are based on constructions that involve encodings of NFWs by SD-tNCWs. Here, however, the constructions are more complicated, and we first need some definitions and notations.

Consider a tNCW $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$. We refer to the SCCs we get by removing $\mathcal{A}$'s α-transitions as the $\overline{\alpha}$-*components* of $\mathcal{A}$; that is, the $\overline{\alpha}$-*components* of $\mathcal{A}$ are the SCCs of the graph $G_{\mathcal{A}^{\overline{\alpha}}} = \langle Q, E^{\overline{\alpha}} \rangle$, where $\langle q, q' \rangle \in E^{\overline{\alpha}}$ iff there is a letter $\sigma \in \Sigma$ such that $\langle q, \sigma, q' \rangle \in \overline{\alpha}$. We say that $\mathcal{A}$ is *normal* if there are no $\overline{\alpha}$-transitions connecting different $\overline{\alpha}$-components. That is, for all states q and s of $\mathcal{A}$, if there is a path of $\overline{\alpha}$-transitions from q to s, then there is also a path of $\overline{\alpha}$-transitions from s to q. Note an accepting run of $\mathcal{A}$ eventually gets trapped in one of $\mathcal{A}$'s $\overline{\alpha}$-components. In particular, accepting runs in $\mathcal{A}$ traverse transitions that leave $\overline{\alpha}$-components only finitely often. Hence, we can add transitions among $\overline{\alpha}$-components to α without changing the language of the automaton. Accordingly, in the sequel we assume that given tNCWs are normal.

We proceed to encoding NFWs by SD-tNCWs. For a language $R \subseteq \Sigma^*$, we define the language $\bowtie_{\$}(R) \subseteq (\Sigma \cup \{\$\})^\omega$, by

$$\bowtie_{\$}(R) = \{w : \text{if } w \text{ has infinitely many } \$, \text{ then it has a suffix in } (\$R)^\omega\}.$$

Also, we say that a finite word $x \in \Sigma^*$ is a *bad infix* for R if for all words $w \in \Sigma^*$ that have x as an infix, it holds that $w \in \overline{R}$.

Note that for every language $R \subseteq \Sigma^*$, we have that $\bowtie_{\$}(R) = \overline{\infty(\$ \cdot \overline{R} \cdot \$)}$. Thus, $\bowtie_{\$}(R)$ complements $\infty(\$ \cdot \overline{R} \cdot \$)$. Yet, unlike tDCWs and tDBWs, which dualize each other, SD-tNBWs and SD-tNCWs are not dual. Hence, adjusting Theorems 3 and 4 to the co-Büchi setting, requires different, in fact more complicated, constructions.

▶ **Theorem 12.** *Given an NFW $\mathcal{N}$, one can obtain, in linear time, an SD-tNCW $\mathcal{A}$ such that $L(\mathcal{A}) = \bowtie_{\$}(L(\mathcal{N}))$ and $|\mathcal{A}| = |\mathcal{N}|$.*

Proof. Given $\mathcal{N} = \langle \Sigma, Q, Q_0, \delta, F \rangle$, we obtain $\mathcal{A}$ by adding \$-transitions from all states to Q_0. The new transitions are in α iff they leave a state in $Q \setminus F$. Formally, $\mathcal{A} = \langle \Sigma \cup \{\$\}, Q, Q_0, \delta', \alpha \rangle$, where for all $s \in Q$ and $\sigma \in \Sigma$, we have that $\delta'(s, \sigma) = \delta(s, \sigma)$, and $\delta'(s, \$) = Q_0$. Then, $\alpha = \{\langle s, \$, q \rangle : q \in Q_0 \text{ and } s \in Q \setminus F\}$. It is easy to see that $|\mathcal{A}| = |\mathcal{N}|$. In order to prove that $\mathcal{A}$ is SD and $L(\mathcal{A}) = \bowtie_{\$}(L(\mathcal{N}))$, we prove in the full version that for every state $q \in Q$, it holds that $L(\mathcal{A}^q) = \bowtie_{\$}(L(\mathcal{N}))$. Essentially, this follows from the fact that $\mathcal{A}$ can avoid traversing α when it reads an input of the form $x \cdot \$$, only when $x \in L(\mathcal{N})$. ◀

▶ **Theorem 13.** *Consider a language $R \subseteq \Sigma^*$ and a letter $\$ \notin \Sigma$. For every tNCW $\mathcal{A}$ such that $L(\mathcal{A}) = \bowtie_{\$}(R)$, there exists an NFW $\mathcal{N}$ such that $L(\mathcal{N}) = R$ and $|\mathcal{N}| \leq |\mathcal{A}| + 1$. In addition, if R has bad infixes, then $|\mathcal{N}| \leq |\mathcal{A}|$.*

Proof. Let $\mathcal{A} = \langle \Sigma \cup \{\$\}, Q, Q_0, \delta, \alpha \rangle$ be a tNCW for $\bowtie_{\$}(R)$. We assume that $\mathcal{A}$ is normal and all of its states are reachable. We define the NFW $\mathcal{N} = \langle \Sigma, Q \cup \{q_{rej}\}, Q_0^{\mathcal{N}}, \delta_{\mathcal{N}}, F_{\mathcal{N}} \rangle$, where

- $Q_0^{\mathcal{N}} = \{q \in Q : \text{there is a state } q' \text{ such that } \langle q', \$, q \rangle \in \overline{\alpha}\}$.
- $F_{\mathcal{N}} = \{q \in Q : \text{there is a state } q' \text{ such that } \langle q, \$, q' \rangle \in \overline{\alpha}\}$.
- The transition function $\delta_{\mathcal{N}}$ is defined as follows: for every two states $q, s \in Q$ and letter $\sigma \in \Sigma$, it holds that $s \in \delta_{\mathcal{N}}(q, \sigma)$ iff $\langle q, \sigma, s \rangle \in \overline{\alpha}$. Also, if $\{q\} \times \{\sigma\} \times \delta(q, \sigma) \subseteq \alpha$, then $\delta_{\mathcal{N}}(q, \sigma) = q_{rej}$. Finally, for all letters $\sigma \in \Sigma$, it holds that $\delta_{\mathcal{N}}(q_{rej}, \sigma) = q_{rej}$; that is, q_{rej} is a rejecting sink.

Thus, $\mathcal{N}$ tries to accept words $x \in \Sigma^*$ for which there is a run in $\mathcal{A}$ that does not traverse α on the word $\$ \cdot x \cdot \$$.

We prove that $L(\mathcal{N}) = R$. We first prove that $R \subseteq L(\mathcal{N})$. Consider a word $x \in R$, and let $r = r_0, r_1, r_2, \ldots$ be an accepting run of $\mathcal{A}$ on $(\$ \cdot x)^\omega$. As r is accepting, there are $i < j$ such that $r_i, r_{i+1}, \ldots, r_j$ is a run that does not traverse α on $\$ \cdot x \cdot \$$. By the definition of $\delta_{\mathcal{N}}$, $r_{i+1} \in Q_0^{\mathcal{N}}, r_{j-1} \in F_{\mathcal{N}}$, and so $r_{i+1}, r_{i+2}, \ldots, r_{j-1}$ is an accepting run of $\mathcal{N}$ on x.

We prove next that $L(\mathcal{N}) \subseteq R$. Consider a word $x = \sigma_1 \cdot \sigma_2 \cdots \sigma_n \in L(\mathcal{N})$ and let $r = r_0, r_1, \ldots r_n$ be an accepting run of $\mathcal{N}$ on x. As r ends an accepting state of $\mathcal{N}$, then it does not visit q_{rej}. Hence, the definition of $\delta_{\mathcal{N}}$ implies that r is a run that does not traverse α in $\mathcal{A}$. Also, as $r_0 \in Q_0^{\mathcal{N}}$ and $r_n \in F_{\mathcal{N}}$, there are states $q_1, q_2 \in Q$ such that $\langle q_1, \$, r_0\rangle, \langle r_n, \$, q_2\rangle \in \overline{\alpha}$. Hence, $q_1, r_0, r_1, \ldots, r_n, q_2$ is a run that does not traverse α in $\mathcal{A}$ on the word $\$ \cdot x \cdot \$$. As $\mathcal{A}$ is normal, there is a word $y \in (\Sigma \cup \{\$\})^*$ such that there is a run that does not traverse α of $\mathcal{A}^{q_2}$ on y that reaches q_1. Therefore, $(\$ \cdot x \cdot \$ \cdot y)^\omega \in L(\mathcal{A}^{q_1})$. As all the states of $\mathcal{A}$ are reachable, it follows that there is a word $z \in (\Sigma \cup \{\$\})^*$ such that $q_1 \in \delta(Q_0, z)$, and thus $z \cdot (\$ \cdot x \cdot \$ \cdot y)^\omega \in L(\mathcal{A})$. Hence, as $L(\mathcal{A}) = \bowtie_{\$}(R)$, we get that $x \in R$.

Since the state space of $\mathcal{N}$ is $Q \cup \{q_{rej}\}$, we have that $|\mathcal{N}| = |\mathcal{A}| + 1$. Moreover, if R has a bad infix x, then $x \in \overline{R}$. Hence, if we consider the word x^ω, then as it has no \$'s, it is accepted by $\mathcal{A}$. Hence, there is a state $q \in Q$ such that $\mathcal{A}^q$ has a run on x^ω that does not leave the $\overline{\alpha}$-component of q. As we detail in the full version, the fact that x is a bad infix of R implies that the $\overline{\alpha}$-component of q does not contain \$-transitions. Hence, since the only states in Q that are reachable in $\mathcal{N}$ lie in $\overline{\alpha}$-components that contain a \$-transition, we can remove the state q_{rej} from $\mathcal{N}$ and make q a rejecting sink instead. Hence, in this case, we have that $|\mathcal{N}| \leq |\mathcal{A}|$, and we are done. ◀

4.1 Succinctness and Complementation

In this section we study the succinctness of SD co-Büchi automata with respect to deterministic ones, and the blow-up involved in their complementation. Recall that unlike Büchi automata, for co-Büchi automata, HD automata are exponentially more succinct than deterministic ones. Accordingly, our results also refer to the succinctness of SD automata with respect to HD ones:

▶ **Theorem 14.** *There is a family $L_1, L_2, L_3, \ldots$ of languages such that for every $n \geq 1$, there is an SD-tNCW with $3n + 3$ states that recognizes L_n, yet every tDCW or HD-tNCW that recognizes L_n needs at least 2^n states.*

Proof. For $n \geq 1$, consider the alphabet $\Sigma_n = [n] \cup \{\#\}$, and the language $R_n = \{x \cdot \# \cdot i : x \in [n]^+ \text{ and } i \text{ appears in } x\} \subseteq \Sigma_n^*$. We define $L_n = \bowtie_{\$}(R_n)$. Recall that a word over $z \in (\Sigma_n \cup \{\$\})^*$ is good if $z = \$ \cdot x \cdot \# \cdot i$, where $x \in [n]^+$ and i appears in x, and note that L_n consists of exactly the words with finitely many \$'s, or that have a suffix that is a concatenation of good words. First, it is not hard to see that R_n can be recognized by an NFW $\mathcal{N}_n$ with $3n + 3$ states, for example, a candidate for $\mathcal{N}_n$ can be obtained from the NFW in Figure 1 by removing the \$-transitions from q_0, and letting q_0 guess to behave as any state in $\bigcup_{i \in [n]} \{q_i^1\}$. By Theorem 12, there is an SD-tNCW for L_n with $3n + 3$ states.

In order to show that an HD-tNCW for L_n needs at least 2^n states, we rely on properties of HD-tNCWs [20, 1] and argue that (1) an HD-tNCW for L_n has a state q such that $L_{\overline{\alpha}}(q) = \{w : \text{there is a run from } q \text{ on } w \text{ that does not traverse } \alpha\}$ is such that $(\$ \cdot R_n)^\omega \subseteq L_{\overline{\alpha}}(q)$, and (2) the $\overline{\alpha}$-component of q is of size at least 2^n. As we detail in the full version, the first claim follows from the fact that HD-tNCWs can be assumed to be $\overline{\alpha}$ *deterministic*, thus all their $\bar{\alpha}$-transitions are deterministic [20, 1]. The second claim follows from the fact that the $\overline{\alpha}$-component of q should detect good subwords, and should remember for this subsets of $[n]$. ◀

▶ Remark 15. As has been the case with Büchi automata, here too, an analysis of the application of the HD-tNCW minimization algorithm on a tDCW for L_n leads to a slightly tighter bound. Specifically, in the full version, we describe a tDCW for L_n with $2^{n+1} + 1$ states, such that the application of the HD-tNCW minimization algorithm on it does not make it smaller.

▶ **Theorem 16.** *There is a family $L_1, L_2, L_3, \ldots$ of languages such that for every $n \geq 1$, there is an SD-tNCW with $O(n)$ states that recognizes L_n, yet every SD-tNBW that recognizes $\overline{L_n}$ needs at least $2^{O(n)}$ states.*

Proof. Let $\Sigma = \{0, 1\}$, and let $R_n = \{w : w \in (0+1)^* \cdot (0 \cdot (0+1)^{n-1} \cdot 1 + 1 \cdot (0+1)^{n-1} \cdot 0) \cdot (0+1)^*\}$. Thus, R_n is the language of all words that contain two letters that are at distance n and are different. We define $L_n = {\bowtie_\$}(R_n)$.

It is easy to see that R_n can be recognized by an NFW with $O(n)$ states. Hence, by Theorem 12, there is an SD-tNCW with $O(n)$ states for L_n. We prove next that every SD-tNBW for $\overline{L_n}$ needs at least $2^{O(n)}$ states. For this, note that $\overline{L_n}$ consists of all words w such that w contains infinitely many \$'s yet has no suffix in $(\$R_n)^\omega$. Thus, w contains infinitely many infixes in $\$\overline{R_n}\$$. Therefore, $\overline{L_n} = \infty(\$ \cdot \overline{R_n} \cdot \$)$. It is not hard to prove that an NFW for $\overline{R_n}$ needs at least $2^{O(n)}$ states. Then, by Theorem 4, this bound is carried over to a $2^{O(n)}$ lower bound on an SD-tNBW for $\overline{L_n}$. ◀

4.2 Decision Problems

We continue to decision problems for SD-tNCWs and SD-NCWs. As in Section 3.2, we state only the lower bounds, and for SD-tNCWs. Matching upper bounds and similar results for SD-NCWs follow from the known upper bounds for tNCWs and the known linear translations between state-based and transition-based automata.

We start with language-containment and universality.

▶ **Theorem 17.** *The language-containment and universality problems for SD-tNCWs are PSPACE-hard.*

Proof. We describe a reduction from universality of NFWs to universality of SD-tNCWs. Given an NFW $\mathcal{N}$ over Σ, the reduction returns the SD-tNCW $\mathcal{A}$ over $\Sigma \cup \{\$\}$ that is obtained by applying the operation from Theorem 12 on $\mathcal{N}$. Thus, $L(\mathcal{A}) = {\bowtie_\$}(L(\mathcal{N}))$.

First, if $L(\mathcal{N}) = \Sigma^*$, then $(\Sigma \cup \{\$\})^* \cdot (\$L(\mathcal{N}))^\omega = \infty\$$, and so $L(\mathcal{A}) = (\Sigma \cup \{\$\})^\omega$. Conversely, if there is a word $x \in \Sigma^* \setminus L(\mathcal{N})$, then the word $w = (\$x)^\omega$ is not in $L(\mathcal{A})$. Indeed, w has infinitely many \$'s, yet it has a word not in $L(\mathcal{N})$ between every two consecutive \$'s. ◀

We continue to the minimization problem. Note that while minimization is PSPACE-complete for NCWs, it is NP-complete for DCWs and in PTIME for HD-tNCWs. Thus, our PSPACE lower bound suggests a significant difference between SD and HD automata. Note also that, as has been the case with Büchi automata, the case of SD-NCWs is easy, as a non-empty SD-NCW is universal iff it has an equivalent SD-NCW with one state, which is not the case for SD-tNCWs (see proof in the full version):

▶ **Theorem 18.** *The minimization problem for SD-tNCWs is PSPACE-hard.*

We continue to the D-to-SD minimization problem, showing it stays PSPACE-hard.

▶ **Theorem 19.** *The D-to-SD minimization problem for co-Büchi automata is PSPACE-hard.*

Proof. We reduce from the D-to-N minimization problem for automata on finite words, which is already PSPACE-hard for languages that have bad infixes[19]. We start with co-Büchi automata with transition-based acceptance.

Consider the construction from Theorem 12. Recall it takes an NFW $\mathcal{A}$ as input, returns an SD-tNCW $\mathcal{A}'$ of the same size for ${\bowtie_\$}(L(\mathcal{A}))$, and preserves determinism.

Consider a DFW $\mathcal{A}$ over Σ such that $L(\mathcal{A})$ has a bad infix, and an integer k. The reduction returns the SD-tNCW $\mathcal{A}'$ constructed from $\mathcal{A}$ in Theorem 12, and the integer k. Recall that $L(\mathcal{A}') = \bowtie_{\$}(L(\mathcal{A}))$, and that the construction in Theorem 12 preserves determinism. Thus, the automaton $\mathcal{A}'$ is really a tDCW.

We prove next that the reduction is correct. That is, the DFW $\mathcal{A}$ has an equivalent NFW with at most k states iff the tDCW $\mathcal{A}'$ has an equivalent SD-tNCW with at most k states. For the first direction, if $\mathcal{B}$ is an NFW equivalent to $\mathcal{A}$ whose size is at most k, then, by applying to it the construction from Theorem 12, we get an SD-tNCW $\mathcal{B}'$ whose size is at most k, and $L(\mathcal{B}') = \bowtie_{\$}(L(\mathcal{B})) = \bowtie_{\$}(L(\mathcal{A})) = L(\mathcal{A}')$.

Conversely, if $\mathcal{B}'$ is an SD-tNCW for $\bowtie_{\$}(L(\mathcal{A}))$ whose size is at most k, then as $L(\mathcal{A})$ has bad infixes, we get by Theorem 13 that there is an NFW $\mathcal{B}$ for $L(\mathcal{A})$ whose size is at most k, and we are done.

In the full version, we extend the proof to co-Büchi automata with state-based acceptance. ◀

5 Semantically Deterministic Weak Automata

By [3], SD-NWWs need not be DBP or even HD. For completeness we describe here the example from [3], as it highlights the challenges in SD-NWW determinization. Consider the automaton $\mathcal{A}$ in Figure 2.

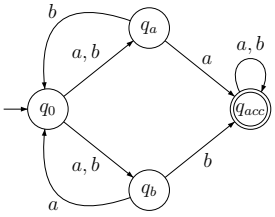


Figure 2 An SD-NWW that is not DBP.

It is easy to see that $\mathcal{A}$ is weak, and all its states are universal, and so it is SD. On the other hand, $\mathcal{A}$ is not HD as every strategy has a word with which it does not reach q_{acc} – a word that forces every visit in q_a and q_b to be followed by a visit in q_0. Below we show that despite not being DBP, SD-NWWs can be determinized in polynomial time. Essentially, our proof is based on redirecting transitions of the SD-NWW to deep components in the automaton. In our example, note that while the SD-NWW $\mathcal{A}$ is not HD, it has a deterministic state q_{acc} that recognizes $L(\mathcal{A})$.

Consider an SD-NWW $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$. We denote the set of $\mathcal{A}$'s SCCs by $\mathcal{C}(\mathcal{A})$, and the SCC containing a state q by $C(q)$. Let $C_1 \leq C_2 \leq \cdots \leq C_m$ be a total order on the SCCs of $\mathcal{A}$, extending the partial order induced by δ. That is, if $q' \in \delta(q, \sigma)$, for some letter σ, then $C(q) \leq C(q')$. When $C \leq C'$, we say that C' is *deeper than* C. Thus, states along runs of a weak automaton proceed from SCCs to deeper ones, and eventually get stuck in some SCC.

If we had an algorithm that checks language equivalence between states in $\mathcal{A}$ in polynomial time, we could have a polynomial determinization algorithm that defines the σ-successor of a state as the deepest state among all states equivalent to its σ-successors (since $\mathcal{A}$ is SD, all these successors agree on their language). Since we still do not have such an algorithm (in fact, it would follow from our construction), we approximate language equivalence by an equivalence that follows from the semantic determinism of $\mathcal{A}$.

For two states $s_1, s_2 \in Q$, we say that s_1 and s_2 are *δ-close*, if there is a state $q \in Q$ and a word $w \in \Sigma^*$ such that $s_1, s_2 \in \delta(q, w)$. Note that the δ-close relation refines equivalence, yet the converse does not hold. Indeed, the SDness property implies that $s_1 \sim_{\mathcal{A}} s_2$ for all δ-close states s_1 and s_2.

▶ **Lemma 20.** *If s_1 and s_2 are δ-close, then there is a state q and word w of length at most $|Q|^2$ such that $s_1, s_2 \in \delta(q, w)$.*

In order to calculate the δ-close relation in polynomial time, we define a sequence $H_0, H_1, H_2, \ldots \subseteq Q \times Q$ of relations, where $\langle s_1, s_2 \rangle \in H_i$ iff there is a state q and word w of length at most i such that $s_1, s_2 \in \delta(q, w)$. By Lemma 20 (see proof in the full version), we are guaranteed to reach a fixed point after at most $|Q|^2$ iterations. The relations H_i are defined as follows.

- $H_0 = \{\langle q, q \rangle : q \in Q\}$.
- For $i \geq 0$, we define $H_{i+1} = H_i \cup \{\langle s_1, s_2 \rangle :$ there is $\langle q_1, q_2 \rangle \in H_i$ and letter $\sigma \in \Sigma$ such that $s_1 \in \delta(q_1, \sigma)$ and $s_2 \in \delta(q_2, \sigma)\}$.

Let $j \geq 0$ be such that $H_{j+1} = H_j$. It is not hard to see that H_j is the δ-close relation. While the δ-close relation is reflexive and symmetric, it is not transitive. Now, let $H \subseteq Q \times Q$ be the closure of H_j under transitivity. That is, $\langle s_1, s_2 \rangle \in H$ iff there is $k \geq 2$ and states $q_1, q_2, \ldots, q_k$ such that $q_1 = s_1$, $q_k = s_2$ and $\langle q_i, q_{i+1} \rangle \in H_j$ for all $1 \leq i < k$.

The following lemma implies that H propagates to successor states (see proof in the full version):

▶ **Lemma 21.** *If $H(s, s')$, then for every letter $\sigma \in \Sigma$ and states $q \in \delta(s, \sigma)$ and $q' \in \delta(s', \sigma)$, we have that $H(q, q')$.*

It is easy to see that H is an equivalence relation. Let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be the set of the equivalence classes of H. For each equivalence class $P \in \mathcal{P}$, we fix the *representative of* P as some state in $P \cap C$, where $C \in \mathcal{C}(\mathcal{A})$ is the deepest SCC that intersects P. Let $p_1, \ldots, p_k$ be the representatives of the sets in $\mathcal{P}$. For a state $q \in Q$, let $\widetilde{q}$ denote the representative of the set $P \in \mathcal{P}$ with $q \in P$. Note that as H refines $\sim_{\mathcal{A}}$, we have that $q \sim_{\mathcal{A}} \widetilde{q}$.

▶ **Theorem 22.** *Given an SD-NWW $\mathcal{A}$ with state space Q, we can construct, in polynomial time, an equivalent DWW $\mathcal{D}$ with state space Q', for $Q' \subseteq Q$.*

Proof. Given $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, we define $\mathcal{D} = \langle \Sigma, Q', q'_0, \delta', \alpha \cap Q' \rangle$, where

- $Q' = \{p_1, \ldots, p_k\}$. Note that indeed $Q' \subseteq Q$.
- $q'_0 = \widetilde{q_0}$.
- For $p \in Q'$ and $\sigma \in \Sigma$, we define $\delta'(p, \sigma) = \tilde{q}$, for some $q \in \delta(p, \sigma)$. Note that all the states in $\delta(p, \sigma)$ are δ-close, and thus belong to the same set in $\mathcal{P}$. Hence, the choice of q is not important, as $\delta'(p, \sigma)$ is the representative of this set.

We prove that $\mathcal{D}$ is a DWW equivalent to $\mathcal{A}$. First, in order to see that $\mathcal{D}$ is weak, consider states p, p' such that $p' \in \delta'(p, \sigma)$. Thus, p' is the representative of a state $q \in \delta(p, \sigma)$. As $\mathcal{A}$ is weak, we have that $C(p) \leq C(q)$. As $p' = \tilde{q}$, we have that $C(q) \leq C(p')$. Hence $C(p) \leq C(p')$, and we are done.

We continue and prove that $L(\mathcal{A}) = L(\mathcal{D})$. We first prove that $L(\mathcal{A}) \subseteq L(\mathcal{D})$. Consider a word w and assume that $w \in L(\mathcal{A})$. Let $r = q_0, q_1, q_2, \ldots$ be an accepting run of $\mathcal{A}$ on w, and let $r' = s_0, s_1, s_2, \ldots$ be the run of $\mathcal{D}$ on w. If r' is accepting, we are done. Otherwise, namely if r' is rejecting, we point to a contradiction. Let $j \geq 0$ be the index in which r'

visits only states in $sinf(r')$. Note that all the states $s_j, s_{j+1}, s_{j+2}, \ldots$ belong to some SCC C of $\mathcal{A}$. Since we assume that r' is rejecting, and acceptance in $\mathcal{D}$ is inherited from $\mathcal{A}$, we get that C is a rejecting SCC of $\mathcal{A}$. We claim that all the runs of $\mathcal{A}^{s_j}$ on the suffix $w[j+1,\infty]$ of w are stuck in C. Thus, $w[j+1,\infty] \notin L(\mathcal{A}^{s_j})$. On the other hand, we claim that for all $i \geq 0$, we have that $q_i \sim_{\mathcal{A}} s_i$. Then, however, $w[j+1,\infty] \notin L(\mathcal{A}^{q_j})$, contradicting the fact that r is accepting:

The easy part is to prove that for all $i \geq 0$, we have that $q_i \sim_{\mathcal{A}} s_i$. Indeed, it follows from the fact that for all $i \geq 0$, we have that $H(q_i, s_i)$ and the fact that H refines $\sim_{\mathcal{A}}$. The proof proceeds by an induction on i. First, as for every state $q \in Q$, we have that $H(q, \widetilde{q})$, then $H(q_0, q'_0)$, and so the claim follows from s_0 being q'_0. Assume now that $H(q_i, s_i)$. Recall that s_{i+1} is $\widetilde{q}$ for some $q \in \delta(s_i, w[i+1])$. Also, $q_{i+1} \in \delta(q_i, w[i+1])$. Then, by Lemma 21, we have that $H(q_{i+1}, q)$. Since, in addition, we have that $H(q, \widetilde{q})$, then the transitivity of H implies that $H(q_{i+1}, s_{i+1})$, and we are done.

We proceed to the other part, namely, proving that all the runs of $\mathcal{A}^{s_j}$ on the suffix $w[j+1,\infty]$ of w are stuck in C. Let $u_{j+1}, u_{j+2}, \ldots$ be such that for all $i \geq 1$, it holds that $u_{j+i} \in \delta(s_{j+i-1}, w[j+i])$ and $H(u_{j+i}, s_{j+i})$. Note that such states exist, as $s_{j+i-1} \xrightarrow{w[j+i]} s_{j+i}$ is a transition of $\mathcal{D}$ and so $s_{j+i} = \widetilde{q}$ for all $q \in \delta(s_{j+i-1}, w[j+i])$. Consider a run $r'' = v_0, v_1, v_2, \ldots$ of $\mathcal{A}^{s_j}$ on $w[j+1,\infty]$. Note that $v_0 = s_j$, and so $H(v_0, s_j)$. Therefore, by Lemma 21, we have that $H(v_1, u_{j+1})$, implying $H(v_1, s_{j+1})$. By repeated applications of Lemma 21, we get that $H(v_i, u_{j+i})$, implying $H(v_i, s_{j+i})$, for all $i \geq 1$. Assume now by way of contradiction that the run r'' leaves the SCC C, and so there is $i \geq 1$ such that $C(v_i) \not\leq C(s_{j+i})$. As $H(v_i, s_{j+i})$, the states v_i and s_{j+i} are in the same equivalence class $P \in \mathcal{P}$. Then, the definition of Q' implies that $C(s_{j+i}) \geq C(v_i)$, and we have reached a contradiction.

It is left to prove that $L(\mathcal{D}) \subseteq L(\mathcal{A})$. Consider a word $w \in L(\mathcal{D})$. Let $r' = s_0, s_1, s_2, \ldots$ be the run of $\mathcal{D}$ on w, and let $j \geq 0$ be the index in which r' visits only states in $sinf(r')$; in particular, as argued above, $sinf(r')$ is included in some SCC C of $\mathcal{A}$. Thus, all the states $s_j, s_{j+1}, s_{j+2}, \ldots$ are in C. Since $w \in L(\mathcal{D})$, then r' is accepting, and so C is an accepting SCC of $\mathcal{A}$. As in the previous direction, it holds that s_j is $\mathcal{A}$-equivalent to all the states in $\delta(q_0, w[1,j])$. Hence, to conclude that $w \in L(\mathcal{A})$, we show that there is an accepting run of $\mathcal{A}^{s_j}$ on $w[j+1,\infty]$. Assume by way of contradiction that all the runs of $\mathcal{A}^{s_j}$ on $w[j+1,\infty]$ are rejecting. In particular, all these runs leave the SCC C. As in the previous direction, this contradicts the choice of the states $s_j, s_{j+1}, s_{j+2}, \ldots$ as representatives of equivalence classes in $\mathcal{P}$. ◀

Since DWWs can be complemented by dualization (that is, by switching α and $\bar{\alpha}$), Theorem 22 implies the following.

▶ **Theorem 23.** *Given an SD-NWW $\mathcal{A}$ with n states, we can construct, in polynomial time, an SD-NWW (in fact, a DWW) that complements $\mathcal{A}$.*

Since the language-containment problem for DWW can be solved in NLOGSPACE, and minimization for DWW is similar to minimization of DFWs and can be solved in polynomial time [31], we have the following.

▶ **Theorem 24.** *The language-containment, universality, and minimality problems for SD-NWWs can be solved in polynomial time.*

References

1 B. Abu Radi and O. Kupferman. Minimizing GFG transition-based automata. In *Proc. 46th Int. Colloq. on Automata, Languages, and Programming*, volume 132 of *LIPIcs*, pages 100:1–100:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

2 B. Abu Radi and O. Kupferman. On semantically-deterministic automata. *CoRR*, abs/2305.15489, 2023. `arXiv:2305.15489`.

3 B. Abu Radi, O. Kupferman, and O. Leshkowitz. A hierarchy of nondeterminism. In *46th Int. Symp. on Mathematical Foundations of Computer Science*, volume 202 of *LIPIcs*, pages 85:1–85:21, 2021.

4 B. Aminof, O. Kupferman, and R. Lampert. Reasoning about online algorithms with weighted automata. *ACM Transactions on Algorithms*, 6(2), 2010.

5 M. Bagnol and D. Kuperberg. Büchi good-for-games automata are efficiently recognizable. In *Proc. 38th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 122 of *LIPIcs*, pages 16:1–16:14, 2018.

6 B. Boigelot, S. Jodogne, and P. Wolper. On the use of weak automata for deciding linear arithmetic with integer and real variables. In *Proc. Int. Joint Conf. on Automated Reasoning*, volume 2083 of *Lecture Notes in Computer Science*, pages 611–625. Springer, 2001.

7 U. Boker, D. Kuperberg, O. Kupferman, and M. Skrzypczak. Nondeterminism in the presence of a diverse or unknown future. In *Proc. 40th Int. Colloq. on Automata, Languages, and Programming*, volume 7966 of *Lecture Notes in Computer Science*, pages 89–100, 2013.

8 U. Boker and K. Lehtinen. When a little nondeterminism goes a long way: an introduction to history-determinism. *ACM SIGLOG News*, 10(1):177–196, 2023.

9 J.R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Int. Congress on Logic, Method, and Philosophy of Science. 1960*, pages 1–12. Stanford University Press, 1962.

10 Olivier Carton and Max Michel. Unambiguous büchi automata. *Theoretical Computer Science*, 297(1):37–81, 2003.

11 T. Colcombet. The theory of stabilisation monoids and regular cost functions. In *Proc. 36th Int. Colloq. on Automata, Languages, and Programming*, volume 5556 of *Lecture Notes in Computer Science*, pages 139–150. Springer, 2009.

12 T. Colcombet, K. Quaas, and M. Skrzypczak. Unambiguity in automata theory (dagstuhl seminar 21452). *Dagstuhl Reports*, 11(10):57–71, 2021.

13 A. Duret-Lutz, A. Lewkowicz, A. Fauchille, Th. Michaud, E. Renault, and L. Xu. Spot 2.0 — a framework for LTL and ω-automata manipulation. In *14th Int. Symp. on Automated Technology for Verification and Analysis*, volume 9938 of *Lecture Notes in Computer Science*, pages 122–129. Springer, 2016.

14 P. Gastin and D. Oddoux. Fast LTL to Büchi automata translation. In *Proc. 13th Int. Conf. on Computer Aided Verification*, volume 2102 of *Lecture Notes in Computer Science*, pages 53–65. Springer, 2001.

15 D. Giannakopoulou and F. Lerda. From states to transitions: Improving translation of LTL formulae to Büchi automata. In *Proc. 22nd International Conference on Formal Techniques for Networked and Distributed Systems*, volume 2529 of *Lecture Notes in Computer Science*, pages 308–326. Springer, 2002.

16 E.M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak. Good-for-MDPs automata for probabilistic analysis and reinforcement learning. In *Proc. 26th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 12078 of *Lecture Notes in Computer Science*, pages 306–323. Springer, 2020.

17 T.A. Henzinger, O. Kupferman, and S. Rajamani. Fair simulation. *Information and Computation*, 173(1):64–81, 2002.

18 T.A. Henzinger and N. Piterman. Solving games without determinization. In *Proc. 15th Annual Conf. of the European Association for Computer Science Logic*, volume 4207 of *Lecture Notes in Computer Science*, pages 394–410. Springer, 2006.

19 T. Jiang and B. Ravikumar. Minimal NFA problems are hard. *SIAM Journal on Computing*, 22(6):1117–1141, 1993.

20 D. Kuperberg and M. Skrzypczak. On determinisation of good-for-games automata. In *Proc. 42nd Int. Colloq. on Automata, Languages, and Programming*, pages 299–310, 2015.

21 O. Kupferman. Avoiding determinization. In *Proc. 21st IEEE Symp. on Logic in Computer Science*, pages 243–254, 2006.

22 O. Kupferman. Automata theory and model checking. In *Handbook of Model Checking*, pages 107–151. Springer, 2018.

23 O. Kupferman. Using the past for resolving the future. *Frontiers in Computer Science*, 4, 2023.

24 O. Kupferman and O. Leshkowitz. On repetition languages. In *45th Int. Symp. on Mathematical Foundations of Computer Science*, Leibniz International Proceedings in Informatics (LIPIcs), 2020.

25 O. Kupferman, S. Safra, and M.Y. Vardi. Relating word and tree automata. *Ann. Pure Appl. Logic*, 138(1-3):126–146, 2006.

26 O. Kupferman and M.Y. Vardi. Safraless decision procedures. In *Proc. 46th IEEE Symp. on Foundations of Computer Science*, pages 531–540, 2005.

27 R.P. Kurshan. Complementing deterministic Büchi automata in polynomial time. *Journal of Computer and Systems Science*, 35:59–71, 1987.

28 R.P. Kurshan. *Computer Aided Verification of Coordinating Processes*. Princeton Univ. Press, 1994.

29 L.H. Landweber. Decision problems for ω–automata. *Mathematical Systems Theory*, 3:376–384, 1969.

30 W. Li, Sh. Kan, and Z. Huang. A better translation from LTL to transition-based generalized Büchi automata. *IEEE Access*, 5:27081–27090, 2017.

31 C. Löding. Efficient minimization of deterministic weak ω-automata. *Information Processing Letters*, 79(3):105–109, 2001.

32 S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.

33 G. Morgenstern. Expressiveness results at the bottom of the ω-regular hierarchy. M.Sc. Thesis, The Hebrew University, 2003.

34 D.E. Muller, A. Saoudi, and P. E. Schupp. Weak alternating automata give a simple explanation of why most temporal and dynamic logics are decidable in exponential time. In *Proc. 3rd IEEE Symp. on Logic in Computer Science*, pages 422–427, 1988.

35 M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:115–125, 1959.

36 S. Safra. On the complexity of ω-automata. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 319–327, 1988.

37 S. Schewe. Beyond Hyper-Minimisation—Minimising DBAs and DPAs is NP-Complete. In *Proc. 30th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 400–411, 2010.

38 S. Schewe. Minimising good-for-games automata is NP-complete. In *Proc. 40th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 182 of *LIPIcs*, pages 56:1–56:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

39 S. Schewe, Q. Tang, and T. Zhanabekova. Deciding what is good-for-MDPs. *CoRR*, abs/2202.07629, 2022. `arXiv:2202.07629`.

40 S. Sickert, J. Esparza, S. Jaax, and J. Křetínský. Limit-deterministic Büchi automata for linear temporal logic. In *Proc. 28th Int. Conf. on Computer Aided Verification*, volume 9780 of *Lecture Notes in Computer Science*, pages 312–332. Springer, 2016.

41 M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.

Checking Refinement of Asynchronous Programs Against Context-Free Specifications

Pascal Baumann
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Moses Ganardi
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Rupak Majumdar
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Ramanathan S. Thinniyam
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Georg Zetzsche
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Abstract

In the language-theoretic approach to refinement verification, we check that the language of traces of an implementation all belong to the language of a specification. We consider the refinement verification problem for asynchronous programs against specifications given by a Dyck language. We show that this problem is EXPSPACE-complete – the same complexity as that of language emptiness and for refinement verification against a regular specification. Our algorithm uses several technical ingredients. First, we show that checking if the coverability language of a succinctly described vector addition system with states (VASS) is contained in a Dyck language is EXPSPACE-complete. Second, in the more technical part of the proof, we define an ordering on words and show a downward closure construction that allows replacing the (context-free) language of each task in an asynchronous program by a regular language. Unlike downward closure operations usually considered in infinite-state verification, our ordering is not a well-quasi-ordering, and we have to construct the regular language ab initio. Once the tasks can be replaced, we show a reduction to an appropriate VASS and use our first ingredient. In addition to the inherent theoretical interest, refinement verification with Dyck specifications captures common practical resource usage patterns based on reference counting, for which few algorithmic techniques were known.

2012 ACM Subject Classification Theory of computation → Concurrency; Software and its engineering → Software verification

Keywords and phrases Asynchronous programs, VASS, Dyck languages, Language inclusion, Refinement verification

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.110

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: https://arxiv.org/abs/2306.13058

Funding Funded by the European Union (ERC, FINABIS, 101077902). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them. Partially funded by the DFG project 389792660 TRR 248–CPEC.

1 Introduction

Asynchronous programs are a common programming idiom for multithreaded shared memory concurrency. An asynchronous program executes tasks atomically; each task is a sequential recursive program that can read or write some shared state, emit events (such as calling an API), and, in addition, can spawn an arbitrary number of new tasks for future execution. A cooperative scheduler iteratively picks a previously spawned task and executes it atomically to completion. Asynchronous programs occur in many software systems with stringent correctness requirements. At the same time, they form a robustly decidable class of infinite-state systems closely aligned with other concurrency models. Thus, algorithmic verification of asynchronous programs has received a lot of attention from both theoretical and applied perspectives [25, 13, 10, 8, 9, 15, 11, 16, 20].

We work in the language-theoretic setting, where we treat asynchronous programs as generators of languages, and reduce verification questions to decision problems on these languages. Thus, an execution of a task yields a word over the alphabet of its events and task names. An execution of the asynchronous program concatenates the words of executing tasks and further ensures that any task executing in the concatenation was spawned before and not already executed. The trace of an execution projects the word to the alphabet of events and the language of the program is the set of all traces. With this view, reachability or safety verification questions reduce to language emptiness, and refinement verification reduces to language inclusion of a program in a given specification language over the alphabet of events.

We consider the language inclusion problem for asynchronous programs when the specification language is given by a Dyck language. Our main result shows that this problem is EXPSPACE-complete. The language emptiness problem for asynchronous programs, as well as language inclusion in a regular language, are already EXPSPACE-complete [10]. Thus, there is no increase in complexity even when the specifications are Dyck languages. However, as we shall see below, our proof of membership in EXPSPACE requires several new ingredients.

In addition to the inherent language-theoretic interest, the problem is motivated by the practical "design pattern" of reference counting and barrier synchronization in concurrent event-driven programs. In this pattern, each global shared resource maintains a counter of *how many* processes have access to it. Before working with the shared resource, a task acquires access to the resource by incrementing a counter (the reference count). Later, a possibly different task can release the resource by decrementing the reference count. When the count is zero, the system can garbage collect the resource. For example, device drivers in the kernel maintain such reference counts, and there are known bugs arising out of incorrect handling of reference counts [21]. Here is a small snippet that shows the pattern in asynchronous code:

```
start :   { t := inc(); if (t) spawn(work); }
          // arbitrarily many requests may start concurrently
work :    { in this code, we can assert that the reference count is positive ;
            spawn(cleanup); }
cleanup : { dec(); if zeroref() { garbage collect the resource }}
```

Here, `inc` and `dec` increment and decrement the reference count associated with a shared resource, `inc` succeeds if the resource has not been garbage collected. `spawn` starts a new task, and `zeroref` checks if the reference count is zero. There are three tasks, `start`, `work`, and `cleanup`; each invocation of a task executes atomically. Initially, an arbitrary number of `start` tasks are spawned.

Our goal is to ensure the device is not garbage collected while some instance of `work` is pending. Intuitively, the reason for this is clear: each `work` is spawned by a previous `start` that takes a reference count and this reference is held until a later `cleanup` runs. However,

it is difficult for automated model checking tools to perform this reasoning, and existing techniques require manual annotations of invariants [11, 15]. Dyck languages allow specifying correct handling of reference counts [1], and our algorithm provides as a special case an algorithmic analysis of correct reference counting for asynchronous programs.

Since there is a simple reduction from language emptiness to inclusion, we immediately inherit EXPSPACE-hardness. Let us therefore focus on the challenges in obtaining an EXPSPACE upper bound. The EXPSPACE algorithm for language emptiness proceeds as follows (see [10, 20]). First, we can ignore the alphabet of events and only consider words over the alphabet of task names. Second, we notice that (non-)emptiness is preserved if we "lose" some spawns along an execution; this allows us to replace the language of each task by its downward closure. By general results about well-quasi orderings, the downward closure is a regular language which, moreover, has a succinct representation. Thus, we can reduce the language emptiness problem to checking (coverability) language emptiness of an associated vector addition system with states (VASS). This problem can be solved in EXPSPACE, by a result of Rackoff [22].

Unfortunately, this outline is not sufficient in our setting. First, unlike for language emptiness or regular language inclusion, we cannot simply replace tasks with their downward closures (w.r.t. the subword ordering). While we can drop spawns as before, dropping letters from the event alphabet does not preserve membership in a Dyck language. Second, even if each handler is regular, we are left with checking if a VASS language is contained in a Dyck language. We provide new constructions to handle these challenges.

Our starting point is the characterization of inclusion in Dyck languages [23]: A language L is not included in a Dyck language if and only if there is a word $w \in L$ with either an *offset violation* (number of open brackets does not match the number of closed brackets), a *dip violation* (some prefix with more closed brackets than open ones), or a *mismatch violation* (an open bracket of one kind matched with a closed bracket of a different kind).

Checking VASS Language Inclusion. Our first technical construction shows how to check language inclusion of a VASS coverability language in a Dyck language in EXPSPACE. (In a *coverability language*, acceptance is defined by reaching a final control state.) In fact, our result carries over when the control states of the VASS are *succinctly represented*, for example by using transducers and binary encodings of numbers.

We first check that the VASS language is *offset-uniform*, that is, every word in the language has exactly the same offset (difference between open brackets and closed brackets), and that this offset is actually zero. (If this condition is not true, there is already an offset violation.) We show that the offset of every prefix of a word in any offset-uniform VASS language is bounded by a doubly exponential number, and therefore, this number can be tracked by adding double exponentially bounded counters (as in Lipton's construction [18]) in the VASS itself. Moreover, we can reduce the checking of dip or mismatch violations to finding a *marked Dyck factor*: an infix of the form $\#w\bar{\#}$ for a Dyck word w. Finally, for offset-uniform VASS, finding a marked Dyck factor reduces to coverability in succinctly represented VASS, which can be checked in EXPSPACE [2]. Offset uniformity is important – finding a marked Dyck factor in an arbitrary VASS language is equivalent to VASS reachability, which is Ackermann-complete [7, 17]. In fact, checking whether a given VASS language is included in the set of *prefixes* of the one-letter Dyck language is already equivalent to VASS reachability (see the long version of the paper for a proof).

A consequence of our result is that given a VASS coverability language K and a *reachability language* (i.e. acceptance requires all counters to be zero in the end) L of a *deterministic* VASS, deciding whether $K \subseteq L$ is EXPSPACE-complete. This is in contrast (but not in contradiction[1]) to recent Ackermann-completeness results for settings where both K and L are drawn from subclasses of VASS coverability languages [6].

Downward Closure of Tasks. Next, we move to asynchronous programs. We define a composite ordering on words that is a combination of two different orderings: the subword ordering for task names, and the syntactic preorder on the events projected to a single set $\{x, \bar{x}\}$ of Dyck letters. In our case, the latter means a word u is less than v iff they both have the same offset, but v has at most the dip of u. The composite order is defined so as to preserve the existence of marked Dyck factors. In contrast to the subword ordering, this (composite) ordering is not a well-quasi-ordering (since, e.g., $\bar{x}x, \bar{x}\bar{x}xx, \bar{x}\bar{x}\bar{x}xxx, \ldots$ forms an infinite descending chain). Nevertheless, our most difficult technical construction shows that for any context-free language (satisfying an assumption, which we call tame-pumping) there exists a regular language with the same downward closure in this ordering. The case of general context-free languages reduces to this special case since the presence of a non-tame pump immediately results in a Dyck-violation and can easily be detected in PSPACE. For the tame-pumping grammars, a succinct description of the corresponding automaton can be computed in PSPACE. This key observation allows us to replace the context-free languages of tasks with regular sets, and thereby reduce the problem to checking VASS language inclusion.

Related Work. Language inclusion in Dyck languages is a well-studied problem. For example, inclusion in a Dyck language can be checked in polynomial time for context-free languages [26] or for ranges of two-copy tree-to-string transducers [19]. Our work extends the recent result that the language noninclusion problem for context-bounded multi-pushdown systems in Dyck languages is NP-complete [1]. Our result is complementary to that of [1]: their model considers a *fixed* number of threads but allows the threads to be interrupted and context-switched a fixed number of times. In contrast, we allow dynamic spawning of threads but assume each thread is atomically run to completion. A natural open question is whether our results continue to hold if threads can be interrupted up to a fixed number of times.

Inclusion problems have recently also been studied when both input languages are given as VASS coverability languages [6]. Since in our setting, the supposedly larger language is always a Dyck language (which is not a coverablity VASS language), those results are orthogonal.

2 Language-Theoretic Preliminaries

General Definitions. We assume familiarity with basic language theory, see the textbook [14] for more details. For an alphabet $\Sigma \subseteq \Theta$, let $\pi_\Sigma \colon \Theta^* \to \Sigma^*$ denote the projection onto Σ^*. In other words, for $w \in \Theta^*$, the word $\pi_\Sigma(w)$ is obtained from w by deleting every occurrence of a letter in $\Theta \setminus \Sigma$. If Σ contains few elements, e.g. $\Sigma = \{x, y\}$, then instead of writing $\pi_{\{x,y\}}$ we also write $\pi_{x,y}$, leaving out the set brackets. We write $|w|_\Sigma$ for the number of occurrences of letters $x \in \Sigma$ in w, and similarly $|w|_x$ if $\Sigma = \{x\}$.

[1] For general VASS, every coverability language is also a reachability language. However, *deterministic* VASS with reachability acceptance cannot accept all coverability languages.

Context-Free Languages. A *context-free grammar* (CFG) $\mathcal{G} = (N, \Theta, P, S)$ consists of an alphabet of *nonterminals* N, an alphabet of *terminals* Θ with $N \cap \Theta = \emptyset$, a finite set of *productions* $P \subseteq N \times (N \cup \Theta)^*$, and the start symbol $S \in N$. We usually write $A \to v$ to denote a production $(A, v) \in P$. The size of the CFG $\mathcal{G}$ is defined as $|\mathcal{G}| = \sum_{A \to v \in P}(|v| + 1)$. We denote the *derivation relation* by $\Rightarrow_{\mathcal{G}}$ and its reflexive, transitive closure by $\overset{*}{\Rightarrow}_{\mathcal{G}}$. We drop the subscript $\mathcal{G}$ if it is clear from the context. We also use *derivation trees* labelled by $N \cup \Theta$ for derivations of the form $A \overset{*}{\Rightarrow} w$ for some $A \in N$. Here we start with the root labelled by A, and whenever we apply a production $B \to v$ with $v = a_1 \dots a_n$, we add n children labelled by $a_1, \dots, a_n$ (in that order from left to right) to a leaf labelled by B. A *pump* is a derivation of the form $A \overset{*}{\Rightarrow} uAv$ for some nonterminal A. A derivation tree which is pumpfree, i.e., in which no path contains multiple occurrences of the same nonterminal, is referred to as a *skeleton*. We will often see an arbitrary derivation tree as one which is obtained by inserting pumps into a skeleton.

The *language* $L(\mathcal{G}, A)$ of $\mathcal{G}$ starting from nonterminal $A \in N$ contains all words $w \in \Theta^*$ such that there exists a derivation $A \overset{*}{\Rightarrow}_{\mathcal{G}} w$. The language of $\mathcal{G}$ is $L(\mathcal{G}) = L(\mathcal{G}, S)$. A *context-free language* (CFL) L is a language for which there exists a CFG $\mathcal{G}$ with $L = L(\mathcal{G})$.

A CFG $\mathcal{G} = (N, \Theta, P, S)$ is said to be in *Chomsky normal form* if all of its productions have one of the forms $A \to BC$, $A \to a$, or $S \to \varepsilon$, where $B, C \in N \setminus \{S\}$, $a \in \Theta$, and the last form only occurs if $\varepsilon \in L(\mathcal{G})$. It is well known that every CFG can be transformed in polynomial time into one in Chomsky normal form with the same language.

An *extended* context-free grammar (ECFG) $\mathcal{G} = (N, \Theta, P, S)$ is a CFG, which may additionally have productions of the form $A \to \Gamma^* \in P$ for some alphabet $\Gamma \subseteq \Theta$. Productions of this form induce derivations $uAs \Rightarrow_{\mathcal{G}} uvs$, where $u, s \in (N \cup \Theta)^*$ and $v \in \Gamma^*$. Chomsky normal form for ECFG is defined as for CFG, but also allows productions of the form $A \to \Gamma^*$. An ECFG can still be transformed into Chomsky normal form using the same algorithm as for a CFG, treating expressions Γ^* like single terminal symbols. Since the extended productions can be simulated by conventional CFG productions, the language of an ECFG is still a CFL.

Dyck Language. Let X be an alphabet and let $\bar{X} = \{\bar{x} \mid x \in X\}$ be a disjoint copy of X. The *Dyck language (over X)* $\mathsf{Dyck}_X \subseteq (X \cup \bar{X})^*$ is defined by the following context-free grammar:

$$S \to \varepsilon \mid S \to SS \mid S \to xS\bar{x} \quad \text{for } x \in X.$$

Let $\Theta \supseteq X \cup \bar{X}$ be an alphabet. For $w \in \Theta^*$ we define $\mathsf{offset}(w) = |w|_X - |w|_{\bar{X}}$. A language $L \subseteq \Theta^*$ is called *offset-uniform* if for any $u, v \in L$, we have $\mathsf{offset}(u) = \mathsf{offset}(v)$.

The *dip* of $w \in \Theta^*$ is defined as $\mathsf{dip}(w) = \max\{-\mathsf{offset}(u) \mid u \text{ is a prefix of } w\}$. We define $e(w) = (\mathsf{dip}(w), \mathsf{offset}(w))$. Observe that for $w \in (X \cup \bar{X})^*$ with $|X| = 1$ we have $w \in \mathsf{Dyck}_X$ if and only if $e(w) = (0, 0)$.

A language $L \subseteq (X \cup \bar{X})^*$ is *not* included in Dyck_X if and only if there exists a word $w \in L$ that satisfies one of the following violation conditions [23]:

(OV) an *offset violation* $\mathsf{offset}(w) \neq 0$,

(DV) a *dip violation*, where $\mathsf{dip}(w) > 0$, i.e., there is a prefix u of w with $\mathsf{offset}(u) < 0$, or

(MV) a *mismatch violation*, where there exists a pair $x, \bar{y}$ (for some $x \neq y$) of *mismatched* letters in w, i.e., w contains an infix $xv\bar{y}$ where $e(v) = (0, 0)$.

For example, $w_1 = x\bar{x}\bar{x}x$ has a dip violation due to the prefix $u = x\bar{x}\bar{x}$; $w_2 = xx\bar{x}$ has an offset violation and $w_3 = xx\bar{x}\bar{y}$ has a mismatch violation.

3 Asynchronous Programs

An *asynchronous program* [10], henceforth simply called a *program*, is a tuple $\mathscr{P} = (Q, \Sigma, \Gamma, \mathcal{G}, \Delta, q_0, q_f, \gamma_0)$, where Q is a finite set of *global states*, Σ is an alphabet of *event letters*, Γ is an alphabet of *handler names* with $\Sigma \cap \Gamma = \emptyset$, $\mathcal{G}$ is a CFG over the terminal symbols $\Sigma \cup \Gamma$, Δ is a finite set of transition rules (described below), $q_0 \in Q$ is the *initial state*, $q_f \in Q$ is the *final state*, and γ_0 is the *initial handler*.

Transition rules in Δ are of the form $q \xhookrightarrow{a,A} q'$, where $q, q' \in Q$ are global states, $a \in \Gamma$ is a handler name, and A is a nonterminal symbol in $\mathcal{G}$.

Let $\mathbb{M}[S]$ denote the set of all multisets of elements from the set S. A *configuration* $(q, \mathbf{m}) \in Q \times \mathbb{M}[\Gamma]$ of $\mathscr{P}$ consists of a global state q and a multiset $\mathbf{m} : \Gamma \to \mathbb{N}$ of pending handler instances. The *initial* configuration of $\mathscr{P}$ is $c_0 = (q_0, [\![\gamma_0]\!])$, where $[\![\gamma_0]\!]$ denotes the singleton multiset containing γ_0. A configuration is considered *final* if its global state is q_f. The rules in Δ induce a transition relation on configurations of $\mathscr{P}$: We have $(q, \mathbf{m}) \xrightarrow{w} (q', \mathbf{m}')$ iff there is a rule $q \xhookrightarrow{a,A} q' \in \Delta$ and a word $u \in L(\mathcal{G}, A)$ such that $\pi_\Sigma(u) = w$ and $\mathbf{m}' = (\mathbf{m} \ominus [\![a]\!]) \oplus \mathsf{Parikh}(\pi_\Gamma(u))$, where $\mathbf{m}'' = \mathbf{m} \oplus \mathbf{m}'$ is the multiset which satisfies $\mathbf{m}''(a) = \mathbf{m}'(a) + \mathbf{m}(a)$ for each $a \in \Gamma$. Similarly $\mathbf{m}'' = \mathbf{m} \ominus \mathbf{m}'$ is the multiset which satisfies $\mathbf{m}''(a) = \mathbf{m}'(a) - \mathbf{m}(a)$ for each $a \in \Gamma$ with the implicit assumption that $\mathbf{m}'(a) \geq \mathbf{m}(a)$. Here, $\mathsf{Parikh}(w) : \Gamma \to \mathbb{N}$ is the *Parikh image* of w that maps each handler in Γ to its number of occurrences in w. Note that the transition is feasible only if $\mathbf{m}$ contains at least one instance of the handler a.

Intuitively, a program consists of a set of atomic event handlers that communicate over a shared global state Q. Each handler is a piece of sequential code that generates a word over a set of events Σ and, in addition, posts new instances of handlers from Γ. A configuration $(q, \mathbf{m})$ represents the current value of the shared state q and a task buffer $\mathbf{m}$ containing the posted, but not yet executed, handlers. At each step, a scheduler non-deterministically picks and removes a handler from the multiset of posted handlers and "runs" it. Running a handler changes the global state and produces a sequence of events over Σ as well as a multiset of newly posted handlers. The newly posted handlers are added to the task buffer.

We consider asynchronous programs as generators of words over the set of events. A *run* of $\mathscr{P}$ is a finite sequence of configurations $c_0 = (q_0, [\![\gamma_0]\!]) \xrightarrow{w_1} c_1 \xrightarrow{w_2} \ldots \xrightarrow{w_\ell} c_\ell$. It is an *accepting* run if it ends in a final configuration.

The *language* of $\mathscr{P}$ is defined as

$$L(\mathscr{P}) = \{w \in \Sigma^* \mid w = w_1 \cdots w_\ell, \text{ there is an accepting run } c_0 \xrightarrow{w_1} \ldots \xrightarrow{w_\ell} c_\ell\}.$$

The size of the program $\mathscr{P}$ is defined as $|\mathscr{P}| = |Q| + |\mathcal{G}| + |\Delta|$, i.e., the combined size of states, grammar, and transitions.

The *Dyck inclusion problem* for programs asks, given a program $\mathscr{P}$ over a set $(X \cup \bar{X})$ of events, whether every word in $L(\mathscr{P})$ belongs to the Dyck language Dyck_X. We show the following main result.

▶ **Theorem 3.1** (Main Theorem). *Given a program $\mathscr{P}$ with $L(\mathscr{P}) \subseteq (X \cup \bar{X})^*$, deciding if $L(\mathscr{P}) \subseteq \mathsf{Dyck}_X$ is* EXPSPACE*-complete.*

EXPSPACE-hardness follows easily from the following result on language emptiness (by simply adding a loop with a letter $\bar{x} \in \bar{X}$ at the final state). Therefore, the rest of the paper focuses on the EXPSPACE upper bound.

▶ **Proposition 3.2** (Theorem 6.2, Ganty and Majumdar [10]). *Given a program $\mathscr{P}$, checking if $L(\mathscr{P}) = \emptyset$ is* EXPSPACE*-complete.*

A nonterminal B in the grammar $\mathcal{G}$ of a program $\mathscr{P}$ is called *useful* if there exists a run ρ of $\mathscr{P}$ reaching q_f in which there exists a derivation tree containing B. More precisely, there are two successive configurations $(q, \mathbf{m}) \xrightarrow{w} (q', \mathbf{m}')$ in ρ such that there is a rule $q \xhookrightarrow{a,A} q'$ and a word $u \in L(\mathcal{G}, A)$ with $\pi_\Sigma(u) = w$, $\mathbf{m}' = (\mathbf{m} \ominus [\![a]\!]) \oplus \mathsf{Parikh}(\pi_\Gamma(u))$, and B occurs in some derivation tree with root A and yield u. There is a simple reduction from checking if a nonterminal is useful to checking language emptiness (see the full version) so we can check if a nonterminal is useful also in EXPSPACE. Therefore, in the following, we shall assume that all nonterminals are useful.

4 Checking Dyck Inclusion for VASS Coverability Languages

As a first technical construction, we show how to check Dyck inclusion for (succinctly defined) VASS languages. We shall reduce the problem for programs to this case.

4.1 Models: VASS and Succinct Versions

Vector Addition Systems with States. A *vector addition system with states* (VASS) is a tuple $\mathcal{V} = (Q, \Sigma, I, E, q_0, q_f)$ where Q is a finite set of *states*, Σ is a finite alphabet of *input letters*, I is a finite set of *counters*, $q_0 \in Q$ is the *initial state*, $q_f \in Q$ is the *final state*, and E is a finite set of *edges* of the form $q \xrightarrow{x,\delta} q'$, where $q, q' \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, and $\delta \in \{-1, 0, 1\}^I$.[2]

A *configuration* of $\mathcal{V}$ is a pair $(q, \mathbf{u}) \in Q \times \mathbb{M}[I]$. The elements of $\mathbb{M}[I]$ and $\{-1, 0, 1\}^I$ can also be seen as vectors of length $|I|$ over $\mathbb{N}$ and $\{-1, 0, 1\}$, respectively, and we sometimes denote them as such. The edges in E induce a transition relation on configurations: there is a transition $(q, \mathbf{u}) \xrightarrow{x} (q', \mathbf{u}')$ if there is an edge $q \xrightarrow{x,\delta} q'$ in E such that $\mathbf{u}'(i) = \mathbf{u}(i) + \delta(i) \geq 0$ for all $i \in I$. A *run* of the VASS is a finite sequence of configurations $c_0 \xrightarrow{x_1} c_1 \xrightarrow{x_2} \dots \xrightarrow{x_\ell} c_\ell$ where $c_0 = (q_0, \mathbf{0})$. A run is said to reach a state $q \in Q$ if the last configuration in the run is of the form $(q, \mathbf{m})$ for some multiset $\mathbf{m}$. An *accepting* run is a run whose final configuration has state q_f. The *(coverability) language* of $\mathcal{V}$ is defined as

$$L(\mathcal{V}) = \{w \in \Sigma^* \mid \text{there exists a run } (q_0, \mathbf{0}) = c_0 \xrightarrow{x_1} \dots \xrightarrow{x_\ell} c_\ell = (q_f, \mathbf{u}) \text{ with } w = x_1 \cdots x_\ell\}.$$

The size of the VASS $\mathcal{V}$ is defined as $|\mathcal{V}| = |I| \cdot |E|$.

Models with Succinct Control. In this paper we need various models with *doubly succinct* control, i.e., models with doubly exponentially many states. Informally speaking, a machine with finite control $\mathcal{B}$, e.g. an NFA or a VASS, is doubly succinct if its set of control states is Λ^M where $M \in \mathbb{N}$ is an exponential number given in binary encoding, and Λ is a finite alphabet. The initial and final state of $\mathcal{B}$ are the states 0^M and 1^M for some letters $0, 1 \in \Lambda$. Finally, the transitions of $\mathcal{B}$ are given by *finite-state transducers* $\mathcal{T}$, i.e., asynchronous multitape automata recognizing relations $R \subseteq (\Lambda^M)^k$. For example, a *doubly succinct* NFA (dsNFA in short) contains binary transducers $\mathcal{T}_a$ for each $a \in \Sigma \cup \{\varepsilon\}$ where Σ is the input alphabet, and $\mathcal{B}$ contains a transition $p \xrightarrow{x} q$ if and only if (p, q) is accepted by $\mathcal{T}_x$. A *doubly succinct VASS* (dsVASS, for short) contains binary transducers $\mathcal{T}_{x,i}, \mathcal{T}_{x,\bar{i}}, \mathcal{T}_{x,\varepsilon}$ for each $x \in \Sigma \cup \{\varepsilon\}$ and $i \in I$, where I is the set of counters. A state pair (p, q) accepted by $\mathcal{T}_{x,i}$ specifies

[2] A more general definition of VASS would allow each transition to add an arbitrary vector over the integers. We instead restrict ourselves to the set $\{-1, 0, 1\}$, since this suffices for our purposes, and the EXPSPACE-hardness result by Lipton [18] already holds for VASS of this form.

a transition $p \xrightarrow{x,\mathbf{e}_i} q$ in $\mathcal{B}$, where $\mathbf{e}_i$ only increments counter i and leaves other counters the same. Similarly $\mathcal{T}_{x,\bar{i}}$ and $\mathcal{T}_{x,\varepsilon}$ specify decrementing transitions and transitions without counter updates.

Later we will also use *(singly) succinct* ECFGs, which are extended context-free grammars whose set of nonterminals is Λ^M where M is a unary encoded number. The set of productions is given in a suitable fashion by transducers. Let us remark that the precise definition of (doubly) succinct automata or grammars is not important for our paper, e.g. one could also use circuits instead of transducers to specify the transitions/productions.

4.2 Checking Dyck Inclusion for dsVASS

We prove our first technical contribution: an EXPSPACE procedure to check non-inclusion of a VASS language in a Dyck language. This involves checking if one of (OV), (DV), or (MV) occurs. We begin by showing how these violations can be detected for a (non-succinct) VASS.

To this end, first we show that offset-uniformity of a VASS language implies a doubly exponential bound B on the offset values for prefixes of accepted words (Theorem 4.1). Given an alphabet X and a number $k \in \mathbb{N}$, we define the language

$$\mathscr{B}(X,k) = \{w \in (X \cup \bar{X})^* \mid \text{for every prefix } v \text{ of } w\colon\ |\mathsf{offset}(v)| \leq k\}.$$

▶ **Theorem 4.1.** *Let $\mathcal{V}$ be a* VASS *with $L(\mathcal{V}) \subseteq (X \cup \bar{X})^*$. If $L(\mathcal{V})$ is offset-uniform, then $L(\mathcal{V}) \subseteq \mathscr{B}(X, 2^{2^{p(|\mathcal{V}|)}})$ for some polynomial function p.*

Proof. Let $\mathcal{V} = (Q, X \cup \bar{X}, I, E, q_0, q_f)$ be a VASS where $L(\mathcal{V}) \neq \emptyset$ is offset-uniform. The unique offset of $L(\mathcal{V})$ is bounded double exponentially in $|\mathcal{V}|$ since $L(\mathcal{V})$ contains some word that is at most double exponentially long, a fact that follows from Rackoff's bound on covering runs [22]. Let $C \subseteq Q \times \mathbb{M}[I]$ be the set of configurations that are reachable from $(q_0, \mathbf{0})$ and from which the final state can be reached. Observe that for any configuration $c \in C$ the language $L(c) = \{w \in (X \cup \bar{X})^* \mid \exists \mathbf{u}\colon c \xrightarrow{w} (q_f, \mathbf{u})\}$ is also offset-uniform since $L(c) \subseteq \{w \in (X \cup \bar{X})^* \mid vw \in L(\mathcal{V})\}$ where $v \in (X \cup \bar{X})^*$ is any word with $(q_0, \mathbf{0}) \xrightarrow{v} c$. Define the function $f\colon C \to \mathbb{Z}$ where $f(c)$ is the unique offset of the words in $L(c)$. It remains to show that $|f(c)|$ is bounded double exponentially for all $c \in C$.

Let M be the set of all configurations from which the final state can be reached (hence $C \subseteq M$). Consider the following order on VASS configurations $Q \times \mathbb{M}[I]$: $(q, \mathbf{u}) \leq (q', \mathbf{u}')$ iff $q = q'$ and $\mathbf{u}(i) \leq \mathbf{u}'(i)$ for each $i \in I$. The cardinality of the set $\min(M)$ of minimal elements in M with respect to this order is bounded doubly exponentially in the size of $\mathcal{V}$. This follows directly from the fact that Rackoff's doubly-exponential bound [22] on the length of a covering run does not depend on the start configuration (but only the size of the VASS and the final configuration). An explicit bound for $|\min(M)|$ is given in [4, Theorem 2].

Observe that if $c_1 \in M$ and $c_2 \in C$ with $c_1 \leq c_2$ then $L(c_1) \subseteq L(c_2)$ and therefore $L(c_1)$ is also offset-uniform, having the same offset as $L(c_2)$. Hence, if for two configurations $c_1, c_2 \in C$ there exists a configuration $c \in M$ with $c \leq c_1$ and $c \leq c_2$, then $f(c_1) = f(c_2)$. Since for every $c_2 \in C$ there exists $c_1 \in \min(M)$ with $c_1 \leq c_2$, the function f can only assume doubly exponentially many values on C.

Finally, we claim that $f(C) \subseteq \mathbb{Z}$ is an interval containing 0, which proves that the norms of elements in $f(C)$ are bounded by the number of different values, i.e., double exponentially. Since we assumed $L(\mathcal{V}) \neq \emptyset$, some final configuration $(q_f, \mathbf{u}) \in C$ is reachable from $(q_0, \mathbf{0})$, and therefore $0 \in f(C)$ since $\varepsilon \in L((q_f, \mathbf{u}))$. Consider the configuration graph $\mathcal{C}$ of $\mathcal{V}$ restricted to C. For any edge $c_1 \to c_2$ in $\mathcal{C}$ we have $|f(c_1) - f(c_2)| \leq 1$ since VASS transitions consume at most one input symbol. Moreover, the underlying undirected graph of $\mathcal{C}$ is connected since any configuration is reachable from $(q_0, \mathbf{0}) \in C$. Therefore $f(C)$ is an interval, which concludes the proof. ◀

Note that although $\mathscr{B}(X,k)$ is a regular language for each X and k, Theorem 4.1 does not imply that every offset-uniform VASS language is regular. For example, the VASS language $\{(x\bar{x})^m(y\bar{y})^n \mid m \geq n\}$ is offset-uniform, but it is not regular. This is because Theorem 4.1 only implies boundedness of the number of occurrences of letters in the input words, but the VASS's own counters might be unbounded.

The main consequence of Theorem 4.1 is that in a VASS we can track the offset using a doubly succinct control state. Thus, we have the following corollary.

▶ **Corollary 4.2.** *The following problems can be decided in* EXPSPACE*: Given a* VASS *or* dsVASS $\mathcal{V}$*, does* $\mathsf{offset}(w) = 0$ *hold for all* $w \in L(\mathcal{V})$*?*

Proof. First assume $\mathcal{V}$ is a VASS. We show that the problem can be reduced to the intersection non-emptiness problem for a VASS and a doubly succinct NFA, i.e., given a VASS $\mathcal{V}$ and a doubly succinct NFA $\mathcal{A}$, is the intersection $L(\mathcal{V}) \cap L(\mathcal{A})$ nonempty? One can construct in polynomial time a doubly succinct VASS for $L(\mathcal{V}) \cap L(\mathcal{A})$, as a product construction between $\mathcal{V}$ and $\mathcal{A}$. Since the emptiness problem for dsVASS is in EXPSPACE ([2, Theorem 5.1]), we can also decide emptiness of $L(\mathcal{V}) \cap L(\mathcal{A})$ in EXPSPACE.

Define the number $M = 2^{2^{p(|\mathcal{V}|)}}$ where p is the polynomial from Theorem 4.1. Let $K_0 = \{w \in (X \cup \bar{X})^* \mid \mathsf{offset}(w) = 0\}$. According to Theorem 4.1, we have $L(\mathcal{V}) \subseteq K_0$ if and only if $L(\mathcal{V}) \subseteq K_0 \cap \mathscr{B}(X,M)$. By the remarks above, it suffices to construct a doubly succinct NFA for the complement of $K_0 \cap \mathscr{B}(X,M)$. The following doubly succinct deterministic finite automaton $\mathcal{A}$ recognizes $K_0 \cap \mathscr{B}(X,M)$: Given an input word over $X \cup \bar{X}$, the automaton tracks the current offset in the interval $[-M,M]$, stored in the control state as a binary encoding of length $\log M = 2^{p(|\mathcal{V}|)}$ together with a bit indicating the sign. If the absolute value of the offset exceeds M, the automaton moves to a rejecting sink state. The state representing offset 0 is the initial and the only final state. Finally, we complement $\mathcal{A}$ to obtain a doubly succinct NFA $\bar{\mathcal{A}}$, with a unique final state, for the complement of $K_0 \cap \mathscr{B}(X,M)$.

Now assume $\mathcal{V}$ is a dsVASS. Using Lipton's construction simulating doubly exponential counter values [18], we can construct a (conventional) VASS $\mathcal{V}'$, size polynomial in $|\mathcal{V}|$, with the same language (similar to [2, Theorem 5.1]). We can now apply the above construction. ◀

Next, we check for (DV) or (MV), assuming offset uniformity. We will reduce both kinds of violations to the problem of searching for *marked Dyck factors*. A word of the form $u\#v\bar{\#}w$ is called a *marked Dyck factor* if $u,v,w \in \{x,\bar{x}\}^*$ and $v \in \mathsf{Dyck}_x$.

Intuitively, if a (DV) occurs in a word w, there is a first time that the offset reaches -1. Placing a $\bar{\#}$ at the place where this happens, and a $\#$ right at the beginning, we have a word of the form $\#u\bar{\#}v$ where $u \in \mathsf{Dyck}_x$. Similarly for (MV), we replace two letters $z \in X$ and $\bar{y} \in X$ with $z \neq y$ by $\#$ and $\bar{\#}$, respectively, and look for a word $u\#v\bar{\#}w$, where $v \in \mathsf{Dyck}_x$.

▶ **Proposition 4.3.** *The following problems can be decided in* EXPSPACE*: Given an offset-uniform* VASS *or* dsVASS $\mathcal{V}$*, does* $L(\mathcal{V})$ *contain a marked Dyck factor?*

Proof. As in Corollary 4.2, given a dsVASS, we can convert to a polynomial-sized VASS with the same language and apply the following algorithm.

We again reduce to the intersection nonemptiness problem between a VASS and a doubly succinct NFA, and use the fact that nonemptiness of dsVASS is in EXPSPACE [2, Theorem 5.1]. As above, define the number $M = 2^{2^{p(|\mathcal{V}|)}}$ where p is the polynomial from Theorem 4.1. The automaton keeps track of the offset and also verifies that the input has the correct format $u\#v\bar{\#}w$ where $u,v,w \in \{x,\bar{x}\}^*$. Furthermore, upon reaching $\#$ it starts tracking the current offset and verifies that (i) the offset stays nonnegative, (ii) the offset never exceeds $2M$,

and (iii) the offset is zero when reaching $\bar{\#}$. If $L(\mathcal{V})$ intersects $L(\mathcal{A})$, then clearly $\mathcal{V}$ is a positive instance of the problem. Conversely, assume that $L(\mathcal{V})$ contains a word $u\#v\bar{\#}w$ with $v \in \mathsf{Dyck}_x$. By offset-uniformity of $\mathcal{V}$ and by Theorem 4.1, each prefix v' of v satisfies $\mathsf{offset}(v') = \mathsf{offset}(uv') - \mathsf{offset}(u) \leq M - (-M) = 2M$. Therefore $u\#v\bar{\#}w \in L(\mathcal{A})$. ◀

Let us put everything together. Let $\rho\colon (X \cup \bar{X})^* \to \{x, \bar{x}\}^*$ be the morphism that replaces all letters from X (resp., $\bar{X}$) by the letter x (resp., $\bar{x}$). Given a dsVASS $\mathcal{V}$ over $X \cup \bar{X}$ we can construct in polynomial time three dsVASS $\mathcal{V}_\mathsf{o}, \mathcal{V}_\mathsf{d}, \mathcal{V}_\mathsf{m}$ where

$$L(\mathcal{V}_\mathsf{o}) = \rho(L(\mathcal{V})),$$
$$L(\mathcal{V}_\mathsf{d}) = \{\#\rho(v)\bar{\#}\rho(\bar{y}w) \mid v\bar{y}w \in L(\mathcal{V}) \text{ for some } v, w \in (X \cup \bar{X})^*,\ y \in X\},$$
$$L(\mathcal{V}_\mathsf{m}) = \{\rho(u)\#\rho(v)\bar{\#}\rho(w) \mid uyv\bar{z}w \in L(\mathcal{V}) \text{ for some } u, v, w \in (X \cup \bar{X})^*,\ y \neq z \in X\}.$$

Observe that $L(\mathcal{V}) \subseteq \mathsf{Dyck}_x$ if and only if $L(\mathcal{V}_\mathsf{o})$ has uniform offset 0 and $L(\mathcal{V}_\mathsf{d})$ and $L(\mathcal{V}_\mathsf{m})$ do not contain marked Dyck factors.

Hence, to decide whether $L(\mathcal{V}) \subseteq \mathsf{Dyck}_X$ we first test that $L(\mathcal{V}_\mathsf{o})$ has uniform offset 0, using Corollary 4.2, rejecting if not. Otherwise, we can apply Proposition 4.3 to test whether $L(\mathcal{V}_\mathsf{d})$ or $L(\mathcal{V}_\mathsf{m})$ contain marked Dyck factors. If one of the tests is positive, we know $L(\mathcal{V}) \not\subseteq \mathsf{Dyck}_X$, otherwise $L(\mathcal{V}) \subseteq \mathsf{Dyck}_X$.

▶ **Theorem 4.4.** *Given a* dsVASS $\mathcal{V}$ *over the alphabet* $X \cup \bar{X}$*, checking whether* $L(\mathcal{V}) \subseteq \mathsf{Dyck}_X$ *is* EXPSPACE*-complete.*

Let us remark that Theorem 4.4 can also be phrased slightly more generally. Above, we have defined the language of a VASS to be the set of input words for which a final state is reached. Such languages are also called *coverability languages*. Another well-studied notion is the *reachability language* of a VASS, which consists of those words for which a configuration $(q_f, \mathbf{0})$ is reached. Moreover, a VASS is *deterministic* if for each input letter x and each state q, there is at most one x-labeled transition starting in q (and there are no ε-transitions). We can now phrase Theorem 4.4 as follows: Given a VASS coverability language K and a reachability language L of a deterministic VASS, it is EXPSPACE-complete to decide whether $K \subseteq L$. This is in contrast to inclusion problems where K is drawn from a subclass of the coverability languages: This quickly leads to Ackermann-completeness [6]. In fact, even if we replace Dyck_X in Theorem 4.4 with the set of prefixes of $\mathsf{Dyck}_{\{x\}}$, the problem becomes Ackermann-complete (see the full version of this work).

5 Checking Dyck Inclusion for Programs

We now describe our algorithm for checking inclusion in Dyck_X for programs. Our argument is similar to the case of dsVASS: we first construct three auxiliary programs $\mathscr{P}_\mathsf{o}$, $\mathscr{P}_\mathsf{d}$, and $\mathscr{P}_\mathsf{m}$, and then we use them to detect each type of violation in the original program. We construct the program $\mathscr{P}_\mathsf{o}$ for checking offset violation by projecting the Dyck letters to the one-dimensional Dyck alphabet $\{x, \bar{x}\}$. The programs $\mathscr{P}_\mathsf{d}$ and $\mathscr{P}_\mathsf{m}$ are constructed by first placing two markers like for VASS, and then projecting to $\{x, \bar{x}\}$.

As in the algorithm for VASS, we check whether $L(\mathscr{P}_\mathsf{o})$ has uniform offset 0, and whether $L(\mathscr{P}_\mathsf{d})$ and $L(\mathscr{P}_\mathsf{m})$ contain marked Dyck factors. For these checks, we convert the three programs into dsVASS $\mathcal{V}_\mathsf{o}$, $\mathcal{V}_\mathsf{d}$, and $\mathcal{V}_\mathsf{m}$, respectively, in such a way that violations are preserved. To be more precise, this conversion from programs to dsVASS will preserve the *downward closure* with respect to a specific order that we define below. The global downward closure procedure is obtained by composing a local downward closure procedure applied to each task.

On the task level, the order $\sqsubseteq$ is a combination of the subword order on the handler names in Γ and the syntactic order of Dyck_X over the event letters. The core technical result is a transformation from context-free grammars into dsNFA which preserve the downward closure with respect to $\sqsubseteq$.

One key aspect of our downward closure construction is an important condition on the pumps that appear in the context-free grammar.

▶ **Definition 5.1.** *A context-free grammar $\mathcal{G}$ is* tame-pumping *if for every pump $A \overset{*}{\Rightarrow} uAv$, we have $\mathsf{offset}(u) \geq 0$ and $\mathsf{offset}(v) = -\mathsf{offset}(u)$. A derivation $A \overset{*}{\Rightarrow} uAv$ is called an* increasing pump *if $\mathsf{offset}(u) > 0$, otherwise it is called a* zero pump*. An asynchronous program is* tame-pumping *if its grammar is tame-pumping.*

Note that while our definition of a tame-pumping grammar is syntactic, it actually only depends on the generated language, assuming every nonterminal occurs in a derivation: In that case, a grammar is tame-pumping if and only if (i) the set of offsets and (ii) the set of dips of words in its language are both finite.

The following lemma summarizes some properties of tame-pumping and why it is useful for our algorithm. The proof can be found in the full version.

▶ **Lemma 5.2.**

1. *We can check in* coNP *whether a given context-free grammar over $\{x, \bar{x}\}$ is tame-pumping. Furthermore, given a nonterminal A_0, we can check in* NP *whether A_0 has a zero pump (resp., increasing pump).*
2. *There exists a polynomial p such that, if $\mathcal{G}$ is tame-pumping, then for every nonterminal A of $\mathcal{G}$ and every $w \in L(\mathcal{G}, A)$ we have $\mathsf{dip}(w) \leq 2^{p(|\mathcal{G}|)}$.*
3. *If $\mathscr{P}$ does not have tame-pumping, then $L(\mathscr{P}) \not\subseteq \mathsf{Dyck}_X$.*

Thus, if $\mathscr{P}$ is not tame-pumping, the refinement checking algorithm rejects immediately. From now on, we assume that $\mathscr{P}$ is tame-pumping.

5.1 Combining the subword order and the syntactic order

Suppose Γ is an alphabet and let $\Theta = \Gamma \cup \{x, \bar{x}\}$. Define $\bar{a} = a$ for $a \in \Gamma$. By $\preccurlyeq$, we denote the *subword ordering* on Γ^*, i.e. $u \preccurlyeq v$ if and only if u can be obtained from v by deleting some letters. Formally there exist words $u_1, \ldots, u_n, v_0, \ldots, v_n \in \Gamma^*$ such that $u = u_1 \cdots u_n$ and $v = v_0 u_1 v_1 \cdots u_n v_n$. For $u, v \in \{x, \bar{x}\}^*$, we write $u \trianglelefteq v$ if $\mathsf{offset}(u) = \mathsf{offset}(v)$ and $\mathsf{dip}(u) \geq \mathsf{dip}(v)$. In fact, $\trianglelefteq$ is the *syntactic order* with respect to the Dyck language, i.e. if $u \trianglelefteq v$ and $rus \in \mathsf{Dyck}_x$ then $rvs \in \mathsf{Dyck}_x$ for all r, s. We define the ordering $\sqsubseteq'$ on Θ^* by $z_1 \sqsubseteq' z_2$ if and only if $\pi_{x,\bar{x}}(z_1) \trianglelefteq \pi_{x,\bar{x}}(z_2)$, and $\pi_\Gamma(z_1) \preccurlyeq \pi_\Gamma(z_2)$. For example, $a\bar{x}xc \sqsubseteq' xabc\bar{x}$ because ac is a subword of abc, and both $\bar{x}x$ and $x\bar{x}$ have offset 0, but $\bar{x}x$ has a larger dip.

Let $\#, \bar{\#}$ be two fresh letters, called *markers*. The set of *marked words* is defined as

$$\mathscr{M} = \Theta^*\{\varepsilon, \#\}\Theta^*\{\varepsilon, \bar{\#}\}\Theta^*.$$

A marked word should be viewed as an infix of a larger word $u\#v\bar{\#}w$. The set of *admissible* marked words, denoted by $\mathscr{A}$, consists of those words $z \in \mathscr{M}$ which are an infix of a word $u\#v\bar{\#}w$ where $v \in \mathsf{Dyck}_x$. For example, a marked word $u\#v$ is admissible if v is a prefix of a Dyck word.

On the set of admissible marked words, we define an ordering $\sqsubseteq$. To do so, we first define for each marked word $z \in \mathscr{M}$ two words $\mathsf{inside}(z)$ and $\mathsf{outside}(z)$ in Θ^* as follows: Let $u, v, w \in \Theta^*$ such that either $z = v$, $z = u\#v$, $z = v\bar{\#}w$, or $z = u\#v\bar{\#}w$. Then we define $\mathsf{inside}(z) = v$ and $\mathsf{outside}(z) = uw$ (here, $u = \varepsilon$ if it is not part of z, same for w). Given

two admissible marked words $z_1, z_2 \in \mathscr{A}$ we define $w \sqsubseteq w'$ if and only if z_1 and z_2 contain the same markers, and $\mathsf{inside}(z_1) \sqsubseteq' \mathsf{inside}(z_2)$, and $\mathsf{outside}(z_1) \sqsubseteq' \mathsf{outside}(z_2)$. For example, $a\bar{x}xc\#a \sqsubseteq xabc\bar{x}\#ab$ because $a\bar{x}xc \sqsubseteq' xabc\bar{x}$ and $a \sqsubseteq' ab$.

For a language $L \subseteq \mathscr{M}$ we denote by $L{\downarrow}$ the downward closure of L within $\mathscr{A}$ with respect to the ordering $\sqsubseteq$. Thus, we define:

$$L{\downarrow} = \{u \in \mathscr{A} \mid \exists v \in L \cap \mathscr{A} \colon u \sqsubseteq v\}.$$

▶ **Theorem 5.3.** *Given a tame-pumping* CFG $\mathcal{G}$*, we can compute in polynomial space a doubly succinct* NFA $\mathcal{A}$ *such that* $L(\mathcal{A}){\downarrow} = L(\mathcal{G}){\downarrow}$ *and* $|\mathcal{A}|$ *is polynomially bounded in* $|\mathcal{G}|$*.*

We explain how to prove Theorem 5.3 in Section 6. Let us make a few remarks. While downward closed sets with respect to the subword ordering are always regular, this does not hold for $\sqsubseteq$. Consider the language $L = (ax)^*$ where $a \in \Gamma$ is a handler name and $x \in X$ is an event letter. Then $L{\downarrow}$ consists of all words $w \in \{a, x, \bar{x}\}^*$ where $|w|_a \leq |w|_x - |w|_{\bar{x}}$, which is not a regular language. Furthermore, the automaton in Theorem 5.3 may indeed require double exponentially many states. For example, given a number n, consider the language $L = \{u\bar{x}^{2^n}\#x^{2^n}\bar{u} \mid u \in \{ax, bx\}^*\}$ where $\Gamma = \{a, b\}$ is the set of handler names and $X = \{x\}$. Here we define $\overline{a_1 a_2 \cdots a_n} = \bar{a}_n \cdots \bar{a}_2 \bar{a}_1$ for a word $a_1 \cdots a_n \in \{a, b, x\}^*$ where $\bar{a} = a$ and $\bar{b} = b$. This is generated by a tame-pumping context-free grammar of size linear in n. However, for any $\mathcal{A}$ with $L(\mathcal{A}){\downarrow} = L{\downarrow}$, projecting to just a and b yields the language $K = \{uu^{\mathsf{rev}} \mid u \in \{a, b\}^*,\ |u| \leq 2^n\}{\downarrow}$, for which an NFA requires at least 2^{2^n} states.

Finally, note that the restriction to admissible words is crucial: If we defined the ordering $\sqsubseteq$ on all words of $\mathscr{M}$, then for the tame-pumping language $L = \{x^n\#\bar{x}^n \mid n \in \mathbb{N}\}$, the downward closure would not be regular, because an NFA would be unable to preserve the unbounded offset at the separator $\#$. A key observation in this work is that in combination with tame pumping, admissibility guarantees that the offset at the borders $\#$ and $\bar{\#}$ is bounded (see Lemma 6.7), which enables a finite automaton to preserve it.

Given a tame-pumping asynchronous program $\mathscr{P}$, we can now compute a dsVASS $\mathcal{V}$ with the same downward closure: Its counters are the handler names $a \in \Gamma$ in $\mathscr{P}$. For each nonterminal A we apply Theorem 5.3 to $\mathcal{G}_A$, which is the grammar of $\mathscr{P}$ with start symbol A, and obtain a dsNFA $\mathcal{B}_A$. We replace each transition $q \xhookrightarrow{a,A} q'$ by the following gadget: First, it decrements the counter for the handler name a. Next, the gadget simulates the dsNFA $\mathcal{B}_A$ where handlers $b \in \Gamma$ are interpreted as counter increments. Finally, when reaching the final state of $\mathcal{B}_A$ we can non-deterministically switch to q'.

▶ **Corollary 5.4.** *Given an asynchronous program* $\mathscr{P}$ *with tame-pumping, we can compute in polynomial space a doubly succinct* VASS $\mathcal{V}$ *such that* $L(\mathscr{P}){\downarrow} = L(\mathcal{V}){\downarrow}$ *and* $|\mathcal{V}|$ *is polynomially bounded in* $|\mathscr{P}|$*.*

The details of the proof are given in the full version.

5.2 The algorithm

We are now ready to explain the whole algorithm. Given an asynchronous program $\mathscr{P} = (Q, X \cup \bar{X}, \Gamma, \mathcal{G}, \Delta, q_0, q_f, \gamma_0)$, we want to check if $L(\mathscr{P}) \subseteq \mathsf{Dyck}_X$. Recall that, wlog, we can assume all nonterminals are useful, meaning every nonterminal is involved in some accepting run. The algorithm is presented in Algorithm 1. As a first step, the algorithm verifies that $\mathscr{P}$ is tame-pumping using Lemma 5.2. Next we construct the following auxiliary asynchronous programs $\mathscr{P}_{\mathsf{o}}$, $\mathscr{P}_{\mathsf{d}}$, $\mathscr{P}_{\mathsf{m}}$, to detect offset, dip, and mismatch violations in $L(\mathscr{P})$.

Algorithm 1 Checking non-inclusion of $L(\mathscr{P})$ in the Dyck language Dyck_X in EXPSPACE.

```
Asynchronous program 𝒫 for a language L ⊆ (X ∪ X̄)*
if 𝒫 does not have tame-pumping (Lemma 5.2) then return L ⊈ Dyck_X;
Construct asynchronous programs 𝒫_o, 𝒫_d, 𝒫_m (Equation (1)).
Construct dsVASS 𝒱_o, 𝒱_d, 𝒱_m with L(𝒱_x)↓ = L(𝒫_x)↓ for x ∈ {o, d, m} (Corollary 5.4).
if 𝒱_o does not have uniform offset 0 (Corollary 4.2) then return L ⊈ Dyck_X;
if L(𝒱_d) or L(𝒱_m) contains a marked Dyck factor (Proposition 4.3) then return
  L ⊈ Dyck_X;
return L ⊆ Dyck_X
```

Let $\rho\colon (X \cup \bar{X})^* \to \{x, \bar{x}\}^*$ be the morphism which replaces all letters in X by unique letter x and all letters in $\bar{X}$ by unique letter $\bar{x}$. The programs $\mathscr{P}_\mathsf{o}$, $\mathscr{P}_\mathsf{d}$, $\mathscr{P}_\mathsf{m}$ recognize the following languages over the alphabet $\{x, \bar{x}, \#, \bar{\#}\}$:

$$\begin{aligned} L(\mathscr{P}_\mathsf{o}) &= \{\rho(w) \mid w \in L(\mathscr{P})\}, \\ L(\mathscr{P}_\mathsf{d}) &= \{\#\rho(v)\bar{\#}\rho(\bar{y}w) \mid v\bar{y}w \in L(\mathscr{P}) \text{ for some } v, w \in (X \cup \bar{X})^*,\ y \in X\}, \\ L(\mathscr{P}_\mathsf{m}) &= \{\rho(u)\#\rho(v)\bar{\#}\rho(w) \mid uyv\bar{z}w \in L(\mathscr{P}), \\ &\qquad\qquad \text{for some } u, v, w \in (X \cup \bar{X})^*,\ y \neq z \in X\}. \end{aligned} \tag{1}$$

In fact, if the original asynchronous program $\mathscr{P}$ is tame-pumping, we can ensure that $\mathscr{P}_\mathsf{o}$, $\mathscr{P}_\mathsf{d}$, $\mathscr{P}_\mathsf{m}$ are also tame-pumping (see the full version for details).

It remains to verify whether $L(\mathscr{P}_\mathsf{o})$ has uniform offset 0, and $L(\mathscr{P}_\mathsf{d})$ and $L(\mathscr{P}_\mathsf{m})$ do not contain marked Dyck factors. By Corollary 5.4 we can compute for each $\mathsf{x} \in \{\mathsf{o}, \mathsf{d}, \mathsf{m}\}$ a dsVASS $\mathcal{V}_\mathsf{x}$ with $L(\mathcal{V}_\mathsf{x})\downarrow = L(\mathscr{P}_\mathsf{x})\downarrow$. Since $\sqsubseteq$ preserves offsets we know that $L(\mathscr{P}_\mathsf{o})$ has uniform offset 0 if and only if $L(\mathcal{V}_\mathsf{o})$ has uniform offset 0, which can be decided in exponential space by Corollary 4.2. Finally, we check whether $L(\mathcal{V}_\mathsf{d})$ or $L(\mathcal{V}_\mathsf{m})$ contain a marked Dyck factor by Proposition 4.3. This is correct, because a language L contains a marked Dyck factor if and only if $L\downarrow$ contains a marked Dyck factor: On the one hand, the "only if" direction is clear because $L \subseteq L\downarrow$. On the other hand, if $u\#v\bar{\#}w \in L\downarrow$ is a marked Dyck word then there exists a word $u'\#v'\bar{\#}w' \in L$ with $v \trianglelefteq v'$, and therefore $v' \in \mathsf{Dyck}_x$.

6 Computing Downward Closures and the Proof of Theorem 5.3

It remains to show how the automaton $\mathcal{A}$ for the downward closure in Theorem 5.3 is constructed. As a warm-up, let us illustrate how to construct from a context-free grammar $\mathcal{G}$ an NFA $\mathcal{A}$ for the subword closure of $L(\mathcal{G})$, cf. [5]. Here, *subword closure* refers to the downward closure with respect to the subword ordering $\preccurlyeq$. Notice that this is a special case of Theorem 5.3, namely where $L(\mathcal{G}) \subseteq \Gamma^*$. The basic idea is that every derivation tree of $\mathcal{G}$ can be obtained by inserting pumps into a *skeleton* – a derivation tree without vertical repetitions of nonterminals. The skeleton can be guessed by an (exponentially large) automaton $\mathcal{A}$ and the effects of pumps are abstracted as follows: For each nonterminal A one can compute the subalphabets $\Gamma_{A,\mathsf{L}}, \Gamma_{A,\mathsf{R}} \subseteq \Gamma$ containing all letters occurring on the left side u and the right side v of a pump $A \overset{*}{\Rightarrow} uAv$. Instead of inserting pumps, the automaton for the subword closure inserts arbitrary words $u' \in \Gamma_{A,\mathsf{L}}^*$ and $v' \in \Gamma_{A,\mathsf{R}}^*$ on the left or right side of A, respectively. This is sufficient because for any word w, the subword closure of the language w^* contains exactly those words that consist only of letters present in w.

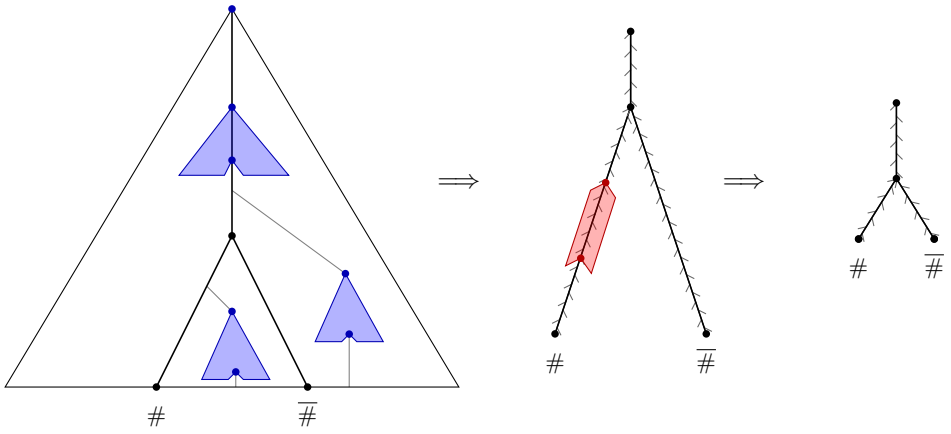


Figure 1 Abstracting undivided pumps (in blue) and divided pumps (in red).

The difficulty in proving Theorem 5.3 is to preserve, not only the subword closure, but also the downward closure with respect to the syntactic order $\trianglelefteq$ on the letters $\{x, \bar{x}\}$. To do so, we need to distinguish between two types of pumps. Consider the derivation tree for a marked word $z = u\#v\bar{\#}w$, depicted left in Figure 1. Observe that removing one of the three pumps in blue does not change the offset of $\mathsf{inside}(z) = v$ or $\mathsf{outside}(z) = uw$, because $\mathcal{G}$ is tame-pumping. Such pumps, which are completely contained in $\mathsf{inside}(z)$ or $\mathsf{outside}(z)$, will be called *undivided*. However, one needs to be more careful when removing *divided* pumps, e.g., the red pump in the second derivation tree of Figure 1. Removing the red pump decreases the offset of $\mathsf{outside}(z)$, while increasing the offset of $\mathsf{inside}(z)$ by the same amount.

We will proceed in two transformations, which preserve the downward closure w.r.t. $\sqsubseteq$. In the first transformation we obtain a grammar whose derivation trees do not contain any undivided pumps. In the second step we additionally eliminate divided pumps.

6.1 Abstracting undivided pumps

Recall that $\mathscr{M} = \Theta^*\{\#, \varepsilon\}\Theta^*\{\bar{\#}, \varepsilon\}\Theta^*$ where $\Theta = \Gamma \cup \{x, \bar{x}\}$. In the following we only consider *uniformly marked* grammars $\mathcal{G}$, that is, we assume $L(\mathcal{G})$ is contained in one of the subsets $\Theta^*\#\Theta^*\bar{\#}\Theta^*$, $\Theta^*\#\Theta^*$, $\Theta^*\bar{\#}\Theta^*$, or Θ^*. This is not a restriction since we can split the given grammar $\mathcal{G}$ into four individual grammars, covering the four types of marked words, and treat them separately. This allows us to partition the set of nonterminals N into $N_{\#\bar{\#}} \cup N_{\#} \cup N_{\bar{\#}} \cup N_0$ where $N_{\#\bar{\#}}$-nonterminals only produce marked words in $\Theta^*\#\Theta^*\bar{\#}\Theta^*$, $N_{\#}$-nonterminals only produce marked words in $\Theta^*\#\Theta^*$, etc. A pump $A \overset{*}{\Rightarrow} uAv$ is *undivided* if $A \in N_{\#\bar{\#}} \cup N_0$, and *divided* otherwise. Our first goal will be to eliminate undivided pumps. A derivation tree without undivided pumps may still contain exponentially large subtrees below N_0-nonterminals. Such subtrees will also be "flattened" in this step, see the first transformation step in Figure 1.

▶ **Definition 6.1.** *A context-free grammar $\mathcal{G} = (N, \Theta \cup \{\#, \bar{\#}\}, P, S)$ is* almost-pumpfree *iff*
(C1) *$\mathcal{G}$ does not have undivided pumps, and*
(C2) *for all productions $A \to \alpha$ with $A \in N_0$ either $\alpha = a \in \Theta$ or $\alpha = (\Gamma')^*$ for some $\Gamma' \subseteq \Gamma$.*

We will now explain how to turn any uniformly marked CFG into an almost-pumpfree one. The resulting (extended) grammar will be exponentially large but can be represented succinctly. Recall that a *succinct ECFG* (sECFG) is an extended context-free grammar

$\mathcal{G}$ whose nonterminals are polynomially long strings and whose productions are given by finite-state transducers. For example, one of the transducers accepts the finite relation of all triples (A, B, C) such that there exists a production $A \to BC$. Productions either adhere to Chomsky normal form or have the form $A \to B$. The latter enables us to simulate PSPACE-computations in the grammar without side effects, see Observation 6.5 below.

▶ **Proposition 6.2.** *Given a uniformly marked tame-pumping* CFG $\mathcal{G}$*, one can compute in polynomial space a tame-pumping almost-pumpfree* sECFG $\mathcal{G}'$ *such that* $L(\mathcal{G})\downarrow = L(\mathcal{G}')\downarrow$ *and* $|\mathcal{G}'|$ *is polynomially bounded in* $|\mathcal{G}|$*.*

To prove Proposition 6.2, we first need some auxiliary results, which are mainly concerned with computing the minimal dips and letter occurrences within undivided pumps of a grammar $\mathcal{G}$. Recall that for the subword closure we computed for each nonterminal A the subalphabets $\Gamma_{\mathsf{L},A}$ and $\Gamma_{\mathsf{R},A}$, and inserted arbitrary words over $\Gamma_{\mathsf{L},A}$ and $\Gamma_{\mathsf{R},A}$ left and right to the nonterminal A. For the refined order $\sqsubseteq$ we may only use a letter $a \in \Gamma$ after simulating the minimal dip which is required to produce the letter a.

For a word $w \in \Theta^*$ we define the set $\psi(w)$ of all pairs $(n, m) \in \mathbb{N}^2$ such that $n \geq \mathsf{dip}(w)$ and $m = n + \mathsf{offset}(w)$. In other words, $\psi(w)$ is the reachability relation induced by w, interpreted as counter instructions. Recall that *Presburger arithmetic* is the first-order theory of $(\mathbb{N}, +, <, 0, 1)$. As an auxiliary step, we will compute existential Presburger formulas capturing the relation $\psi(u) \times \psi(v)$ for all pumps $A \overset{*}{\Rightarrow} uAv$ of a nonterminal A.

In the following lemma, when we say that we can *compute a formula for a relation* $R \subseteq \mathbb{N}^k$ *in polynomial space*, we mean that there is non-deterministic polynomial-space algorithm, where each non-deterministic branch computes a polynomial-size formula for a relation R_i such that if $R_1, \ldots, R_n$ are the relations of all the branches, then $R = \bigcup_{i=1}^{n} R_i$. Here we tacitly use the fact that NPSPACE = PSPACE [24].

▶ **Lemma 6.3.** *Given an offset-uniform* CFG *with* $L(\mathcal{G}) \subseteq \Theta^*\$\Theta^*$*, where* $\$ \notin \Theta$*, we can compute in polynomial space an existential Presburger formula for the relation*

$$\bigcup_{u\$v \in L} \psi(u) \times \psi(v) \subseteq \mathbb{N}^4.$$

Proof sketch. The result of Lemma 6.3 was already proved in [1, Proposition 3.8], under the additional assumption that the given context-free grammar $\mathcal{G}$ for L is *annotated* (they even show that in this case the formula can be computed in NP). We call $\mathcal{G}$ annotated if for every nonterminal A the minimal dip that can be achieved by a word in $L(\mathcal{G}, A)$ is given as an input, denoted by $\mathsf{mindip}(A)$. Hence, it remains to show how to compute the annotation of an offset-uniform grammar in PSPACE, which is possible using a simple saturation algorithm. For each nonterminal A, the algorithm stores a number $D(A)$ satisfying $D(A) \geq \mathsf{mindip}(A)$. Initially, $D(A)$ is set to an upper bound for $\mathsf{mindip}(A)$, which by Lemma 5.2 (2) can be chosen to be exponentially large in $|\mathcal{G}|$. In each round the function D is updated as follows: For each production $A \to BC$ we set $D(A)$ to the minimum of $D(A)$ and $\max\{D(B), D(C) - \mathsf{offset}(B)\}$, where $\mathsf{offset}(B)$ is the unique offset of $L(\mathcal{G}, B)$. Clearly, the algorithm can be implemented in polynomial space since the numbers are bounded exponentially. Termination of the algorithm is guaranteed since the numbers $D(A)$ are non-increasing. ◀

With Lemma 6.3 in hand, we can now prove the following lemma, which allows us to check whether pumps with certain letter occurrences exist for certain minimal dips.

▶ **Lemma 6.4.** *Given a tame-pumping* CFG $\mathcal{G}$ *such that* $L(\mathcal{G}) \subseteq \mathscr{M}$*, a nonterminal* A *in* $\mathcal{G}$*, a letter* $a \in \Gamma$ *and two numbers* $d_{\mathsf{L}}, d_{\mathsf{R}} \in \mathbb{N}$*, we can decide in* PSPACE *if there exists a derivation* $A \overset{*}{\Rightarrow} uAv$ *such that* u *contains the letter* a *(or symmetrically, whether* v *contains the letter* a*),* $\mathsf{dip}(u) \leq d_{\mathsf{L}}$*, and* $\mathsf{dip}(v) \leq d_{\mathsf{R}}$*. Furthermore, we can also decide in* PSPACE *whether a derivation with the above properties exists that also satisfies* $\mathsf{offset}(u) > 0$*.*

Proof sketch. We first construct the CFG $\mathcal{G}_A$ for the language of pumps of the nonterminal A, meaning for $L(\mathcal{G}_A) = \{u\$v \mid A \overset{*}{\Rightarrow}_{\mathcal{G}} uAv\}$. Then we intersect with the regular language $\Theta^* a \Theta^* \$ \Theta^*$, and apply Lemma 6.3 to the resulting grammar. This is possible, because tame-pumping implies that the grammar for the pumps has a uniform offset of zero. We can modify the resulting Presburger formula from Lemma 6.3 to check for the required dips, and modify it further to check for the positive offset for u. Finally, we use the fact that testing satisfiability of an existential Presburger formula is in NP [3]. ◀

Now we are almost ready to prove Proposition 6.2. The last thing we need is for an sECFG to perform PSPACE-computations on paths in its derivation trees:

▶ **Observation 6.5.** *An* sECFG *can simulate* PSPACE*-computations on exponentially long paths in its derivation trees. This is because the nonterminals are polynomially long strings and can therefore act as polynomial space Turing tape configurations. Moreover, the transducers of the* sECFG *can easily be constructed to enforce the step-relation of a Turing machine. If we apply this enforcement to productions of the form* $A \to B$*, then the path that simulates the* PSPACE*-computation will not even have any additional side paths until after the computation is complete. Thus, only the result of the computation will affect the derived word.*

Since grammars and transducers are non-deterministic (and NPSPACE = PSPACE*), we can even implement non-determinism and guessing within such computations.*

We are ready to present a proof sketch of Proposition 6.2. The main idea is that $\mathcal{G}'$ simulates derivation trees of $\mathcal{G}$ by keeping track of at most polynomially many nodes, and abstracting away pumps via the previous auxiliary results.

If a nonterminal A of $\mathcal{G}$ does not belong to N_0 (i.e., it produces a marker), then $\mathcal{G}'$ guesses a production $A \to BC$ to apply. If A furthermore belongs to $N_{\#\bar{\#}}$, then $\mathcal{G}'$ also guesses a pump to apply in the form of a 4-tuple consisting of two dip values $d_{\mathsf{L}}, d_{\mathsf{R}} \in \mathbb{N}$ and two alphabets $\Gamma_{\mathsf{L}}, \Gamma_{\mathsf{R}} \subseteq \Gamma$. Guessing and storing the dip values is possible in PSPACE, since they are exponentially bounded by Lemma 5.2 (2). For each $a \in \Gamma_{\mathsf{L}}$, Lemma 6.4 is used on input $A, a, d_{\mathsf{L}}, d_{\mathsf{R}}$ to check in PSPACE whether a matching pump exists. A symmetric version of Lemma 6.4 is also used for each $a \in \Gamma_{\mathsf{R}}$. Then, if all checks succeed, $\mathcal{G}'$ simulates the pump as $A \to \bar{x}^{d_{\mathsf{L}}} x^{d_{\mathsf{L}}} \Gamma_{\mathsf{L}}^* BC \bar{x}^{d_{\mathsf{R}}} x^{d_{\mathsf{R}}} \Gamma_{\mathsf{R}}^*$. This simulation clearly preserves minimal dips and handler names, whereas by tame-pumping the combined offset of a pump is zero anyway, and therefore need not be computed.

If a nonterminal A belongs to N_0, then $\mathcal{G}'$ abstracts away its entire subtree. To this end it generates a pumpfree subtree on-the-fly using depth-first search, which is possible in PSPACE since without pumps the tree has polynomial height. During this process pumps are simulated using the same strategy as before.

We also need to ensure that nonterminals of $\mathcal{G}'$ in N_0 only have productions that allow for a single leaf node below them. To this end $\mathcal{G}'$ only ever derives letters and alphabets Γ'^* one at a time. Consider the up to two *main paths* in a derivation tree of $\mathcal{G}'$, by which we mean the paths leading from the root to a marker. Whenever $\mathcal{G}'$ simulates a pump as $A \to u'Av'$ in the above process, it extends the main path by $|uv|$ and in each step only derives a single nonterminal from N_0 to the left or right. When $\mathcal{G}'$ abstracts an entire subtree of a nonterminal in N_0, then this subtree is also produced to the left or right of the main path, without leaving said path.

Additionally, whenever $\mathcal{G}'$ simulates a pump of some A, then $\mathcal{G}'$ assumes that this pump is the combination of all pumps that occur in the original derivation tree for that instance of A. Thus, below such a pump, it remembers in polynomial space, that A is not allowed to occur anymore. Finally, whenever $\mathcal{G}'$ checks by Lemma 6.4 that a pump exists with $\mathsf{offset}(u) > 0$, then this is a so-called increasing pump, and it can be repeated to achieve an infix with arbitrary high offset. Thus, dip values below this pump cannot make up for this offset and therefore will no longer be simulated.

6.2 Abstracting divided pumps

We have now removed all the undivided pumps and are left with derivation trees as in the middle picture of Figure 1. In this subsection, we will show the following:

▶ **Lemma 6.6.** *Given a tame-pumping almost-pumpfree* sECFG $\mathcal{G}$ *with* $L(\mathcal{G}) \subseteq \mathscr{M}$*, one can construct in polynomial space a* dsNFA $\mathcal{B}$ *such that* $L(\mathcal{B})\downarrow = L(\mathcal{G})\downarrow$ *and* $|\mathcal{B}|$ *is polynomially bounded in* $|\mathcal{G}|$*.*

We give a proof sketch here, the details can be found in the full version of the paper. Our starting point in the proof of Lemma 6.6 is the following key observation: The offsets which occur during the production of any *admissible* marked word w which contains exactly one marker are bounded. This allows us to keep track of the offset precisely, which is necessary for us to solve the marked Dyck factor (MDF) problem.

For a node t in a derivation tree T, let $w(t)$ denote the word derived by the subtree rooted at t and let $u(t) = \mathsf{inside}(w(t))$, $v(t) = \mathsf{outside}(w(t))$.

▶ **Lemma 6.7.** *There exists a polynomial p such that for any uniformly marked, tame-pumping, almost-pumpfree* sECFG $\mathcal{G}$ *the following holds. Let T be a derivation tree of $\mathcal{G}$ which produces an admissible marked word containing $\#$ or $\bar{\#}$, but not both. Then we have* $|\mathsf{offset}(u(t))|, |\mathsf{offset}(v(t))| \leq 2^{p(|\mathcal{G}|)}$*.*

Proof. We consider the case when the word derived is of the form $u\#v$, the case for $v\bar{\#}w$ being symmetric. Our derivation tree T has a skeleton T' into which pumps are inserted to form T. This means $u\#v = u'_k\hat{u}_k \cdots u'_1\hat{u}_1u'_0\#v'_0\hat{v}_1v'_1 \cdots \hat{v}_kv'_k$, where $u'_k \cdots u'_0\#v'_0 \cdots v'_k$ is the word generated by T' and each pair $(\hat{u}_i, \hat{v}_i)$ is derived using a pump. Then we have

$$\mathsf{offset}(u) = \overbrace{\mathsf{offset}(u'_k \cdots u'_0)}^{=:U_0} + \overbrace{\sum_{i=1}^{k} \mathsf{offset}(\hat{u}_i)}^{=:U_1},$$

$$\mathsf{offset}(v) = \underbrace{\mathsf{offset}(v'_0 \cdots v'_k)}_{=:V_0} + \underbrace{\sum_{i=1}^{k} \mathsf{offset}(\hat{v}_i)}_{=:V_1}.$$

We claim that each of the numbers $|U_0|, |U_1|, |V_0|, |V_1|$ is bounded by $n(\mathcal{G})$, the number of nonterminals of $\mathcal{G}$. This clearly implies the lemma: Since $\mathcal{G}$ is a succinct grammar, it has at most exponentially many nonterminals in the size of its description. We begin with U_0, V_0. The tree T' contains each nonterminal of $\mathcal{G}$ at most once, and by property (C2) in Definition 6.1, we know that the subtree under each nonterminal in T' not containing $\#$ has offset -1, 0, or 1. Thus, $|U_0|, |V_0| \leq n(\mathcal{G})$. The bound on $|U_1|, |V_1|$ is due to admissibility of $u\#v$: It yields $V_0 + V_1 = \mathsf{offset}(v) \geq 0$ and thus $V_1 \geq -V_0$. Moreover, by tame-pumping, we know that $\mathsf{offset}(\hat{v}_i) \leq 0$ for each $i \in [1, k]$, and thus $V_1 \leq 0$. Together, we obtain $V_1 \in [-V_0, 0]$. Finally, tame-pumping also implies $\mathsf{offset}(\hat{u}_i) = -\mathsf{offset}(\hat{v}_i)$ for each $i \in [1, k]$ and hence $U_1 = -V_1$. ◀

▶ Remark 6.8. Note that the bound only holds under the condition of admissibility. An easy counterexample is the tame-pumping language $L = \{x^n \# \bar{x}^n \mid n \in \mathbb{N}\}$.

The dsNFA $\mathcal{B}$ of Lemma 6.6 can now be constructed in three steps as follows:

Step I: Tracking counter effects. We first observe that since $\mathcal{G}$ is almost-pumpfree, its pumps $A \overset{*}{\Rightarrow} uAv$ can be simulated by a transducer that traverses the derivation tree bottom-up. Thus, we can construct a *singly* succinct finite-state transducer $\mathcal{T}_A$ with size polynomial in $|\mathcal{G}|$ that captures all pumps $A \overset{*}{\Rightarrow} uAv$. To be precise, $\mathcal{T}_A$ accepts exactly those pairs (u, v) for which $A \overset{*}{\Rightarrow} u^{\mathsf{rev}}Av$. The transducer $\mathcal{T}_A$ has one state for each nonterminal of $\mathcal{G}$.

Since $\mathcal{B}$ will need to preserve offset and dip, we need to expand $\mathcal{T}_A$ to track them as well. Here, it is crucial that we only need to do this for $A \in N_{\#} \cup N_{\bar{\#}}$ and pumps $A \overset{*}{\Rightarrow} uAv$ that are used to derive an admissible word. According to Lemma 6.7 tells us that in such a pump, the absolute values of offsets and dips of u and v are bounded by $2^{q(|\mathcal{G}|)}$ for some polynomial q. Thus, we can modify $\mathcal{T}_A$ so as to track the dip and offset of the two words it reads. Therefore, for each $A \in N_{\#} \cup N_{\bar{\#}}$ and each quadruple $\mathbf{x} = (d_{\mathsf{L}}, \delta_{\mathsf{L}}, d_{\mathsf{R}}, \delta_{\mathsf{R}})$ of numbers with absolute value at most $2^{q(\mathcal{G})}$, we can construct in PSPACE a transducer $\mathcal{T}_{A,\mathbf{x}}$ with

$$(u, v) \text{ is accepted by } \mathcal{T}_{A,\mathbf{x}} \quad \text{iff} \quad A \overset{*}{\Rightarrow} u^{\mathsf{rev}}Av \text{ and } e(u^{\mathsf{rev}}) = (d_{\mathsf{L}}, \delta_{\mathsf{L}}), \text{ and } e(v) = (d_{\mathsf{R}}, \delta_{\mathsf{R}}).$$

Moreover, $\mathcal{T}_{A,\mathbf{x}}$ is singly succinct, polynomial-size, and can be computed in PSPACE. Observe that by Lemma 6.7, if a pump $A \overset{*}{\Rightarrow} uAv$ is used in a derivation of an admissible word, then for some quadruple $\mathbf{x}$, the pair (u^{rev}, v) is accepted by $\mathcal{T}_{A,\mathbf{x}}$.

Step II: Skeleton runs. The automaton $\mathcal{B}$ has to read words from left to right, rather than two factors in parallel as $\mathcal{T}_A$ and $\mathcal{T}_{A,\mathbf{x}}$ do. To this end, it will guess a run of $\mathcal{T}_{A,\mathbf{x}}$ without state repetitions; such a run is called a *skeleton run*. For a fixed skeleton run ρ, the set of words read in each component of $\mathcal{T}_{A,\mathbf{x}}$ is of the shape $\Gamma_0^*\{a_1, \varepsilon\}\Gamma_1^* \cdots \{a_k, \varepsilon\}\Gamma_k^*$, where each a_i is read in a single step of ρ and Γ_i is the set of letters from Γ seen in cycles in a state visited in ρ. Sets of this shape are called *ideals* [12]. The ideal for the left (right) component is called the *left* (*right*) *ideal* of the skeleton run. Note that since $\mathcal{T}_{A,\mathbf{x}}$ has exponentially many states, the skeleton run is at most exponentially long.

Step III: Putting it together. The dsNFA $\mathcal{B}$ guesses and verifies an exponential size skeleton T of the sECFG $\mathcal{G}$. Moreover, for each node t that is above $\#$ or $\bar{\#}$—but not both—it guesses a quadruple $\mathbf{x} = (d_{\mathsf{L}}, \delta_{\mathsf{L}}, d_{\mathsf{R}}, \delta_{\mathsf{R}})$ with $d_{\mathsf{L}}, d_{\mathsf{R}} \in [0, 2^{q(|\mathcal{G}|)}]$, $\delta_{\mathsf{L}}, \delta_{\mathsf{R}} \in [-2^{q(\mathcal{G})}, 2^{q(\mathcal{G})}]$ and a skeleton run ρ_t of the transducer $\mathcal{T}_{A,\mathbf{x}}$, where A is t's label. The automaton $\mathcal{B}$ then traverses the skeleton T in-order; i.e. node, left subtree, right subtree, node; meaning each inner node is visited exactly twice. Whenever $\mathcal{B}$ visits a node t as above, it produces an arbitrary word from an ideal of ρ_t: For the first (resp. second) visit of t, it uses the left (resp. right) ideal of ρ_t. Moreover, in addition to the word from the left ideal, $\mathcal{B}$ outputs a string $w \in \{x, \bar{x}\}^*$ with $e(w) = (d_{\mathsf{L}}, \delta_{\mathsf{L}})$, where $\mathbf{x} = (d_{\mathsf{L}}, \delta_{\mathsf{L}}, d_{\mathsf{R}}, \delta_{\mathsf{R}})$ is the quadruple guessed for t (and similarly for the right ideal). This way, it preserves offset and dip at the separators $\#$ and $\bar{\#}$.

Since the skeleton T has exponentially many nodes (in $|\mathcal{G}|$) and each skeleton run ρ_t requires exponentially many bits, the total number of bits that $\mathcal{B}$ has to keep in memory is also bounded by an exponential in $|\mathcal{G}|$.

References

1. Pascal Baumann, Moses Ganardi, Rupak Majumdar, Ramanathan S. Thinniyam, and Georg Zetzsche. Context-bounded verification of context-free specifications. *Proc. ACM Program. Lang.*, 7(POPL):2141–2170, 2023. doi:10.1145/3571266.
2. Pascal Baumann, Rupak Majumdar, Ramanathan S. Thinniyam, and Georg Zetzsche. Context-bounded verification of thread pools. *Proc. ACM Program. Lang.*, 6(POPL):1–28, 2022. doi:10.1145/3498678.
3. I. Borosh and L. B. Treybig. Bounds on positive integral solutions of linear diophantine equations. *Proceedings of the American Mathematical Society*, 55(2):299–304, 1976.
4. Laura Bozzelli and Pierre Ganty. Complexity analysis of the backward coverability algorithm for VASS. In Giorgio Delzanno and Igor Potapov, editors, *Reachability Problems – 5th International Workshop, RP 2011, Genoa, Italy, September 28-30, 2011. Proceedings*, volume 6945 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2011. doi:10.1007/978-3-642-24288-5_10.
5. Bruno Courcelle. On constructing obstruction sets of words. *Bulletin of the EATCS*, 44:178–186, 1991.
6. Wojciech Czerwinski and Piotr Hofman. Language inclusion for boundedly-ambiguous vector addition systems is decidable. In Bartek Klin, Slawomir Lasota, and Anca Muscholl, editors, *33rd International Conference on Concurrency Theory, CONCUR 2022, September 12-16, 2022, Warsaw, Poland*, volume 243 of *LIPIcs*, pages 16:1–16:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CONCUR.2022.16.
7. Wojciech Czerwiński and Łukasz Orlikowski. Reachability in vector addition systems is Ackermann-complete. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1229–1240. IEEE, 2021. doi:10.1109/FOCS52979.2021.00120.
8. Ankush Desai, Pranav Garg, and P. Madhusudan. Natural proofs for asynchronous programs using almost-synchronous reductions. In Andrew P. Black and Todd D. Millstein, editors, *Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications, OOPSLA 2014, part of SPLASH 2014, Portland, OR, USA, October 20-24, 2014*, pages 709–725. ACM, 2014. doi:10.1145/2660193.2660211.
9. Ankush Desai and Shaz Qadeer. P: modular and safe asynchronous programming. In Shuvendu K. Lahiri and Giles Reger, editors, *Runtime Verification – 17th International Conference, RV 2017, Seattle, WA, USA, September 13-16, 2017, Proceedings*, volume 10548 of *Lecture Notes in Computer Science*, pages 3–7. Springer, 2017. doi:10.1007/978-3-319-67531-2_1.
10. Pierre Ganty and Rupak Majumdar. Algorithmic verification of asynchronous programs. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 34(1):6, 2012. doi:10.1145/2160910.2160915.
11. Ivan Gavran, Filip Niksic, Aditya Kanade, Rupak Majumdar, and Viktor Vafeiadis. Rely/guarantee reasoning for asynchronous programs. In Luca Aceto and David de Frutos-Escrig, editors, *26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1.4, 2015*, volume 42 of *LIPIcs*, pages 483–496. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.CONCUR.2015.483.
12. Jean Goubault-Larrecq, Simon Halfon, P. Karandikar, K. Narayan Kumar, and Philippe Schnoebelen. The ideal approach to computing closed subsets in well-quasi-orderings. In Peter M. Schuster, Monika Seisenberger, and Andreas Weiermann, editors, *Well-Quasi Orders in Computation, Logic, Language and Reasoning*, volume 53 of *Trends In Logic*, pages 55–105. Springer, 2020. doi:10.1007/978-3-030-30229-0_3.
13. Ranjit Jhala and Rupak Majumdar. Interprocedural analysis of asynchronous programs. In *POPL '07: Proc. 34th ACM SIGACT-SIGPLAN Symp. on Principles of Programming Languages*, pages 339–350. ACM Press, 2007.
14. Dexter Kozen. *Automata and computability*. Undergraduate texts in computer science. Springer, 1997.

15 Bernhard Kragl, Constantin Enea, Thomas A. Henzinger, Suha Orhun Mutluergil, and Shaz Qadeer. Inductive sequentialization of asynchronous programs. In Alastair F. Donaldson and Emina Torlak, editors, *Proceedings of the 41st ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2020, London, UK, June 15-20, 2020*, pages 227–242. ACM, 2020. doi:10.1145/3385412.3385980.

16 Bernhard Kragl, Shaz Qadeer, and Thomas A. Henzinger. Synchronizing the asynchronous. In Sven Schewe and Lijun Zhang, editors, *29th International Conference on Concurrency Theory, CONCUR 2018, September 4-7, 2018, Beijing, China*, volume 118 of *LIPIcs*, pages 21:1–21:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CONCUR.2018.21.

17 Jérôme Leroux. The Reachability Problem for Petri Nets is Not Primitive Recursive. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1241–1252, February 2022. doi:10.1109/FOCS52979.2021.00121.

18 Richard Lipton. The reachability problem is exponential-space hard. *Yale University, Department of Computer Science, Report*, 62, 1976.

19 Raphaela Löbel. *Linear Tree Transducers: From Equivalence to Balancedness*. PhD thesis, Technical University of Munich, Germany, 2020. URL: https://nbn-resolving.org/urn:nbn:de:bvb:91-diss-20201127-1552125-1-5.

20 Rupak Majumdar, Ramanathan S. Thinniyam, and Georg Zetzsche. General decidability results for asynchronous shared-memory programs: Higher-order and beyond. In Jan Friso Groote and Kim Guldstrand Larsen, editors, *Tools and Algorithms for the Construction and Analysis of Systems – 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 – April 1, 2021, Proceedings, Part I*, volume 12651 of *Lecture Notes in Computer Science*, pages 449–467. Springer, 2021. doi:10.1007/978-3-030-72016-2_24.

21 Shaz Qadeer and Dinghao Wu. KISS: keep it simple and sequential. In William W. Pugh and Craig Chambers, editors, *Proceedings of the ACM SIGPLAN 2004 Conference on Programming Language Design and Implementation 2004, Washington, DC, USA, June 9-11, 2004*, pages 14–24. ACM, 2004. doi:10.1145/996841.996845.

22 Charles Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6(2):223–231, 1978.

23 Robert W Ritchie and Frederick N Springsteel. Language recognition by marking automata. *Information and Control*, 20(4):313–330, 1972.

24 Walter J Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of computer and system sciences*, 4(2):177–192, 1970.

25 Koushik Sen and Mahesh Viswanathan. Model checking multithreaded programs with asynchronous atomic methods. In *CAV '06: Proc. 18th Int. Conf. on Computer Aided Verification*, volume 4144 of *LNCS*, pages 300–314. Springer, 2006.

26 Akihiko Tozawa and Yasuhiko Minamide. Complexity results on balanced context-free languages. In Helmut Seidl, editor, *Foundations of Software Science and Computational Structures, 10th International Conference, FOSSACS 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007, Braga, Portugal, March 24-April 1, 2007, Proceedings*, volume 4423 of *Lecture Notes in Computer Science*, pages 346–360. Springer, 2007. doi:10.1007/978-3-540-71389-0_25.

On the Limits of Decision: the Adjacent Fragment of First-Order Logic

Bartosz Bednarczyk
Computational Logic Group, Technische Universität Dresden, Germany
Institute of Computer Science, University of Wrocław, Poland

Daumantas Kojelis
Department of Computer Science, University of Manchester, UK

Ian Pratt-Hartmann
Department of Computer Science, University of Manchester, UK
Institute of Computer Science, University of Opole, Poland

Abstract

We define the *adjacent fragment* $\mathcal{AF}$ of first-order logic, obtained by restricting the sequences of variables occurring as arguments in atomic formulas. The adjacent fragment generalizes (after a routine renaming) two-variable logic as well as the fluted fragment. We show that the adjacent fragment has the finite model property, and that its satisfiability problem is no harder than for the fluted fragment (and hence is TOWER-complete). We further show that any relaxation of the adjacency condition on the allowed order of variables in argument sequences yields a logic whose satisfiability and finite satisfiability problems are undecidable. Finally, we study the effect of the adjacency requirement on the well-known guarded fragment ($\mathcal{GF}$) of first-order logic. We show that the satisfiability problem for the guarded adjacent fragment ($\mathcal{GA}$) remains 2EXPTIME-hard, thus strengthening the known lower bound for $\mathcal{GF}$.

2012 ACM Subject Classification Theory of computation → Finite Model Theory

Keywords and phrases decidability, satisfiability, variable-ordered logics, complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.111

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `https://arxiv.org/abs/2305.03133` [5]

Funding *Bartosz Bednarczyk*: supported by the ERC Consolidator Grant No. 771779 (DeciGUT).
Ian Pratt-Hartmann: supported by the NCN grant 2018/31/B/ST6/03662.

Acknowledgements B. Bednarczyk thanks R. Jaakkola for many inspiring discussions.

1 Introduction

The quest to find fragments of first-order logic for which satisfiability is algorithmically decidable has been a central undertaking of mathematical logic since the appearance of Hilbert and Ackermann's *Grundzüge der theoretischen Logik* [15, 16] almost a century ago. The great majority of such fragments so far discovered, however, belong to just three families: (i) quantifier prefix fragments [8], where we are restricted to prenex formulas with a specified quantifier sequence; (ii) two-variable logics [13], where the only logical variables occurring as arguments of predicates are x_1 and x_2; and (iii) guarded logics [1], where quantifiers are relativized by atomic formulas featuring all the free variables in their scope.

There is, however, a fourth family of decidable logics, originating in the work of W.V.O. Quine [33], and based on restricting the allowed sequences of variables occurring as arguments in atomic formulas. This family of logics, which includes the *fluted fragment*,

the *ordered fragment* and the *forward fragment*, has languished in relative obscurity. In this paper, we investigate the potential for obtaining decidable fragments in this way, identifying a new fragment, which we call the *adjacent fragment*. This fragment not only includes the fluted, ordered and forward fragments, but also subsumes, in a sense we make precise, the two-variable fragment. We show that the satisfiability problem for the adjacent fragment is decidable, and determine bounds on its complexity.

To explain how restrictions on argument orderings work, we consider presentations of first-order logic without equality over purely relational signatures, employing individual variables from the alphabet $\{x_1, x_2, x_3, \ldots\}$. Any atomic formula in this logic has the form $p(\bar{x})$, where p is a predicate of arity $m \geq 0$ and $\bar{x}$ is a word over the alphabet of variables of length m. Call a first-order formula φ *index-normal* if, for any quantified sub-formula $Qx_k\,\psi$ of φ, ψ is a Boolean combination of formulas that are either atomic with free variables among $x_1, \ldots, x_k$, or have as their major connective a quantifier binding x_{k+1}. By re-indexing variables, any first-order formula can easily be written as a logically equivalent index-normal formula. In the *fluted fragment*, denoted $\mathcal{FL}$, as defined by W. Purdy [32], we confine attention to index-normal formulas, but additionally insist that any atom occurring in a context in which x_k is quantified have the form $p(x_{k-m+1} \cdots x_k)$, i.e. $p(\bar{x})$ with $\bar{x}$ a *suffix* of $x_1 \cdots x_k$. In the *ordered fragment*, due to A. Herzig [14], by contrast, we insist that $\bar{x}$ be a *prefix* of $x_1 \cdots x_k$. In the *forward fragment* [2], we insist only that $\bar{x}$ be an *infix* of $x_1 \cdots x_k$.

All these logics have the finite model property, and hence are decidable for satisfiability. Denoting by $\mathcal{FL}^k$ the sub-fragment of $\mathcal{FL}$ involving at most k variables (free or bound), the satisfiability problem for $\mathcal{FL}^k$ is known to be in $(k{-}2)$-NExpTime for all $k \geq 3$, and $\lfloor k/2 \rfloor$-NExpTime-hard for all $k \geq 2$ [30]. Thus, satisfiability for the whole fluted fragment is Tower-complete, in the system of trans-elementary complexity classes due to [35]. By contrast, the satisfiability problem for the ordered fragment is known to be PSpace-complete [14, 18]. On the other hand, the apparent liberalization afforded by the forward fragment yields no difference in expressive power [4], and moreover there is a polynomial time satisfiability-preserving reduction of the forward fragment to the fluted fragment [2]. The term "fluted" originates with Quine [34], and presumably invites us to imagine the atoms in formulas aligned in such a way that the variables form columns. Note that none of these fragments can state that a relation is reflexive or symmetric (see [4] for a discussion of their expressivity).

Say that a word $\bar{x}$ over the alphabet $\{x_1, \ldots, x_k\}$ $(k \geq 0)$ is *adjacent* if the indices of neighbouring letters differ by at most 1. For example, $x_3x_2x_1x_2x_2x_2x_3x_4x_3$ is adjacent, but $x_1x_3x_2$ is not. The *adjacent fragment*, denoted $\mathcal{AF}$, is analogous to the fluted, ordered and forward fragments, but we allow any atom $p(\bar{x})$ to occur in a context where x_k is available for quantification as long as $\bar{x}$ is an adjacent word over $\{x_1, \ldots, x_k\}$. (A formal definition is given in Sec. 2.) As a simple example, the formula

$$\forall x_1 \forall x_2 \forall x_3 \exists x_4 \forall x_5 \; \big(p(x_1x_2x_3x_2x_3x_4x_5) \rightarrow p(x_1x_2x_3x_4x_3x_4x_5)\big) \tag{1}$$

is a validity of $\mathcal{AF}$, as can be seen by assigning x_4 the same value as x_2. Evidently, $\mathcal{AF}$ includes the fluted, ordered and forward fragments; the inclusion is strict, since the formulas $\forall x_1\, r(x_1x_1)$ and $\forall x_1x_2(r(x_1x_2) \rightarrow r(x_2x_1))$, stating that r is reflexive and symmetric, respectively, are in $\mathcal{AF}$. As every word over $\{x_1, x_2\}$ is adjacent, we may transform any formula of the two-variable fragment without equality, FO^2, in polynomial time, to a logically equivalent formula of $\mathcal{AF}$. The converse is true over signatures with predicates of arity at most two. Since the system of basic multimodal propositional logic is, under the standard translation to first-order logic, included within FO^2, this logic is similarly subsumed by $\mathcal{AF}$, as indeed is its notational variant, the description logic $\mathcal{ALC}$ (see, e.g. [17]).

We show that the satisfiability problem for the restriction of the adjacent fragment to formulas involving at most k variables (free or bound) is in $(k-2)$-NExpTime for all $k \geq 3$ – and hence no more difficult than the k-variable fluted fragment, which it properly contains. The critical step in our analysis is a lemma on the combinatorics of strings (Theorem 3.1), which may be of independent interest. We also consider minimal relaxations of adjacency involving the fragment with just three variables, and show that, in all cases of interest, the satisfiability and finite satisfiability problems for the resulting logics are undecidable. Thus, adjacency is as far as we can go in seeking decidable fragments based on straightforward argument ordering restrictions of the type envisaged by Quine.

The adjacent fragment is incomparable in expressive power to the guarded fragment. Moreover, the satisfiability problem for the union of $\mathcal{GF}$ and $\mathcal{AF}$ is undecidable, as one can use adjacent formulas to introduce any k-ary universal relations, which makes $\mathcal{GF}$ as expressive as first-order logic. Therefore, we study the effect of the adjacency restriction on $\mathcal{GF}$. We investigate the complexity of satisfiability for the resulting logic, $\mathcal{GA}$, showing that the problem is 2ExpTime-complete, thus sharpening the existing 2ExpTime-hardness proof for $\mathcal{GF}$ [11].

2 Preliminaries

Let m and k be non-negative integers. For any integers i and j, we write $[i, j]$ to denote the set of integers h such that $i \leq h \leq j$. A function $f\colon [1, m] \to [1, k]$ is *adjacent* if $|f(i+1)-f(i)| \leq 1$ for all i $(1 \leq i < m)$. We write $\mathbf{A}_k^m$ to denote the set of adjacent functions $f\colon [1, m] \to [1, k]$. Since $[1, 0] = \emptyset$, we have $\mathbf{A}_k^0 = \{\emptyset\}$, and $\mathbf{A}_0^m = \emptyset$ if $m > 0$. Let A be a non-empty set. A word $\bar{a}$ over the alphabet A is simply a tuple of elements from A; we alternate freely in the sequel between these two ways of speaking as the context requires. Accordingly, A^k denotes the set of words over A having length exactly k, and A^* is the set of all finite words over A. If $\bar{a} \in A^*$, denote the length of $\bar{a}$ by $|\bar{a}|$, and the reversal of $\bar{a}$ by $\bar{a}^{-1}$. Any function $f\colon [1, m] \to [1, k]$ (adjacent or not) induces a natural map from A^k to A^m defined by $\bar{a}^f = a_{f(1)} \cdots a_{f(m)}$, where $\bar{a} = a_1 \cdots a_k$. If $f \in \mathbf{A}_k^m$ (i.e. if f is adjacent), we may think of $\bar{a}^f$ as the result of a "walk" on the tuple $\bar{a}$, starting at the element $a_{f(1)}$, and moving left, right, or remaining stationary according to the sequence of values $f(i+1)-f(i)$ $(1 \leq i < m)$.

For any $k \geq 0$, denote by $\mathbf{x}_k$ the fixed word $x_1 \cdots x_k$ (if $k = 0$, this is the empty word). A *k-atom* is an expression $p(\mathbf{x}_k^f)$, where p is a predicate of some arity $m \geq 0$, and $f\colon [1, m] \to [1, k]$. Thus, in a k-atom, each argument is a variable chosen from $\mathbf{x}_k$. If f is adjacent, we speak of an *adjacent k-atom*. Thus, in an adjacent k-atom, the indices of neighbouring arguments differ by at most one. When $k \leq 2$, the adjacency requirement is vacuous, and in this case we prefer to speak simply of *k-atoms*. Proposition letters (predicates of arity $m = 0$) count as (adjacent) k-atoms for all $k \geq 0$, taking f to be the empty function. When $k = 0$, we perforce have $m = 0$, since otherwise, there are no functions from $[1, m]$ to $[1, k]$; thus the 0-atoms are precisely the proposition letters.

We define the sets of first-order formulas $\mathcal{AF}^{[k]}$ by simultaneous structural induction:

1. every adjacent k-atom is in $\mathcal{AF}^{[k]}$;
2. $\mathcal{AF}^{[k]}$ is closed under Boolean combinations;
3. if φ is in $\mathcal{AF}^{[k+1]}$, $\exists x_{k+1}\, \varphi$ and $\forall x_{k+1}\, \varphi$ are in $\mathcal{AF}^{[k]}$.

Now let $\mathcal{AF} = \bigcup_{k \geq 0} \mathcal{AF}^{[k]}$ and define $\mathcal{AF}^k$ to be the set of formulas of $\mathcal{AF}$ featuring no variables other than $\mathbf{x}_k$, free or bound. We call $\mathcal{AF}$ the *adjacent fragment* and $\mathcal{AF}^k$ the *k-variable adjacent fragment*. Note that formulas of $\mathcal{AF}$ contain no individual constants, function symbols or equality. The primary objects of interest here are the languages $\mathcal{AF}$

and $\mathcal{AF}^k$; however, the sets of formulas $\mathcal{AF}^{[k]}$ play an important auxiliary role in their analysis. Thus, for example, formula (1) is in $\mathcal{AF}^k$ for all $k \geq 5$, but in $\mathcal{AF}^{[k]}$ only for $k = 0$. On the other hand, the *quantifier-free* formulas of $\mathcal{AF}^{[k]}$ and $\mathcal{AF}^k$ are the same.

We silently assume the variables $\mathbf{x}_k = x_1 \cdots x_k$ to be ordered in the standard way. That is: if φ is a formula of $\mathcal{AF}^{[k]}$, $\mathfrak{A}$ a structure interpreting its signature, and $\bar{a} = a_1 \cdots a_k \in A^k$, we say simply that $\bar{a}$ *satisfies* φ in $\mathfrak{A}$, and write $\mathfrak{A} \models \varphi[\bar{a}]$ to mean that $\bar{a}$ satisfies φ in $\mathfrak{A}$ under the assignment $x_i \leftarrow a_i$ $(1 \leq i \leq k)$. (This does not necessarily mean that each of the variables of $\mathbf{x}_k$ actually appears in φ.) If φ is true under all assignments in all structures, we write $\models \varphi$; the notation $\varphi \models \psi$ means the same as $\models \varphi \to \psi$ (i.e. variables are consistently instantiated in φ and ψ). The notation $\varphi(\bar{v})$, where $\bar{v} = v_1 \cdots v_k$ are variables, will always be used to denote the formula that results from substituting v_i for x_i $(1 \leq i \leq k)$ in φ. We write $\forall \mathbf{x}_{k;\ell}$ in place of $\forall x_k \forall x_{k+1} \cdots \forall x_\ell$ (and just $\forall \mathbf{x}_\ell$ if $k = 1$). A *sentence* is a formula with no free variables. Necessarily, all formulas of $\mathcal{AF}^{[0]}$ are sentences. For a sentence φ we write simply $\mathfrak{A} \models \varphi$ to mean that φ is true in $\mathfrak{A}$. We call the set of predicates used in φ *the signature of* φ (denoted $\mathsf{sig}(\varphi)$). By routine renaming of variables we establish:

▶ **Lemma 2.1.** *Every* FO^2*-formula is logically equivalent to an* $\mathcal{AF}$*-formula. The converse holds for* $\mathcal{AF}$*-formulas featuring predicates of arity at most two.*

We adapt the standard notion of (atomic) k-types for the fragments studied here. Fix some signature σ. An *adjacent k-literal over σ* is an *adjacent k-atom* or its negation, featuring a predicate in σ. An *adjacent k-type over σ* is a maximal consistent set of adjacent k-literals over σ. Reference to σ is suppressed where clear from context. We use the letters ζ and η to range over adjacent k-types for various k. We denote by Atp_k^σ the set of all adjacent k-types over σ. For finite σ, we identify members of Atp_k^σ with their conjunctions, and treat them as (quantifier-free) $\mathcal{AF}^k$-formulas, writing ζ instead of $\bigwedge \zeta$. When $k \leq 2$, the adjacency requirement is vacuous, and in this case we shall simply speak simply of *k-types*. It is obvious that every quantifier-free $\mathcal{AF}^k$-formula χ is logically equivalent to a disjunction of adjacent k-types, in essence the adjacent disjunctive normal form of χ. In particular, if χ is satisfiable, then there is an adjacent k-type which entails it. If $\mathfrak{A}$ is a σ-structure and $\bar{a}$ a k-tuple of elements from A, there is a unique adjacent k-type ζ such that $\mathfrak{A} \models \zeta[\bar{a}]$; we denote this adjacent k-type by $\mathrm{atp}^{\mathfrak{A}}[\bar{a}]$, and call it the *adjacent type of $\bar{a}$ in $\mathfrak{A}$*. If $\tau \subseteq \sigma$, we use $\mathrm{atp}_\tau^{\mathfrak{A}}[\bar{a}]$ to denote the adjacent type of $\bar{a}$ in $\mathfrak{A}$ restricted to predicates from τ. It is not required that the elements $\bar{a}$ be distinct. Again, if $k \leq 2$, adjacency is vacuous, and we write $\mathrm{tp}^{\mathfrak{A}}[\bar{a}]$ rather than $\mathrm{atp}^{\mathfrak{A}}[\bar{a}]$, and refer to $\mathrm{tp}^{\mathfrak{A}}[\bar{a}]$ as the *type of $\bar{a}$ in $\mathfrak{A}$*.

Since adjacent formulas do not contain equality, we may freely duplicate elements in their models. Let $\mathfrak{B}$ be a σ-structure, and I a non-empty set of indices. We define the structure $\mathfrak{B} \times I$ over the Cartesian product $B \times I$ by setting, for any $p \in \sigma$ of arity m, and any m-tuples $b_1 \cdots b_m$ from B and $i_1 \cdots i_m$ from I^m, $\mathfrak{B} \times I \models p[\langle b_1, i_1 \rangle \cdots \langle b_m, i_m \rangle]$ if and only if $\mathfrak{B} \models p[b_1 \cdots b_m]$. By routine structural induction:

▶ **Lemma 2.2.** *Let ψ be an equality-free first-order formula all of whose free variables occur in $\mathbf{x}_k$, $\mathfrak{B}$ a structure interpreting the signature of ψ, and I a non-empty set. Then, for any tuples $\bar{b} = b_1 \cdots b_k$ from B and $i_1 \cdots i_k$ from I, $\mathfrak{B} \models \psi[\bar{b}]$ if and only if $\mathfrak{B} \times I \models \psi[\langle b_1, i_1 \rangle \cdots \langle b_k, i_k \rangle]$.*

The following combinatorial lemma allows us to extend the technique of "circular witnessing" [12], frequently used in the analysis of two-variable logics, to the languages $\mathcal{AF}^k$.

▶ **Lemma 2.3.** *For any integer $k > 0$ there is a set J with $|J| = (k^2 + k + 1)^{k+1}$ and a function $g\colon J^k \to J$ such that, for any tuple $\bar{t} \in J^k$ consisting of the elements $t_1, \ldots, t_k$ in some order:* (i) *$g(\bar{t})$ is not in $\bar{t}$;* (ii) *if $\bar{t}' \in J^k$ consists of the elements $\{t_2, \ldots, t_k, g(\bar{t})\}$ in some order, then $g(\bar{t}')$ is not in $\bar{t}$ either.*

3 Primitive generators of words

The upper complexity bounds obtained below depend on an observation concerning the combinatorics of words, which may be of independent interest. For words $\bar{a}, \bar{c} \in A^*$ with $|\bar{a}| = k$ and $|\bar{c}| = m$, say that $\bar{a}$ *generates* $\bar{c}$ if $\bar{c} = \bar{a}^f$ for some *surjective* function $f \in \mathbf{A}_k^m$. As explained above, it helps to think of $\bar{a}^f$ as the sequence of letters encountered on an m-step "walk" backwards and forwards on the tuple $\bar{a}$, with $f(i)$ giving the index of our position in $\bar{a}$ at the ith step. The condition that f is adjacent ensures that we never change position by more than one letter at a time; the condition that f is surjective ensures that we visit every position of $\bar{a}$. We may picture a walk as a piecewise linear function, with the generat*ed* word superimposed on the abscissa and the generat*ing* word on the ordinate, c.f. Figure 1.

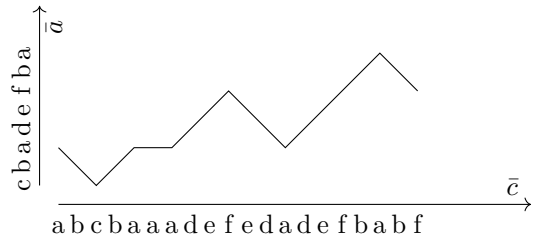


Figure 1 Generation of abcbaaadefedadefbabf from cbadefba.

Every word generates both itself and its reversal. Moreover, if $\bar{a}$ generates $\bar{c}$, then $|\bar{c}| \geq |\bar{a}|$; in fact, $\bar{a}$ and $\bar{a}^{-1}$ are the only words of length $|\bar{a}|$ generated by $\bar{a}$. Finally, generation is transitive: if $\bar{a}$ generates $\bar{b}$ and $\bar{b}$ generates $\bar{c}$, then $\bar{a}$ generates $\bar{c}$. We call $\bar{a}$ *primitive* if it is not generated by any word shorter than itself, equivalently, if it is generated only by itself and its reversal. For example, *babcd* and *abcbcd* are not primitive, because they are generated by *abcd*; but *abcbda* is primitive. Note that an infix (factor) of a primitive word need not be primitive. Define a *primitive generator* of $\bar{c}$ to be a generator of $\bar{c}$ that is itself primitive. From the foregoing remarks, it is obvious that every word $\bar{c}$ has some primitive generator $\bar{a}$, and indeed, $\bar{a}^{-1}$ as well, since the reversal of a primitive generator is clearly a primitive generator. The following result, by contrast, is anything but obvious. Notwithstanding the naturalness of the question it answers, we believe it to be new. Since it is concerned only with the combinatorics of strings, however, we refer the reader to [29] for the proof.

▶ **Theorem 3.1.** *The primitive generator of any word is unique up to reversal.*

Remarkably, while primitive generators are unique up to reversal, modes of generation are not. The word $\bar{c} = abcbcbd$ has primitive generator $\bar{a} = abcbd$. But there are *distinct* surjective functions $f, g \in \mathbf{A}_5^7$ such that $\bar{c} = \bar{a}^f = \bar{a}^g$, as is easily verified. Define the *primitive length* of any word $\bar{c}$ to be the length of any primitive generator of $\bar{c}$. By Theorem 3.1, this notion is well-defined; it will play a significant role in our analysis of the adjacent fragment. Obviously, the primitive length of $\bar{c}$ is at most $|\bar{c}|$, but will be strictly less if $\bar{c}$ is not primitive.

Let χ be a quantifier-free $\mathcal{AF}^\ell$-formula, and let $g \in \mathbf{A}_k^\ell$. We denote by χ^g the formula $\chi(x_{g(1)} \cdots x_{g(\ell)})$. We claim that $\chi^g \in \mathcal{AF}^k$. Indeed, any atom α appearing in χ is of the form $p(\mathbf{x}_k^f)$, where p is a predicate of some arity m and $f \in \mathbf{A}_\ell^m$. But then the corresponding atom in χ^g has the form $\beta \coloneqq \alpha(x_{g(1)} \cdots x_{g(\ell)}) = p(x_{g(f(1))} \cdots x_{g(f(m))}) = p(\mathbf{x}_k^{(g \circ f)})$. Since the composition of adjacent functions is adjacent, the claim follows. The following (almost trivial) lemma is useful when manipulating adjacent formulas. Recall in this regard that any function $g \in \mathbf{A}_k^\ell$ maps a k-tuple $\bar{a}$ over some set to an ℓ-tuple $\bar{a}^g$ over the same set.

▶ **Lemma 3.2.** *Let χ be a quantifier-free formula of $\mathcal{AF}^\ell$, and $g \in \mathbf{A}_k^\ell$. For any $\mathsf{sig}(\chi)$-structure $\mathfrak{A}$ and any $\bar{a} \in A^k$, we have $\mathfrak{A} \models \chi^g[\bar{a}]$ if and only if $\mathfrak{A} \models \chi[\bar{a}^g]$.*

Proof. We may assume without loss of generality that $\chi = p(\mathbf{x}_k^f)$ is atomic, with $f \in \mathbf{A}_\ell^m$; the general case follows by an easy structural induction. But then, writing $\bar{a} = a_1 \cdots a_m$, both sides of the bi-conditional amount to the statement $a_{g(f(1))} \cdots a_{g(f(m))} \in p^{\mathfrak{A}}$. ◀

The adjacent type of any tuple in $\mathfrak{A}$ is thus completely determined by that of its primitive generator. Indeed, let $\mathfrak{A}$ be a σ-structure, and $\bar{a}$ an ℓ-tuple from A. Then $\bar{a}$ has a primitive generator, say $\bar{b}$ of length $k \leq \ell$, with $\bar{a} = \bar{b}^g$ for some surjective $g \in \mathbf{A}_k^\ell$. Now consider any atomic $\mathcal{AF}^\ell$-formula α. By Lemma 3.2 we have $\mathfrak{A} \models \alpha[\bar{a}]$ if and only if $\mathfrak{A} \models \alpha^g[\bar{b}]$.

When evaluating $\mathcal{AF}^\ell$-formulas, for fixed ℓ, we can disregard any tuples whose primitive length is greater than ℓ. Indeed, consider a pair of σ-structures $\mathfrak{A}$ and $\mathfrak{A}'$ over a common domain A. We write $\mathfrak{A} \approx_\ell \mathfrak{A}'$, if, for any predicate p (of any arity $m \geq 0$), and any m-tuple $\bar{a}$ from A of primitive length at most ℓ, $\bar{a} \in p^{\mathfrak{A}}$ if and only if $\bar{a} \in p^{\mathfrak{A}'}$. That is, $\mathfrak{A} \approx_\ell \mathfrak{A}'$ just in case $p^{\mathfrak{A}}$ and $p^{\mathfrak{A}'}$ agree on all those m-tuples whose primitive length is at most ℓ. The following may be proved by structural induction, using Lemma 3.2.

▶ **Lemma 3.3.** *Let φ be an $\mathcal{AF}^\ell$-sentence, and suppose $\mathfrak{A}$ and $\mathfrak{A}'$ are $\mathsf{sig}(\varphi)$-structures over a common domain A such that $\mathfrak{A} \approx_\ell \mathfrak{A}'$. Then $\mathfrak{A} \models \varphi \Rightarrow \mathfrak{A}' \models \varphi$.*

Proof. Let ψ be a *formula* of $\mathcal{AF}^\ell$ (possibly featuring free variables), and let k ($0 \leq k \leq \ell$) be such that $\psi \in \mathcal{AF}^{[k]}$. (We may as well take the smallest such k.) We claim that, for any k-tuple of elements $\bar{a}$, $\mathfrak{A} \models \psi[\bar{a}]$ if and only if $\mathfrak{A}' \models \psi[\bar{a}]$. To see this, suppose first that ψ is atomic. We may write $\psi := p(\mathbf{x}_k^f)$, where p is a predicate (of arity, say, m), and $f \in \mathbf{A}_k^m$. If $\bar{a}$ is a k-tuple of elements from A, then $\mathfrak{A} \models \psi[\bar{a}]$ if and only if $\bar{a}^f \in p^{\mathfrak{A}}$. But the primitive length of $\bar{a}^f$ is certainly at most $k = |\bar{a}|$. This proves our claim for all k ($0 \leq k \leq \ell$) and for all *atomic* $\psi \in \mathcal{AF}^{[k]}$. The general case follows simply by structural induction. The statement of the lemma is the special case where ψ has no free variables. ◀

In view of Lemma 3.3, when considering models of $\mathcal{AF}^\ell$-sentences, it will be useful to take the extensions of predicates (of whatever arity) to be *undefined in respect of tuples whose primitive length is greater than ℓ*, since these cannot affect the outcome of semantic evaluation. That is, where ℓ is clear from context, we typically suppose any model $\mathfrak{A}$ of φ to determine whether $\bar{a} \in p^{\mathfrak{A}}$ for any m-ary predicate p and any m-tuple $\bar{a}$ *of primitive length at most ℓ*; but with respect to m-tuples $\bar{a}$ having greater primitive length, $\mathfrak{A}$ remains agnostic. To make it clear that the structure $\mathfrak{A}$ need not be fully defined, we speak of such a structure $\mathfrak{A}$ as a *layered* structure, and we refer to ℓ as its *primitive length*. Notice that the notion of primitive length is independent of the arities of the predicates interpreted. A layered structure $\mathfrak{A}$ may have primitive length, say 3, but still interpret a predicate p of arity, say, 5. In this case, it is determined whether $\mathfrak{A} \models p[babcc]$, because the primitive generator of *babcc* is *abc*; however, it will not be determined whether $\mathfrak{A} \models p[abcab]$, because *abcab* is primitive.

One of the intriguing aspects of layered structures is that they allow us to build up models of $\mathcal{AF}$-formulas layer by layer. Suppose $\mathfrak{A}$ has primitive length k; and we wish to construct a layered structure $\mathfrak{A}^+$ of primitive length $k{+}1$ over the same domain A, agreeing with the assignments made by $\mathfrak{A}$. Clearly, it suffices to fix the adjacent type of each primitive $(k{+}1)$-tuple $\bar{b}$ from A. To fix the adjacent type of $\bar{b}$ – and hence that of its reversal, $\bar{b}^{-1}$ – we consider each predicate p in turn – of arity, say, m – and decide, for any m-tuple $\bar{c}$ from A whose primitive generator is $\bar{b}$, whether $\mathfrak{A} \models p[\bar{c}]$. Now repeat this process for all pairs of mutually inverse primitive words $(\bar{b}, \bar{b}^{-1})$ from A having primitive length $k{+}1$. Since every

tuple $\bar{c}$ considered for inclusion in the extension of some predicate has primitive length $k+1$, these assignments will not clash with any previously made in the original structure $\mathfrak{A}$. Moreover, since, by Theorem 3.1, every m-tuple $\bar{c}$ assigned in this process has a unique primitive generator $\bar{b}$ (up to reversal), these assignments will not clash with each other. Thus, to increment the primitive length of $\mathfrak{A}$, one takes each inverse pair $(\bar{b}, \bar{b}^{-1})$ of primitive $(k+1)$-tuples in turn, and fixes the adjacent type of each $\bar{b}$ consistently with the existing assignments of all tuples generated by proper infixes of $\bar{b}$, as given in the original structure $\mathfrak{A}$.

We finish this section with an easy technical observation that will be needed in the sequel. Denote by $\vec{\mathbf{A}}_k^m$ the set of all functions $f \in \mathbf{A}_k^m$ such that $f(m) = k$. Thus, if $f \in \vec{\mathbf{A}}_k^m$ is used to define a walk of length m on some word $\bar{a}$ of length k, then the walk in question ends at the final position of $\bar{a}$.

▶ **Lemma 3.4.** *Let $\bar{c}$ be a word of length $m \geq 0$ over some alphabet A, and d an element of A that does not appear in $\bar{c}$. If $\bar{c}d$ is not primitive, then neither is $\bar{c}$. In fact, there is a word $\bar{a}$ of length $k < m$ and a function $f \in \vec{\mathbf{A}}_k^m$ such that $\bar{a}^f = \bar{c}$.*

Proof. Suppose $\bar{c}d = \bar{b}^g$ for some word $\bar{b}$ of length $k+1 \leq m$ and some surjective $g \in \mathbf{A}_{k+1}^{m+1}$. Since d does not occur in $\bar{c}$, it is immediate that d occupies either the first or last position in $\bar{b}$ for, otherwise, it would be encountered again in the entire traversal of $\bar{b}$ (as g is adjacent and surjective). By reversing $\bar{b}$ if necessary, assume the latter, so that we may write $\bar{b} = \bar{a}d$, with $g(m+1) = k+1$. By adjacency, $g(m) = k$, so that setting $f = g \setminus \{\langle m+1, k+1\rangle\}$, we have the required $\bar{a}$ and f. ◀

Finally, we remark that, if $f \in \vec{\mathbf{A}}_k^m$, then the function $f^+ = f \cup \langle m+1, k+1\rangle$ satisfies $f^+ \in \vec{\mathbf{A}}_{k+1}^{m+1}$. That is, we can extend f by setting $f(m+1) = k+1$, retaining adjacency.

4 Upper bounds for $\mathcal{AF}$ and $\mathcal{AF}^k$

In this section, we establish a small model property for each of the fragments $\mathcal{AF}^k$ with $k \geq 3$. Define the function $\mathfrak{t}(k, n)$ inductively by $\mathfrak{t}(0, n) = n$ and $\mathfrak{t}(k+1, n) = 2^{\mathfrak{t}(k,n)}$. We show that, for some fixed polynomial p, if φ is a satisfiable formula of $\mathcal{AF}^k$, then φ is satisfied in a structure of size at most $\mathfrak{t}(k-2, p(\|\varphi\|))$. We proceed by induction, establishing first the base case for $k = 3$, and then reducing the case $k+1$ to the case k. It follows that the satisfiability problem (= finite satisfiability problem) for $\mathcal{AF}^k$ is in $(k-2)$-NExpTime for all $k \geq 3$. The best lower complexity bound is $\lfloor k/2 \rfloor$-NExpTime-hard, from the k-variable fluted fragment [30]. For $k \leq 2$, the adjacency restriction has no effect on the complexity of satisfiability. Thus satisfiability for $\mathcal{AF}^2$ is NExpTime-complete, while for $\mathcal{AF}^1$ and $\mathcal{AF}^0$ it is NPTime-complete. We begin by establishing a normal form lemma for $\mathcal{AF}$.

▶ **Lemma 4.1.** *Let φ be a sentence of $\mathcal{AF}^{\ell+1}$, where $\ell \geq 2$. We can compute, in polynomial time, an $\mathcal{AF}^{\ell+1}$-formula ψ satisfiable over the same domains as φ, of the form*

$$\bigwedge_{i \in I} \forall \mathbf{x}_\ell \exists x_{\ell+1}\, \gamma_i \wedge \forall \mathbf{x}_{\ell+1}\, \delta, \tag{2}$$

where I is a finite index set, the formulas γ_i and δ are quantifier-free.

Let φ be a normal-form $\mathcal{AF}^{\ell+1}$-formula as given in (2), over signature σ. Recall the operation $\cdot^f$ on quantifier-free adjacent formulas employed in Lemma 3.2, as well as the sets of functions $\vec{\mathbf{A}}_k^\ell$ employed in Lemma 3.4. For any $f \in \vec{\mathbf{A}}_k^\ell$, we continue to write f^+ for the function (in $\vec{\mathbf{A}}_{k+1}^{\ell+1}$) extending f by setting $f(\ell+1) = k+1$. Now define the *adjacent closure of* φ, denoted $\varphi^\#$, to be:

$$\bigwedge_{i\in I}\bigwedge_{k=1}^{\ell-1}\bigwedge_{f\in\vec{\mathbf{A}}_k^\ell}\forall\mathbf{x}_k\exists x_{k+1}\,\gamma_i^{f^+}\wedge\bigwedge_{k=1}^{\ell}\bigwedge_{g\in\mathbf{A}_k^{\ell+1}}\forall\mathbf{x}_k\,\delta^g.$$

Observe that the conjunctions for the $\forall^\ell\exists$-formulas range over $\vec{\mathbf{A}}_k^\ell$, while the conjunctions for the purely universal formula range over the whole of $\mathbf{A}_k^{\ell+1}$. Up to trivial logical rearrangement and re-indexing of variables, $\varphi^\#$ is actually a normal-form $\mathcal{AF}^\ell$-formula. In effect, $\varphi^\#$ is the result of identifying various universally quantified variables in φ in a way which preserves adjacency. The following lemma is therefore immediate.

▶ **Lemma 4.2.** *Let $\varphi\in\mathcal{AF}^{\ell+1}$ be in normal-form. Then $\varphi\models\varphi^\#$.*

The following notation will be useful. If χ is any quantifier-free $\mathcal{AF}^{\ell+1}$-formula, we denote by χ^{-1} the formula $\chi(x_{\ell+1},\ldots,x_1)$ obtained by simultaneously replacing each variable x_h by $x_{\ell-h+2}$ $(1\leq h\leq\ell+1)$; and we denote by $\hat{\chi}$ the formula $\chi\wedge\chi^{-1}$. Obviously χ^{-1} and $\hat{\chi}$ are also in $\mathcal{AF}^{\ell+1}$. If η is an adjacent ℓ-type, we denote by η^+ the quantifier-free $\mathcal{AF}^{\ell+1}$-formula $\eta(x_2,\ldots,x_{\ell+1})$ obtained by incrementing the index of each variable. Finally, if χ is a quantifier-free $\mathcal{AF}^{\ell+1}$-formula over some signature σ (which we take to be given by context), we denote by χ° the quantifier-free $\mathcal{AF}^\ell$-formula $\bigvee\{\eta\in\mathrm{Atp}_\ell^\sigma\mid\chi\wedge\eta^+\text{ is consistent}\}$. The intuition in this last case is that, if a is an element and $\bar{a}$ an ℓ-tuple of elements such that $a\bar{a}$ satisfies χ in some structure, then χ° is the strongest statement that follows regarding $\bar{a}$.

Now we are in a position to tackle the main task of this section, namely, to bound the complexity of the satisfiability problem for $\mathcal{AF}^k$ $(k\geq 3)$. Certainly the satisfiability problem for $\mathcal{AF}^2$ is in NExpTime, since any normal-form $\mathcal{AF}^2$-formula is in FO^2. Here, we strengthen that result to $\mathcal{AF}^3$ (which will sharpen the bound of Theorem 4.9 by one exponential). The proof is similar to an analogous result for the three-variable *fluted* fragment, $\mathcal{FL}^3$ [30, Lemma 4.5]

Let σ be a relational signature. If π is a 1-type over σ, define the 2-type π^2, over the same signature, to be $\{\lambda\mid\lambda\text{ a literal in }\mathcal{AF}^{[2]}\text{ s.t. }\lambda(x_1,x_1)\in\pi\}$. The intuition here is that if π is the type of an element a in some structure, then π^2 is the type of the pair aa. A *connector-type* (*over* σ) is a set ω of 2-types over σ subject to the condition that there exists some 1-type π over σ such that $\pi^2\in\omega$ and $\zeta\models\pi$ for all $\zeta\in\omega$. This 1-type π is clearly unique, and we denote it by $\mathrm{tp}(\omega)$. If $\mathfrak{A}$ is any structure interpreting σ and $a\in A$, then a defines a connector-type ω over σ in a natural way by setting $\omega=\{\mathrm{tp}^{\mathfrak{A}}[a,b]\mid b\in A\}$. We refer to ω as *the connector-type of a in $\mathfrak{A}$*, and denote it $\mathrm{con}^{\mathfrak{A}}[a]$. It follows immediately from the above definitions that $\mathrm{tp}(\mathrm{con}^{\mathfrak{A}}[a])=\mathrm{tp}^{\mathfrak{A}}[a]$. When speaking of connector-types, we suppress reference to σ if irrelevant or clear from context.

Let φ be a normal-form formula of $\mathcal{AF}^3$, as given in (2), with $\ell=2$, with $\sigma=\mathsf{sig}(\varphi)$. In the sequel we refer freely to the subformulas γ_i $(i\in I)$ and δ of φ. Say that a connector-type ω is *compatible* with φ if the following conditions hold:

L$\exists_1$: for all $i\in I$, there exists $\eta\in\omega$ s.t. $\eta\models\gamma_i(x_1,x_1,x_2)$.

L$\exists_2$: for all ζ such that $\zeta^{-1}\in\omega$ and all $i\in I$, there exists $\eta\in\omega$ such that the $\mathcal{AF}^3$-formula $\zeta\wedge\eta^+\wedge\gamma_i\wedge\hat{\delta}$ is consistent;

L$\forall_1$: for all $\eta\in\omega$ and all $f\in\mathbf{A}_2^3$, $\eta\models\delta^f$;

L$\forall_2$: for all ζ such that $\zeta^{-1}\in\omega$ and all $\eta\in\omega$, the $\mathcal{AF}^3$-formula $\zeta\wedge\eta^+\wedge\hat{\delta}$ is consistent.

The proofs of Lemmas 4.3–4.5 are straightforward and will be omitted.

▶ **Lemma 4.3.** *If φ is a normal-form $\mathcal{AF}^3$-formula, $\mathfrak{A}\models\varphi$ and $a\in A$, then $\mathrm{con}^{\mathfrak{A}}[a]$ is compatible with φ.*

A set Ω of connector-types is said to be *coherent* if the following conditions hold:
G$\exists$: for all $\omega \in \Omega$ and all $\zeta \in \omega$, there exists $\omega' \in \Omega$ such that $\zeta^{-1} \in \omega'$;
G$\forall$: for all $\omega, \omega' \in \Omega$, there exists a 2-type ζ such that $\zeta \in \omega$ and $\zeta^{-1} \in \omega'$.

▶ **Lemma 4.4.** *Let $\mathfrak{A}$ be a structure. Then $\Omega = \{\mathrm{con}^{\mathfrak{A}}[a] \mid a \in A\}$ is coherent.*
Define a *certificate* for φ to be a non-empty, coherent set of connector-types, all of which are compatible with φ.

▶ **Lemma 4.5.** *Any satisfiable normal-form $\mathcal{AF}^3$-formula has a certificate Ω such that both $|\Omega|$ and $|\bigcup \Omega|$ are $2^{O(\|\varphi\|)}$.*

We are now in a position to obtain a bound on the size of models of $\mathcal{AF}^3$-formulas.

▶ **Lemma 4.6.** *Let φ be a normal-form $\mathcal{AF}^3$-formula over a signature σ. If φ is satisfiable, then it has a model of size $2^{O(\|\varphi\|)}$.*

Proof. We may assume without loss of generality that σ features no proposition letters. Let φ be as given by (2). By Lemma 4.5, φ has a certificate Ω of cardinality at most $2^{O(\|\varphi\|)}$; moreover the set of 2-types T occurring anywhere in Ω is $2^{O(\|\varphi\|)}$. Let $H = \{0, 1, 2\}$, let I be the index set occurring in φ, let J be a set of cardinality $343 = 7^3$, and let $g\colon J^2 \to J$ a function satisfying the conditions of Lemma 2.3 with $k = 2$. Defining $A = \Omega \times T \times H \times I \times J$, we see that $|A|$ is $2^{O(\|\varphi\|)}$, as required by the lemma. We write any element $a \in A$ as (ω, ζ, h, i, j). We shall construct a layered model $\mathfrak{A} \models \varphi$ of primitive length 3 over this domain, proceeding layer by layer. In the sequel, bear in mind that a pair or triple of elements is primitive if and only if those elements are distinct.

Stage 1. We set the 1-type of any $a = (\omega, \zeta, h, i, j)$ to be $\mathrm{tp}^{\mathfrak{A}}[a] = \mathrm{tp}(\omega)$. Clearly, all these determinations can be made independently, since σ features no proposition letters. At this point, we have a layered structure of primitive length 1.

Stage 2. Now consider any $a = (\omega, \zeta, h, i, j) \in A$ and any $\eta \in \omega$. By (G$\exists$), there exists $\omega_\eta \in \Omega$ such that $\eta^{-1} \in \omega_\eta$. For each $i' \in I$ and $j' \in J$ set $\mathrm{tp}^{\mathfrak{A}}[a, a_{i',j'}] = \eta$, where $a_{i',j'}$ denotes the element $(\omega_\eta, \eta, h{+}1, i', j')$. (Here the addition in "$h{+}1$" is taken modulo 3.) The index η ensures that the $a_{i',j'}$ are chosen to be distinct for distinct $\eta \in \omega$. Moreover, the index $h{+}1$ ensures that this process can be carried out for every $a \in A$ without danger of clashes. (This is the familiar technique of "circular witnessing" [12].) Finally, suppose $a = (\omega, \zeta, h, i, j)$ and $a' = (\omega', \zeta', h', i', j')$ are distinct elements of A for which $\mathrm{tp}^{\mathfrak{A}}[a, a']$ has not yet been defined. By (G$\forall$), there exists $\eta \in \omega$ such that $\eta^{-1} \in \omega'$, and we set $\mathrm{tp}^{\mathfrak{A}}[a, a'] = \eta$. At the end of this process, all 1- and 2-types have been defined, and we thus have a layered structure of primitive length 2. From the foregoing construction, if $a = (\omega, \zeta, h, i, j) \in A$ and $\eta \in \omega$, then there exists a constructor-type ω' such that $\mathrm{tp}^{\mathfrak{A}}[a, b] = \eta$ for each $b \in A$ of the form $(\omega', \eta, h + 1, i', j')$ (where $i' \in I$, $j' \in J$); moreover, for all $a = (\omega, \zeta, h, i, j)$ and $b = (\omega', \zeta', h', i', j')$ with $\mathrm{tp}^{\mathfrak{A}}[a, b] = \eta$, we are guaranteed that $\eta \in \omega$ and $\eta^{-1} \in \omega'$. We remark that, in particular, $\mathrm{con}^{\mathfrak{A}}[a] = \omega$. It follows from L$\exists_1$ that, for every $a \in A$ and every $i \in I$, there exists $b \in A$ such that $\gamma_i[a, a, b]$. Another way of saying this is that, for every pair of elements a_1, a_2 whose primitive length is 1 (i.e. $a_1 = a_2$), $\mathfrak{A}$ provides a witness for the formula $\exists x_3\, \gamma_i$. Likewise, it follows from L$\forall_1$ that, for every triple $\bar{a}$ whose primitive length is either 1 or 2, $\mathfrak{A} \models \delta[\bar{a}]$. Indeed, if $\bar{a} = \bar{b}^f$ where $|\bar{b}| \leq 2$, we have $\mathfrak{A} \models \delta^f[\bar{b}]$, whence $\mathfrak{A} \models \delta[\bar{a}]$ by Lemma 3.2.

Stage 3. We now increment the primitive length of $\mathfrak{A}$ to 3 by setting the adjacent 3-types of all primitive triples in $\mathfrak{A}$. Fix any pair of distinct elements $a = (\omega, \tau, h, i, j)$ and $a' = (\omega', \tau', h', i', j')$. Let us write $\zeta = \mathrm{tp}^{\mathfrak{A}}[a, a']$, so that, by construction of $\mathfrak{A}$ in the previous stage, $\zeta \in \omega$ and $\zeta^{-1} \in \omega'$. By (L$\exists_2$), there exists some $\eta \in \omega'$ such that the $\mathcal{AF}^3$-formula $\psi \coloneqq \zeta \wedge \eta^+ \wedge \gamma_i \wedge \hat{\delta}$ is consistent; let θ_i be an adjacent 3-type entailing this formula. By the construction of the previous stage again, we can find an element $b_i \coloneqq (\omega'', \eta, h' + 1, i, g(j, j')) \in A$ such that $\mathrm{tp}^{\mathfrak{A}}[a', b_i] = \eta$. We shall set $\mathrm{atp}^{\mathfrak{A}}[a, a', b_i] = \theta_i$ for all $i \in I$. From the index i, the elements b_i are distinct, and so these assignments do not clash with each other. Since θ_i entails $\zeta \wedge \eta^+$, they do not clash with the 2-types assigned so far. Since θ_i entails γ_i, the pair a, a' now has a witness in respect of the formula $\exists x_3\, \gamma_i$. From property (i) of g secured by Lemma 2.3, the triple a, a', b_i is primitive; hence the only primitive triples whose adjacent types are thereby defined are a, a', b_i and b_i, a', a. But since θ_i entails $\hat{\delta}$, neither of these triples violates $\forall x_1 x_2 x_3\, \delta$. Now repeat this construction for all pairs of distinct elements $a = (\omega, \tau, h, i, j)$ and $a' = (\omega', \tau', h', i', j')$. We claim that no tuple $\bar{c}$ is assigned to the extensions of any predicates twice in this process. Since $\bar{c}$ must have some primitive generators $a_1a_2a_3$ and $a_3a_2a_1$, the only possibility for double assignment of $\bar{c}$ is if a_3 is chosen as some witness for the pair a_1, a_2, and a_1 is chosen as some witness for the pair a_3, a_2. Remembering that a_1, a_2 and a_3 are actually quintuples, let their final components be, respectively, j, j', j''. By the choice of witnesses, $j'' = g(j, j')$ and $j = g(j'', j')$. But this contradicts property (ii) of g secured by Lemma 2.3, thus establishing the claim that no primitive triple is assigned to extensions of predicates twice. At this point, for every pair of elements a_1a_2 (of primitive length either 1 or 2) and every $i \in I$, $\mathfrak{A}$ provides a witness for the formula $\exists x_3\, \gamma_i$. Moreover, no adjacent 3-type so-far assigned violates δ. To complete the extension of $\mathfrak{A}$ to primitive length 3, it remains only to assign adjacent types to all remaining primitive triples without violating δ. Suppose, then a, a', a'' are distinct elements whose adjacent type in $\mathfrak{A}$ has not yet been defined. Let $\zeta = \mathrm{tp}^{\mathfrak{A}}[a_1, a_2]$ and $\eta = \mathrm{tp}^{\mathfrak{A}}[a_2, a_3]$. By the previous stage, $\zeta \wedge \eta^+ \wedge \hat{\delta}$ is consistent, so let θ be an adjacent 3-type entailing this formula, and set $\mathrm{tp}^{\mathfrak{A}}[a_1, a_2, a_3] = \theta$. Observe that we are also thereby assigning the adjacent 3-type of $\mathrm{tp}^{\mathfrak{A}}[a_3, a_2, a_1]$, but are assigning no other adjacent 3-types. Since θ entails $\zeta \wedge \eta^+$, this assignment does not clash with the assignments of the previous step. Since θ entails $\hat{\delta}$, no newly assigned triple violates δ. This completes the construction of the model $\mathfrak{A}$. ◀

Extending Lemma 4.6 to the whole of $\mathcal{AF}$ represents a greater challenge. For the next two lemmas (4.7 and 4.8), fix a normal-form $\mathcal{AF}^{\ell+1}$-formula φ over some signature σ, as given in (2), with $\ell \geq 3$. We construct a normal-form formula $\varphi' \in \mathcal{AF}^{\ell}$ such that: (i) if φ is satisfiable over some domain A, then so is φ'; and (ii) if φ' is satisfiable over some domain B, then φ is satisfiable over a domain A, with $|A|/|B|$ bounded by some exponential function of $\|\varphi\|$.

Recall that the adjacent closure, $\varphi^{\#}$ of φ, may be regarded as a normal-form $\mathcal{AF}^{\ell}$-formula over the same signature. For every adjacent ℓ-type ζ over σ, let p_ζ be a fresh predicate of arity $\ell{-}1$. Intuitively, we shall think of $p(x_2 \cdots x_\ell)$ as stating "*for some* x_1, the ℓ-tuple $\mathbf{x}_\ell = x_1 \cdots x_\ell$ is of adjacent type ζ". Now define φ' to be the conjunction of $\varphi^{\#}$ with the following $\mathcal{AF}^{\ell}$-formulas:

$$\bigwedge_{\zeta \in \mathrm{Atp}^{\sigma}_{\ell}} \forall \mathbf{x}_\ell \big(\zeta \to p_\zeta(x_2 \cdots x_\ell)\big) \tag{3}$$

$$\bigwedge_{\zeta \in \mathrm{Atp}^{\sigma}_{\ell}} \bigwedge_{i \in I} \forall \mathbf{x}_{\ell-1} \exists x_\ell \big(p_\zeta(\mathbf{x}_{\ell-1}) \to (\zeta \wedge \hat{\delta} \wedge \gamma_i)^{\circ}\big) \tag{4}$$

$$\bigwedge_{\zeta \in \mathrm{Atp}^{\sigma}_{\ell}} \forall \mathbf{x}_{\ell} \big(p_{\zeta}(\mathbf{x}_{\ell-1}) \to (\zeta \wedge \hat{\delta})^{\circ} \big) \tag{5}$$

▶ **Lemma 4.7.** *Suppose $\mathfrak{A} \models \varphi$. Then we can expand $\mathfrak{A}$ to a model $\mathfrak{A}^{+} \models \varphi'$.*

Proof. Set $p_{\zeta}^{\mathfrak{A}^{+}} = \{\bar{a} \in A^{\ell-1} \colon \mathfrak{A} \models \zeta[a\bar{a}] \text{ for some } a \in A\}$, for every $\zeta \in \mathrm{Atp}^{\sigma}_{\ell}$. The truth of (3) in $\mathfrak{A}^{+}$ is then immediate. To see the same for (4), fix $\zeta \in \mathrm{Atp}^{\sigma}_{\ell}$ and $i \in I$, and suppose $\mathfrak{A}^{+} \models p_{\zeta}(\bar{a})$, where $\bar{a} \in A^{\ell-1}$. Then there exists $a \in A$ such that $\mathfrak{A} \models \zeta[a\bar{a}]$. Moreover, since $\mathfrak{A} \models \varphi$, there exists $b \in A$ such that $\mathfrak{A} \models \gamma_i[a\bar{a}b]$ and $\mathfrak{A} \models \hat{\delta}[a\bar{a}b]$. Now let $\eta = \mathrm{atp}^{\mathfrak{A}}[\bar{a}b]$. Writing χ for $\zeta \wedge \hat{\delta} \wedge \gamma_i$, we have $\mathfrak{A} \models \chi[a\bar{a}b]$, whence χ is consistent; and since $\mathfrak{A} \models \eta[\bar{a}b]$, it follows that $\eta \models \chi^{\circ}$. Thus, b is a witness for the $(\ell-1)$-tuple $\bar{a}$ required by the relevant conjunct of (4). This secures the truth of (4) in $\mathfrak{A}^{+}$. Formula (5) is handled similarly. ◀

▶ **Lemma 4.8.** *Suppose $\mathfrak{B} \models \varphi'$. Then we can construct a model $\mathfrak{C}^{+} \models \varphi$ such that $|C^{+}|/|B| \leq |I| \cdot (\ell^2 + \ell + 1)^{\ell}$.*

Proof. Since $\varphi' \in \mathcal{AF}^{\ell}$, we may assume by Lemma 3.3 that $\mathfrak{B}$ is a layered structure of primitive length ℓ – that is, does not specify the extensions of predicates in respect of tuples whose primitive length is greater than ℓ. Let $\mathfrak{B}^{-}$ be the reduct of $\mathfrak{B}$ to the signature σ (i.e. we forget the predicates p_{ζ}). Thus, every ℓ-tuple from B satisfies a unique element of $\mathrm{Atp}^{\sigma}_{\ell}$. We first define a collection of "witness" functions $v_i \colon B^{\ell} \to B$, where $i \in I$. For any ℓ-tuple $\bar{b} = b_1 \cdots b_{\ell}$, let $\zeta = \mathrm{atp}^{\mathfrak{B}^{-}}[\bar{b}]$. By (3), $\mathfrak{B} \models p_{\zeta}[b_2 \cdots b_{\ell}]$, whence, by (4), we may select $b \in B$ such that $\mathfrak{B} \models (\zeta \wedge \hat{\delta} \wedge \gamma_i)^{\circ}[(b_2 \cdots b_k b)]$. Set $v_i(\bar{b}) = b$. Now let J be a set of cardinality $\ell^2 + \ell + 1$ and let $g \colon J^{\ell} \to J$ a function satisfying conditions (i) and (ii) guaranteed by Lemma 2.3. We inflate the structure $\mathfrak{B}^{-}$ using the product construction of Lemma 2.2. Specifically, we define $\mathfrak{C} = \mathfrak{B}^{-} \times (I \times J)$, writing elements of $\mathfrak{C}$ as triples (b, i, j), where $b \in B$, $i \in I$ and $j \in J$. Now, predicate extensions featuring tuples of primitive length greater than ℓ can be safely disregarded in the structure $\mathfrak{C}$. We next define a collection of witness functions $w_i \colon C^{\ell} \to C$, based on the functions v_i defined above. The motivation is that these functions will allow us to choose witnesses in $\mathfrak{C}$ for the conjuncts (4) that do not, as it were, tread on each others' toes. Consider any ℓ-tuple $\bar{c} = c_1 \cdots c_{\ell}$ of elements in C, with $c_h = (b_h, i_h, j_h)$ for each h $(1 \leq h \leq \ell)$. Writing $\bar{b} = b_1 \cdots b_{\ell}$, we define $w_i(\bar{c})$ to be the element $(v_i(\bar{b}), i, g(j_1 \cdots j_{\ell}))$. Since $\mathfrak{B}^{-} \models (\zeta \wedge \hat{\delta} \wedge \gamma_i)^{\circ}[b_2 \cdots b_{\ell}\, w_i(\bar{b})]$, it follows from Lemma 2.2 that $\mathfrak{C} \models (\zeta \wedge \hat{\delta} \wedge \gamma_i)^{\circ}[c_2 \cdots c_{\ell}\, w_i(\bar{c})]$. In addition, the functions w_i satisfy the following two additional properties:

(w1) for fixed $\bar{c}$, the $w_i(\bar{c})$ are distinct as i varies over I;
(w2) $w_i(\bar{c})$ does not occur in $\bar{c}$;
(w3) if $\bar{c}'$ is an ℓ-tuple consisting of the elements $c_2, \ldots, c_{\ell}, w_i(\bar{c})$ in some order, then $w_{i'}(\bar{c}')$ does not occur in $\bar{c}$ for any $i' \in I$.

Indeed, (w1) is immediate from the fact $w_i(\bar{c})$ contains i as its second element; (w2) and (w3) follow, respectively, from conditions (i) and (ii) on g guaranteed in Lemma 2.3.

We are now ready to extend $\mathfrak{C}$ to a structure $\mathfrak{C}^{+}$ of primitive length $\ell+1$ such that $\mathfrak{C}^{+} \models \varphi$. We first manufacture witnesses required by the conjuncts $\forall \bar{x} \exists x_{\ell+1} \gamma_i$, insofar as these are not already present. Fix any ℓ-tuple $\bar{c} = c_1 \cdots c_{\ell}$, and let $\zeta = \mathrm{atp}^{\mathfrak{C}}[\bar{c}]$. Now consider any $i \in I$, and write $c = w_i(\bar{c})$. We have two cases, depending on whether the word $\bar{c}c$ is primitive. Suppose first that it is not. By (w2), c is not an element of $\bar{c}$, whence by Lemma 3.4 there is some k-tuple $\bar{d}$ $(k < \ell)$ and $f \in \vec{\mathbf{A}}^{\ell}_{k}$ such that $\bar{d} = \bar{c}^{f}$. As before, define $f^{+} \in \vec{\mathbf{A}}^{\ell+1}_{k+1}$ extending f by setting $f(k+1) = \ell + 1$. Since $k < \ell$, and $\mathfrak{C} \models \varphi^{\#}$, there exists $c' \in C$ such that $\mathfrak{C} \models \gamma_i^{f^{+}}[\bar{d}c']$. By Lemma 3.2, $\mathfrak{C} \models \gamma_i[(\bar{d}c')^{f^{+}}]$, or in other words, $\mathfrak{C} \models \gamma_i[\bar{c}c']$, so that a witness c' is already present in respect of the tuple $\bar{c}$ and the index i. (Notice that we are throwing

our original witness, c, away.) Suppose on the other hand that $\bar{c}c$ is primitive. Since $\mathfrak{C}$ has primitive length ℓ, no tuple with primitive generator $\bar{c}c$ has been assigned to the extension of any predicate in $\mathfrak{C}$. Let $\eta = \mathrm{atp}^{\mathfrak{C}}[c_2 \cdots c_\ell c]$. Writing χ for the $\mathcal{AF}^\ell$-formula $\zeta \wedge \hat{\delta} \wedge \gamma_i$ it follows from the choice of c that $\mathfrak{C} \models \chi^\circ[\bar{c}c]$, whence, by the definition of the operator $(\cdot)^\circ$, the $\mathcal{AF}^{\ell+1}$-formula $\eta^+ \wedge (\zeta \wedge \hat{\delta} \wedge \gamma_i)$ is consistent. Therefore, there exists an adjacent $(\ell+1)$-type ω entailing it, and we may fix $\mathrm{atp}^{\mathfrak{C}^+}[\bar{c}c] = \omega$. To see that this assignment makes sense and extends $\mathfrak{C}$, recall that $\mathrm{atp}^{\mathfrak{C}^+}[\bar{c}c]$ specifies whether $\mathfrak{C}^+ \models q[\bar{d}]$ for any m-tuple $\bar{d}$ whose primitive generator is an infix, say $\bar{e}$, of $\bar{c}c$. If $\bar{e}$ is of length ℓ or less, then its adjacent type has already been fixed in $\mathfrak{C}$ consistently with ζ or η^+. Otherwise, the primitive generator of $\bar{d}$ is $\bar{c}c$, so that $\mathfrak{C}$ does not determine satisfaction of q by $\bar{d}$; writing $\bar{d} = (\bar{c}c)^f$, then, we may set $\mathfrak{C}^+ \models q[\bar{d}]$ if and only if $\models \omega \to q((\mathbf{x}_\ell x_{\ell+1})^f)$. Since $\omega \models \gamma_i$, we see that, following these assignments, $\mathfrak{C}^+$ has been provided with a witness in respect of the tuple $\bar{c}$ and the index i. We claim in addition that the newly assigned tuples do not violate $\forall \mathbf{x}_\ell x_{\ell+1}\, \delta$. For suppose that $\bar{d}$ is an $(\ell+1)$-tuple whose adjacent type in $\mathfrak{C}^+$ has been defined. If the primitive length of $\bar{d}$ is ℓ or less, then we have $\bar{d} = \bar{e}^g$ for some primitive $\bar{e}$ of length $k \leq \ell$ and some $g \in \mathbf{A}_k^{\ell+1}$. Since $\mathfrak{C} \models \varphi^\#$, we have $\mathfrak{C} \models \delta^g[\bar{e}]$, whence by Lemma 3.2, $\mathfrak{C} \models \delta[\bar{d}]$. If, however, the primitive length of $\bar{d}$ is $\ell+1$, then $\bar{d}$ is either $\bar{c}c$ or its reversal, and by the fact that $\omega \models \hat{\delta}$, we have $\mathfrak{C}^+ \models \delta[\bar{d}]$ as required. Still keeping $\bar{c}$ fixed for the moment, we may carry out the above procedure for all $i \in I$. To see that these assignments do not interfere with each other, we simply note property (w1) of the functions w_i.

Now make these assignments as just described for *each* word $\bar{c} \in C^\ell$. To ensure that these assignments do not interfere with each other, we make use of properties (w1) and (w3) of the functions w_i. If $\bar{d}$ is an m-tuple that has been assigned (or not) to the extensions of various predicates by the process described above, then the two primitive generators of $\bar{d}$ must be of the form $\bar{c}c$ and $(\bar{c}c)^{-1}$, where $c = w_i(\bar{c})$ for some $i \in I$. Since primitive generators are unique up to reversal by Theorem 3.1, it suffices to show that, for distinct pairs $(\bar{c}, i)$ and $(\bar{c}', i')$, the corresponding $(\ell{+}1)$-tuples $(\bar{c}\, w_i(\bar{c}))$ and $(\bar{c}'\, w_{i'}(\bar{c}'))$ are not the same up to reversal. Now $\bar{c}\, w_i(\bar{c}) = \bar{c}'\, w_{i'}(\bar{c}')$ implies $\bar{c} = \bar{c}'$, whence i and i' are distinct, whence $w_i(\bar{c}) \neq w_{i'}(\bar{c}')$ by (w1), a contradiction. On the other hand if $\bar{c}\, w_i(\bar{c}) = (\bar{c}'\, w_{i'}(\bar{c}))^{-1}$, then $\bar{c}' = w_i(\bar{c}), c_\ell \cdots c_2$, whence $w_{i'}(\bar{c}')$ does not occur in $\bar{c}$ by (w3), again a contradiction.

At this point, we have assigned a collection of tuples with primitive length $\ell{+}1$ to the extensions of predicates in σ so as to guarantee that $\mathfrak{C}^+ \models \forall \mathbf{x}_\ell \exists x_{\ell+1}\, \gamma_i$ for all $i \in I$. In addition, no adjacent $(\ell{+}1)$-types thus defined violate $\forall \mathbf{x}_{\ell+1}\, \delta$. It remains to complete the specification of $\mathfrak{C}^+$ by defining the adjacent types of all remaining primitive $\ell{+}1$-tuples, and showing that, in the resulting structure, every $(\ell{+}1)$-tuple (primitive or not) satisfies δ. Let $c_1 \cdots c_{\ell+1}$ be a primitive $(\ell{+}1)$-tuple whose adjacent type has not yet been defined. Let $\zeta = \mathrm{atp}^{\mathfrak{C}}[c_1 \cdots c_\ell]$ and $\eta = \mathrm{atp}^{\mathfrak{C}}[c_2 \cdots c_{\ell+1}]$. Writing $c_h = (b_h, i_h, j_h)$ for all h $(1 \leq h \leq \ell+1)$, we have $\zeta = \mathrm{atp}^{\mathfrak{B}^-}[b_1 \cdots b_\ell]$ and $\eta = \mathrm{atp}^{\mathfrak{B}^-}[b_2 \cdots b_{\ell+1}]$. By (3), $\mathfrak{B} \models p_\zeta[b_2 \cdots b_{\ell+1}]$, and hence by (5), $\mathfrak{B} \models (\zeta \wedge \hat{\delta})^\circ[b_2 \cdots b_{\ell+1}]$, whence $\zeta \wedge \eta^+ \wedge \hat{\delta}$ is consistent, by the definition of the operator $(\cdot)^\circ$. So let $\omega \in \mathrm{Atp}^\sigma_{\ell+1}$ entail this formula, and set $\mathrm{atp}^{\mathfrak{C}^+}[\bar{c}c] = \omega$. Carrying this procedure out for all remaining primitive $\ell{+}1$-tuples, we obtain a layered structure $\mathfrak{C}^+$ of primitive length $\ell+1$. Let $\bar{d}$ be any $(\ell{+}1)$-tuple of elements from C. If $\bar{d}$ is primitive, then we have just ensured that $\mathfrak{C}^+ \models \delta[\bar{d}]$. If, on the other hand, $\bar{d} = \bar{e}^f$ for some k-tuple $\bar{e}$ and some $f \in \mathbf{A}_k^{\ell+1}$, where $k \leq \ell$, then, since $\varphi^\#$, we have $\mathfrak{C} \models \delta^f[\bar{e}]$ and hence, by Lemma 3.2, $\mathfrak{C} \models \delta[\bar{d}]$. This completes the construction of $\mathfrak{C}^+$. We have shown that $\mathfrak{C}^+ \models \varphi$. ◀

Lemma 4.6 establishes the decidability of satisfiability for $\mathcal{AF}^3$. Lemmas 4.7 and 4.8, on the other hand, reduce the satisfiability problem for $\mathcal{AF}^{\ell+1}$ to that for $\mathcal{AF}^\ell$ $(\ell \geq 3)$, though with exponential blow-up. Putting these together, we obtain the decidability of satisfiability

for the whole of $\mathcal{AF}$. More precisely:

▶ **Theorem 4.9.** *If φ is a satisfiable $\mathcal{AF}^{\ell+1}$-formula, with $\ell \geq 2$, then φ is satisfied in a structure of size at most $\mathfrak{t}(\ell{-}1, O(\|\varphi\|))$. Hence the satisfiability problem for $\mathcal{AF}^{\ell}$ is in $(\ell{-}2)$-NExpTime for all $\ell \geq 3$, and the adjacent fragment is Tower-complete.*

Proof. Fix $\ell \geq 2$ and suppose φ is a satisfiable $\mathcal{AF}^{\ell+1}$-formula over a signature σ. By Lemma 4.1, we may assume that φ is in normal form. Writing $\varphi_{\ell+1}$ for φ, let φ_ℓ be the formula φ' given by the conjunction of $\varphi^{\#}$ and formulas (3)–(5) as described before Lemma 4.7. Repeating this process, we obtain a sequence of formulas $\varphi_{\ell+1}, \ldots, \varphi_3$. By Lemma 4.7, φ_3 is satisfiable. For all k, $(3 \leq k \leq \ell+1)$, let φ_k have signature σ_k, and for $k \leq \ell$, consider the construction of φ_k from φ_{k+1}. Since $\sum_{k'=1}^{k+1} |\mathbf{A}_{k'}^{k+1}|$ is bounded by a constant, we see that $\|\varphi_{k+1}^{\#}\|$ is $O(\|\varphi_{k+1}\|)$. Turning now to the formulas corresponding to (3)–(5), we employ the same technique used in the proof of Lemma 4.5. When considering the adjacent k-types over σ_{k+1}, we may disregard all adjacent atoms whose argument sequence is not a substitution instance of some argument sequence $\mathbf{x}_k^g$ occurring in an atom of φ_{k+1}, as these cannot affect the evaluation of φ_{k+1}. And since $k \leq \ell$, the number of functions from $\mathbf{x}_k$ to itself is again bounded by a constant, so that the number of adjacent k-atoms over σ_{k+1} that we need to consider is $O(\|\varphi_{k+1}\|)$. Thus, the number of adjacent k-types over σ_{k+1} that we need to consider is $2^{O(\|\varphi_{k+1}\|)}$; and this bounds the number of conjuncts in (3)–(5) taken together. Some care is needed when calculating the sizes of these conjuncts themselves, as they feature the subformulas $(\zeta \wedge \hat{\delta} \wedge \gamma_i)^\circ$ and $(\zeta \wedge \hat{\delta})^\circ$. However, these are simply, in effect, disjunctive normal forms over atoms contained in φ_{k+1}, and hence have cardinality $2^{O(\|\varphi_{k+1}\|)}$, whence $\|\varphi_k\|$ is $2^{O(\|\varphi_{k+1}\|)}$. By an easy induction, then, $\|\varphi_3\|$ is $\mathfrak{t}(\ell-2, O(\|\varphi_{\ell+1}\|))$, i.e. $\mathfrak{t}(\ell-2, O(\|\varphi\|))$.

By Lemma 4.6, φ_3 has a model of cardinality $\mathfrak{t}(\ell-1, O(\|\varphi\|))$. Moreover, by Lemma 4.7, each of the formulas φ_k $(3 \leq k \leq \ell)$ has a model over a set, say B_k, such that $|B_{k+1}| \leq |B_k| \cdot \|\varphi_{k+1}\| \cdot (\ell^2+\ell+1)^\ell$. Since $\|\varphi_k\|$ is $\mathfrak{t}(\ell-3, O(\|\varphi\|))$ for all k and remembering that ℓ is a constant, we see that $\varphi = \varphi_{\ell+1}$ has a model of cardinality $O(\|\varphi\|^{\ell-1}\mathfrak{t}(\ell-1, O(\|\varphi\|)))$, that is to say $\mathfrak{t}(\ell-1, O(\|\varphi\|))$. ◀

5 The Guarded Subfragment

We next shift our attention to the *guarded subfragment* of the adjacent fragment, denoted $\mathcal{GA}$, defined as the intersection of the guarded fragment $\mathcal{GF}$ and $\mathcal{AF}$. Recall that in $\mathcal{GF}$, quantification is relativized by atoms, e.g. all universal quantification takes the form $\forall \bar{x}(\alpha \to \psi)$, where α (a *guard*) is an atom featuring all the variables in $\bar{x}$ and all the free variables of ψ. We show that the satisfiability problem for $\mathcal{GA}$, in contrast to $\mathcal{GF}^2$ (the two-variable guarded fragment), is 2ExpTime-complete, and thus as difficult as full $\mathcal{GF}$.

Our proof employs the same strategy as the 2ExpTime-hardness proof for $\mathcal{GF}$ by Grädel [11]. The novel part of the reduction here concerns a feature characteristic of hardness results for guarded logics [11, 19]. However, the fact that we are working in the guarded *adjacent* fragment means that existing techniques are not directly available.

Let $m \in \mathbb{N}$ and consider the following adjacent functions (the upper index is mapped to the lower one):

$$\lambda_1 \coloneqq \begin{pmatrix} 1 & 2 & 3 & 4 & \ldots & m{+}2 \\ 1 & 2 & 2 & 3 & \ldots & m{+}1 \end{pmatrix}, \lambda_2 \coloneqq \begin{pmatrix} 1 & 2 & 3 & 4 & \ldots & m{+}2 \\ 1 & 2 & 1 & 2 & \ldots & m \end{pmatrix}, \lambda_3 \coloneqq \begin{pmatrix} 1 & 2 & 3 & 4 & \ldots & m{+}2 \\ 1 & 2 & 3 & 3 & \ldots & m{+}1 \end{pmatrix}.$$

We show that repeated application of λ_1–λ_3 on the bit-string $\mathbf{011}^m$ yields the whole of $\mathbf{01}\{\mathbf{0},\mathbf{1}\}^m$.

▶ **Lemma 5.1.** *Let* $W_0 \subseteq \{\mathbf{0},\mathbf{1}\}^*$ *contain* $\mathbf{011}^m$ *and* $W_i \coloneqq W_{i-1} \cup \{\bar{w}^{\lambda_1}, \bar{w}^{\lambda_2}, \bar{w}^{\lambda_3} \mid \bar{w} \in W_{i-1}\}$. *Setting* $W \coloneqq \bigcup_{i\geq 0} W_i$, *we have* $\mathbf{01}\{\mathbf{0},\mathbf{1}\}^m \subseteq W$.

Proof. We inductively prove that, for any $i \in [0,m]$ and any $\bar{c} \in \{\mathbf{0},\mathbf{1}\}^i$, we have $\mathbf{01}\bar{c}\mathbf{1}^{m-i} \in W$. The base case ($i = 0$) follows from the assumption that W_0 contains $\mathbf{011}^m$, so let $i > 0$. We aim at generating any word $u = \mathbf{01}x\bar{c}\mathbf{1}^{m-i-1}$ for $x \in \{\mathbf{0},\mathbf{1}\}$. By induction hypothesis both $\bar{v} = \mathbf{01}\bar{c}\mathbf{1}^{m-i-1}\mathbf{1}$ and $\bar{w} = \mathbf{01}\bar{d}\mathbf{1}^{m-i-1}\mathbf{11}$ (where $\bar{c} = c_1\bar{d}$) are in W. We consider cases: (i) if $x = \mathbf{1}$ then $\bar{u} = \bar{v}^{\lambda_1}$, (ii) if both x and $c_1 = \mathbf{0}$ then $\bar{u} = \bar{v}^{\lambda_3}$, and otherwise, (iii) $x = \mathbf{0}$, $c_1 = \mathbf{1}$ and $\bar{u} = \bar{w}^{\lambda_2}$. Thus $\bar{u} \in W$. ◀

Let G_m and P be, respectively, $(m+2)$-ary and binary predicates. We define ζ_m^P to be the sentence below:

$$\forall xy\,\Big(P(xy) \to G_m(xy\underbrace{y\cdots y}_{m})\Big) \wedge \bigwedge_{i=1,2,3} \forall \mathbf{z}_{m+2}\,\Big(G_m(\mathbf{z}_{m+2}) \to G_m(\mathbf{z}_{m+2}^{\lambda_i})\Big).$$

Let $\mathfrak{A}$ be a model of ζ_m^P, and take any $(a,b) \in P^{\mathfrak{A}}$. By Lemma 5.1 we conclude that $G_m^{\mathfrak{A}}$ contains every word of the form $ab\{a,b\}^m$. Let R be some 4-ary relation symbol. In the forthcoming proof we also consider a $(2m+4)$-ary predicate F_m described by ϵ_m^R which is a conjunction of the following two sentences:

$$\forall yxx'y'\,\Big(R(yxx'y') \to F_m(\underbrace{y\ldots y}_{m}yxx'y'\underbrace{y'\ldots y'}_{m})\Big)$$

$$\bigwedge_{i,j\in\{0,1,2,3\}} \forall \mathbf{z}_{m+2}^{-1}\mathbf{z}'_{m+2}\,\Big(F_m(\mathbf{z}_{m+2}^{-1}\mathbf{z}'_{m+2}) \to F_m(z_{\lambda_i(m+2)}\cdots z_{\lambda_i(1)}z'_{\lambda_j(1)}\cdots z'_{\lambda_j(m+2)})\Big).$$

Here λ_0 is the identity function (i.e. $k \mapsto k$ for each $k \in [1,m]$). The intended meaning is that whenever $\mathfrak{A} \models \zeta_m^R$ holds, this implies that for any quadruple $baa'b' \in R^{\mathfrak{A}}$ we have that $\mathfrak{A} \models G[\bar{c}baa'b'\bar{c}']$ holds for all $\bar{c} \in \{a,b\}^m$ and $\bar{c}' \in \{a',b'\}^m$.

ATMs. An *Alternating Turing Machine* (ATM) $\mathcal{M}$ is a tuple $\langle \mathcal{Q}, \mathcal{S}, \mathcal{T}_l, \mathcal{T}_r, q_0, \kappa\rangle$, where $\mathcal{Q}$ is a finite set of states, $\mathcal{S}$ is a finite alphabet containing an empty-cell symbol "␣", $\mathcal{T}_l, \mathcal{T}_r : \mathcal{Q}\times\mathcal{S} \to \mathcal{Q}\times\mathcal{S}\times[-1,1]$ are, respectively, the left and right transition functions, $q_0 \in Q$ is the initial state, and $\kappa : \mathcal{Q} \to \{\forall,\exists\}$ is the state descriptor function stating if a given state is, respectively, *universal* $\forall$ or *existential* $\exists$. We say that a state q is *accepting* if and only if $\kappa(q) = \forall$, and there are no possible transitions given by $\mathcal{T}_l(q,s)$ and $\mathcal{T}_r(q,s)$ for any symbol $s \in \mathcal{S}$. Dually, q is *rejecting* if and only if $\kappa(q) = \exists$, and, again, there are no possible transitions. By tracking every step of the computation by $\mathcal{M}$ on $\bar{w} \in \mathcal{S}^*$ we obtain a binary tree structure $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}_l, \mathcal{E}_r\rangle$. Each vertex $v \in \mathcal{V}$ is labelled with some *configuration* $\langle q, \bar{s}, h\rangle$, where $q \in \mathcal{Q}$ is a machine state, $\bar{s} \in \mathcal{S}^m$ a word indicating the contents of the tape, and $h \in [0, m-1]$ an integer indicating the position of the head at some point in the computation. Each edge $(v,u) \in \mathcal{E}_\eta$ (where $\eta = l, r$) is labelled by a transition $\mathcal{T}_\eta(q, s_h)$, enabled in the configuration labelling of v. We call $\mathcal{G}$ the *configuration tree* of the computation by $\mathcal{M}$ on $\bar{w}$. Assuming $\mathcal{G}$ is finite, we say that it is *accepting* if no vertex is associated with a rejecting state. We identify the following three properties of $\mathcal{G}$. The root node is labelled with a configuration in which the machine is in state q_0, the head position is 0, and the tape is written with the string $\bar{w}$ followed by blanks. We call this property *initial configuration* (**IC**). Let u be a vertex labelled with a configuration in which the machine is in state q. If q is universal, then u will have two children; if q is existential, then u will have a single child;

if q is accepting, or rejecting then u will have no children. We call this property *successor existence* (**SE**). Suppose further that, in the configuration labelling of some node u we have that the head is reading the symbol s whilst in state q. Then any child v_η (s.t. $(u, v_\eta) \in \mathcal{T}_\eta$, where $\eta = l$ or $\eta = r$) represents the result of a single transition $\mathcal{T}_\eta(q, s) = (p, s', k)$, and thus is labelled with a configuration in which the machine state is p, s' is written in place of s, and the head is moved by a distance of k. We call this property *configuration succession* (**CS**).

Encoding numbers. Let $\text{BIN}^m_{x,y}$ be the canonical map from $[0, 2^n - 1]$ to bit-string representations of length $m \in \mathbb{N}$, using x as the *zero bit* and y as the *unit bit*. In the sequel, we will consider structures $\mathfrak{A}$ with elements labelled by a unary predicate O. If $\mathfrak{A} \models \neg O[a] \wedge O[b]$ we say that a, b *act as zero and unit bits*. We thus associate every word $\bar{c} \in \{a, b\}^+$ with an integer value given by the canonical map $\text{VAL}^{\mathfrak{A}}$ (this function depends on $\mathfrak{A}$, because it is $\mathfrak{A}$ that determines which is the zero bit and which is the unit bit.) Given two bit-strings $\bar{c}$ and $\bar{d}$ (not necessarily composed of the same elements) there is a classical way to define the following properties in the monadic fragment of FO (hence also in $\mathcal{GA}$):

- $\mathfrak{A} \models \text{LESS}(\bar{c}, \bar{d})$ iff $\text{VAL}^{\mathfrak{A}}(\bar{c}) < \text{VAL}^{\mathfrak{A}}(\bar{d})$
- $\mathfrak{A} \models \text{EQ}(\bar{c}, \bar{d})$ iff $\text{VAL}^{\mathfrak{A}}(\bar{c}) = \text{VAL}^{\mathfrak{A}}(\bar{d})$
- $\mathfrak{A} \models \text{EQ}(\bar{c}, \bar{d} + k)$ iff $\text{VAL}^{\mathfrak{A}}(\bar{c}) = \text{VAL}^{\mathfrak{A}}(\bar{d}) + k$, where $k \in [-1, 1]$.

Formally, the formulas are defined as follows:

$$\text{LESS}(\mathbf{z}_m, \mathbf{z}'_m) \coloneqq \bigvee_{i=1}^{m} \Big(\neg O(z_i) \wedge O(z'_i) \wedge \bigwedge_{j=i+1}^{m} \big(O(z_j) \leftrightarrow O(z'_j) \big) \Big)$$

$$\text{EQ}(\mathbf{z}_m, \mathbf{z}'_m) \coloneqq \bigwedge_{i=1}^{m} \Big(O(z_i) \leftrightarrow O(z'_i) \Big)$$

$$\text{EQ}(\mathbf{z}_m, \mathbf{z}'_m + 1) \coloneqq \bigwedge_{i=1}^{m} \Big(\big(O(z_i) \leftrightarrow O(z'_i) \big) \leftrightarrow \bigvee_{j=1}^{i-1} O(z_j) \Big)$$

and where $\text{EQ}(\mathbf{z}_m, \mathbf{z}'_m + 0) \coloneqq \text{EQ}(\mathbf{z}_m, \mathbf{z}'_m)$ and $\text{EQ}(\mathbf{z}_m, \mathbf{z}'_m - 1) \coloneqq \text{EQ}(\mathbf{z}'_m, \mathbf{z}_m + 1)$.

Fix an ATM $\mathcal{M}$ working in exponential space w.r.t any given input $\bar{w}$. Our goal is to construct a polynomial-size $\mathcal{GA}$-sentence $\varphi_{\mathcal{M},\bar{w}}$ which is satisfiable if and only if $\mathcal{M}$ has an accepting configuration tree on a given input $\bar{w}$. Utilising the fact that AExpSpace equals 2ExpTime, the reduction yields the desired bound on $\mathcal{GA}$. Now, take an accepting configuration tree $\mathcal{G} = (\mathcal{V}, \mathcal{E}_l, \mathcal{E}_r)$ for $\mathcal{M}$ and $\bar{w}$, and fix $n = |\bar{w}|$. We consider structures interpreting binary predicates V, R, Q_q (for each $q \in \mathcal{Q}$), quaternary predicates E_l, E_r and n-ary predicates H, S_s (for each $s \in \mathcal{S}$). We say that $\mathfrak{A}_0$ *embeds* $\mathcal{G}$ if there is $f : \mathcal{V} \to A_0^2$ such that for all $v \in \mathcal{V}$

(a) $\mathfrak{A}_0 \models V[f(v)]$,
(b) $\mathfrak{A}_0 \models R[f(v)]$ if v is the root node,
(c) $\mathfrak{A}_0 \models E_\eta[f(u)^{-1}, f(v)]$ if $(u, v) \in \mathcal{E}_\eta$, where $\eta = l, r$,
(d) $\mathfrak{A}_0 \models Q_q[f(v)]$ if the configuration v is in state q,
(e) $\mathfrak{A}_0 \models S_s[\text{BIN}^n_{f(v)}(i)]$ if v's i-th tape cell has symbol s,
(f) $\mathfrak{A}_0 \models H[\text{BIN}^n_{f(v)}(i)]$ if v's head is located over the i-th tape square.

We construct $\mathfrak{A}_0$ embedding $\mathcal{G}$ as follows: the domain A_0 is composed of fresh symbols $0_v, 1_v$ for each vertex $v \in \mathcal{V}$, for which we also put $f(v) = 0_v 1_v$. (Notice that $0_v 1_v$ is a word over A_0 of length 2.) We interpret the predicates V, R, E, Q_q, S_s and H as required by conditions (a)–(f). Then, we construct a $\varphi_{\mathcal{M},\bar{w}}$ in $\mathcal{GA}$ such that: (i) $\mathfrak{A}_0$ can be expanded to a model $\mathfrak{A}$ of $\varphi_{\mathcal{M},\bar{w}}$; and (ii) every model of $\varphi_{\mathcal{M},\bar{w}}$ embeds $\mathcal{G}$.

The first conjunct of $\varphi_{\mathcal{M},\bar{w}}$ requires pairs ab satisfying the predicate V to act as zero bits and unit bits, indicated by the predicate O:

$$\varphi_1 = \forall xy \Big(V(xy) \to \neg O(x) \wedge O(y) \Big)$$

We now add ζ_n^V and ζ_{2n}^V to the main formula $\varphi_{\mathcal{M},\bar{w}}$. Recall that the sentence ζ_m^V features an $(m+2)$-ary predicate G_m, and ensures that, if $\mathfrak{A} \models V[ab]$, then $\mathfrak{A} \models G_m[ab\bar{c}]$ for all $c \in \{a,b\}^m$. Writing φ_2, φ_3 and φ_4 as

$$\bigwedge_{p,q\in\mathcal{Q}}^{p\neq q} \forall xy \Big(V(xy) \to \big(\neg Q_p(xy) \vee \neg Q_q(xy) \big) \Big),$$

$$\bigwedge_{s,s'\in\mathcal{S}}^{s\neq s'} \forall xy\mathbf{z}_n \Big(G_n(xy\mathbf{z}) \to \neg \big(S_s(\mathbf{z}_n) \wedge S_{s'}(\mathbf{z}_n) \big) \Big),$$

$$\forall xy\mathbf{z}_n\mathbf{z}'_n \Big(G_{2n}(xy\mathbf{z}_n\mathbf{z}'_n) \to \Big(\big(H(\mathbf{z}_n) \wedge H(\mathbf{z}'_n) \big) \to \textsc{eq}(\mathbf{z}_n, \mathbf{z}'_n) \Big) \Big)$$

respectively, we ensure that every configuration is in at most one state at a time, every tape square of a configuration has at most one symbol, and the read-write head of any configuration is pointing to a single square at a time. Note that all of these formulas are guarded. However, the advertised behaviour of the guard predicates G_n and G_{2n} means, in essence, that the guards have no semantic effect.

We now secure the property (**IC**). Let μ_1 abbreviate the formula $Q_{q_0}(xy) \wedge H(\textsc{bin}^n_{xy}(0)) \wedge \bigwedge_{i=1}^{|\bar{w}|} S_{w_i}(\textsc{bin}^n_{x,y}(i{-}1))$, and μ_2 the formula $\textsc{less}(\textsc{bin}^n_{x,y}(|\bar{w}|{-}1), \mathbf{z}_n) \to S_{\sqcup}(\mathbf{z}_n)$. Writing

$$\varphi_5 \coloneqq \exists xy \Big(V(xy) \wedge R(xy) \Big) \wedge \forall xy \Big(R(xy) \to \big(\mu_1 \wedge \forall \mathbf{z}_n (G(xy\mathbf{z}_n) \to \mu_2) \big) \Big),$$

we ensure that there is a root configuration in which the machine state is q_0, the head is scanning square "0", and the tape is written with the string $\bar{w}$ followed by the requisite number of blanks.

Let $\mathrm{K}_\forall$ be the formula $\bigvee_{q\in\mathcal{Q}}^{\kappa(q)=\forall} Q_q(xy)$, and define $\mathrm{K}_\exists$ analogously. Similarly, we define $\mathrm{K}_\times$ to be a disjunction of rejecting states. The formula $\varphi_6 \coloneqq \forall xy(V(xy) \to \neg \mathrm{K}_\times(xy))$ ensures that no configuration is labelled with a rejecting state.

We next encode the transitions of $\mathcal{M}$, securing the property (**SE**). Let $\psi_\forall$ abbreviate the formula $\exists x'y' E_l(yxx'y') \wedge \exists x'y' E_r(yxx'y')$, and $\psi_\exists$ the formula $\exists x'y' E_l(yxx'y') \vee \exists x'y' E_r(yxx'y')$. Writing

$$\varphi_7 \coloneqq \bigwedge_{k=\forall,\exists} \forall yx \Big(V(xy) \to \Big(\mathrm{K}_k(xy) \to \psi_k \Big) \Big).$$

we ensure that, if $\mathfrak{A} \models V[a,b] \wedge \mathrm{K}_k[ab]$, then $\mathfrak{A}$ contains pairs encoding the appropriate successor configurations.

We next ensure that the transitions have the expected effect on the configurations they connect, securing the property (**CS**). For this, we need a further predicate, F_n, to act as a dummy guard. By adding $\epsilon_n^{E_\eta}$ to the main formula (for both $\eta = l, r$), we secure $\mathfrak{A} \models F_n[\bar{c}baa'b'\bar{c}']$ for all a, b, a', b' such that $\mathfrak{A} \models E_\eta[baa'b']$ with $\bar{c} \in \{a,b\}^n$, $\bar{c}' \in \{a',b'\}^n$. The formula φ_8 then ensures that any pair of parent and successor configurations have identical tape contents except (possibly) for the position scanned by the head, thus:

$$\varphi_8 \coloneqq \forall \mathbf{z}_n yxx'y'\mathbf{z}'_n \Big(F_n(\mathbf{z}_n yxx'y'\mathbf{z}'_n) \rightarrow \Big(\Big(\big(\neg H(\mathbf{z}_n) \wedge \textsc{eq}(\mathbf{z}_n, \mathbf{z}'_n)\big) \rightarrow \Big(\bigwedge_{s\in\mathcal{S}} \big(S_s(\mathbf{z}_n) \rightarrow S_s(\mathbf{z}'_n)\big)\Big)\Big)\Big).$$

Now let χ_1 abbreviate the formula $Q_p(x'y')$, χ_2 the formula $\textsc{eq}(\mathbf{z}'_n, \mathbf{z}_n) \rightarrow S_{s'}(\mathbf{z}'_n)$, and χ_3 the formula $\textsc{eq}(\mathbf{z}'_n, \mathbf{z}_n + k) \rightarrow H(\mathbf{z}'_n)$. In addition, we write ξ_{τ_η} for the sentence

$$\forall \mathbf{z}_n yxx'y'\mathbf{z}'_n \Big(G(\mathbf{z}_n yxx'y'\mathbf{z}'_n) \rightarrow \Big(\big(E_\eta(yxx'y') \wedge Q_q(xy) \wedge H(\mathbf{z}_n) \wedge S_s(\mathbf{z}_n)\big) \rightarrow \big(\chi_1 \wedge \chi_2 \wedge \chi_3\big)\Big)\Big).$$

Assuming the transition τ_η is of the form $(q, s) \mapsto (p, s', k)$, the formula ξ_{τ_η} states that, if in a certain configuration, the machine state is q and the head is reading symbol s, then in the η-side successor configuration defined by $\mathcal{T}_\eta$, the machine state will be p, the symbol s will have been replaced by s', and the head will have moved by k. To encode all possible transitions, we write φ_9 to be a conjunction of ξ_{τ_η} for each transition $\tau_\eta \in \mathcal{T}_\eta$ for both $\eta = l, r$.

Let $\mathfrak{A}_0$ embed some accepting $\mathcal{G}$ as described in (a)–(f). We expand $\mathfrak{A}_0$ to $\mathfrak{A}$ by setting $A = A_0$ with

1. $\mathfrak{A} \models \neg O[a]$ and $\mathfrak{A} \models O[b]$ if $\mathfrak{A}_0 \models V[ab]$,
2. $\mathfrak{A} \models G_m[ab\bar{c}]$ where $\mathfrak{A}_0 \models V[ab]$ and $\bar{c} \in \{a, b\}^m$ (for $m = n, 2n$),
3. $\mathfrak{A} \models F_n[\bar{c}baa'b'\bar{c}']$ where $\mathfrak{A}_0 \models S_\eta[baa'b']$, $\bar{c} \in \{a, b\}^n$ and $\bar{c}' \in \{a', b'\}^n$ (here $\eta = l, r$).

Recalling that $\mathcal{G}$ contains an initial configuration (**IC**), we have that $\mathfrak{A} \models \varphi_5$. Additionally, $\mathcal{G}$ has the property (**SE**), we see that $\mathfrak{A} \models \varphi_7$. Lastly, since $\mathcal{G}$ has the property (**CS**), we have that $\mathfrak{A} \models \varphi_8$. At this point it is easy to verify that $\mathfrak{A} \models \varphi_{\mathcal{M},\bar{w}}$.

Conversely, suppose $\mathfrak{A} \models \varphi_{\mathcal{M},\bar{w}}$. We construct an embedding $f\colon \mathcal{V} \rightarrow A^2$ for an accepting $\mathcal{G}$ by well-founded induction. The following observations will be used. Suppose $\mathfrak{A} \models V[ab]$. Intuitively, we think of the pair ab as a vertex of the computation tree labelled by some configuration, as determined by the predicates Q_q, S_s and H. By φ_2, there is a unique Q_q (for $q \in \mathcal{Q}$) satisfied by ab. Moreover, by φ_3, any bit-string $\bar{c} \in \{a, b\}^n$ satisfies a unique S_s (for $s \in \mathcal{S}$). Similarly, by φ_4, there is a unique $\bar{c}$ satisfying H.

Proceeding with the induction, for the base case, pick a, b s.t. $\mathfrak{A} \models R[ab]$. By φ_5 we see that $\mathfrak{A} \models Q_{q_0}[ab]$, $\mathfrak{A} \models H[\textsc{bin}^n_{a,b}(0)]$, and $\mathfrak{A} \models S_{w_i}[\bar{c}]$ for each $1 \leq i \leq |\bar{w}|$ with $\textsc{val}^{\mathfrak{A}}(\bar{c}) = i-1$ and $\mathfrak{A} \models S_\sqcup[\bar{c}]$ otherwise. We then set $\mathcal{V} = \{v\}$ and $f(v) = ab$. Labeling v with the state, tape and head position as suggested by (a)–(f), we have secured the property (**IC**).

For the inductive step, let u be vertex which has been added to the tree. Assume u is labelled with a configuration in which the machine state q is universal, and the head is at position h, reading symbol s. If q is accepting, then we stop. Otherwise, φ_7 guarantees that there are words $a_\eta b_\eta \in A^2$ such that $\mathfrak{A} \models S_\eta[f(u)^{-1} a_\eta b_\eta]$ for both $\eta = l, r$. Notice that by φ_8 if the configuration labelling u has the symbol s' written on tape square i (for $i \neq h$), then $\mathfrak{A} \models S_{s'}[\textsc{bin}^n_{a_\eta b_\eta}(i)]$. By φ_9 the pair $a_\eta b_\eta$ satisfies the predicate Q_p that is in accordance with the transition $\mathcal{T}_\eta(q, s) = (p, s', k)$. Additionally, $\mathfrak{A} \models S_{s'}[\textsc{bin}^n_{a_\eta b_\eta}(h)]$ and $\mathfrak{A} \models H[\textsc{bin}^n_{a_\eta b_\eta}(h + k)]$.

We thus set $\mathcal{V} \coloneqq \mathcal{V} \cup \{v_\eta\}$, $f(v_\eta) = (a_\eta, b_\eta)$ and $\mathcal{E}_\eta \coloneqq \mathcal{E}_\eta \cup \{(u, v_\eta)\}$ thus securing (**SE**). By interpreting the state, tape and head position of v_η as suggested by (a)–(f) we see that v_η is a proper successor of u as required by (**CS**). The case for when q is existential is similar.

Since there are no rejecting states in conf. tree (reference φ_6), there is an initial configuration by (**IC**), each parent has children complying with (**SE**), and each parent-child pair conforms to (**CS**), we conclude that $\mathcal{G}$ is an accepting configuration tree.

6 Conclusions

The adjacent fragment $\mathcal{AF}$ is defined as the union of the formulas sets $\mathcal{AF}^{[k]}$, each of which restricts the allowed argument sequences appearing in atomic formulas to *adjacent* words over the alphabet $\mathbf{x}_k$. The question arises as to whether these restrictions might be further relaxed without compromising the decidability of satisfiability. Under reasonable assumptions about the fragment in question, the answer must be no. Indeed, assume, for simplicity, that the argument sequence x_1x_2 is allowed in the 2-variable case, and x_2x_3 in the 3-variable case. Now the only non-adjacent words of length 2 over $\mathbf{x}_3$ are x_1x_3 and x_3x_1. In the first case, this allows us to write the formula $\forall x_1 \forall x_2 (r(x_1x_2) \to \forall x_3 (r(x_2x_3) \to r(x_1x_3)))$, which says that r is transitive. But even two-variable logic with (at least two) transitive relations yields a logic for which satisfiability and finite satisfiability are undecidable [20], since it is simple to write formulas all of whose models embed grids of unbounded size. Similar remarks apply to the formula $\forall x_1 \forall x_2 (r(x_1x_2) \to \forall x_3 (r(x_2x_3) \to r(x_3x_1)))$, and indeed to the case of formulas featuring ternary non-adjacent atoms such as $p(x_1x_3x_2)$. It is therefore difficult to conceive of meaningful fragments of first-order logic defined purely by reference to restrictions on the allowable argument sequences that do not define sub-fragments of the adjacent fragment, and that are at the same time decidable for satisfiability. In this respect, $\mathcal{AF}$ appears to be the end of the road.

On the other hand, the last two decades have witnessed concerted attempts to investigate the decidability of the satisfiability problem for FO^2 over various classes of structures, where some distinguished predicates are interpreted in a special way, e.g. as linear orders [24, 36, 37]; other such semantic constraints have also been investigated [25, 10, 21, 7, 9, 3]. It is therefore natural to ask whether the adjacent fragment remains decidable when subject to similar semantic constraints. Of course, since $\mathcal{AF}$ extends FO^2, all the undecidability results for FO^2 immediately transfer to $\mathcal{AF}$. Thus, $\mathcal{AF}$ extended with two transitive relations [20], or with three equivalence relations [22], or with one transitive and one equivalence relation [24], or with two linear orders and their two corresponding successor relations [25], must all be undecidable. (See [23] for a survey.) Regarding positive results, existing results on the fluted fragment give cause for some hope. Thus, for example, the fluted fragment remains decidable with the addition of one transitive relation (and equality) [31]; moreover, *finite* satisfiability for FO^2 with one transitive relation is also known to be decidable [27].

A second generalization of FO^2 which preserves decidability of satisfiability is the extension with counting quantifiers [26, 28, 6]. (Here, however, the finite model property is lost.) It has been shown that the corresponding extension of the fluted fragment retains the finite model property [28]. Extending the adjacent fragment with counting quantifiers certainly results in loss of the finite model property, because $\mathcal{AF}$ includes FO^2; however, the decidability of the satisfiability and finite satisfiability problems is left for future work.

References

1 Hajnal Andréka, István Németi, and Johan van Benthem. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998. doi:10.1023/a:1004275029985.

2 Bartosz Bednarczyk. Exploiting forwardness: Satisfiability and query-entailment in forward guarded fragment. In Wolfgang Faber, Gerhard Friedrich, Martin Gebser, and Michael Morak, editors, *Logics in Artificial Intelligence – 17th European Conference, JELIA 2021, Virtual Event, May 17-20, 2021, Proceedings*, volume 12678 of *Lecture Notes in Computer Science*, pages 179–193. Springer, 2021. doi:10.1007/978-3-030-75775-5_13.

3 Bartosz Bednarczyk, Witold Charatonik, and Emanuel Kieroński. Extending two-variable logic on trees. In Valentin Goranko and Mads Dam, editors, *26th EACSL Annual Conference on Computer Science Logic, CSL 2017, August 20-24, 2017, Stockholm, Sweden*, volume 82 of *LIPIcs*, pages 11:1–11:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.CSL.2017.11.

4 Bartosz Bednarczyk and Reijo Jaakkola. Towards a model theory of ordered logics: Expressivity and interpolation. In Stefan Szeider, Robert Ganian, and Alexandra Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022, August 22-26, 2022, Vienna, Austria*, volume 241 of *LIPIcs*, pages 15:1–15:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.MFCS.2022.15.

5 Bartosz Bednarczyk, Daumantas Kojelis, and Ian Pratt-Hartmann. On the limits of decision: the adjacent fragment of first-order logic. *ArXiV*, abs/2305.03133, 2023. arXiv:2305.03133.

6 Michael Benedikt, Egor V. Kostylev, and Tony Tan. Two variable logic with ultimately periodic counting. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 112:1–112:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.112.

7 Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data words. *ACM Transactions on Computational Logic*, 12(4), July 2011. doi:10.1145/1970398.1970403.

8 Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer, 1997.

9 Witold Charatonik, Emanuel Kieroński, and Filip Mazowiecki. Decidability of weak logics with deterministic transitive closure. In Thomas A. Henzinger and Dale Miller, editors, *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14–18, 2014*, pages 29:1–29:10. ACM, 2014. doi:10.1145/2603088.2603134.

10 Witold Charatonik and Piotr Witkowski. Two-variable logic with counting and trees. *ACM Transactions on Computational Logic*, 17(4), November 2016. doi:10.1145/2983622.

11 Erich Grädel. On the restraining power of guards. *The Journal of Symbolic Logic*, 64(4):1719–1742, 1999. URL: http://www.jstor.org/stable/2586808.

12 Erich Grädel, Phokion G. Kolaitis, and Moshe Y. Vardi. On the decision problem for two-variable first-order logic. *The Bulletin of Symbolic Logic*, 3(1):53–69, 1997. URL: http://www.jstor.org/stable/421196.

13 Leon Henkin. *Logical Systems Containing Only a Finite Number of Symbols*. Séminaire de mathématiques supérieures. Presses de l'Université de Montréal, 1967. URL: https://books.google.pl/books?id=0jPQAAAAMAAJ.

14 Andreas Herzig. A new decidable fragment of first order logic. In *Abstracts of the 3rd Logical Biennial Summer School and Conference in honour of S. C. Kleene*, Varna, Bulgaria, June 1990.

15 David Hilbert and Wilhelm Ackermann. *Grundzüge der theoretischen Logik*. Springer, Berlin, 1928.

16 David Hilbert and Wilhelm Ackermann. *Principles of Mathematical Logic*. Chelsea, New York, 1950.

17 Ullrich Hustadt, Renate A Schmidt, and Lilia Georgieva. A survey of decidable first-order fragments and description logics. *Journal of Relational Methods in Computer Science*, 1(3):251–276, 2004.

18 Reijo Jaakkola. Ordered fragments of first-order logic. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021, August 23-27, 2021, Tallinn, Estonia*, volume 202 of *LIPIcs*, pages 62:1–62:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.MFCS.2021.62.

19 Emanuel Kieroński. One-dimensional guarded fragments. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, volume 138 of *LIPIcs*, pages 16:1–16:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.MFCS.2019.16.

20 Emanuel Kieroński and Jakub Michaliszyn. Two-variable universal logic with transitive closure. In Patrick Cégielski and Arnaud Durand, editors, *Computer Science Logic (CSL'12) – 26th International Workshop/21st Annual Conference of the EACSL, CSL 2012, September 3-6, 2012, Fontainebleau, France*, volume 16 of *LIPIcs*, pages 396–410. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPIcs.CSL.2012.396.

21 Emanuel Kieroński, Jakub Michaliszyn, Ian Pratt-Hartmann, and Lidia Tendera. Two-variable first-order logic with equivalence closure. *SIAM Journal on Computing*, 43(3):1012–1063, 2014. doi:10.1137/120900095.

22 Emanuel Kieroński and Martin Otto. Small substructures and decidability issues for first-order logic with two variables. *Journal of Symbolic Logic*, 77(3):729–765, 2012. doi:10.2178/jsl/1344862160.

23 Emanuel Kieroński, Ian Pratt-Hartmann, and Lidia Tendera. Two-variable logics with counting and semantic constraints. *ACM SIGLOG News*, 5(3):22–43, 2018. doi:10.1145/3242953.3242958.

24 Emanuel Kieroński and Lidia Tendera. On finite satisfiability of two-variable first-order logic with equivalence relations. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science, LICS 2009, 11-14 August 2009, Los Angeles, CA, USA*, pages 123–132. IEEE Computer Society, 2009. doi:10.1109/LICS.2009.39.

25 Amaldev Manuel. Two variables and two successors. In Petr Hlinený and Antonín Kucera, editors, *Mathematical Foundations of Computer Science 2010, 35th International Symposium, MFCS 2010, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6281 of *Lecture Notes in Computer Science*, pages 513–524. Springer, 2010. doi:10.1007/978-3-642-15155-2_45.

26 Ian Pratt-Hartmann. The two-variable fragment with counting revisited. In Anuj Dawar and Ruy J. G. B. de Queiroz, editors, *Logic, Language, Information and Computation, 17th International Workshop, WoLLIC 2010, Brasilia, Brazil, July 6-9, 2010. Proceedings*, volume 6188 of *Lecture Notes in Computer Science*, pages 42–54. Springer, 2010. doi:10.1007/978-3-642-13824-9_4.

27 Ian Pratt-Hartmann. Finite satisfiability for two-variable, first-order logic with one transitive relation is decidable. *Mathematical Logic Quarterly*, 64(3):218–248, 2018. doi:10.1002/malq.201700055.

28 Ian Pratt-Hartmann. Fluted logic with counting. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 141:1–141:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.141.

29 Ian Pratt-Hartmann. Walking on words. *ArXiV*, abs/2208.08913, 2022. doi:10.48550/arXiv.2208.08913.

30 Ian Pratt-Hartmann, Wieslaw Szwast, and Lidia Tendera. The fluted fragment revisited. *Journal of Symbolic Logic*, 84(3):1020–1048, 2019.

31 Ian Pratt-Hartmann and Lidia Tendera. The fluted fragment with transitive relations. *Annals of Pure and Applied Logic*, 173(1):103042, 2022. doi:10.1016/j.apal.2021.103042.

32 William C. Purdy. Fluted formulas and the limits of decidability. *The Journal of Symbolic Logic*, 61(2):608–620, 1996. URL: http://www.jstor.org/stable/2275678.

33 Willard Van Orman Quine. On the limits of decision. In *Proceedings of the 14th International Congress of Philosophy*, volume III, pages 57–62. University of Vienna, 1969.

34 Willard Van Orman Quine. Algebraic logic and predicate functors. In *The Ways of Paradox*, pages 283–307. Harvard University Press, Cambridge, MA, revised and enlarged edition, 1976.

35 Sylvain Schmitz. Complexity hierarchies beyond elementary. *ACM Transactions on Computational Logic*, 8(1), February 2016. doi:10.1145/2858784.

36 Thomas Schwentick and Thomas Zeume. Two-Variable Logic with Two Order Relations. *Logical Methods in Computer Science*, Volume 8, Issue 1, March 2012. doi:10.2168/LMCS-8(1:15)2012.

37 Thomas Zeume and Frederik Harwath. Order-invariance of two-variable logic is decidable. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 807–816. ACM, 2016. doi:10.1145/2933575.2933594.

The Complexity of Presburger Arithmetic with Power or Powers

Michael Benedikt
Department of Computer Science, University of Oxford, UK

Dmitry Chistikov
Centre for Discrete Mathematics and its Applications (DIMAP) &
Department of Computer Science, University of Warwick, Coventry, UK

Alessio Mansutti
IMDEA Software Institute, Madrid, Spain

Abstract

We investigate expansions of Presburger arithmetic ($\mathcal{P}\boldsymbol{a}$), i.e., the theory of the integers with addition and order, with additional structure related to exponentiation: either a function that takes a number to the power of 2, or a predicate $2^{\mathbb{N}}$ for the powers of 2. The latter theory, denoted $\mathcal{P}\boldsymbol{a}(2^{\mathbb{N}}(\cdot))$, was introduced by Büchi as a first attempt at characterizing the sets of tuples of numbers that can be expressed using finite automata; Büchi's method does not give an elementary upper bound, and the complexity of this theory has been open. The former theory, denoted as $\mathcal{P}\boldsymbol{a}(\lambda x.2^{|x|})$, was shown decidable by Semenov; while the decision procedure for this theory differs radically from the automata-based method proposed by Büchi, Semenov's method is also non-elementary. And in fact, the theory with the power function has a non-elementary lower bound. In this paper, we show that while Semenov's and Büchi's approaches yield non-elementary blow-ups for $\mathcal{P}\boldsymbol{a}(2^{\mathbb{N}}(\cdot))$, the theory is in fact decidable in triply exponential time, similarly to the best known quantifier-elimination algorithm for $\mathcal{P}\boldsymbol{a}$. We also provide a NExpTime upper bound for the existential fragment of $\mathcal{P}\boldsymbol{a}(\lambda x.2^{|x|})$, a step towards a finer-grained analysis of its complexity. Both these results are established by analyzing a single parameterized satisfiability algorithm for $\mathcal{P}\boldsymbol{a}(\lambda x.2^{|x|})$, which can be specialized to either the setting of $\mathcal{P}\boldsymbol{a}(2^{\mathbb{N}}(\cdot))$ or the existential theory of $\mathcal{P}\boldsymbol{a}(\lambda x.2^{|x|})$. Besides the new upper bounds for the existential theory of $\mathcal{P}\boldsymbol{a}(\lambda x.2^{|x|})$ and $\mathcal{P}\boldsymbol{a}(2^{\mathbb{N}}(\cdot))$, we believe our algorithm provides new intuition for the decidability of these theories, and for the features that lead to non-elementary blow-ups.

2012 ACM Subject Classification Theory of computation → Logic and verification

Keywords and phrases arithmetic theories, exponentiation, decision procedures

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.112

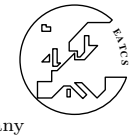
Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Extended Version*: https://arxiv.org/abs/2305.03037

Funding *Michael Benedikt*: funded in part by EPSRC grant EP/T022124/1.
Dmitry Chistikov: acknowledges the support of IMDEA Software Institute.
Alessio Mansutti: funded in part by ERC grant No. 852769 (ARiAT).

1 Introduction

This paper concerns decision problems involving first-order logic sentences over the integers. We are given a sentence in the logic, and want to know if it holds in a certain infinite structure over the integers – we refer to these as "satisfaction problems" below. If the sentence can mention "full arithmetic" – both addition and multiplication on the integers – then it is well-known that the satisfaction problem is undecidable [3]. On the other hand, if the sentence mentions only addition, inequality, and integer constants – *Presburger arithmetic* ($\mathcal{P}\boldsymbol{a}$) –



then the satisfaction problem is decidable [20]. Presburger arithmetic is by no means the maximal decidable arithmetic theory. For instance, adding a "bit predicate" to Presburger arithmetic – a binary predicate holding on (m, n) if m is the largest power of 2 dividing n – does not undermine decidability. This extension is known as *Büchi arithmetic*. A decision procedure for the satisfaction problem of this theory is based on translating each formula into a finite automaton over strings, representing the binary expansions of possible solutions to the formula [2]. Although both are decidable, there is a big difference between Presburger arithmetic and Büchi arithmetic: the satisfaction problem of the former can be decided in triply exponential time [17] and even in doubly exponential space [8], whereas the latter is known to have no elementary bound. See [1] and [25] for a finer-grained analysis of the complexity of Presburger arithmetic, in terms of alternating Turing machines.

Sitting in between Presburger arithmetic and Büchi arithmetic is the extension of $\boldsymbol{\mathcal{Pa}}$ with a predicate for the powers of 2: we refer to this set of numbers as $2^{\mathbb{N}}$ and to the theory as $\boldsymbol{\mathcal{Pa}}(2^{\mathbb{N}}(\cdot))$. This predicate is clearly definable in Büchi arithmetic, so the first-order theory of $\boldsymbol{\mathcal{Pa}}(2^{\mathbb{N}}(\cdot))$ is again decidable with automata. An alternative decision procedure, avoiding automata, was developed by Semenov in [23]. It proceeds by eliminating quantifiers, arriving at a quantifier-free formula in an enhanced signature – including, for example, a predicate for the highest power of 2 below a given integer. Semenov's procedure applies more broadly to extensions of Presburger arithmetic by a unary predicate satisfying a condition "effective sparseness": thus it isolates combinatorial properties of $2^{\mathbb{N}}$ that underlie decidability, rather than automata-theoretic constructions. Semenov's procedure has been refined and extended by Point; see, e.g., [19]. The complexity of the procedure and the complexity of $\boldsymbol{\mathcal{Pa}}(2^{\mathbb{N}}(\cdot))$ has, to our knowledge, received no attention.

Instead of adding a unary predicate, one can add to $\boldsymbol{\mathcal{Pa}}$ the function taking a number n to 2^n: the power function for short, rather than the powers predicate above. The theory, which we denote $\boldsymbol{\mathcal{Pa}}(\lambda x.2^{|x|})$, subsumes $\boldsymbol{\mathcal{Pa}}(2^{\mathbb{N}}(\cdot))$. Semenov proved decidability of this theory as well [24]. But in this case a non-elementary lower bound follows from [6], see [4]. We are not aware of any finer-grained analysis of the complexity of the theory. Note that $\boldsymbol{\mathcal{Pa}}(\lambda x.2^{|x|})$ is incomparable to Büchi arithmetic in expressiveness: in fact the union of the two is undecidable [4].

In this paper we show that the complexity of $\boldsymbol{\mathcal{Pa}}(2^{\mathbb{N}}(\cdot))$ is elementary, and is in fact contained in 3ExpTime. In this sense $\boldsymbol{\mathcal{Pa}}(2^{\mathbb{N}}(\cdot))$ is quite similar to Presburger arithmetic in complexity. We also show that the existential fragment of the theory $\boldsymbol{\mathcal{Pa}}(\lambda x.2^{|x|})$ has elementary complexity: its satisfaction problem is in NExpTime.

We show our results on extending $\boldsymbol{\mathcal{Pa}}$ with powers or the power function using a single parameterized algorithm. The algorithm can be applied to decide satisfaction of a $\boldsymbol{\mathcal{Pa}}(\lambda x.2^{|x|})$ sentence φ in time tower of $|\varphi|$, matching the prior non-elementary complexity. But it can be specialized to the context of either a $\boldsymbol{\mathcal{Pa}}(2^{\mathbb{N}}(\cdot))$ sentence or an existential $\boldsymbol{\mathcal{Pa}}(\lambda x.2^{|x|})$ sentence, giving in each case an elementary bound. The algorithm is based on eliminating quantifiers: it makes heavy use both of existing Presburger quantifier elimination algorithms [25] and the core of the method of Semenov, which involves removing "problematic occurrences" of a variable within a formula. Intuitively, an occurrence of a variable in an atomic formula is unproblematic if it occurs only outside of power functions, or if the atomic formula is just a comparison between two power terms. In the latter case, the exponentiation of the variable can be eliminated by taking logarithms. We factor this core Semenovian idea out into a self-contained subroutine. We give a short top-level procedure that interleaves calls to this subroutine with calls to a variant of Presburger quantifier elimination. The latter enables us to remove quantified variables completely. A meticulous complexity analysis that tracks several parameters of the input formula shows that the procedure achieves the desired

bounds in the special cases of $\mathcal{Pa}(2^{\mathbb{N}}(\cdot))$ and existential $\mathcal{Pa}(\lambda x.2^{|x|})$.

Our work brings the following ideas:

- For $\mathcal{Pa}(2^{\mathbb{N}}(\cdot))$, we rewrite the formula by processing quantifier blocks inside out but do not eliminate quantifier alternation. The resulting formula is put in a nontrivial fragment of $\mathcal{Pa}$: integer octagon arithmetic ($\mathcal{Oct}$), which we observe to be decidable in PSpace.
- In eliminating problematic variables from formulae, our algorithm exploits a new substitution strategy: it tailors substitutions to individual inequalities, rather than applying them in the entire formula globally or in an individual disjunct in the DNF (à la Reddy and Loveland [22]). For both $\mathcal{Pa}(2^{\mathbb{N}}(\cdot))$ and $\exists\mathcal{Pa}(\lambda x.2^{|x|})$, this will be key to obtaining strong bounds on the number of homogeneous terms produced during the transformation, and bounds on these terms will give us bounds on the running times of the algorithms.

We believe that our procedure, in addition to providing the desired bounds, gives a good intuition for the decidability of $\mathcal{Pa}(\lambda x.2^{|x|})$ and the sources of non-elementary blow-up within it. Note that in this work we deal for simplicity with powers of 2, but the same complexity results apply to any other base $k \in \mathbb{N}$, $k > 2$. Expansion with two bases is undecidable [12].

2 Preliminaries

The symbols $\mathbb{Z}$, $\mathbb{N}$ and $\mathbb{N}_+$ denote the set of integers, natural numbers (including zero), and positive integers, respectively. We write $\#A$ for the cardinality of a finite set A. Given $n, m \in \mathbb{Z}$, we define $[n, m] \coloneqq \{n, n+1, \dots, m\}$ and, if $n \in \mathbb{N}_+$, $[n] \coloneqq [0, n-1]$. For two sets D and C, $[D \to C]$ stands for the set of all functions from D to C. We write $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ to denote the *floor* and *ceiling* functions, respectively, $|\cdot|$ to denote the *absolute value* function, and $\log(\cdot)$ to denote the *binary logarithm* function. All these functions take as input a real number. Note that $n \in \mathbb{N}$ can be represented in binary using $\lceil\log(n+1)\rceil$ many bits.

We sometimes apply standard set operations and predicates, such as for instance $\in$, $\subseteq$ and $\setminus$, to vectors $\boldsymbol{v} = (v_1, \dots, v_d)$. In these cases, there is an implicit conversion of $\boldsymbol{v}$ into the set $V = \{v_1, \dots, v_d\}$. As an example, $v \in \boldsymbol{v}$ and $\boldsymbol{v} \setminus A$ stand for $v \in V$ and $V \setminus A$, respectively, where A is a set (or another vector).

Presburger arithmetic with a power function. We consider the structure $\mathcal{Pa}(\lambda x.2^{|x|}) \coloneqq \langle \mathbb{Z}, 1, +, (a \cdot x)_{a\in\mathbb{Z}}, 2^{|x|}, (q \mid x)_{q\in\mathbb{N}_+}, < \rangle$, in which the classical signature of Presburger arithmetic ($\mathcal{Pa}$) is enriched with the unary *power of the absolute value* function $x \mapsto 2^{|x|}$. As usual, 1 is the constant (interpreted as) $1 \in \mathbb{Z}$, $+$ and $<$ stand for addition and strict ordering over $\mathbb{Z}$, respectively, $x \mapsto a \cdot x$ is the unary function multiplying its input by the constant $a \in \mathbb{Z}$, and $x \mapsto q \mid x$ is the unary relation that is true for integers divisible by $q \in \mathbb{N}_+$.

The *first-order formulae* $\Phi, \Psi, \varphi, \psi, \dots$ of $\mathcal{Pa}(\lambda x.2^{|x|})$ are generated from the grammar

$$\Phi, \Psi \coloneqq \alpha \mid \top \mid \bot \mid \neg\Phi \mid \Phi \wedge \Psi \mid \exists x\, \Phi \mid \forall x\, \Phi \qquad \alpha \coloneqq t_1 < t_2 \mid (q \mid t_1),$$

where x is a first-order variable from an infinite countable set $\mathbb{V}$. The elements of α are the *atomic formulae* of the logic, i.e., they are *linear inequalities* $t_1 < t_2$ between terms t_1 and t_2, or *divisibility constraints* $q \mid t_1$, where $q \in \mathbb{N}_+$. Instead of allowing arbitrary terms of the signature, we will deal with a simpler language where *terms* are expressions of the form $\sum_{i\in I} a_i \cdot 2^{|x_i|} + \sum_{j\in J} b_j \cdot x_j + c$ where $c \in \mathbb{Z}$ is the *constant* of the term, $a_i, b_j \in \mathbb{Z} \setminus \{0\}$ are the *coefficients* of the *power terms* $(2^{|x_i|})_{i\in I}$ and of the *linear* variables $(x_j)_{j\in J}$, and I, J are finite sets of indices (which might overlap). We also restrict to terms where no variable occurs linearly twice, or occurs exponentiated twice. It is easy to see that formulae of the full language can be converted to use this restricted term language, and our algorithms will

not take us out of this fragment. When we talk about equality of terms, we mean modulo associativity and commutativity of $+$. A term is said to be *homogeneous* if its constant is 0.

The Boolean connectives $\vee$, $\rightarrow$ and $\leftrightarrow$, and the standard (in)equalities between terms $\leq$, $=$, $\geq$ and $>$ are defined from $\wedge$, $\neg$ and $<$, as usual. We use the absolute value of variables occurring linearly as a shortcut: e.g., $a \cdot |x| < t$ is equivalent to the formula $(x \geq 0 \rightarrow a \cdot x < t) \wedge (x < 0 \rightarrow -a \cdot x < t)$. We write $\Phi(x_1, \ldots, x_d)$ or $\Phi(\boldsymbol{x})$ to highlight the fact that all free variables of the formula Φ are in $\boldsymbol{x}$. A formula without free variables is said to be a *sentence*. We write $\Phi \Leftrightarrow \Psi$ whenever Φ and Ψ are equivalent. A finite set of formulae $S \coloneqq \{\Phi_i : i \in I\}$ is said to be a *cover for* (or *to cover*) a formula Ψ whenever $\Psi \Leftrightarrow \bigvee_{i \in I} \Phi_i$.

The *satisfaction problem* asks whether a given sentence is true.

Term and formula normalization. To simplify the exposition, we often bring terms and formulae to convenient (normal) form, without mentioning this explicitly every time. This normalization does not change our bounds on the asymptotic running time of the algorithms, nor their correctness.

- We assume inequalities have the form $t < 0$, where t is a term. Thus, we convert inequalities of the form $t_1 < t_2$ into $t_1 - t_2 < 0$. The construction of $t_1 - t_2$ follows the convention on terms described above. We will still sometimes refer to more general inequalities $t_1 < t_2$ for brevity, but these should be taken as abbreviations for inequalities of the above form.
- We rearrange terms following associativity and commutativity of $+$. We also evaluate arithmetic expressions, including, e.g., $2^{|a|}$ and $|a|$, where $a \in \mathbb{Z}$.
- In divisibility constraints of the form $q \mid \sum_{i \in I} a_i \cdot 2^{|x_i|} + \sum_{j \in J} b_j \cdot x_j + c$, we always assume $a_i, b_j, c \in [q]$.
- Inequalities $a < b$ and divisibility constraints $q \mid a$ on integers $a, b \in \mathbb{Z}$ and $q \in \mathbb{N}_+$ are evaluated to $\top$ or $\bot$. So are divisibility constraints $1 \mid t$ or $q \mid 0$, where $q \in \mathbb{N}_+$ and t is a term (these are $\top$).
- Inequalities of the form $a \cdot x < b$ with $a, b \in \mathbb{Z}$ and $|a| \geq 2$ are rewritten into $x \leq \lfloor \frac{b-1}{a} \rfloor$ if $a > 0$, and to $x \geq \lceil \frac{b-1}{a} \rceil$ if $a < 0$. This normalization is required in the context of the quantifier elimination (q.e.) procedure for Presburger arithmetic applied to octagons (see *integer octagon arithmetic* below).
- Trivial inequalities involving power terms, where just the fact that $2^{|c|}$ is always positive suffices to evaluate the atomic formula, are also rewritten as $\top$ or $\bot$. For instance, $a \cdot 2^{|x|} < c$ when a and c have different signs or when $c = 0$; or $a \cdot 2^{|x|} < b \cdot 2^{|y|}$ where a and b have different signs.

Beyond normalization, we need the following operations and notation for terms. We write $t(\boldsymbol{x})$ if all variables appearing in the term t are in $\boldsymbol{x}$. Let α_1 be a formula (resp., a term of the form x or $2^{|x|}$). Given a second formula (resp., term) α_2, $\Phi[\alpha_2 \,/\, \alpha_1]$ stands for the formula obtained from Φ by replacing every occurrence of α_1 by α_2. Additionally, when α_1 and α_2 are two terms t_1 and t_2, and given $n \in \mathbb{N}_+$, we write $\Phi[\frac{t_2}{n} \,/\, t_1]$ for the formula obtained from Φ by replacing each inequality $a \cdot t_1 + t' < t''$ by $a \cdot t_2 + n \cdot t' < n \cdot t''$ and each divisibility constraint $q \mid a \cdot t_1 + t'$ by $n \cdot q \mid a \cdot t_2 + n \cdot t'$. This operation can be seen as scaling by n the atomic formulae where t_1 occurs linearly, relying on the equivalences $s_1 < s_2 \Leftrightarrow n \cdot s_1 < n \cdot s_2$ and $q \mid s \Leftrightarrow n \cdot q \mid n \cdot s$, followed by the substitution of each $n \cdot t_1$ by t_2. We will restrict the use of term substitutions to the following cases: $\Phi[t \,/\, x]$ and $\Phi[\frac{t}{n} \,/\, x]$, where x is a variable only occurring linearly in Φ; and $\Phi[t \,/\, 2^{|x|}]$ and $\Phi[\frac{t}{n} \,/\, 2^{|x|}]$. Note that in the last two cases, all linear occurrences of the variable x are left untouched. We extend the notion of substitution to multiple terms or formulae: $\Phi[\beta_i \,/\, \alpha_i : i \in [1, k]] \coloneqq (\ldots(\Phi[\beta_1 \,/\, \alpha_1])[\beta_2 \,/\, \alpha_2]\ldots)[\beta_k \,/\, \alpha_k]$.

Parameters of formulae. As often done for $\mathcal{P}\boldsymbol{a}$, the complexity analysis of our procedure requires the introduction of several parameters for a formula. We define functions $\mathit{lin}(\cdot)$, $\mathit{hom}(\cdot)$, $\mathit{heft}(\cdot)$, $\mathit{mod}(\cdot)$, $\mathcal{B}(\cdot)$, and $\mathit{alt}(\cdot)$, to track various features of a formula Φ from $\mathcal{P}\boldsymbol{a}(\lambda x.2^{|x|})$:

- $\mathit{lin}(\Phi)$ is the set containing the terms 0 and 2 as well as all the terms t that appear in linear inequalities $t < 0$ of Φ (implicitly converting $t_1 < t_2$ into $t_1 - t_2 < 0$);
- $\mathit{hom}(\Phi)$ is the set of *homogeneous linear terms* obtained from the linear terms in $\mathit{lin}(\Phi)$ by eliminating their constant term c (alternatively, updating c to 0);
- $\mathit{heft}(\Phi)$ is the maximum number of variables in a term of Φ;
- $\mathit{mod}(\Phi)$ is the least common multiple of all $q \in \mathbb{N}_+$ appearing in constraints $q \mid t$ of Φ (if the formula Φ has no divisibility constraints, then we postulate $\mathit{mod}(\Phi) = 1$);
- $\mathcal{B}(\Phi)$ denotes the number of occurrences of negations $\neg$ and conjunctions $\wedge$ in Φ (note that the syntax permits binary conjunction only);
- $\mathit{alt}(\Phi)$ is the *quantifier alternation rank* (number of quantifier blocks) of a formula Φ in prenex normal form.

Throughout the paper, we assume an encoding of terms where constants and coefficients are given in binary representation. By $\mathrm{len}(\Phi)$ we denote the *length* of the formula Φ: the number of bits required to write it down. For simplicity, we assume it is always at least 2. We extend the notion of infinity norm to terms. The *infinity norm* $\|t\|$ of a linear term t is the maximum absolute value of a coefficient or constant appearing in t. For a finite set of terms T, we define $\|T\| \coloneqq \max\{\|t\| : t \in T\}$. The 1-*norm* of t, denoted $\|t\|_1$, is the sum of absolute values of all its coefficients and of its constant; this is always non-negative.

3 Summary of main results

This paper focuses on two fragments of $\mathcal{P}\boldsymbol{a}(\lambda x.2^{|x|})$:

- The first-order theory of $\mathcal{P}\boldsymbol{a}(2^{\mathbb{N}}(\cdot)) \coloneqq \langle \mathbb{Z}, 1, +, (a \cdot x)_{a\in\mathbb{Z}}, 2^{\mathbb{N}}(x), (q \mid x)_{q\in\mathbb{N}_+}, < \rangle$, that is, the structure which enriches Presburger arithmetic with the unary relation $x \mapsto 2^{\mathbb{N}}(x)$ that is true for the powers of 2, i.e., $2^{\mathbb{N}}(x) = \top$ iff $x \in \{1, 2, 4, \ldots\}$. Note that the relation $2^{\mathbb{N}}(x)$ can be expressed in $\mathcal{P}\boldsymbol{a}(\lambda x.2^{|x|})$, with the formula $\exists y.\, x = 2^{|y|}$.
- The existential fragment of $\mathcal{P}\boldsymbol{a}(\lambda x.2^{|x|})$, denoted by $\exists\mathcal{P}\boldsymbol{a}(\lambda x.2^{|x|})$. Formulae of this fragment are of the form $\exists \boldsymbol{x}.\Phi$, where Φ is quantifier-free (q.f., in short).

The main results of this paper are summarized below:

▶ **Theorem 1.** *The satisfaction problem for* $\exists\mathcal{P}\boldsymbol{a}(\lambda x.2^{|x|})$ *is in* NExpTime.

▶ **Theorem 2.** *The satisfaction problem for* $\mathcal{P}\boldsymbol{a}(2^{\mathbb{N}}(\cdot))$ *is in* 3ExpTime.

Theorems 1 and 2 are based on a common core procedure for $\mathcal{P}\boldsymbol{a}(\lambda x.2^{|x|})$ that we introduce in Section 4. The procedure manipulates the subformulae of an input formula so that they (eventually) enter the following fragments of $\mathcal{P}\boldsymbol{a}(\lambda x.2^{|x|})$:

- The *power comparisons fragment*, denoted by $\mathcal{P}\boldsymbol{owCmp}$. In this fragment, inequalities are restricted to the form $a \cdot 2^{|x|} < b \cdot 2^{|y|}$ or $a \cdot 2^{|x|} < b$, where $a, b \in \mathbb{Z}$, and divisibility constraints are of the form $q \mid 2^{|x|} - r$, where $q \in \mathbb{N}_+$ and $r \in [q]$.
- *Integer octagon arithmetic*, denoted by $\mathcal{O}\boldsymbol{ct}$ (see e.g. [13, 16]), that is, the fragment of $\mathcal{P}\boldsymbol{a}$ in which inequalities are restricted to the forms $\pm x \pm y < c$ and $\pm x < c$, where $c \in \mathbb{Z}$, and divisibility constraints are of the form $q \mid x - r$, where $q \in \mathbb{N}_+$ and $r \in [q]$.
- The fragment $\mathcal{S}\boldsymbol{em}$ (short for *Semenov*, as this fragment is related to the one used in [23]). In formulae Φ of this fragment, each variable appears either always linearly or always in a power, and every *bound* variable x appears only in atomic formulae from $\mathcal{P}\boldsymbol{owCmp}$

(hence, x is always in a power). Moreover, divisibility constraints in Φ are *simple*, i.e., they are of the form $q \mid 2^{|x|} - r$ or of the form $q \mid x - r$, where $q \in \mathbb{N}_+$ and $r \in [q]$. Notice that a sentence in this fragment must be in $\boldsymbol{\mathcal{P}owC mp}$.

- The quantifier-free fragment of $\boldsymbol{\mathcal{P}a}(\lambda x.2^{|x|})$, denoted $\boldsymbol{\mathcal{QF}}$, consisting of all q.f. formulae.

4 The core procedure

Overall organization. Our final decision procedures, which will be presented in Section 5, rely on a core procedure (Algorithm 1) that interleaves calls to what are essentially quantifier elimination subroutines à la Presburger [20] and Semenov [23], respectively, to be explained further below. The input of Algorithm 1 is a formula Φ of $\boldsymbol{\mathcal{P}a}(\lambda x.2^{|x|})$ in prenex normal form. The procedure can be run in two modes, taking an additional parameter $\boldsymbol{\mathcal{F}}$ accordingly. This parameter specifies the "target" fragment of the logic:

- For Φ obtained as a translation of a $\boldsymbol{\mathcal{P}a}(2^{\mathbb{N}}(\cdot))$ formula into $\boldsymbol{\mathcal{P}a}(\lambda x.2^{|x|})$, set $\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{S}em}$.
- For general formulae of $\boldsymbol{\mathcal{P}a}(\lambda x.2^{|x|})$, and for the handling of existential $\boldsymbol{\mathcal{P}a}(\lambda x.2^{|x|})$ in non-deterministic exponential time, set $\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{QF}}$.

The output of the algorithm is a simplified formula: more specifically, it is a formula of the form $\exists \boldsymbol{x}.\varphi$ or $\neg\exists \boldsymbol{x}.\varphi$, where φ is in $\boldsymbol{\mathcal{F}}$: "*alternation-free modulo* $\boldsymbol{\mathcal{F}}$" below. If the input is a sentence, then the output has no leading quantifiers and is thus a sentence of $\boldsymbol{\mathcal{F}}$.

The procedure processes blocks of quantifiers at a time, eliminating them one by one. Each block corresponds to one iteration of the outer **while** loop. In line 2 we split the quantifier prefix at the innermost existential block that takes us out of the fragment $\boldsymbol{\mathcal{F}}$. There may be a choice as to whether $\neg$ appears at the beginning of Π, but this introduces no ambiguity to the choice of $\boldsymbol{u}$, because $\forall v.\Psi$ is in $\boldsymbol{\mathcal{F}}$ iff $\exists v.\Psi$ is in $\boldsymbol{\mathcal{F}}$. This follows because both fragments $\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{S}em}$ and $\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{QF}}$ are closed under negation.

The organization of the procedure maintains a DNF-like structure. The set Q acts as a worklist containing the formulae; intuitively, they are the conjunctions (although not necessarily of atomic formulae). The PresQE and SemCover subroutines embed Reddy and Loveland's optimization for $\boldsymbol{\mathcal{P}a}$ [22]: whenever a pair $(\boldsymbol{x}, \varphi_1 \vee \varphi_2)$ could be produced, it is split into two pairs $(\boldsymbol{x}, \varphi_1)$ and $(\boldsymbol{x}, \varphi_2)$ evolving independently for as long as possible. Thus, the DNF-like structure is maintained within each iteration of the outer **while** loop of the Master procedure:

$$\Phi \Leftrightarrow \Pi.\Pi'.\left[\left(\bigvee_{(\boldsymbol{x},\varphi)\in Q} \exists \boldsymbol{x}.\varphi\right) \vee \bigvee_{\varphi\in D} \varphi\right].$$

For each $\varphi \in D \cup \{\varphi : (\boldsymbol{x}, \varphi) \in Q \text{ for some } \boldsymbol{x}\}$, we have $\varphi \in \boldsymbol{\mathcal{F}}$. Pairs from Q are processed in the inner **while** loop one at a time. Formulae from D are "done" and will only be picked up again after leaving the current block: the algorithm will no longer process them within the current block. Thanks to the DNF-like structure, our analysis of the parameter growth for an individual pair $(\boldsymbol{x}, \varphi)$ can ignore the complexity of the big disjunction (i.e., other pairs in Q and D).

Above we have presented Algorithm 1 deterministically: any deterministic choice can be made in line 5 when popping an element from Q, and in lines 7–9 when choosing an appropriate $x \in \boldsymbol{x}$. This implementation will be employed to obtain the claimed triply-exponential bound for $\boldsymbol{\mathcal{P}a}(2^{\mathbb{N}}(\cdot))$, but not the NExpTime bound for the existential fragment of $\boldsymbol{\mathcal{P}a}(\lambda x.2^{|x|})$. In the latter case, we will only perform the outer loop once. The prefix Π will always be empty, and thus the formula we are processing can always be considered

Algorithm 1 Master procedure.

Input: fragment $\mathcal{F} \in \{\mathcal{QF}, \mathcal{Sem}\}$;
formula $\Phi(\boldsymbol{y})$ in $\mathcal{Pa}(\lambda x.2^{|x|})$ in prenex normal form with quantifier-free part from $\mathcal{F}$ in which all divisibility constraints are simple
Output: an equivalent formula $\Phi'(\boldsymbol{y})$, alternation-free modulo $\mathcal{F}$;
if Φ is a sentence, Φ' is a sentence of $\mathcal{F}$

1: **while** true **do**
2: $\quad\Pi \leftarrow$ the shortest quantifier prefix of Φ, possibly with $\neg$ in front, such that $\Phi = \Pi.\exists \boldsymbol{u}.\Psi$ where Ψ is in $\mathcal{F}$ (if necessary, rewrite $\forall \boldsymbol{u}$ as $\neg\exists \boldsymbol{u}.\neg$)
3: $\quad Q \leftarrow \{(\boldsymbol{u}, \Psi)\}$; $D \leftarrow \varnothing$
4: $\quad\Pi' \leftarrow$ empty string of quantifiers ▷ Π' is a **global** variable
5: $\quad$**while** $(\boldsymbol{x}, \varphi) \leftarrow \mathrm{pop}(Q)$ **do**
6: $\quad\quad$**if** $\boldsymbol{x}$ is empty **then** add φ to D
7: $\quad\quad$**else if** some $x \in \boldsymbol{x}$ does not appear in φ **then** add pair $(\boldsymbol{x} \setminus \{x\}, \varphi)$ to Q
8: $\quad\quad$**else if** $\exists x.\varphi$ is in $\mathcal{F}$ for some $x \in \boldsymbol{x}$ **then** add pair $(\boldsymbol{x} \setminus \{x\}, \exists x.\varphi)$ to Q
9: $\quad\quad$**else if** some $x \in \boldsymbol{x}$ appears only linearly in φ **then** add $\mathrm{PresQE}(x, \boldsymbol{x}, \varphi)$ to Q
10: $\quad\quad$**else** add $\mathrm{Linearize}(\mathrm{SemCover}(\boldsymbol{x}, \varphi))$ to Q
11: $\quad\Phi \leftarrow \Pi.\Pi'.\ \bigvee_{\varphi \in D} \varphi$
12: $\quad$**if** Π contains no quantifiers **then return** Φ

Algorithm 2 Function Linearize.

Input: a set S of pairs $(\boldsymbol{x}, \theta)$, with $\boldsymbol{x}$ a vector of variables and θ a formula
Output: if $\mathcal{F} = \mathcal{QF}$: for each $(\boldsymbol{x}, \theta)$, a pair $(\boldsymbol{x}, \theta')$ where $\theta \Leftrightarrow \theta'$ and, for every $x \in \boldsymbol{x}$, if $2^{|x|}$ only occurs in constraints from $\mathcal{PowCmp}$ in θ, then x only occurs linearly in θ'

1: **if** $\mathcal{F} = \mathcal{Sem}$ **then** return S ▷ do nothing unless $\mathcal{F} = \mathcal{QF}$
2: **for** $(\boldsymbol{x}, \theta) \in S$ **do**
3: $\quad\boldsymbol{x}' \leftarrow$ vector of all $x \in \boldsymbol{x}$ s.t. $2^{|x|}$ only occurs in constraints from $\mathcal{PowCmp}$ in θ
4: $\quad$**for** $x \in \boldsymbol{x}'$ **do**
5: $\quad\quad$update θ by applying all of the following replacements:
6: $\quad\quad a \cdot 2^{|x|} < b \ \mapsto \begin{cases} |x| < \lceil \log_2(b/a) \rceil & \text{if } a > 0 \text{ and } b > 0 \\ |x| > \lfloor \log_2(b/a) \rfloor & \text{if } a < 0 \text{ and } b < 0 \end{cases}$
7: $\quad\quad a \cdot 2^{|x|} < b \cdot 2^{|y|} \ \mapsto \begin{cases} |x| < |y| + \lceil \log_2(b/a) \rceil & \text{if } a > 0 \text{ and } b > 0 \\ |x| > |y| + \lfloor \log_2(b/a) \rfloor & \text{if } a < 0 \text{ and } b < 0 \end{cases}$
8: $\quad\quad q \mid 2^{|x|} - r \ \mapsto \begin{cases} q' \mid |x| - r' & \text{if } r' = \min\{s \geq 0 : q \mid 2^s - r\}, \\ & \quad q' = \min\{t > 0 : q \mid r \cdot (2^t - 1)\} \\ |x| = r' & \text{if } r' = \min\{s \geq 0 : q \mid 2^s - r\}, \\ & \quad \{t > 0 : q \mid r \cdot (2^s - 1)\} = \varnothing \\ \bot & \text{otherwise, i.e., } \{s \geq 0 : q \mid 2^s - r\} = \varnothing \end{cases}$
▷ in the replacements in line 8, search for $s, t \leq q - 1$ only
9: $\quad$**return** S

Algorithm 3 Function PresQE.

Input: variable x; vector of variables $\boldsymbol{x}$, where $x \in \boldsymbol{x}$;
formula $\varphi(x, \boldsymbol{y})$ of $\mathscr{F}$ where $\boldsymbol{x} \setminus \{x\} \subseteq \boldsymbol{y}$ and x appears only linearly in atomic formulae
Output: a set of pairs $(\boldsymbol{x}, \psi(\boldsymbol{y}))$ where $\psi \in \mathscr{F}$ and the set of all ψ is a cover for $\exists x.\varphi$

$T \leftarrow \{(a, -t(\boldsymbol{y})) : a > 0,\ a \cdot x + t \in \mathit{hom}(\varphi)\} \cup \{(-a, t(\boldsymbol{y})) : a < 0,\ a \cdot x + t \in \mathit{hom}(\varphi)\}$
$g \leftarrow \Pi\{a : (a, t) \in T \text{ for some } t\}$ ▷ product of all elements of non-empty set
$\Gamma \leftarrow \{\varphi[\frac{t+k}{a} / x] \wedge (a \mid t+k) \ :\ (a, t) \in T, k \in [-r, r] \text{ where } r \coloneqq a \cdot (2 \cdot \|\mathit{lin}(\varphi)\| + g \cdot \mathit{mod}(\varphi))\}$
return $\{(\boldsymbol{x}, \psi) : \psi \in \mathrm{SimplifyDiv}(\gamma), \gamma \in \Gamma\}$

Algorithm 4 Function SemCover.

Input: vector $\boldsymbol{x}$ of variables; formula $\varphi(\boldsymbol{x}, \boldsymbol{z})$ of $\mathscr{F}$, containing $2^{|x|}$ for each $x \in \boldsymbol{x}$
Output: a set of pairs $(\boldsymbol{x}, \psi(\boldsymbol{x}, \boldsymbol{z}))$, where $\psi \in \mathscr{F}$ and the set of all $\Pi'.\exists \boldsymbol{x}.\psi$ covers $\Pi'.\exists \boldsymbol{x}.\varphi$; in every ψ some $2^{|x|}$ $(x \in \boldsymbol{x})$ only occurs in constraints from $\boldsymbol{\mathcal{P}ow\mathcal{C}mp}$
Side effect: update global variable Π' (string of quantifiers)

for $x \in \boldsymbol{x}$ **do**
  $I \leftarrow$ set of inequalities in φ outside $\boldsymbol{\mathcal{P}ow\mathcal{C}mp}$ in which x appears as a power
  $H \leftarrow \{(\eta, \sigma) : \eta(\boldsymbol{x}) + \sigma(\boldsymbol{z}) + c < 0 \text{ in } I;\ \eta \text{ and } \sigma \text{ homogeneous}\}$
  $\Gamma_x \leftarrow \{\varphi\}$
  for $(\eta, \sigma) \in H$ **do**
    $A \leftarrow$ subset of I with these η and σ (only c varies)
    $2^g \leftarrow 2^7 \cdot \big(\lambda(\|\eta\|_1 + \max\{|c| : (\eta + \sigma + c < 0) \in A\})\big)^2$
      ▷ factors up to 2^g are considered "small"
    $a \leftarrow$ coefficient at $2^{|x|}$ in η
    $V \leftarrow$ variables in η except x
    $\beta \leftarrow 2^{|x|} > 2^g \wedge (\bigwedge_{u \in V} 2^{|x|} > 2^g \cdot 2^{|u|})$
    $\Gamma_x \leftarrow \big\{2^{|x|} = 2^j \wedge \gamma[\alpha[2^j \,/\, 2^{|x|}] \,/\, \alpha : \alpha \in A]\,,$
      $2^{|x|} > 2^g \wedge 2^{|x|} = 2^j \cdot 2^{|v|} \wedge \gamma[\alpha[2^j \cdot 2^{|v|} \,/\, 2^{|x|}] \,/\, \alpha : \alpha \in A]\,,$
      $\beta \wedge \lambda(a) \cdot 2^{|x|} < \lambda(\sigma) \wedge \sigma < 0 \wedge \gamma[\top \,/\, \alpha : \alpha \in A]\,,$
      $\beta \wedge \lambda(a) \cdot 2^{|x|} < \lambda(\sigma) \wedge \sigma \geq 0 \wedge \gamma[\bot \,/\, \alpha : \alpha \in A]\,,$
      $\beta \wedge \lambda(a) \cdot 2^{|x|} = \lambda(\sigma) \wedge \gamma[\alpha[\frac{\lambda(\sigma)}{\lambda(a)} \,/\, 2^{|x|}] \,/\, \alpha : \alpha \in A]\,,$
      $\beta \wedge \lambda(a) \cdot 2^{|x|} = 2 \cdot \lambda(\sigma) \wedge \gamma[\alpha[\frac{2 \cdot \lambda(\sigma)}{\lambda(a)} \,/\, 2^{|x|}] \,/\, \alpha : \alpha \in A]\,,$
      $\beta \wedge \lambda(a) \cdot 2^{|x|} > 2 \cdot \lambda(\sigma) \wedge a < 0 \wedge \gamma[\top \,/\, \alpha : \alpha \in A]\,,$
      $\beta \wedge \lambda(a) \cdot 2^{|x|} > 2 \cdot \lambda(\sigma) \wedge a > 0 \wedge \gamma[\bot \,/\, \alpha : \alpha \in A]$
      $:\ \ \gamma \in \Gamma_x,\ 0 \leq j \leq g,\ v \in V\big\}$
$\Gamma \leftarrow \bigcup_{x \in \boldsymbol{x}} \{(\bigwedge_{y \in \boldsymbol{x}} 2^{|x|} \geq 2^{|y|}) \wedge \gamma : \gamma \in \Gamma_x\}$ ▷ we next remove all occurrences of λ
$\Sigma \leftarrow \{\sigma : \lambda(\sigma) \text{ is a subterm of some } \gamma \in \Gamma\} \setminus \{0\}$
for $\sigma \in \Sigma$ **do**
  if $\forall w_\sigma$ is not in Π' **then**
    $w_\sigma \leftarrow$ fresh variable; add $\forall w_\sigma$ to Π' ▷ update **global** Π'
$\Theta \leftarrow \{(\sigma \neq 0 \wedge \neg(2^{|w_\sigma|} \leq |\sigma| < 2 \cdot 2^{|w_\sigma|})) : \sigma \in \Sigma\}$
for each $\Sigma' \subseteq \Sigma$ and each $\gamma \in \Gamma$ **do**
  add to Θ the following formula:
  $(\bigwedge_{\sigma \in \Sigma'} 2^{|w_\sigma|} \leq |\sigma| < 2 \cdot 2^{|w_\sigma|}) \wedge (\bigwedge_{\sigma \in \Sigma \setminus \Sigma'} \sigma = 0) \wedge \gamma[2^{|w_\sigma|} \,/\, \lambda(\sigma) : \sigma \in \Sigma'][0 \,/\, \lambda(\sigma) : \sigma \in \Sigma \setminus \Sigma']$
return $\{(\boldsymbol{x}, \theta) : \theta \in \Theta\}$

Algorithm 5 Function SimplifyDiv.

Input: formula φ *almost* in $\mathscr{F}$: may contain non-simple divisibility constraints
Output: a cover for φ of formulae from $\mathscr{F}$, in which all divisibility constraints are simple

1: $G \leftarrow$ set of non-simple divisibilities in φ
2: $d \leftarrow$ least common multiple of all divisors in G
3: $\boldsymbol{t} \leftarrow$ all variables x and powers $2^{|y|}$ appearing in G
4: $\Gamma \leftarrow \varnothing$
5: **for** $r \in [\boldsymbol{t} \to [d]]$ **do**
6: add $\big((\bigwedge_{t \in \boldsymbol{t}} d \mid t - r(t)) \wedge \varphi[r(\alpha) \,/\, \alpha : \alpha \in G]\big)$ to Γ
7: **where** $r(q \mid \sum_{i=1}^{n} a_i \cdot t_i + c) \coloneqq q \mid \sum_{i=1}^{n} a_i \cdot r(t_i) + c$ ▷ simplifies to $\top$ or $\bot$
8: **return** Γ

an existentially quantified DNF, or equivalently a disjunction of existentials. It suffices to guess one disjunct, corresponding to one element of Q that is satisfiable. Thus, we will replace a deterministic inner loop that maintains a set of pairs in Q with a non-deterministic algorithm that maintains a single pair from Q. In the deterministic interpretation, calls to the subroutines in lines 9 and 10 replace a single element of Q with a set of pairs. In the non-deterministic interpretation, we guess one pair in the output of the subroutine as the new element of Q.

Subroutines. We turn from the Master procedure to its subroutines. The core of the subroutine PresQE (Algorithm 3) corresponds to Weispfenning's quantifier elimination for $\mathcal{P}\boldsymbol{a}$ [25], while Linearize given in Algorithm 2 is a simple procedure taking $\mathcal{P}\boldsymbol{owC}\boldsymbol{mp}$ atomic formulae like $2^{|x|} < 2^{|y|}$ and transforming them to Presburger formulae $x < y$ by "taking logs". We remark that all three types of divisibility replacements in line 8 of Linearize are possible: e.g., $(7 \mid 2^{|x|} - 4) \mapsto (3 \mid |x| - 2)$, and $(20 \mid 2^{|x|} - 2) \mapsto (|x| = 1)$, and $(6 \mid 2^{|x|} - 3) \mapsto \bot$.

The SemCover subroutine (Algorithm 4) is a variation of procedures dating back to Semenov's [23]. This will be less familiar to most readers, and so we discuss it in detail here.

The purpose of subroutine SemCover is to ensure that, in each of the pairs $(\boldsymbol{x}, \psi)$ in its output, for some variable $x \in \boldsymbol{x}$ every occurrence is either linear or in an atomic formula from $\mathcal{P}\boldsymbol{owC}\boldsymbol{mp}$. Across all outputs, the identity of the variable x may differ. Thus, the subroutine is essentially "$\mathcal{P}\boldsymbol{owC}\boldsymbol{mp}$-ifying" the formula. The significance of converting atomic formulae to $\mathcal{P}\boldsymbol{owC}\boldsymbol{mp}$ is that powers can then be eliminated by just "taking logarithms", i.e., by invoking Linearize. And once a quantified variable is so heavily processed that it occurs only linearly (outside powers), then by applying standard Presburger arithmetic quantifier elimination, we can eliminate the variable completely using PresQE.

To be more precise about how SemCover assists the Master procedure, consider what happens when $(\boldsymbol{x}, \psi)$ from the output of SemCover gets popped from Q in the Master procedure. Our actions depend on the chosen fragment (unless x is eliminated from ψ entirely, in which case line 7 takes care of it).

If $\mathscr{F} = \mathcal{S}em$: for some $x \in \boldsymbol{x}$, we can move $\exists x$ into ψ while still staying in the fragment, since x occurs only in power comparisons (line 8).

If $\mathscr{F} = \mathcal{QF}$: all occurrences of x became linear after the execution of Linearize on the output of SemCover, so the variable x can be eliminated by PresQE (line 9).

A look inside subroutine SemCover. Intuitively, the overall workflow of the Master procedure is repeated processing of atomic formulae lying within the scope of a particular block of quantifiers. The constraints we process will be those containing "problematic quantified variables": those that appear in atomic formulae involving powers, but are outside the fragment $\mathcal{PowCmp}$. We exhibit the idea using the following subformula:

$$\exists x.\exists y. \quad 3 \cdot 2^{|x|} - 5 \cdot 2^{|y|} - z < 0. \tag{1}$$

Here both x and y are problematic within the sole atomic subformula of the quantified formula. A major component of all prior procedures is to replace such a formula with a quantified DNF corresponding to a case analysis on the relative values of the problematic variables. These cases correspond to lines 11 to 19 of Algorithm 4 and are a cover for the formula under analysis, hence the name "Semenov cover" given to the algorithm. Each case is defined by a $\mathcal{PowCmp}$ "guard" and, under the assumption specified in a given case, we will be able to eliminate one problematic variable within a constraint, without introducing new problematic existentially-quantified variables. Thus, by applying the procedure repeatedly, we can expunge all problematic quantified variables.

The case analysis includes a guess as to which existentially quantified variable is the largest. In the example, one such guess is that $2^{|x|}$ is the largest. In all the subcases for this guess, we will make x unproblematic. The Semenov cover breaks up this guess into several subcases. One subcase is where $2^{|x|}$ is not much bigger than one of the other power terms, say $2^{|x|} = 4 \cdot 2^{|y|}$. In such cases we can substitute $2^{|x|}$ by a constant multiple of the other term, where the constant is itself a power of 2. Returning to the subcase mentioned just above, where $2^{|x|} = 4 \cdot 2^{|y|}$, we can replace $2^{|x|}$ by $4 \cdot 2^{|y|}$. The remaining case is where $2^{|x|}$ is significantly bigger than all other power terms like $2^{|y|}$; the threshold for "significantly" is set by line 7 of Algorithm 4. In this case we further analyze the most significant digit of the binary expansion for each term.

▶ **Definition.** For any integer N, let $\lambda(N)$ denote the *highest power of* 2 *below* $|N|$ [1]; we have $\lambda(0) = 0$ and $\lambda(N) \leq |N| < 2\lambda(N)$.

Algorithm 4 will make use of intermediate terms that contain λ's – for example, $\lambda(\sigma)$ for σ an ordinary $\mathcal{Pa}(\lambda x.2^{|x|})$ term. The semantics of such terms is the obvious one, which could be formalized by translation into $\mathcal{Pa}(\lambda x.2^{|x|})$, where the function λ is definable.

Returning to the example, our "significantly bigger" hypothesis implies that

$$\lambda(3 \cdot 2^{|x|} - 5 \cdot 2^{|y|}) = \lambda(3 \cdot 2^{|x|}) = 2^{x+1}.$$

This equality in turn implies that, when $\lambda(3 \cdot 2^{|x|})$ is strictly below $\lambda(z)$, the corresponding inequality in Equation (1) is true: $\lambda(3 \cdot 2^{|x|} - 5 \cdot 2^{|y|})$ is strictly below $\lambda(z)$, and while each term can differ from the corresponding λ, the difference cannot be large enough to make the inequality go the other way. By a similar argument, in the subcase where $\lambda(3 \cdot 2^{|x|})$ is at least four times greater than $\lambda(z)$, the inequality must be false. Here we reason that if $\lambda(3 \cdot 2^{|x|} - 5 \cdot 2^{|y|})$ is at least four times greater than $\lambda(z)$, then the offset of each term from its λ value cannot change the inequality from true to false.

This leaves some subcases where $\lambda(3 \cdot 2^{|x|})$ is close to $\lambda(z)$, and in these cases we can substitute away $2^{|x|}$ as well. For example, in the subcase where $\lambda(3 \cdot 2^{|x|}) = \lambda(z)$, we note that $\lambda(3 \cdot 2^{|x|}) = 2 \cdot 2^{|x|}$, and thus we could replace $2^{|x|}$ with $\lambda(z)/2$. By multiplying through the inequality by 2, we can eliminate the division by 2.

[1] We will be mostly concerned with this function on positive integers, but using absolute values gives us the convenience of avoiding partial functions.

Using the output of SemCover. The procedure above removed x, but there are several caveats. Firstly, each case was associated with a condition, where the problematic variable x still appears! However, these conditions are in $\boldsymbol{\mathcal{P}owC}\boldsymbol{mp}$, and therefore all occurrences of x are now unproblematic. Secondly, in some of our substitutions to eliminate x, we introduced λ terms, which appear both in the condition describing the case and in the formula obtained by substituting (assuming the condition). One solution to this problem, applied in earlier procedures such as Point's [19], is to extend the signature with several functions such as λ, and declare that such conditions are acceptable. In this way one can obtain quantifier elimination in the extended signature. In our SemCover subroutine, we proceed slightly differently, eliminating λ terms in favor of new variables that are bound by *definitional quantifiers*. That is, the new variables are associated with additional conditions which define them from the free variables. For example, $\lambda(z)$ can be replaced by 2^w, with additional conditions $2^w \leq |z| < 2 \cdot 2^w$. There is a unique such w for a given z, so the quantification over w can be thought of simultaneously as an existential conjoined with this condition and as a universal relativized to this condition. Such quantifications take us out of $\boldsymbol{\mathcal{P}owC}\boldsymbol{mp}$. But when considered as leading universal quantifiers, they will not increase the quantifier alternation of the global formula – they will add on variables to the next quantified block considered in the Master procedure. And since Algorithm 1 will process from inner quantifier blocks outward, the fattening of outer quantifier blocks does not jeopardize termination of our procedure. Note that at the end of processing quantifier blocks outward with Algorithm 1, we will have only an outermost block of definitional quantifiers; if there are no free variables in the top-level input formula Φ, the quantified variables will depend only on constants, and thus can be replaced by numbers, leading to a quantifier-free sentence.

Analysis of the core procedure. We analyze the procedure, showing in particular that each of Algorithms 1–5 correctly implements its specification. In the sequel we will also need the following facts.

▶ **Lemma 3.** *All divisibility constraints in $D \cup \{\varphi : (\boldsymbol{x}, \varphi) \in Q$ for some $\boldsymbol{x}\}$ are simple.*

▶ **Lemma 4.** *The Master procedure always terminates and, on a formula $\Phi(\boldsymbol{y})$, returns an equivalent formula $\Phi'(\boldsymbol{y})$ such that:*

- *$\Phi'(\boldsymbol{y})$ is equal to either $\exists \boldsymbol{w}.\varphi'(\boldsymbol{w}, \boldsymbol{y})$ or $\neg\exists \boldsymbol{w}.\varphi'(\boldsymbol{w}, \boldsymbol{y})$, where $\varphi' \in \boldsymbol{\mathcal{F}}$,*
- *if Φ is a sentence, then $\Phi' \in \boldsymbol{\mathcal{F}}$ (in other words, if $\boldsymbol{y}$ is empty, then $\boldsymbol{w}$ is empty).*

  In fact, Φ' starts with $\neg$ iff the outermost quantifier block of Φ is existential.

▶ **Lemma 5.** *Consider a prenex formula $\Pi.\Phi$, with Φ from $\boldsymbol{\mathcal{F}}$, in which all variables from the quantifier prefix Π appear only linearly. When running the Master procedure on Φ,* SemCover *is never invoked. Moreover, if no variable in Φ occurs in a power term (i.e., it is a formula from $\boldsymbol{\mathcal{P}a}$), then the quantifier-free formula returned by the procedure is in $\boldsymbol{\mathcal{P}a}$.*

5 Decision procedures and their complexity

In this section, we provide our top-level decision procedures, which make use of the algorithms presented in Section 4. We then provide a complexity analysis that establishes Theorems 1 and 2. To simplify the exposition, the growth of the formulae returned by the procedure is described with the help of "parameter tables" having the following shape:

	$\mathrm{p}_1(\cdot)$	$\mathrm{p}_2(\cdot)$	$\cdots$	$\mathrm{p}_n(\cdot)$
φ	a_1	a_2	$\cdots$	a_n
ψ_1	$f_{1,1}(a_1,\ldots,a_n)$	$f_{1,2}(a_1,\ldots,a_n)$	$\cdots$	$f_{1,n}(a_1,\ldots,a_n)$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
ψ_m	$f_{m,1}(a_1,\ldots,a_n)$	$f_{m,2}(a_1,\ldots,a_n)$	$\cdots$	$f_{m,n}(a_1,\ldots,a_n)$

In this table, φ,ψ_1,…,ψ_m are formulae, $\mathrm{p}_1(\cdot)$,…,$\mathrm{p}_n(\cdot)$ are parameter functions from formulae to $\mathbb{N}$, $a_1,\ldots,a_n \in \mathbb{N}_+$, and all $f_{j,k}$ are functions from $\mathbb{N}^n$ to $\mathbb{N}$. The table states that

if $\mathrm{p}_i(\varphi) \leq a_i$ for all $i \in [1,n]$, then $\mathrm{p}_k(\psi_j) \leq f_{j,k}(a_1,\ldots,a_n)$ for all $j \in [1,m]$ and $k \in [1,n]$.

We sometimes assume lower bounds on the values $a_1,\ldots,a_n$ (e.g., $h \geq 2$ and $a \geq 2$ in the table of Lemma 6) in order to simplify the definition of the functions $f_{j,k}$. Note that this does not change the semantics of the table. We sometimes write the ditto mark 〃 inside a cell of the table. In that case, the ditto mark represents the value of the cell directly above it (e.g., the rightmost 〃 appearing in the table of Lemma 6 is short for $b+2\cdot v+1$).

Theorem 1: NExpTime upper bound for existential $\mathscr{P}\boldsymbol{a}(\lambda x.2^{|x|})$

Before arguing for a NExpTime decision procedure for $\exists\mathscr{P}\boldsymbol{a}(\lambda x.2^{|x|})$, we analyze the growth of formulae resulting from calls to PresQE, SemCover and Linearize.

Our analysis of PresQE simply merges the analysis of Weispfenning's quantifier elimination for Presburger arithmetic from [25] (implemented in lines 1 to 3 of PresQE) with an analysis of SimplifyDiv. Here are the resulting bounds:

▶ **Lemma 6.** *On input $(x,\boldsymbol{x},\varphi(\boldsymbol{x},\boldsymbol{z}))$ where x only occurs linearly in φ,* PresQE *returns a set $\{(\boldsymbol{x},\psi_1),\ldots,(\boldsymbol{x},\psi_k)\}$ whose formulae satisfy the parameter table below ($i \in [1,k]$):*

	$\#hom$	$heft$	$\|hom(\cdot)\|$	$\|lin(\cdot)\|$	mod	$\mathcal{B}$
φ	$h \geq 2$	v	$a \geq 2$	c	m	b
ψ_i	h	$2\cdot v$	$2\cdot a^2$	$a^{h+2}(c+m)$	$a\cdot m$	$b+2\cdot v+1$
$\bigvee_{j=1}^{k}\psi_j$	h^2	〃	〃	〃	$a^h m$	$k(\,〃+1)$

and $k \leq h\cdot c\cdot m^{2v+1}\cdot a^{2v+h+4}$. The running time is in $(len(\varphi)\cdot c\cdot m)^{\mathrm{poly}(h,v)}$.

A simple analysis of SemCover yields the following bounds.

▶ **Lemma 7.** *Let $\Theta = \{(\boldsymbol{x},\theta_1),\ldots,(\boldsymbol{x},\theta_k)\}$ be the output of* SemCover$(\boldsymbol{x},\varphi(\boldsymbol{x},\boldsymbol{z}))$*, where $\boldsymbol{x}=(x_1,\ldots,x_n)$ with $n\geq 1$. Then, the following parameter table holds, where $i\in[1,k]$:*

	$\#hom$	$heft$	$\|lin(\cdot)\|$	mod	$\mathcal{B}$
φ	h	$v \geq 2$	$c \geq 2$	m	b
θ_i	$h\cdot(v+10)+n$	v	$2^{11}\cdot v^2\cdot c^4$	m	$b+h\cdot(v+11)+n$
$\bigvee_{j=1}^{k}\theta_j$	$h\cdot 2^8\cdot v^3\log(c)+n^2$	〃	〃	〃	$k\cdot(\,〃+1)$

where $k \leq (v+1)^{8h}\cdot\log(c)^h\cdot n$. Moreover, (i) at most h universal quantifiers are added to the global variable Π'; (ii) the running time is in $(len(\varphi)\cdot n)^{\mathrm{poly}(h)}$; and (iii) for every $i\in[1,k]$ there are at most h terms $t\in hom(\theta_i)$ that contain some variable from $\boldsymbol{x}$ and satisfy $(t+c'<0)\in lin(\theta_i)$ with $t+c'<0$ not in $\mathscr{P}\boldsymbol{owCmp}$, for some $c'\in\mathbb{Z}$.

Note that an estimate of $\|hom(\theta_i)\|$ is missing from Lemma 7. For SemCover, this parameter grows similarly to $\|lin(\theta_i)\|$, which by definition always bounds $\|hom(\theta_i)\|$. We also note that Lemma 7 gives an upper bound on the number of global variables added to Π'. This bound is later required to analyze $\mathscr{Pa}(2^{\mathbb{N}}(\cdot))$, but is not needed in the context of deciding sentences from $\exists\mathscr{Pa}(\lambda x.2^{|x|})$. Indeed, from Lemma 4, in this latter case Π' is empty.

In computing the upper bounds on $\|hom(\theta_i)\|$ and $\|hom(\bigvee_{j=1}^{k}\theta_j)\|$ keep in mind that in lines 11 to 18 of SemCover we perform "tailored substitutions": we only replace $2^{|x|}$ in linear terms $\alpha \in A$ with either a constant, a unique (given α) expression $\lambda(\sigma)$, or a multiple of a power $2^{|y|}$, where y appears in α. Our analysis tracks the impact of iterating these types of replacements on the number of homogeneous terms.

We continue by analyzing Linearize. Here the bounds are quite simple, but one observation is in order: line 8 might require iterating through the $q-1$ residue classes of q in order to find suitable q' and r'. Since q is encoded in binary, this yields an exponential running time for Linearize (see m below), as shown in the following lemma.

▶ **Lemma 8.** *Consider a set $S = \{(\boldsymbol{x},\theta_1),\dots,(\boldsymbol{x},\theta_k)\}$ where $\boldsymbol{x} = (x_1,\dots,x_n)$, and let r be the maximum number of variables appearing in some θ_i. On input S,* Linearize *returns a set $\{(\boldsymbol{x},\theta'_1),\dots,(\boldsymbol{x},\theta'_k)\}$ with bounds as in the following table, for all $j \in [1,k]$:*

	$\#hom$	$heft$	$\|hom(\cdot)\|$	$\|lin(\cdot)\|$	mod	$\mathcal{B}$
θ_j	h	v	a	$c \geq 2$	$m \geq 2$	b
θ'_j	$h + (6\cdot r+2)\cdot n$	v	a	c	m^2	$22\cdot b$

The running time is in $\mathrm{poly}(\max_{i=1}^{k} len(\theta_i), m, n, \#S)$.

We now complete the description of the non-deterministic algorithm deciding $\exists\mathscr{Pa}(\lambda x.2^{|x|})$ in NExpTime, completing the informal comments in Section 4. As a preliminary step, the algorithm runs SimplifyDiv on the matrix of the input existential sentence Φ, guessing a residue class for each variable and power, and obtaining an existential sentence where all the divisibilities are simple. The algorithm puts that sentence in prenex form. Afterwards, the algorithm follows Algorithm 1 (with $\mathscr{F} = \mathcal{Q}\mathscr{F}$), but replaces $\mathrm{pop}(Q)$ in line 5, as well as other forms of iteration inside PresQE and SemCover, with non-deterministic guesses. More precisely, when SemCover is called, it guesses a variable $x \in \boldsymbol{x}$ in line 1, iterates (deterministically) over every $(\eta,\sigma) \in H$ in line 5, and guesses only one of the cases in lines 11 to 18. As a result, in the non-deterministic version of SemCover, the variable Γ in line 20 contains a single formula γ. Since Φ is an existential sentence, the various terms σ considered by SemCover are always 0. Hence, $\lambda(\sigma)$ is 0 and lines 21 to 28 have no effect on the subroutine, which simply returns a singleton set containing the pair $(\boldsymbol{x},\gamma)$. For PresQE, non-deterministic guesses replace the iterations done in line 3, as well as the ones performed in line 5 of SimplifyDiv (as done in the aforementioned preliminary step of the algorithm).

The non-deterministic versions of PresQE and SemCover described above always return singleton sets containing a pair of the form $(\boldsymbol{x},\theta)$. Then, by the correctness of PresQE and SemCover, we conclude that on an input sentence Φ containing n quantified variables, the non-deterministic version of Algorithm 1 never calls each of the subroutines PresQE, SemCover and Linearize more than n times. By looking at the bounds on ψ_i, θ_i and θ'_j from Lemmas 6–8 we conclude that these $3n$ subroutine calls (non-deterministically) return a formula that never requires more than exponential space to be represented. Since Φ is a sentence and $\mathscr{F} = \mathcal{Q}\mathscr{F}$, the non-deterministic algorithm will eventually obtain a formula Ψ with no variables, only constants, which can be evaluated in exponential time. As usual, the algorithm returns true if one such (non-deterministically derived) formula Ψ is valid.

Theorem 2: 3ExpTime upper bound for $\mathcal{Pa}(2^{\mathbb{N}}(\cdot))$

We now move to $\mathcal{Pa}(2^{\mathbb{N}}(\cdot))$. Let Φ be obtained by translating a sentence of $\mathcal{Pa}(2^{\mathbb{N}}(\cdot))$ into a prenex sentence of $\mathcal{Pa}(\lambda x.2^{|x|})$ without divisibility constraints (i.e., replace each $2^{\mathbb{N}}(x)$ with $\exists y.\, x = 2^{|y|}$, each $q \mid t$ with $\exists z.\, t = q \cdot z$, and bring the resulting sentence in prenex form). Note that this translation is in polynomial time, and that each variable in Φ appears either always linearly or always in a power. The algorithm to decide Φ is described below:

1: $\Psi_1 \leftarrow$ run Algorithm 1 with $\mathcal{F} = \mathcal{Sem}$, on Φ ▷ as Φ is a sentence, $\Psi_1 \in \mathcal{PowCmp}$.
2: $\Pi.\Psi_2(\boldsymbol{x}) \leftarrow$ prenex form of Ψ_1 ▷ Ψ_2 q.f.; $\boldsymbol{x}$ are the variables appearing in Π
3: $\{(\boldsymbol{x}, \Psi_3)\} \leftarrow$ Linearize($\{(\boldsymbol{x}, \Psi_2)\}$) ▷ $\Psi_3(\boldsymbol{x})$ belongs to $\mathcal{Oct}$
4: $\Omega \leftarrow$ run Algorithm 1 with $\mathcal{F} = \mathcal{QF}$, on $\Pi.\Psi_3$ ▷ Ω does not contain variables
5: evaluate truth of Ω

Above we highlight the fact that, after the first invocation to Algorithm 1, we obtain a formula from $\mathcal{PowCmp}$ which is then manipulated by Linearize into a formula from integer octagon arithmetic ($\mathcal{Oct}$). Then, in order to estimate the running time of this algorithm, we need to study the running time of Algorithm 1 on inputs that either come from $\mathcal{Pa}(2^{\mathbb{N}}(\cdot))$ or are from $\mathcal{Oct}$. Let us discuss the latter case first.

Since $\mathcal{Oct}$ is a fragment of Presburger arithmetic, line 4 above fundamentally runs Weispfenning's quantifier elimination procedure for Presburger arithmetic (see Lemma 5), plus calls to SimplifyDiv. It turns out that on formulae from $\mathcal{Oct}$, this procedure only runs in exponential time, as summarized in the following proposition.

▶ **Proposition 9.** *Let $\mathcal{F} = \mathcal{QF}$. Consider a formula $\Phi(\boldsymbol{z})$ from integer octagon arithmetic ($\mathcal{Oct}$) in prenex form and having $alt(\varphi) = \ell \geq 1$ quantifier blocks, each with $n \geq 1$ many variables. On input Φ, Algorithm 1 returns a formula Ψ with bounds:*

	$\#hom$	$heft$	$\|hom(\cdot)\|$	$\|lin(\cdot)\|$	mod	$\mathcal{B}$
Φ	$h \geq 2$	2	1	$c \geq 2$	$m \geq 2$	b
Ψ	$4 \cdot \#\boldsymbol{z}^2$	2	1	$4^{\ell \cdot n}(c + 2 \cdot m)$	m	$k(b + \ell \cdot n + 1)$

and $k \leq 2^{2^5 \ell^2 n^2}(\#\boldsymbol{z}^2 \cdot c \cdot m)^{\ell \cdot n}$. The running time of the procedure is in $(len(\Phi) \cdot c \cdot m)^{\mathrm{poly}(\ell, n)}$.

The proof of this proposition is by induction on $alt(\varphi)$, and essentially follows the standard arguments to bound the running time of the quantifier elimination procedure for Presburger arithmetic. The key ingredient that leads to the bounds above is that, for $\mathcal{Oct}$, the natural numbers a in line 3 and g in line 2 of PresQE are always 1. This has two effects. Firstly, it shows that $\mathcal{Oct}$ admits quantifier elimination, i.e., while running the procedure no atomic formulae outside $\mathcal{Oct}$ can arise. This is best witnessed by looking at line 3 in PresQE. There, the divisibility constraints $a \mid t + k$ are trivially satisfied, and we are replacing x with a term of the form $\pm y + c$ for some $c \in \mathbb{Z}$. From these substitutions, only constraints from $\mathcal{Oct}$ or constraints of the form $\pm 2 \cdot y < b$ can arise, and the latter are normalized to $y \leq \lfloor \frac{b-1}{2} \rfloor$ or $y \geq \lceil \frac{1-b}{2} \rceil$ as explained in Section 2. The second effect is on the growth of the constants. The variable r in line 3 only depends on $mod(\Phi)$, which now does not grow during the procedure, and on $\|lin(\Phi)\|$, which grows only exponentially in the number of variables in Φ.

We now move to the running time of Algorithm 1 on inputs that come from $\mathcal{Pa}(2^{\mathbb{N}}(\cdot))$. The properties of this procedure are summarized in the next proposition.

▶ **Proposition 10.** *Let $\mathcal{F} = \mathcal{Sem}$. Let $\Phi(\boldsymbol{y})$ be a formula from $\mathcal{Pa}(\lambda x.2^{|x|})$ in prenex normal form, with no divisibility constraints, and in which each quantified variable appears either only linearly or only in powers. Suppose Φ has $alt(\Phi) = B$ quantifier blocks, each with at most $L \geq 1$ variables occurring linearly and each block having at most $E \geq 1$ variables occurring in powers. On input Φ, Algorithm 1 returns a formula Ψ with bounds as in the following table:*

	$\#hom$	*heft*	$\|lin(\cdot)\|$	*mod*	$\mathcal{B}$
Φ	$h \geq 2$	$v \geq 2$	$c \geq 4$	$m \geq 4$	b
Ψ	$H \coloneqq (E \cdot h \cdot \log(c \cdot m))^{(2 \cdot v)^{2^5 \cdot L \cdot B^3}}$	$2^{B \cdot L} v$	2^H	2^H	$b \cdot 2^H$

and the number of quantifiers added to Π' is at most H. The running time of the procedure is in $len(\Phi) \uparrow (E \cdot h \cdot \log(c \cdot m)) \uparrow v \uparrow \mathrm{poly}(L, B)$.

In the above proposition, $a \uparrow b \coloneqq a^b$ is the exponentiation function and, following Knuth's up-arrow notation, it is right-associative. In view of our bounds for one iteration of SemCover given in Lemma 7, the bound on $\#hom(\Psi)$ should seem somewhat surprising. We know from the correctness of SemCover that, after a call to SemCover, one of the variables appearing in powers will only occur in constraints from $\mathcal{PowCmp}$. Since all these variables need to have this property before moving to the next quantifier block, SemCover must be chained at least E times within a block, E being the number of variables occurring in powers in the block. Then, from the bound $\#hom(\theta_j) \leq h \cdot (v + 10) + n$ in Lemma 7, one might expect $\#hom(\Psi)$ to be roughly $h \cdot v^E$, thus exponential in E even for a single block of quantifiers. Proposition 10, however, proves otherwise: $\#hom(\Psi)$ is only polynomial in E. For this, we need to sharpen the correctness statement for SemCover.

We already know that in each output formula some variable x is made unproblematic, and no new problematic occurrences of variables are created. We prove a stronger statement: namely, that every *rewriting* of a homogeneous term (by a substitution in lines 11–18) makes the number of problematic variables in that term decrease. This is possible thanks to the tailoring of these substitutions to individual inequalities, which allows us to track the evolution of each term independently of the rest of the formula. As a result:

- a formula that is placed into D at the end of processing a single quantifier block can be a result of E chained calls to SemCover, but
- the number of calls to SemCover that rewrite an individual term during its evolution is bounded by the heft.

Therefore, we can iterate the above-mentioned bound $\#hom(\theta_j) \leq h \cdot (v + 10) + n$ just $2^L \cdot v$ times instead of E times. Thus $\#hom(\Psi)$ in Proposition 10 is found to be exponential in v and only polynomial in E. We remark that if $\#hom(\Psi)$ were instead found to be exponential in E, then the algorithm would not have any hope of running in elementary time. This is because, after a block of quantifiers is considered, E increases by the number of variables introduced in SemCover, which from Lemma 7 is roughly the number of homogeneous terms.

To prove Theorem 2 it suffices to chain the bounds and running times of Proposition 10, Lemma 8 and Proposition 9, according to the algorithm given at the beginning of the section.

Avoiding quadruply exponential numbers. It may not be immediately evident from the bounds in the various tables why we do not perform quantifier elimination eagerly and instead run Algorithm 1 without fully eliminating quantifiers first (in mode $\mathcal{Sem}$), then call Linearize, and only afterwards eliminate the remaining quantifiers by running Algorithm 1 again (now in mode $\mathcal{QF}$). In fact, this sequence is fundamental for obtaining a 3ExpTime procedure.

Consider the formula $\Psi \coloneqq q \mid 2^{|x|} - r \wedge y \geq 2^{|x|}$. For specific values of q and r, the smallest $|x|$ satisfying Ψ might be $q-1$. If Ψ is a subformula obtained during quantifier elimination, then, according to Proposition 10, q might have a triply exponential magnitude relative to the input size. This means that the smallest y satisfying Ψ might have a quadruply exponential magnitude. Eliminating x and y in this case would lead to a quadruply exponential blow-up in the number of disjuncts to be considered during quantifier elimination. Our strategy avoids this problem by delaying (if necessary) the elimination of x and y until we obtain a formula in $\mathcal{PowCmp}$. Calling Linearize reduces the reasoning to the exponents, which are triply exponential at worst.

An observation on $\mathcal{Oct}$. The bounds for integer octagon arithmetic presented in Proposition 9 reveal not only that this logic admits an exponential-time quantifier elimination procedure, but also that the satisfaction problem for this theory can be solved in PSpace. Indeed, observe that the bound on $\|lin(\Psi)\|$ given in Proposition 9 implies that all constants and coefficients appearing in the output formula Ψ have polynomial bit-length. Then, one can apply the standard quantifier relativization algorithm from $\mathcal{Pa}$ to obtain a PSpace procedure for $\mathcal{Oct}$. Briefly, the quantifier relativization procedure for $\mathcal{Pa}$ first replaces every quantifier $\exists x.\varphi$ in the input formula with a *bounded quantifier* $\exists x \in [-f(\varphi), f(\varphi)].\, \varphi$, where $f\colon \mathcal{Pa} \to \mathbb{N}$, and then iterates through all (finitely many) values the quantified variable can take, searching for a solution to the formula. The bound on $\|lin(\Psi)\|$ obtained for $\mathcal{Oct}$ implies that $f(\Psi)$ has bit-length that is at most polynomial in $\mathrm{len}(\Psi)$. See [22] for more information on quantifier relativization.

On the non-elementary bound for $\mathcal{Pa}(\lambda x.2^{|x|})$. To conclude, we provide some insights on why our procedure runs in non-elementary time on formulae from the Tower-complete logic $\mathcal{Pa}(\lambda x.2^{|x|})$. One of the ingredients that guarantee that our procedure for $\mathcal{Pa}(2^{\mathbb{N}}(\cdot))$ runs in 3ExpTime is that we are able to postpone calls to Linearize to after Algorithm 1. In $\mathcal{Pa}(\lambda x.2^{|x|})$ this is not possible: since each variable can appear both linearly and in powers, Linearize must be invoked after each call to SemCover, in order to "linearize" a variable, and then eliminate it with PresQE. However, SemCover adds, in the worst case, a number of additional variables that is roughly the number h of homogeneous terms in the formula (see Lemma 7). When the next quantifier block is considered, these variables must all be linearized and eliminated with PresQE. As indicated in the table of Lemma 6 (leftmost column of the last row), in doing this, the number of homogeneous terms of the resulting formula becomes exponential in h. This "h^h" dependency makes the algorithm run in non-elementary time (in fact Tower).

6 Conclusion

We have proven new elementary upper bounds for $\mathcal{Pa}(2^{\mathbb{N}}(\cdot))$, and for the existential fragment of $\mathcal{Pa}(\lambda x.2^{|x|})$. We believe this is a step towards understanding which decidable arithmetic theories have elementary bounds, and moreover that our method extends to provide elementary bounds for any prefix class of $\mathcal{Pa}(\lambda x.2^{|x|})$, but we leave this for future work. Our results open several research directions, which we now summarize.

Tighter bounds for $\mathcal{Pa}(2^{\mathbb{N}}(\cdot))$. It is well-known that, using the bounds on the formulae returned by quantifier elimination procedures for $\mathcal{Pa}$, one can derive a 2AExpTime(poly) quantifier relativization procedure for $\mathcal{Pa}$ [25]. Here 2AExpTime(poly) is the class of

all problems that can be decided with an alternating Turing machine running in doubly exponential time and performing a polynomial number of alternations. In fact, $\mathcal{P}\boldsymbol{a}$ is complete for this class under polynomial-time reductions [1]. Our 3ExpTime procedure for $\mathcal{P}\boldsymbol{a}(2^{\mathbb{N}}(\cdot))$ shows that, in terms of deterministic time complexity, this theory is not harder to decide than $\mathcal{P}\boldsymbol{a}$. However, at this stage obtaining a 2AExpTime(poly) quantifier relativization algorithm from the bounds of our procedure seems not easy.

Automata-based decision procedures. As $\mathcal{P}\boldsymbol{a}(2^{\mathbb{N}}(\cdot))$ is a fragment of Büchi arithmetic, it also admits a representation by finite automata. It appears plausible that the automata-based procedure for $\mathcal{P}\boldsymbol{a}$ [14, 7] could be adapted to $\mathcal{P}\boldsymbol{a}(2^{\mathbb{N}}(\cdot))$. However, having the procedure run in 3ExpTime might be very challenging. This is due to the fact that, as observed above, $\mathcal{P}\boldsymbol{a}(2^{\mathbb{N}}(\cdot))$ formulae may require numbers of quadruply exponential magnitude; instead of triply exponential as in the case of $\mathcal{P}\boldsymbol{a}$.

Geometric decision procedures. A class of regular expressions corresponding to $\mathcal{P}\boldsymbol{a}(2^{\mathbb{N}}(\cdot))$ was already defined by Semenov [23, Theorem 5]. These expressions can be seen as an extension of semilinear sets [18, 9], so it is conceivable that there is an elementary decision procedure for $\mathcal{P}\boldsymbol{a}(2^{\mathbb{N}}(\cdot))$ which is based on geometry and manipulates these objects directly. However, similarly to the automata-based approach, making such a procedure run in 3ExpTime, as the recent one for $\mathcal{P}\boldsymbol{a}$ [5] does, appears challenging.

Tighter bounds for $\exists\mathcal{P}\boldsymbol{a}(\lambda x.2^{|x|})$. An obvious question is whether our upper bound for the existential fragment can be improved. For comparison, the existential fragment of Büchi arithmetic is known to be in NP [11]. While the same may be true for $\exists\mathcal{P}\boldsymbol{a}(\lambda x.2^{|x|})$, it would be very surprising if such a result were to be proved with a technique similar to the one in our paper. In our NExpTime algorithm for $\exists\mathcal{P}\boldsymbol{a}(\lambda x.2^{|x|})$, the main source of blow-up is the use of Weispfenning's quantifier elimination procedure to eliminate linearly occurring variables. Quantifier elimination is known to be often non-optimal when it comes to deciding existential fragments of logics, and this is the case for $\exists\mathcal{P}\boldsymbol{a}$, the existential fragment of Presburger arithmetic. A possible avenue to improve the NExpTime upper bound would be to look at geometric procedures, which in the context of $\exists\mathcal{P}\boldsymbol{a}$ perform much better.

Improving the NP lower bound is also challenging. There are several extensions of $\exists\mathcal{P}\boldsymbol{a}$ that currently fall between NP and NExpTime. These include $\exists\mathcal{P}\boldsymbol{a}$ with pre-quadratic constraints [10, 21] and $\exists\mathcal{P}\boldsymbol{a}$ with divisibility constraints [15]. One idea is to exploit the ability of $\exists\mathcal{P}\boldsymbol{a}(\lambda x.2^{|x|})$ to express a pairing function, that is, an injection from $\mathbb{N}\times\mathbb{N}$ to $\mathbb{N}$, with, e.g., the formula $z = 2^{|2x|} + 2^{|2y+1|}$ [6, p. 55]. Pairing functions are known to lead to non-elementary lower bounds in the presence of quantifier alternation, and an interesting direction would be to study their effect on existential theories.

References

1 Leonard Berman. The complexity of logical theories. *Theor. Comput. Sci.*, 11(1):71–77, 1980.

2 Alexis Bès. A survey of arithmetic definability. *Soc. Math. Belgique*, pages 1–54, 2002.

3 Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Springer, 1997.

4 Gregory Cherlin and Françoise Point. On extensions of Presburger arithmetic. In *4th Easter Conference on Model Theory*, volume 86 of *Humboldt-Univ. Berlin Seminarberichte*, pages 17–34, 1986. URL: `https://webusers.imj-prg.fr/~francoise.point/papiers/cherlin_point86.pdf`.

5 Dmitry Chistikov, Christoph Haase, and Alessio Mansutti. Geometric decision procedures and the VC dimension of linear arithmetic theories. In *LICS*, 2022.

6 Kevin J. Compton and C. Ward Henson. A uniform method for proving lower bounds on the computational complexity of logical theories. *APAL*, 48(1):1–79, 1990.

7 Antoine Durand-Gasselin and Peter Habermehl. On the use of non-deterministic automata for Presburger arithmetic. In *CONCUR*, 2010.

8 Jeanne Ferrante and Charles Rackoff. A decision procedure for the first order theory of real addition with order. *SIAM J. Comput.*, 4(1):69–76, 1975.

9 Seymour Ginsburg and Edwin H. Spanier. Semigroups, Presburger formulas, and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966.

10 Robert Givan, David McAllester, Carl Witty, and Dexter Kozen. Tarskian set constraints. *Information and Computation*, 174(2):105–131, 2002.

11 Florent Guépin, Christoph Haase, and James Worrell. On the existential theories of Büchi arithmetic and linear p-adic fields. In *LICS*, 2019.

12 Philipp Hieronymi and Christian Schulz. A strong version of Cobham's theorem. In *STOC*, 2022.

13 Deepak Kapur, Zhihai Zhang, Matthias Horbach, Hengjun Zhao, Qi Lu, and ThanhVu Nguyen. Geometric quantifier elimination heuristics for automatically generating octagonal and max-plus invariants. In *Automated Reasoning and Mathematics - Essays in Memory of William W. McCune*, volume 7788 of *LNCS*, pages 189–228. Springer, 2013.

14 Felix Klaedtke. Bounds on the automata size for Presburger arithmetic. *ACM Trans. Comput. Log.*, 9(2):11:1–11:34, 2008.

15 Antonia Lechner, Joël Ouaknine, and James Worrell. On the complexity of linear arithmetic with divisibility. In *LICS*, 2015.

16 Antoine Miné. The octagon abstract domain. *High. Order Symb. Comput.*, 19(1):31–100, 2006.

17 Derek C. Oppen. A $2^{2^{2^{pn}}}$ upper bound on the complexity of Presburger arithmetic. *JCSS*, 16(3):323–332, 1978.

18 Rohit J. Parikh. On context-free languages. *J. ACM*, 13(4):570–581, 1966.

19 Françoise Point. On decidable extensions of Presburger arithmetic: from A. Bertrand numeration systems to Pisot numbers. *JSL*, 65(3):1347–1374, 2000.

20 Mojżesz Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt, 1929. In *Comptes Rendus du I Congrès des Mathématiciens des Pays Slaves*, pages 92–101.

21 Rodrigo Raya, Jad Hamza, and Viktor Kunčak. On the complexity of convex and reverse convex prequadratic constraints. In *LPAR*, 2023.

22 C. R. Reddy and D. W. Loveland. Presburger arithmetic with bounded quantifier alternation. In *STOC*, 1978.

23 Aleksei L. Semenov. On certain extensions of the arithmetic of addition of natural numbers. *Math. USSR Izv.*, 15(2):401–418, 1980.

24 Aleksei L. Semenov. Logical theories of one-place functions on the set of natural numbers. *Math. USSR Izv.*, 22(3):587–618, 1984.

25 Volker Weispfenning. The Complexity of Almost Linear Diophantine Problems. *J. Symb. Comput.*, 10(5):395–404, 1990.

A Dichotomy for Succinct Representations of Homomorphisms

Christoph Berkholz
Technische Universität Ilmenau, Germany

Harry Vinall-Smeeth
Humboldt-Universität zu Berlin, Germany

Abstract

The task of computing homomorphisms between two finite relational structures $\mathcal{A}$ and $\mathcal{B}$ is a well-studied question with numerous applications. Since the set $\mathrm{Hom}(\mathcal{A}, \mathcal{B})$ of all homomorphisms may be very large having a method of representing it in a succinct way, especially one which enables us to perform efficient enumeration and counting, could be extremely useful.

One simple yet powerful way of doing so is to decompose $\mathrm{Hom}(\mathcal{A}, \mathcal{B})$ using union and Cartesian product. Such data structures, called d-representations, have been introduced by Olteanu and Závodný [32] in the context of database theory. Their results also imply that if the treewidth of the left-hand side structure $\mathcal{A}$ is bounded, then a d-representation of polynomial size can be found in polynomial time. We show that for structures of bounded arity this is optimal: if the treewidth is unbounded then there are instances where the size of any d-representation is superpolynomial. Along the way we develop tools for proving lower bounds on the size of d-representations, in particular we define a notion of reduction suitable for this context and prove an almost tight lower bound on the size of d-representations of all k-cliques in a graph.

2012 ACM Subject Classification Theory of computation → Data structures and algorithms for data management; Theory of computation → Complexity theory and logic; Theory of computation → Data compression

Keywords and phrases homomorphism problem, CSP, succinct representations, enumeration, lower bound, treewidth

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.113

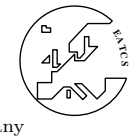
Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `https://arxiv.org/abs/2209.14662`

Funding *Christoph Berkholz*: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 414325841.
Harry Vinall-Smeeth: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 385256563.

1 Introduction

The task of computing homomorphisms between two finite relational structures has a long history and numerous applications. Most notably, as pointed out by Feder and Vardi [17], it is the right abstraction for the constraint satisfaction problem (CSP) – a framework for search problems that generalised Boolean satisfiability. Moreover, evaluating conjunctive queries on a relational database is equivalent to computing homomorphisms from the query structure to the database. While deciding the existence of a homomorphism from a structure $\mathcal{A}$ to a structure $\mathcal{B}$ is a classical NP-complete problem, several restrictions of the input instance have been considered in order to understand the landscape of tractability. One line of research investigates *right-hand-side* restrictions, where it is asked for which classes of structures $\mathcal{B}$ the CSP becomes tractable and when it remains hard. This culminated in



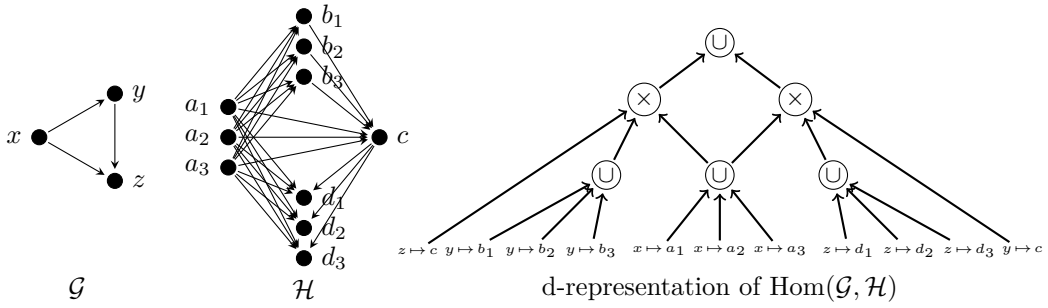


Figure 1 A deterministic d-representation of all homomorphisms from $\mathcal{G}$ to $\mathcal{H}$.

the solution [9, 38] of the CSP-dichotomy conjecture [17] that characterises those $\mathcal{B}$ where finding a homomorphism from a given structure $\mathcal{A}$ can be done in polynomial time (assuming P≠NP).

Another line of research, to which we contribute in this paper, focuses on *left-hand-side* restrictions: for which classes of structures $\mathcal{A}$ can we efficiently find a homomorphism from $\mathcal{A}$ to a given $\mathcal{B}$? In this scenario, a dichotomy is only known when all relations have bounded arity, as is the case for graphs, digraphs, or k-uniform hypergraphs. Grohe [20] showed that, modulo complexity theoretic assumptions, for any class of structures $\mathfrak{A}$ of bounded arity the decision problem, "Given a structure $\mathcal{A} \in \mathfrak{A}$ and a structure $\mathcal{B}$, is there a homomorphism from $\mathcal{A}$ to $\mathcal{B}$?" is in polynomial time if and only if the homomorphic core of every structure in $\mathfrak{A}$ has bounded treewidth. For classes of unbounded arity, polynomial time tractability has been shown for fractional hypertreewidth [5, 21], but a full characterisation of tractability has only been obtained in the parameterised setting using submodular width [29]. Besides deciding the existence of a homomorphism, the complexity of counting all homomorphism has also been characterised in the right-hand-side regime [8] and for bounded-arity classes of left-hand-side structures [14]. A third task, that is less well understood, is to *enumerate* all homomorphisms; here only partial results on the complexity are known (e. g. [10, 13, 37, 19]).

In this work we consider the task of *representing* the set Hom($\mathcal{A}, \mathcal{B}$) of all homomorphisms in a succinct and accessible way. In particular, we want to store all, potentially exponentially many, homomorphisms, in a data structure of polynomial size that enables us to, e. g. generate a stream of all homomorphisms. The data structures we are interested in – so-called d-representations – were first introduced to represent homomorphisms in the context of join evaluation under the name *factorised databases* [32]. They are conceptually very simple: the set of homomorphisms is represented by a circuit, where the "inputs" are mappings of single vertices and larger sets of mappings are generated by combining local mappings using Cartesian product $\times$ and union $\cup$. In the circuit previously computed sets of local homomorphisms are represented by gates and can be used several times, see Figure 1 for an example. Such a representation is called *deterministic* if every $\cup$-gate is guaranteed to combine disjoint sets. Deterministic representations have the advantage that the number of homomorphisms can be efficiently counted by adding the sizes of the local homomorphism sets on every $\cup$-gate and multiplying them on every $\times$-gate. Moreover, all homomorphisms can be efficiently enumerated where the delay between two outputs is only linear in the size of every produced homomorphism (= size of the universe of $\mathcal{A}$) [2]. It is known that if the treewidth of the left-hand side structure is bounded, then a deterministic d-representation of polynomial size can be found in polynomial time [32]. Our main theorem shows that for structures of bounded arity this is optimal: if the treewidth is unbounded, then there are instances where the size of any (not necessarily deterministic) representation is superpolynomial.

▶ **Theorem 1.** *Let $r \in \mathbb{N}$, σ a signature of arity $\leq r$ and $\mathfrak{A}$ a class of σ-structures. Then the following are equivalent:*

1. *There is a $w \in \mathbb{N}$ such that every structure in $\mathfrak{A}$ has treewidth at most w.*
2. *A deterministic d-representation of $\mathrm{Hom}(\mathcal{A}, \mathcal{B})$ can be computed in polynomial time, for any $\mathcal{A} \in \mathfrak{A}$ and any $\mathcal{B}$.*
3. *There is a $c \in \mathbb{N}$ such that for any $\mathcal{A} \in \mathfrak{A}$ and any $\mathcal{B}$ there exists a (not necessarily deterministic) d-representation of $\mathrm{Hom}(\mathcal{A}, \mathcal{B})$ of size $O\big((\|\mathcal{A}\| + \|\mathcal{B}\|)^c\big)$.*

Related work. The research on succinct data structures for homomorphism problems has emerged from the two different perspectives. When fixing the right-hand-side structure $\mathcal{B}$, then data structures like multi-valued decision diagrams (MDD) [4], AND/OR multi-valued decision diagrams (AOMDD) [30], and multi-valued decomposable decision graphs (MDDG) [25] have been proposed, which arose from representations for Boolean functions that are studied in *knowledge compilation* (see, e. g., [15]). The (deterministic) d-representations studied in this paper can be interpreted as (deterministic) DNNF circuits with zero-suppressed semantics [2, Lemma 7.4], where a $\cup$-gate corresponds to a (deterministic) $\vee$-gate and a $\times$-gate corresponds to a *decomposable* $\wedge$-gate.

In the left-hand-side regime, representations have been introduced in the context of enumerating query results. Most notably, Olteanu and Závodnỳ [32] introduced the notion of *factorised databases* that are used to decompose the result relation of a conjunctive query using Cartesian product and union. Their findings imply the upper bound part of our dichotomy theorem: if $\mathcal{A}$ has bounded treewidth, its tree decomposition defines a so-called *d-tree*, which structures the polynomial size d-representation. They have also shown a limited lower bound for *structured* representations ("d-representations respecting a d-tree"). However, this lower bound considers only a small subclass of all possible d-representations. In a similar vein, in knowledge compilation there exist several restrictions of DNNFs e. g. requiring $\vee$-gates to be decision or deterministic [15], or enforcing structuredness [33]. In this light, the significance of our lower bound comes from the fact that it holds for the most general notion of representations (d-representations), which correspond to unrestricted DNNFs.

The proof of our lower bound has some connections to the conditional lower bound for the counting complexity of homomorphisms [14], which in turn builds upon the construction of Grohe [20]. The essence of these proofs is to rely on an assumption about the hardness of the parametrised clique problem and reduce this to all structures of unbounded treewidth. We take a similar route: in Section 5 we prove an unconditional lower bound for representing cliques and obtain our main lower bound using a sequence of reductions in Section 6.

The circuit notion for representing the set of homomorphisms between two given structures (or, equivalently, the result relation of a multiway join query) in a succinct data structure might be confused with previous work on the (Boolean or arithmetic) circuit complexity for deciding or counting homomorphisms or subgraphs. In this research branch, a structure $\mathcal{B}$ over a universe of size n is given as input to a circuit $C_{\mathcal{A},n}$, which decides the existence of or counts the number of homomorphisms (or subgraph-embeddings) from $\mathcal{A}$ to $\mathcal{B}$. Examples include monotone circuits for finding cliques [1, 36], bounded-depth circuits for finding cliques and other small subgraphs [35, 26] as well as graph polynomials and monotone arithmetic circuits [16, 7, 24] for counting homomorphisms. In particular, the recent work of Komarath, Pandey and Rahul [24] studies monotone arithmetic circuits that have, for each pattern $\mathcal{G}$ and each n, an input indicator variable $x_{\{u,v\}}$ for each potential edge $\{u, v\} \in [n]^2$ in the second graph $\mathcal{H}$. For every input (i.e. setting indicator variables according to a graph $\mathcal{H}$

on n vertices), the arithmetic circuit has to compute the number of homomorphisms from $\mathcal{G}$ to $\mathcal{H}$. Interestingly, Komorath et al. prove a tight bound and show that such arithmetic circuits need size $n^{\mathrm{tw}(\mathcal{G})+1}$. Unfortunately, this and related results from circuit complexity (such as lower bounds for the clique problem) do not translate to the knowledge compilation approach. Part of the reason is that we crucially have a different representation *for each pair* $\mathcal{G}$, $\mathcal{H}$ and having, e. g. an arithmetic circuit computing the constant number $|\mathrm{Hom}(\mathcal{G},\mathcal{H})|$ is trivial. Moreover, due to monotonicity, the worst-case right-hand-side instances $\mathcal{H}$ in [24] are complete graphs, whereas d-representations lack this form of monotonicity: adding edges to $\mathcal{H}$ can make factorisation simpler and in particular occurences of patterns in complete graphs can be succinctly factorised.

Despite this, some techniques on a more general level (e. g. arguing about the transversal of a circuit or using random graphs as bad examples) are useful in circuit complexity as well as for proving lower bounds on representations.

2 Preliminaries

We write $\mathbb{N}$ for the set of non-negative integers and define $[n] := \{1, \ldots, n\}$ for any positive integer n. Given a set S we write 2^S to denote the power set of S. Whenever writing a to denote a k-tuple, we write a_i to denote the tuple's i-th component; i. e., $a = (a_1, \ldots, a_k)$. For a function $f\colon X \to Y$ and $X' \subset X$ we write $\pi_{X'} f$ to denote the projection of f to X'. Given a set of functions, each of which has a domain containing X', we write $\pi_{X'} F := \{\pi_{X'} f \mid f \in F\}$.

Graphs, Minors, Structures, Tree Decompositions. Whenever $\mathcal{G}$ is a graph or a hypergraph we write $V(\mathcal{G})$ and $E(\mathcal{G})$ for the set of nodes and the set of edges, respectively, of $\mathcal{G}$. We let $\mathcal{K}_k$ be the complete graph on k vertices, $\mathcal{C}_k$ the k-cycle graph, and $\mathcal{G}_k$ the $k \times k$-grid graph. Given a graph $\mathcal{G}$ and $\{u, v\} \in E(\mathcal{G})$, we can form a new graph by *edge contraction*: replacing u and v be a new vertex w adjacent to all neighbours of u and v. A graph $\mathcal{H}$ is a *minor* of $\mathcal{G}$ if $\mathcal{H}$ can be obtained from $\mathcal{G}$ by repeatedly deleting vertices, deleting edges and contracting edges.

A *tree decomposition* of a graph $\mathcal{G}$ is a pair (T, β) where T is a tree and $\beta\colon V(T) \to 2^{V(\mathcal{G})}$ associates to every node $t \in V(T)$ a bag $\beta(t)$ such that the following is satisfied: (1) For every $v \in V(\mathcal{G})$ the set $\{t \in V(T) \mid v \in \beta(t)\}$ is non-empty and forms a connected set in T. (2) For every $\{u, v\} \in E(\mathcal{G})$ there is some $t \in V(T)$ such that $\{u, v\} \subseteq \beta(t)$. The width of a tree decomposition is $\max_{t \in V(T)} |\beta(t)| - 1$ and the *treewidth of* $\mathcal{G}$ is the minimum width of any tree decomposition of $\mathcal{G}$.

A (relational) signature σ is a set of relation symbols R, each of which is equipped with an arity $r = r(R)$. A (finite, relational) σ-structure $\mathcal{A}$ consists of a finite universe A and relations $R^{\mathcal{A}} \subseteq A^r$ for every r-ary relation symbol $R \in \sigma$. We will write $\|\mathcal{A}\| := \sum_{R \in \sigma} |R^{\mathcal{A}}|$. The *Gaifman graph* of $\mathcal{A}$ is the graph with vertex set A and edges $\{u, v\}$ for any distinct u, v that occur together in a tuple of a relation in $\mathcal{A}$. The *treewidth* of a structure is the treewidth of its Gaifman graph. We say $\mathcal{A}$ is *connected* if its Gaifman graph is connected and we will henceforth assume, without loss of generality, that all structures in this paper are connected.

Enumeration. An enumeration algorithm for $\mathrm{Hom}(\mathcal{A}, \mathcal{B})$ proceeds in two stages. In the *preprocessing* stage the algorithm does some preprocessing on $\mathcal{A}$ and $\mathcal{B}$. In the *enumeration phase* the algorithm enumerates, without repetition, all homomorphisms in $\mathrm{Hom}(\mathcal{A}, \mathcal{B})$, followed by the end of enumeration message. The *delay* is the maximum of three times: the time between the start of the enumeration phase and the first output homomorphism, the

maximum time between the output of two consecutive homomorphisms and between the last tuple and the end of enumeration message. The *preprocessing time* is the time the algorithm spends in the preprocessing stage, which may be 0. Similarly given a d-representation C for $\mathrm{Hom}(\mathcal{A}, \mathcal{B})$, an enumeration algorithm for C has a preprocessing stage, where it can do some preprocessing on C, and an enumeration phase defined as above.

3 Homomorphisms and the complexity of constraint satisfaction

A *homomorphism* $h\colon \mathcal{A} \to \mathcal{B}$ between two σ-structures $\mathcal{A}$ and $\mathcal{B}$ is a mapping from A to B that preserves all relations, i. e., for every r-ary $R \in \sigma$ and $(a_1, \ldots, a_r) \in A^r$ it holds that if $(a_1, \ldots, a_r) \in R^{\mathcal{A}}$, then $(h(a_1), \ldots, h(a_r)) \in R^{\mathcal{B}}$. We let $\mathrm{Hom}(\mathcal{A}, \mathcal{B})$ be the set of all homomorphisms from $\mathcal{A}$ to $\mathcal{B}$. A (homomorphic) core of a structure $\mathcal{A}$ is an inclusion-wise minimal substructure $\mathcal{A}' \subseteq \mathcal{A}$ such that there is a homomorphism from $\mathcal{A}$ to $\mathcal{A}'$. It is well known that all cores of a structure are isomorphic, hence we will also speak of *the* core of a structure.

Following common notation we fix a (potentially infinite) signature σ and define for classes of σ-structures $\mathfrak{A}$ and $\mathfrak{B}$ the (promise) decision problem $\mathrm{CSP}(\mathfrak{A}, \mathfrak{B})$ to be: "Given two σ-structures $\mathcal{A} \in \mathfrak{A}$ and $\mathcal{B} \in \mathfrak{B}$, is there a homomorphism from $\mathcal{A}$ to $\mathcal{B}$?" Similarly, the counting problem $\#\mathrm{CSP}(\mathfrak{A}, \mathfrak{B})$ asks: "Given two σ-structures $\mathcal{A} \in \mathfrak{A}$ and $\mathcal{B} \in \mathfrak{B}$, what is the number of homomorphisms from $\mathcal{A}$ to $\mathcal{B}$?" A lot of work has been devoted towards classifying the classes of structures for which the problems are solvable in polynomial time. Normally either the left-hand-side $\mathfrak{A}$ or the right-hand-side $\mathfrak{B}$ is restricted and the other part ($\mathfrak{B}$ or $\mathfrak{A}$) is the class _ of all structures. A related problem is Enum-$\mathrm{CSP}(\mathfrak{A}, \mathfrak{B})$ [10], which is the following task: "Given two σ-structures $\mathcal{A} \in \mathfrak{A}$ and $\mathcal{B} \in \mathfrak{B}$, enumerate all homomorphisms from $\mathcal{A}$ to $\mathcal{B}$". One way of defining tractability for enumeration algorithms is *polynomial delay enumeration*, where the preprocessing time and the delay is polynomial in $\mathcal{A}$ and $\mathcal{B}$.

In this paper we focus on "left-hand-side" restrictions, where $\mathfrak{B}$ is the class of all structures. Moreover, we assume that the arity of each symbol in σ is bounded by some constant r. In this setting the complexity of $\mathrm{CSP}(\mathfrak{A}, \_)$ and $\#\mathrm{CSP}(\mathfrak{A}, \_)$ is fairly well understood: the decision problem is polynomial time tractable iff the core of every structure in $\mathfrak{A}$ has bounded treewidth, while the counting problem is tractable if every structure from $\mathfrak{A}$ itself has bounded treewidth. This is made precise by the following two theorems.

▶ **Theorem 2** ([20]). *Let $r \in \mathbb{N}$, σ be a signature of arity $\leq r$ and $\mathfrak{A}$ a class of σ-structures. Under the assumption that there is no $c \in \mathbb{N}$ such that for every $k \in \mathbb{N}$ there is an algorithm that finds a k-clique in an n-vertex graph in time $O(n^c)$ the following two statements are equivalent.*

1. *There is a $w \in \mathbb{N}$ such that the core of every structure in $\mathfrak{A}$ has treewidth at most w.*
2. $\mathrm{CSP}(\mathfrak{A}, \_)$ *is solvable in polynomial time.*

▶ **Theorem 3** ([14]). *Let $r \in \mathbb{N}$, σ be a signature of arity $\leq r$ and $\mathfrak{A}$ a class of σ-structures. Under the assumption that there is no $c \in \mathbb{N}$ such that for every $k \in \mathbb{N}$ there is an algorithm that counts the number of k-cliques in an n-vertex graph in time $O(n^c)$ the following two statements are equivalent.*

1. *There is a $w \in \mathbb{N}$ such that every structure in $\mathfrak{A}$ has treewidth at most w.*
2. $\#\mathrm{CSP}(\mathfrak{A}, \_)$ *is solvable in polynomial time.*

To understand the difference between these characterisations, consider the class $\mathfrak{A}$ of all structures $\mathcal{A}_k$ that are complete graphs on k vertices with an additional vertex with a self-loop. The homomorphic core of such structures is just the self-loop and finding one

homomorphism from $\mathcal{A}_k$ to $\mathcal{B}$ is equivalent to finding a self-loop in $\mathcal{B}$. However, counting homomorphisms from $\mathcal{A}_k$ to $\mathcal{B}$ is as hard as counting k-cliques: if $\mathcal{B}$ is a simple graph $\mathcal{G}$ with one additional vertex with a self-loop, then the number of homomorphisms from $\mathcal{A}_k$ to $\mathcal{B}$ is the number of k-cliques in $\mathcal{G}$ plus one.

The complexity of the corresponding enumeration problem Enum-CSP($\mathfrak{A}$, _) is still open. It has been shown that polynomial delay enumeration is possible if $\mathfrak{A}$ has bounded treewidth [10]. On the other hand, polynomial delay enumeration implies solvability of the decision problem in polynomial time (because either the first solution or an end-of-enumeration message has to appear after polynomial time). Hence it follows from Theorem 2, under the same complexity assumption, that there is no polynomial delay enumeration algorithm if the cores of the structures in $\mathfrak{A}$ have unbounded treewidth. For further discussion on this topic we refer the reader to [10].

Our main result (Theorem 1) can be viewed as an unconditional dichotomy for enumeration and counting in a restricted class of algorithms: when the algorithm relies on local decompositions into union and product, then the tractable instances are exactly those that have bounded treewidth. Interestingly, this matches the conditional dichotomy for the counting case (Theorem 3).

4 Factorised Representations

In this section we formally introduce the factorisation formats for CSPs. These formats agree with the factorised representations of relations introduced by Olteanu and Závodný [32] in the context of evaluating conjunctive queries on relational databases. While we stick to the naming conventions introduced there we provide a slightly different circuit-based definition that is very much inspired by [2] and the notion of *set circuits* introduced in [3].

A *factorisation circuit* C for two sets A and B is an acyclic directed graph with node labels and a unique sink. Each node without incoming edges is called an *input gate* and labelled by $\{a \mapsto b\}$ for some $a \in A$ and $b \in B$. Every other node is labelled by either $\cup$ or $\times$ and called a $\cup$-*gate* or $\times$-*gate*, respectively. For each gate g in the circuit we inductively define its *domain* $\mathrm{dom}(g) \subseteq A$ by $\mathrm{dom}(g) = \{a\}$ if g is an input gate with label $\{a \mapsto b\}$ and $\mathrm{dom}(g) = \bigcup_{i=1}^{r} \mathrm{dom}(g_i)$ if g is a non-input gate with child gates $g_1, \dots, g_r$.

A factorisation circuit is *well-defined* if for every gate g with child gates $g_1, \dots, g_r$ it holds that $\mathrm{dom}(g) = \mathrm{dom}(g_1) = \cdots = \mathrm{dom}(g_r)$ if g is a $\cup$-gate and $\mathrm{dom}(g_i) \cap \mathrm{dom}(g_j) = \emptyset$ for all $i \neq j$ if g is a $\times$-gate. For every gate g in a well-defined factorisation circuit we let S_g be a set of mappings $h\colon \mathrm{dom}(g) \to B$ defined by

$$S_g := \begin{cases} \{\{a \mapsto b\}\} & \text{if } g \text{ is an input labelled by } \{a \mapsto b\} \\ S_{g_1} \cup \cdots \cup S_{g_r} & \text{if } g \text{ is a } \cup\text{-gate with children } g_1, \dots, g_r, \\ \{\, h_1 \cup \cdots \cup h_r \mid h_i \in S_{g_i},\, i \in [r] \,\} & \text{if } g \text{ is a } \times\text{-gate with children } g_1, \dots, g_r. \end{cases} \quad (1)$$

We define $S_C := S_s$ for the sink s of C. For each gate g we let C_g be the sub-circuit with sink g. By $\|C\|$ we denote the size of a factorisation circuit C, which is defined to be the number of gates plus the number of wires. The number of gates in C is denoted by $|C|$.

Before defining factorised representations for CSP-instances, we introduce two special types of circuits. A factorisation circuit is *treelike* if the underlying graph is a tree, i. e., every non-sink gate has exactly one parent. Moreover, a well-defined factorisation circuit is *deterministic* if for every $\cup$-gate g the set S_g is a *disjoint* union of its child sets $S_{g_1}, \dots, S_{g_r}$. Note that while treelikeness is a syntactic property of the circuit structure, being deterministic is a semantic property that depends on the valuations of the gates. Now we are ready to state a circuit-based definition of the factorised representations defined in [32].

▶ **Definition 4.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two σ-structures.*

1. *A (deterministic)* d-representation *for $\mathcal{A}$ and $\mathcal{B}$ is a well-defined (deterministic) factorisation circuit over $V(\mathcal{A})$ and $V(\mathcal{B})$ where $S_C = \mathrm{Hom}(\mathcal{A}, \mathcal{B})$.*
2. *A (deterministic)* f-representation *is a (deterministic) d-representation with the additional restriction that the circuit is treelike.*

For brevity we will sometimes refer to d/f-representations as d/f-reps. Note that a d-rep can be more succinct than a f-rep and we will mostly deal with d-reps in this paper. However, in the proofs it will sometimes be convenient to expand out the circuit in order to make it treelike. More formally, the *transversal* $\mathrm{Trans}(C)$ of a d-rep C is the f-rep obtained from C as follows: using a top-down transversal starting at the output gate, we replace each gate g with parents $p_1, \ldots, p_d$ by d copies $g_1, \ldots, g_d$ such that the in-edges of each g_i are exactly the children of g and g_i has exactly one out-edge going to p_i. This procedure produces a treelike circuit that is well-defined/deterministic if C was well-defined/deterministic. Finally it can easily be verified that $S_{\mathrm{Trans}(C)} = S_C$.

We will often want to construct new factorised circuits from old ones. The following lemma introduces two constructions that will be particularly useful, the proof of correctness can be found in the full version of this paper.

▶ **Lemma 5.** *Let $\mathcal{A}, \mathcal{B}$ be σ-structures and C be a d-rep of $\mathrm{Hom}(\mathcal{A}, \mathcal{B})$. Let $X = \{x_1, \ldots, x_\ell\} \subseteq A$, $Y_1, \ldots, Y_\ell \subseteq B$, $\ell \geq 1$. Then one can construct the following factorised circuits in time $O(\|C\|)$.*

1. *C', such that $S_{C'} = \pi_X \mathrm{Hom}(\mathcal{A}, \mathcal{B})$ and $\|C'\| \leq \|C\|$.*
2. *C'', such that $S_{C''} = \{h \in \mathrm{Hom}(\mathcal{A}, \mathcal{B}) \mid h(x_i) \in Y_i, i \in [\ell]\}$ and $\|C''\| \leq \|C\|$.*

A special f-rep is the *flat representation*: a depth-2 circuit with a single $\cup$-gate at the top followed by a layer of $\times$-gates. Note that for any pair $\mathcal{A}$, $\mathcal{B}$, of σ-structures the flat representation has size $1 + |\mathrm{Hom}(\mathcal{A}, \mathcal{B})| \cdot (2|A| + 2)$. Intuitively, this representation corresponds to listing all homomorphisms and provides a trivial upper bound on representation size.

Deterministic d-representations have two desirable properties: they allow us to compute $|\mathrm{Hom}(\mathcal{A}, \mathcal{B})|$ in time $O(\|C\|)$ and to enumerate all homomorphisms with $O(|A|)$ delay after $O(\|C\|)$ preprocessing. Efficient counting is possible by computing bottom-up the number $|S_g|$ for each gate using multiplication on every $\times$-gate and summation on every (deterministic) $\cup$-gate. If, additionally, C is normal – i. e., no parent of a $\cup$-gate is a $\cup$-gate and the in-degree of every $\cup$- and $\times$-gate is at least 2 – Olteanu and Závodný [32, Theorem 4.11] show enumeration with $O(|A|)$ delay and *no* preprocessing is possible by sequentially enumerating the sets S_{g_i} of every child of a (deterministic) $\cup$-gate and by a nested loop to generate all combinations of child elements at $\times$-gates. The case where C is not normal is shown in [2, Theorem 7.5] and is more involved. Note that the delay is optimal in the sense that every homomorphism that is enumerated is of size $O(|A|)$.

In the other direction this means that constructing a deterministic d-rep is at least as hard as counting the number of homomorphisms. Our main theorem implies that, modulo the same assumptions as Theorem 3, the opposite is also true: for a class $\mathfrak{A}$ of structures of bounded arity there is a polynomial time algorithm that constructs a d-representation of polynomial size for two given structures $\mathcal{A} \in \mathfrak{A}$ and $\mathcal{B}$ if and only if, there is a polynomial-time algorithm that counts the number of homomorphisms between $\mathcal{A} \in \mathfrak{A}$ and $\mathcal{B}$.

Upper bounds on representation size. We have already argued that there is always a flat representation of size $O(|A| \cdot |\mathrm{Hom}(\mathcal{A}, \mathcal{B})|)$. Thus, as a corollary of [5] we get an upper bound of $O(|A| \cdot \|\mathcal{B}\|^{\rho^*(\mathcal{A})})$, where $\rho^*(\mathcal{A})$ is the *fractional edge cover number of* $\mathcal{A}$. Note that, however, the fractional edge cover number for structures of bounded arity is quite large. More precisely, if all relations in $\mathcal{A}$ have arity at most r, then $\rho^*(\mathcal{A}) \geq \frac{1}{r}|A|$.

Luckily in many cases we can do better: the results by Olteanu and Závodný in [32] imply that given a tree-decomposition of $\mathcal{A}$ of width $w-1$ we can construct a d-rep of $\mathrm{Hom}(\mathcal{A},\mathcal{B})$ of size $O(\|\mathcal{A}\|^2\|\mathcal{B}\|^w)$ in time $O(\mathrm{poly}(\|\mathcal{A}\|)\|\mathcal{B}\|^w\log(\|\mathcal{B}\|))$. Moreover the d-reps produced are normal and deterministic, meaning they allow us to perform efficient enumeration and counting. Therefore if $\mathfrak{A}$ is a class of bounded treewidth this gives us one method for solving $\#\mathrm{CSP}(\mathfrak{A},\_)$ in polynomial time. In fact, the same holds true if w is the more general *fractional hypertreewidth*, although for the case of bounded arity structures the two measure differ only by a constant. We discuss the unbounded arity case in the conclusion and, in more detail, in the full version of this paper.

5 A near-optimal bound for cliques

The goal of this section is to prove the following two theorems:

▶ **Theorem 6.** *For any $k\in\mathbb{N}$ there exist arbitrary large graphs $\mathcal{G}$ with m edges such that any f-rep of $\mathrm{Hom}(\mathcal{K}_k,\mathcal{G})$ has size $\Omega(m^{k/2}/\log^k(m))$.*

▶ **Theorem 7.** *For any $k\in\mathbb{N}$ there exist arbitrary large graphs $\mathcal{G}$ with m edges such that any d-rep of $\mathrm{Hom}(\mathcal{K}_k,\mathcal{G})$ has size $\Omega(m^{k/2}/\log^{3k-1}(m))$.*

These bounds are almost tight since the number of k-cliques in a graph with m edges is bounded by $m^{k/2}$. Moreover Theorem 7 is a crucial ingredient for proving our main theorem in Section 6. We will first prove Theorem 6 and then show how this implies the bound for d-reps.

The main idea is to exploit a correspondence between the structure of a (simple) graph $\mathcal{G}$ and f-reps of $\mathrm{Hom}(\mathcal{K}_k,\mathcal{G})$. To illustrate this consider the case $k=2$, where $V(\mathcal{K}_2)=\{x_1,x_2\}$ and each $h\in\mathrm{Hom}(\mathcal{K}_2,\mathcal{G})$ corresponds to an edge of $\mathcal{G}$. Let C be a f-rep of $\mathrm{Hom}(\mathcal{K}_2,\mathcal{G})$, with $\times$-gates $g_1,\dots,g_\alpha$. Each g_i has two children g_i^1, g_i^2 with $\mathrm{dom}(g_i^1)=x_1$ and $\mathrm{dom}(g_i^2)=x_2$. Since no $\times$-gates can occur in $C_{g_i^1}$ or $C_{g_i^2}$, $S^1_{g_i}=\{\{x_1\mapsto a\}\mid a\in A_i\}$ and $S_{g_i^2}=\{\{x_2\mapsto b\}\mid b\in B_i\}$ for some disjoint $A_i,B_i\subseteq V(\mathcal{G})$. Therefore $A_i\times B_i$ is a complete bipartite subgraph of $\mathcal{G}$. Since the ancestors of each $\times$-gate can only be $\cup$-gates, each f-rep of $\mathrm{Hom}(\mathcal{K}_2,\mathcal{G})$ corresponds to a set of complete bipartite subgraphs that cover every edge of $\mathcal{G}$. Finding such sets and investigating their properties has been studied in various contexts, for example see [12, 18, 22, 31].

Moreover, the number of input gates appearing in C is $\sum_{i=1}^{\alpha}|A_i|+|B_i|$ and so finding a f-rep of $\mathrm{Hom}(\mathcal{K}_2,\mathcal{G})$ of minimum size corresponds to minimising the sum of the sizes of the partitions in our complete bipartite covering of $\mathcal{G}$, call this the cost of the covering. Proving Theorem 6 for the case $k=2$, corresponds to finding graphs where every covering of the edges by complete bipartite subgraphs has high cost. This is a problem investigated by Chung et al. in [12], where one key idea is that if a graph contains no large complete bipartite subgraphs and a large number of edges then the cost of any cover must be high. We deploy this idea in our more general context. This motivates the following lemma, which follows from a simple probabilistic argument.

▶ **Lemma 8.** *For every $k\in\mathbb{N}$ there exists some $c_k\in\mathbb{R}^+$ such that for every sufficiently large integer n there is a graph $\mathcal{G}$ with n vertices, such that*

1. *$\mathcal{G}$ has $m\geq\frac{1}{8}n^2$ edges,*
2. *$\mathcal{G}$ contains no complete bipartite subgraph $\mathcal{K}_{a,a}$ for $a\geq 3\log(n)$, and*
3. *the number of k-cliques in $\mathcal{G}$ is at least c_kn^k.*

Proof. We first prove the following claim.

▷ Claim 9. Let $\mathcal{G}_n$ be a random graph on n vertices with edge probability $\frac{1}{2}$. Let $\epsilon > 0$. Then for any $a = a(n) \geq (2+\epsilon)\log(n)$,

$$P_a := \mathbb{P}(\mathcal{G}_n \text{ has } \mathcal{K}_{a,a} \text{ as a subgraph}) \to 0 \text{ as } n \to \infty.$$

Proof of Claim. By the union bound and the bound on a we get

$$P_a \leq \binom{n}{a}^2 2^{-a^2} \leq n^{2a} 2^{-a^2} = 2^{2a\log(n) - a^2} \leq 2^{-(\epsilon^2+2\epsilon)\log^2 n}.$$ ◁

Now let $\mathcal{G}_n$ be as above, $s = s(k) := \binom{k}{2} + 1$ and p be the probability that such a graph has at least $\binom{n}{k}2^{-s}$ k-cliques. The expected number of k-cliques in $\mathcal{G}_n$ is $\binom{n}{k}2^{-\binom{k}{2}}$. Therefore,

$$\binom{n}{k}2^{-\binom{k}{2}} \leq \binom{n}{k}2^{-s}(1-p) + \binom{n}{k}p$$

and so $p \geq 1/(2^s - 1)$. Moreover, by the Chernoff bound, (1) from the statement of the Lemma fails only with exponentially small probability. By Claim 9 there must exist a $\mathcal{G}$ satisfying (1), (2), and (3) for sufficiently large n. ◀

Equipped with Lemma 8 we are already in a position to prove Theorem 6.

Proof of Theorem 6. Let $\mathcal{G}$ be an n-vertex graph provided by Lemma 8 and suppose that C is a f-rep for $\mathcal{K}_k$ and $\mathcal{G}$. If $\max_{x \in \mathrm{dom}(g)} |\{a \mid h(x) = a, h \in S_g\}| \leq 3\log(n)$ for a gate g we say that g is *small*. Otherwise we say g is *big*. Note that a $\times$-gate cannot have two big children g_1 and g_2 because otherwise there would be $x_1 \in \mathrm{dom}(g_1)$ and $x_2 \in \mathrm{dom}(g_2)$ such that

$$\{a \mid h(x_1) = a, h \in S_{g_1}\} \times \{a \mid h(x_2) = a, h \in S_{g_2}\}$$

forms a complete bipartite subgraph with partitions bigger than $3\log n$ in $\mathcal{G}$, contradicting (2) from Lemma 8.

If g is small, then C_g represents $|S_g| \leq 3^{|\mathrm{dom}(g)|}\log^{|\mathrm{dom}(g)|}(n)$ homomorphisms. We claim that for any gate g of C, $|S_g| \leq |C_g| \cdot 3^{|\mathrm{dom}(g)|}\log^{|\mathrm{dom}(g)|}(n)$. Clearly this holds for input gates. We can therefore induct bottom up on C. Suppose our claim holds for all children $g_1, \ldots, g_r$ of some gate g. If g is a $\times$-gate then we know at most one of the g_i is big, say g_1. Define $b := \sum_{i=2}^{r} |\mathrm{dom}(g_i)|$. Then,

$$|S_g| = \prod_{i=1}^{r} |S_{g_i}| \leq |C_{g_1}| \cdot 3^{|\mathrm{dom}(g_1)|}\log^{|\mathrm{dom}(g_1)|}(n) \cdot 3^b \log^b(n) \leq |C_g| \cdot 3^{|\mathrm{dom}(g)|}\log^{|\mathrm{dom}(g)|}(n),$$

The $\cup$-gate case follows immediately from the induction hypothesis because the circuit is treelike so if g has children $g_1, \ldots, g_r$ then $|C_g| = 1 + \sum_{i=1}^{r} |C_{g_i}|$.

From the claim we infer in particular that $|\mathrm{Hom}(\mathcal{K}_k, \mathcal{G})| = |S_s| \leq |C| \cdot 3^k \log^k(n)$ for the sink s of C. By (3) from Lemma 8 it follows that $|C| \geq c_k n^k/(3^k \log^k(n))$ which, combined with (1) from Lemma 8, implies the claimed result. ◀

We now transfer this bound to d-reps, by showing that, for the same graphs used above, any d-rep cannot be much smaller than the smallest f-rep.

Proof of Theorem 7. Let $\mathcal{G}$ be an n-vertex graph provided by Lemma 8 as above and C a d-rep of $\mathrm{Hom}(\mathcal{K}_k, \mathcal{G})$ with sink s. If a gate has out-degree of more than one we call it a *definition*. As in the proof of Theorem 6, if $\max_{x \in \mathrm{dom}(g)} |\{a \mid h(x) = a, h \in S_g\}| \leq 3\log(n)$ for a gate g we say that g is *small*. Otherwise we say g is *big*.

Our strategy is to convert C into an equivalent f-rep that is not much bigger than C. For ease of analysis and exposition we will do this by first eliminating all small definitions and then all big definitions. First if s is small replace the whole circuit with its equivalent flat representation. Otherwise, we mark all small gates g that have a big parent and compute the equivalent flat representation F_g of C_g. Since every unmarked small gate is a descendant of some marked gate, we can now safely delete all unmarked small gates. Afterwards we consider every wire between a marked gate g and one of its big parents p and replace it by a copy of F_g as input to p. We obtain an equivalent circuit $\hat{C}$ where every small gate has only one parent. The size (number of gates plus number of wires) increases only by a factor determined by the maximum size of a flat representation:

▷ Claim 10. $\|\hat{C}\| \leq \|C\| \cdot (2k+3)3^k \log^k(n)$.

When we try and eliminate big definitions one challenge is that if g is big, then $\|\hat{C}_g\|$ can be large and so making lots of copies of it could blow up the size of our circuit. To overcome this we introduce the notion of an *active parent*. We then show that non-active parents are effectively redundant and that there can't be too many active ones, which allows us to construct an equivalent treelike circuit of the appropriate size.

So let g be a definition with parents $p_1, \ldots, p_\alpha$, $\alpha > 1$, and suppose there is a unique path from p_i to the sink s for every i. Then for every gate v on the unique path from g to s which passes through p_i, we inductively define a set of (partial) homomorphisms $A_i^v = A_i^v(g)$ as follows, where $\hat{v}$ refers to the child of v also lying on this path.

- $A_i^g := S_g$,
- if v is a $\cup$-gate $A_i^v := A_i^{\hat{v}}$,
- otherwise v is a $\times$-gate with children $u_1, \ldots, u_{r-1}, \hat{v}$ and $A_i^v := \{h_1 \cup \ldots \cup h_r \mid h_i \in S_{u_i},\, i \in [r-1],\, h_r \in A_i^{\hat{v}}\}$.

Write $A_i := A_i^s$, intuitively this is the set of homomorphisms that the wire from g to p_i contributes to. We say that a parent p_i of g is *active* if $A_i \not\subseteq \cup_{j \neq i} A_j$. Now using a top-down traversal starting at the output gate of $\hat{C}$, we replace each gate with active parents $p_1, \ldots, p_\beta$, by β copies $g_1, \ldots, g_\beta$ such that the children of each g_i are exactly the children of g and g_i has exactly one out-edge going to p_i. At each stage we also clean-up the circuit by iteratively deleting all gates which have no incoming wires, as well as all the wires originating from such gates. We can think of this process as constructing a slimmed down version of the traversal, where at each stage we only keep wires going to active parents. Call the resulting circuit C'.

We first note that this process is well-defined, as there is a unique path from the sink to itself and since whenever we visit a gate we have already visited all of its parents. Moreover, by construction this results in a treelike circuit. In the next claim we bound the size of C' and show it is indeed an equivalent circuit. The idea is that firstly a gate cannot have too many active parents, as otherwise we would get a large biclique in $\mathcal{G}$ which is ruled out by Lemma 8, and secondly that since only active parents contribute new homomorphisms we really do get an equivalent circuit, see the full version of this paper for details.

▷ Claim 11. C' is a f-rep of $\mathrm{Hom}(\mathcal{K}_k, \mathcal{G})$ and $\|C'\| \leq 3^k \log^{k-1}(n)\|\hat{C}\|$.

Pulling everything together we get that

$$\|C\| \underset{\text{(Claim 10)}}{\geq} \frac{\|\hat{C}\|}{(2k+3)3^k \log^k(n)} \underset{\text{(Claim 11)}}{\geq} \frac{\|C'\|}{(2k+3)3^{2k} \log^{2k-1}(n)} = \Omega\left(\frac{m^{k/2}}{\log^{3k-1}(m)}\right),$$

where the final equality follows by Theorem 6 since C' is a f-rep of $\mathrm{Hom}(\mathcal{K}_k, \mathcal{G})$. ◀

6 The representation dichotomy for structures of bounded arity

In this section we lift the lower bound for cliques to all classes of graphs with unbounded treewidth. We first introduce a notion of reductions between representations and show that having lower bounds for all graphs of unbounded treewidth immediately implies our main dichotomy theorem for bounded-arity structures.

Afterwards, we introduce minor and almost-minor reductions and use them to obtain a lower bound for representing homomorphisms from large grids and from graphs having large grids as a minor. The superpolynomial representation lower bound for all graph classes with unbounded treewidth then follows from the excluded grid theorem.

6.1 Reductions between representations

In order to define reductions between representations we fix some notation. For two structures $\mathcal{A}$ and $\mathcal{B}$ we let $\mathsf{D}(\mathcal{A},\mathcal{B})$ be the set of all d-representations of $\mathrm{Hom}(\mathcal{A},\mathcal{B})$ and $\mathsf{d}(\mathcal{A},\mathcal{B}) = \min_{C\in\mathsf{D}(\mathcal{A},\mathcal{B})} \|C\|$ be the size of the smallest such representation.

For a class $\mathfrak{C}$ of structures the function $\mathsf{d}_{\mathcal{A},\mathfrak{C}}\colon \mathbb{N}\to\mathbb{N}$ expresses the required size of a d-representation of homomorphisms between $\mathcal{A}$ and $\mathcal{C}\in\mathfrak{C}$ in terms of the size m of $\mathcal{C}$, i. e., $\mathsf{d}_{\mathcal{A},\mathfrak{C}}(m) = \max_{\{\mathcal{C}\in\mathfrak{C}\,:\,\|\mathcal{C}\|\le m\}} \mathsf{d}(\mathcal{A},\mathcal{C})$. We write $\mathsf{d}_{\mathcal{A}}$ as an abbreviation for $\mathsf{d}_{\mathcal{A},\mathfrak{C}}$ when $\mathfrak{C}$ is the class of all structures. Translated to this notation, [32] showed that $\mathsf{d}_{\mathcal{A}} = O(m^{\mathrm{tw}(\mathcal{A})+1})$, whereas Theorem 7 states the lower bound $\mathsf{d}_{K_k} = \Omega(m^{k/2}/\log^{3k-1}(m))$. We also write, for a signature σ, $\mathfrak{C}_\sigma$ to denote the class of all σ-structures.

The main goal of this section is to prove, for some increasing function f, a lower bound of the form $\mathsf{d}_{\mathcal{A}} = \Omega(m^{f(\mathrm{tw}(\mathcal{A}))/\,\mathrm{ar}(\mathcal{A})})$ for every structure $\mathcal{A}$, which immediately implies our main theorem. To achieve this we use reductions with our k-clique lower bound as a starting point. Suppose we already have a lower bound on $\mathsf{d}_{\mathcal{A},\mathfrak{C}}$ for a class $\mathfrak{C}$ of arbitrarily large hard instances (implying a lower bound on $\mathsf{d}_{\mathcal{A}}$), then we can use the following reduction *from* $\mathcal{A}$ *to* $\mathcal{B}$ *via* $\mathfrak{C}$ to obtain a lower bound on $\mathsf{d}_{\mathcal{B}}$.

▶ **Definition 12.** *Let $\mathcal{A}$ be a σ-structure and let $\mathfrak{C}$ be a class of σ-structures. Let $\mathcal{B}$ be a σ'-structure and $c\colon\mathbb{R}^+\to\mathbb{R}^+$ be a strictly increasing function. Then a* c-reduction *from $\mathcal{A}$ to $\mathcal{B}$ via $\mathfrak{C}$ is a pair $(\phi,(\psi_{\mathcal{C}})_{\mathcal{C}\in\mathfrak{C}})$, where $\phi\colon\mathfrak{C}\to\mathfrak{C}_{\sigma'}$ and $\psi_{\mathcal{C}}\colon\mathsf{D}(\mathcal{B},\phi(\mathcal{C}))\to\mathsf{D}(\mathcal{A},\mathcal{C})$ such that:*

1. *for every $n\in\mathbb{N}$ there is a $\mathcal{C}\in\mathfrak{C}$ such that $\|\phi(\mathcal{C})\|\ge n$,*
2. *$\|\phi(\mathcal{C})\|\le c(\|\mathcal{C}\|)$ for all $\mathcal{C}\in\mathfrak{C}$, and*
3. *$\|\psi_{\mathcal{C}}(C)\|\le\|C\|$ for every structure $\mathcal{C}\in\mathfrak{C}$ and circuit $C\in\mathsf{D}(\mathcal{B},\phi(\mathcal{C}))$.*

If $c(m)=\alpha m$ for some $\alpha\in\mathbb{R}^+$, we say we have a linear reduction.

▶ **Lemma 13.** *Suppose there is a c-reduction $(\phi,(\psi_{\mathcal{C}})_{\mathcal{C}\in\mathfrak{C}})$ from $\mathcal{A}$ to $\mathcal{B}$ via $\mathfrak{C}$, let $\mathfrak{D}=\{\phi(\mathcal{C})\mid\mathcal{C}\in\mathfrak{C}\}$ be the image of ϕ. Then $\mathsf{d}_{\mathcal{B},\mathfrak{D}} = \Omega(\mathsf{d}_{\mathcal{A},\mathfrak{C}}\circ\lfloor c^{-1}\rfloor)$.*

Proof. Fix $m\in\mathbb{N}$, where $m\ge\min_{\{\mathcal{C}\in\mathfrak{C}\}}\|\mathcal{C}\|$. Let $\mathcal{C}\in\mathfrak{C}$ with $\|\mathcal{C}\|\le m$. Then $\psi_{\mathcal{C}}$ witnesses that $\mathsf{d}(\mathcal{A},\mathcal{C})\le\mathsf{d}(\mathcal{B},\phi(\mathcal{C}))$. Also $\|\phi(\mathcal{C})\|\le c(m)$, since c is an increasing function. So $\mathsf{d}_{\mathcal{B},\mathfrak{D}}(c(m)) = \max_{\{\mathcal{C}\,:\,\|\phi(\mathcal{C})\|\le c(m)\}}\mathsf{d}(\mathcal{B},\phi(\mathcal{C}))\ge\max_{\{\mathcal{C}\,:\,\|\mathcal{C}\|\le m\}}\mathsf{d}(\mathcal{A},\mathcal{C}) = \mathsf{d}_{\mathcal{A},\mathfrak{C}}(m)$. Since $\mathfrak{C}$ and $\mathfrak{D}$ contain arbitrarily large structures, the asymptotic bound from the lemma follows. ◀

We start illustrating the power of these reductions by making two simplifications. First, we reduce the general problem of representing homomorphisms to representing homomorphisms that respect a partition. Second, we further reduce to graph homomorphisms that respect a partition. All proofs from this subsection can be found in the full version of the paper.

For the first reduction we need the notion of the *individualisation* of a σ-structure $\mathcal{A}$, which is obtained from $\mathcal{A}$ by giving every element of the universe a distinct color. More precisely, we extend the vocabulary σ with unary relations (= colours) $\sigma_A = \{P_a \colon a \in A\}$ and let $\mathcal{A}^{\mathrm{id}}$ be the $\sigma \cup \sigma_A$-expansion of $\mathcal{A}$ by adding $P_a^{\mathcal{A}^{\mathrm{id}}} = \{a\}$.

▶ **Lemma 14.** *Let $\mathcal{A}$ be a σ-structure and let $\mathfrak{C}$ be the class of all $\sigma \cup \sigma_A$-structures where $\{P_a^{\mathcal{C}} \mid a \in A\}$ is a partition of the universe. Then $\mathsf{d}_{\mathcal{A}} = \Omega(\mathsf{d}_{\mathcal{A}^{id},\mathfrak{C}})$.*

We call structures and (vertex-coloured) graphs *individualised* if every vertex has a distinct colour. In the next lemma we reduce from individualised structures to individualised graphs. Recall the definition of the Gaifman graph $\mathcal{G}_{\mathcal{A}}$ from the preliminaries.

▶ **Lemma 15.** *Let $\mathcal{A}$ be an individualised structure and $\mathcal{G}_{\mathcal{A}}^{id}$ the individualisation of its Gaifman graph. Let $\mathfrak{C}$ be the class of all structures $\mathcal{C}$ where $\{P_a^{\mathcal{C}} \mid a \in A\}$ is a partition of its universe and $\mathfrak{H}$ be the class of all vertex-coloured graphs $\mathcal{H}$ where $\{P_a^{\mathcal{H}} \mid a \in A\}$ is a partition of its vertex set. Then $\mathsf{d}_{\mathcal{A},\mathfrak{C}}(m) = \Omega\big((\mathsf{d}_{\mathcal{G}_{\mathcal{A}}^{id},\mathfrak{H}}(m))^{2/\operatorname{ar}(\mathcal{A})} \big)$.*

Taking both lemmas into account, we can now focus on individualised graphs $\mathcal{G}$ on the left-hand side and on graphs $\mathcal{H}$ with the corresponding colouring $\{P_a^{\mathcal{H}} \mid a \in V(\mathcal{G})\}$ that partitions its vertex set on the right-hand side. We call such graphs *$V(\mathcal{G})$-partitioned graphs*. However we would also like to deploy our lower bound from Section 5; the next lemma allows to transfer this lower bound to individualised structures.

▶ **Lemma 16.** *Let $\mathcal{G}$ be a graph and $\mathfrak{C}$ be the class of all $V(\mathcal{G})$-partitioned graphs. Then $\mathsf{d}_{\mathcal{G}^{id},\mathfrak{C}} = \Omega(\mathsf{d}_{\mathcal{G}})$.*

6.2 Minor reductions

In this subsection we show that we can reduce $\mathcal{G}'$ to $\mathcal{G}$ if $\mathcal{G}$ is a minor of $\mathcal{G}'$. We start by illustrating how to handle edge contractions via an example.

▶ **Example 17** (Reduction from 4-cycle to 3-cycle)**.** Consider the 3-cycle $\mathcal{K}_3$ on vertices x_1, x_2, x_3, which is a minor of the 4-cycle $\mathcal{C}_4$ on vertices x_1, x_2, x_3, x_4 by contracting one edge $\{x_4, x_1\}$. We show that we can lift the lower bound for $\mathcal{K}_3^{\mathrm{id}}$ (Theorem 7 + Lemma 16) to $\mathcal{C}_4^{\mathrm{id}}$ (and hence $\mathcal{C}_4$ by Lemma 14) by a simple linear reduction from $\mathcal{K}_3^{\mathrm{id}}$ to $\mathcal{C}_4^{\mathrm{id}}$ via the class of all $\{x_1, x_2, x_3\}$-partitioned graphs. Let $\mathcal{H}$ be a $\{x_1, x_2, x_3\}$-partitioned graph. We define the $\{x_1, x_2, x_3, x_4\}$-partitioned graph $\mathcal{H}' = \phi(\mathcal{H})$ by $P_x^{\mathcal{H}'} := P_x^{\mathcal{H}}$ for $x \in \{x_1, x_2, x_3\}$, $P_{x_4}^{\mathcal{H}'} := \{\widehat{v} \mid v \in P_{x_1}^{\mathcal{H}}\}$ and $E(\mathcal{H}') =$

$$
\begin{aligned}
&\big\{\{v, \widehat{v}\} \mid v \in P_{x_1}^{\mathcal{H}}\big\} \\
\cup &\big\{\{v, w\} \mid v \in P_{x_1}^{\mathcal{H}},\, w \in P_{x_2}^{\mathcal{H}},\, \{v, w\} \in E(\mathcal{H})\big\} \\
\cup &\big\{\{v, w\} \mid v \in P_{x_2}^{\mathcal{H}},\, w \in P_{x_3}^{\mathcal{H}},\, \{v, w\} \in E(\mathcal{H})\big\} \\
\cup &\big\{\{v, \widehat{w}\} \mid v \in P_{x_3}^{\mathcal{H}},\, w \in P_{x_1}^{\mathcal{H}},\, \{v, w\} \in E(\mathcal{H})\big\}.
\end{aligned}
$$

Note that the size of $\mathcal{H}'$ is linear in the size of $\mathcal{H}$. The construction ensures that any mapping $h' \colon \{x_1, \dots, x_4\} \to V(\mathcal{H}')$ is a homomorphism from $\mathcal{C}_4^{\mathrm{id}}$ to $\mathcal{H}'$ if, and only if, $h'(x_4) = \widehat{h'(x_1)}$ and $h(x_i) := h'(x_i)$, for $i \in [3]$, is a homomorphism from $\mathcal{K}_3^{\mathrm{id}}$ to $\mathcal{H}$. Therefore, $\operatorname{Hom}(\mathcal{K}_3^{\mathrm{id}}, \mathcal{H}) = \pi_{\{x_1,x_2,x_3\}} \operatorname{Hom}(\mathcal{C}_4^{\mathrm{id}}, \mathcal{H}')$ and a representation C' of $\operatorname{Hom}(\mathcal{K}_3^{\mathrm{id}}, \mathcal{H})$ can be obtained from a representation C of $\operatorname{Hom}(\mathcal{C}_4^{\mathrm{id}}, \mathcal{H}')$ by Lemma 5 which, moreover, guarantees that $\|C'\| \leq \|C\|$. Therefore we do have a linear reduction from $\mathcal{C}_4^{\mathrm{id}}$ to $\mathcal{K}_3^{\mathrm{id}}$. It follows that $\mathsf{d}_{\mathcal{C}_4}(m) = \Omega(\mathsf{d}_{\mathcal{C}_4^{\mathrm{id}},\mathfrak{C}}(m)) = \Omega(\mathsf{d}_{\mathcal{K}_3^{\mathrm{id}},\mathfrak{H}}(m)) = \Omega(\mathsf{d}_{\mathcal{K}_3}(m)) = \Omega(m^{3/2}/\log^7(m))$,

where $\mathfrak{C}$ is the class of $V(\mathcal{C}_4^{\mathrm{id}})$-partitioned graphs and $\mathfrak{H}$ is the class of $V(\mathcal{K}_3^{\mathrm{id}})$-partitioned graphs. The first equality follows by Lemma 14, the second by Lemma 13, the third by Lemma 16 and the last by Theorem 7.

So to handle edge contractions we take the partitioned hard right-hand side instance and "re-introduce" the edge $\{x, y\}$ contracted to x by copying P_x to P_y and adding a perfect matching between the two partitions P_x and P_y. Handling edge deletions is even simpler: suppose that $\{x, y\}$ is deleted from $\mathcal{G}'$ to $\mathcal{G}$ and we want to reduce $\mathcal{G}'$ to $\mathcal{G}$. Then we take a partitioned hard instance for $\mathcal{G}$ and just introduce the complete bipartite graph between the partitions P_x and P_y; this may square the size of the graph. Since the sets of (partition-respecting) homomorphisms are the same for both instances, we do not even have to modify the representations in the reduction. The next lemma summarises these findings. Its proof is omitted as it is subsumed by Lemma 22.

▶ **Lemma 18.** *Let $\mathcal{G}_X, \mathcal{G}_Y$ be graphs with vertex sets X and Y respectively such that $\mathcal{G}_X$ is a minor of $\mathcal{G}_Y$. Let $\mathfrak{H}$ be the class of all $V(\mathcal{G}_X)$-partitioned graphs and $\mathfrak{H}'$ the class of all $V(\mathcal{G}_Y)$-partitioned graphs. Then there is a c-reduction $(\phi, (\psi_{\mathcal{H}})_{\mathcal{H}\in\mathfrak{H}})$ from $\mathcal{G}_Y^{id}$ to $\mathcal{G}_X^{id}$ via $\mathfrak{H}$ with $\phi(\mathfrak{H}) \subseteq \mathfrak{H}'$ and $c(m) = O(m^2)$.*

This yields together with Lemmas 13, 14 and 16 along with Theorem 7 the following corollary.

▶ **Corollary 19.** *If $\mathcal{G}$ has $\mathcal{K}_k$ as a minor, then $\mathsf{d}_{\mathcal{G}} = \Omega(m^{k/4}/\log^{(3k-1)/2}(m))$.*

6.3 Relaxation of the minor condition

Every graph having $\mathcal{K}_k$ as a minor has treewidth at least $k-1$, so Corollary 19 provides the desired lower bound of Theorem 1 for certain large-treewidth graphs. However, there are graphs of large treewidth that do not have a large clique as a minor. Instead, the excluded grid theorem [34] and its more efficient version [11] tells us that graphs of large treewidth always have a large $k \times k$-grid as a minor.

▶ **Theorem 20** ([11])**.** *There is a polynomial function $w : \mathbb{N} \to \mathbb{N}$ such that for every k the $(k \times k)$-grid is a minor of every graph of treewidth at least $w(k)$.*

Thus, in order to prove Theorem 1 it suffices to combine Lemma 18 with a lower bound for grid graphs. We cannot reduce immediately to our k-clique lower bound, as the grid does not have a $\mathcal{K}_k$ minor for $k \geq 5$. However, the complete graph $\mathcal{K}_k$ is "almost a minor" of $\mathcal{G}_{2k-2}$ for the following notion of *almost minor* that is good enough to prove a variant of Lemma 18.

▶ **Definition 21.** *For two graphs $\mathcal{G}_X$, $\mathcal{G}_Y$ with vertex sets $X = V(\mathcal{G}_X)$ and $Y = V(\mathcal{G}_Y)$ we say that a map $M\colon Y \to 2^X$ is* almost minor *if the following conditions hold:*

1. *for every $y \in Y$, $|M(y)| \in \{1, 2\}$;*
2. *for every $x \in X$ there is a $y \in Y$ s.t. $M(y) = \{x\}$ and for every x, x' adjacent in $\mathcal{G}_X$ there exists y, y' adjacent in $\mathcal{G}_Y$ such that $M(y) = \{x\}$ and $M(y') = \{x'\}$;*
3. *for each $x \in X$, $\{y\colon x \in M(y)\}$ is connected in $\mathcal{G}_Y$ and*
4. *if $M(y) = \{x, x'\}$ with $x \neq x'$ and y' is adjacent to y in $\mathcal{G}_Y$, then $M(y') = \{x\}$ or $M(y') = \{x'\}$.*

If such a map exists we say $\mathcal{G}_X$ is an almost minor of $\mathcal{G}_Y$.

For the special case when $|M(y)| = 1$ for all y, M is a *minor map* and G_X is a minor of G_Y. The motivation for this definition is that whilst grids are planar, large cliques are not and so we introduce "junctions", i.e. nodes y such that $M(y) = \{x_1, x_2\}$ which allows

$\{v \mid x_i \in M(v)\}$, $i \in \{1,2\}$ to intersect in a controlled way, see Figure 2. We should also observe here that this notion is related to Marx's notion of an *embedding* [27].[1] Now we can state our reduction lemma for almost minors, which extends Lemma 18.

▶ **Lemma 22.** *Let $\mathcal{G}_X$, $\mathcal{G}_Y$ be the graphs with vertex sets X and Y, respectively, such that $\mathcal{G}_X$ is an almost minor of $\mathcal{G}_Y$. Let $\mathfrak{H}$ be the class of all X-partitioned graphs and $\mathfrak{H}'$ be the class of all Y-partitioned graphs, then there is a c-reduction $(\phi, (\psi_{\mathcal{H}})_{\mathcal{H}\in\mathfrak{H}})$ from $\mathcal{G}_Y^{id}$ to $\mathcal{G}_X^{id}$ via $\mathfrak{H}$ with $\phi(\mathfrak{H}) \subseteq \mathfrak{H}'$ and $c = O(m^2)$.*

Proof. We start by defining the Y-partitioned graph $\mathcal{H}^* = \phi(\mathcal{H})$ for an arbitrary X-partitioned graph $\mathcal{H}$. To define the partitions, we consider two cases: if $M(y) = \{x\}$, we let $P_y^{\mathcal{H}^*} := \{v_a^y \mid a \in P_x^{\mathcal{H}}\}$ and if $M(y) = \{x, x'\}$, then $P_y^{\mathcal{H}^*} := \{v_{\{a,b\}}^y \mid a \in P_x^{\mathcal{H}}, b \in P_{x'}^{\mathcal{H}}\}$. For every edge $\{y, y'\} \in E(\mathcal{G}_Y)$ we define the edge set $E_{\{y,y'\}}$ between the partitions $P_y^{\mathcal{H}^*}$ and $P_{y'}^{\mathcal{H}^*}$ by the following exhaustive cases:

1. if $M(y) = M(y') = \{x\}$: $E_{\{y,y'\}} := \left\{\{v_a^y, v_a^{y'}\} \mid a \in P_x^{\mathcal{H}}\right\}$
2. if $M(y) = \{x\}$, $M(y') = \{x'\}$, and $\{x, x'\} \in E(\mathcal{G}_X)$:
   $E_{\{y,y'\}} := \left\{\{v_a^y, v_b^{y'}\} \mid a \in P_x^{\mathcal{H}}, b \in P_{x'}^{\mathcal{H}}, \{a, b\} \in E(\mathcal{H})\right\}$
3. if $M(y) = \{x\}$, $M(y') = \{x'\}$, $x \neq x'$, and $\{x, x'\} \notin E(\mathcal{G}_X)$:
   $E_{\{y,y'\}} := \left\{\{v_a^y, v_b^{y'}\} \mid a \in P_x^{\mathcal{H}}, b \in P_{x'}^{\mathcal{H}}\right\}$
4. if $M(y) = \{x\}$ and $M(y') = \{x, x'\}$: $E_{\{y,y'\}} := \left\{\{v_a^y, v_{\{a,b\}}^{y'}\} \mid a \in P_x^{\mathcal{H}}, b \in P_{x'}^{\mathcal{H}}\right\}$

Finally, we set $E(\mathcal{H}^*) := \bigcup_{e \in E(\mathcal{G}_Y)} E_e$ and note that $\|\mathcal{H}^*\| = O(\|\mathcal{H}\|^2)$. For every homomorphism h from $\mathcal{G}_X^{\mathrm{id}}$ to $\mathcal{H}$ we define the mapping $h^*\colon Y \to V(\mathcal{H}^*)$ by

$$h^*(y) := \begin{cases} v^y_{h(x)}, & \text{if } M(y) = \{x\} \\ v^y_{\{h(x),h(x')\}}, & \text{if } M(y) = \{x, x'\} \end{cases}$$

The next claim provides the key property of our construction: h^* is a homomorphism from $\mathcal{G}_Y^{\mathrm{id}}$ to $\mathcal{H}^*$ and every homomorphism from $\mathcal{G}_Y^{\mathrm{id}}$ to $\mathcal{H}^*$ has this form, see the full version of this paper for a proof.

▷ Claim 23. $\mathrm{Hom}(\mathcal{G}_Y^{\mathrm{id}}, \mathcal{H}^*) = \{h^* \colon h \in \mathrm{Hom}(\mathcal{G}_X^{\mathrm{id}}, \mathcal{H})\}$

We finish the lemma by defining the mapping $\psi_{\mathcal{H}}$ that transforms any d-representation for $\mathrm{Hom}(\mathcal{G}_X^{\mathrm{id}}, \mathcal{H})$ into a d-representation for $\mathrm{Hom}(\mathcal{G}_Y^{\mathrm{id}}, \mathcal{H}^*)$. For each $x \in X$ we fix one $y_x \in Y$ such that $M(y_x) = \{x\}$ (those vertices exist by the definition of an almost minor map). Then we apply Lemma 5 and obtain a d-representation of $\pi_{\{y_x \colon x \in X\}} \mathrm{Hom}(\mathcal{G}_Y^{\mathrm{id}}, \mathcal{H}^*)$. After renaming every y_x to x and every v_a^y to a in the input labels of this circuit, we get a d-representation of $\mathrm{Hom}(\mathcal{G}_X^{\mathrm{id}}, \mathcal{H})$. ◀

With the following lemma we have everything in hand to proof our main theorem.

▶ **Lemma 24.** *For every k, $\mathcal{K}_k$ is an almost minor of $\mathcal{G}_{2k-2}$.*

[1] In particular the definition of a *depth-2 embedding* can be obtained from our definition of an almost minor by the following modifications. First remove clause (4). Second replace (2) with the following condition: for every $x \in X$ there is a $y \in Y$ s.t. $x \in M(y)$ and for every x, x' adjacent in $\mathcal{G}_X$ there exists *either* y, y' adjacent in $\mathcal{G}_Y$ such that $x \in M(y)$ and $x' \in M(y')$ *or* there exists y such that $\{x, x'\} \subseteq M(y)$. If we also remove clause (1) we get the general definition of an embedding.

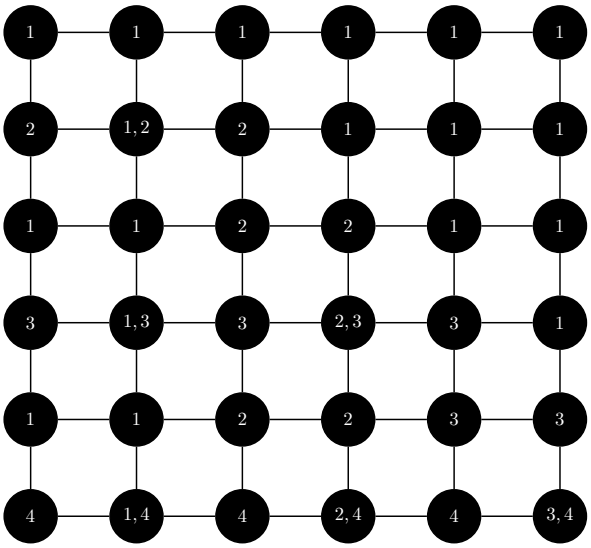


Figure 2 Construction from Lemma 24 for the case $k = 4$. The node in the ith row and jth column is labelled by the $\{a \mid u_a \in M(v_{i,j})\}$.

Proof of Lemma 24. Set

$$X := V(\mathcal{K}_k) = \{u_i \mid i \in [k]\},$$
$$Y := V(\mathcal{G}_{2k-2}) = \{v_{i,j} \mid i, j \in [2k-2]\},$$

where $v_{i,j}$ is the vertex in the ith row and jth column of the grid. Define $M \colon Y \to 2^X$ as follows:

1. if $j - 1 > i$, $M(v_{i,j}) = \{u_1\}$,
2. otherwise if $i \geq j - 1$ then:
   a. if i and j are both odd, $M(v_{i,j}) = \{u_{(j+1)/2}\}$,
   b. if i is odd and is j even, $M(v_{i,j}) = \{u_{j/2}\}$,
   c. if i is even and j is odd, $M(v_{i,j}) = \{u_{(i+2)/2}\}$,
   d. if i and j are both even, $M(v_{i,j}) = \{u_{(i+2)/2}, u_{j/2}\}$.

See Figure 2 for the case $k = 4$. It is easy to see that this map is almost minor, see the full version of this paper for a proof. ◀

Proof of Theorem 1. Let $\mathfrak{A}$ have unbounded treewidth. Then for every k there exists $\mathcal{B}_k \in \mathfrak{A}$ of treewidth at least $w(k)$. Then the Gaifman graph of $\mathcal{B}_k$, $\mathcal{G}_{\mathcal{B}_k}$ also has treewidth at least $w(k)$. By Theorem 20, $\mathcal{G}_{\mathcal{B}_k}$ has $\mathcal{G}_k$ as a minor. Since by Lemma 24, $\mathcal{K}_{(k+2/2)}$ is an almost minor of $\mathcal{G}_k$ we have:

$$\begin{aligned}
\mathsf{d}_{\mathcal{B}_k}(m) &\underset{\text{(Lemma 14)}}{=} \Omega\left(\mathsf{d}_{\mathcal{B}_k^{\mathrm{id}},\mathfrak{C}}(m)\right) \\
&\underset{\text{(Lemma 15)}}{=} \Omega\left((\mathsf{d}_{\mathcal{G}^{\mathrm{id}}_{\mathcal{B}_k},\mathfrak{H}}(m))^{2/\operatorname{ar}(\mathcal{B}_k)}\right) \\
&\underset{\text{(Lemma 18)}}{=} \Omega\left((\mathsf{d}_{\mathcal{G}^{\mathrm{id}}_{k},\mathfrak{H}'}(m))^{1/\operatorname{ar}(\mathcal{B}_k)}\right) \\
&\underset{\text{(Lemma 22)}}{=} \Omega\left((\mathsf{d}_{\mathcal{K}^{\mathrm{id}}_{(k+2)/2},\mathfrak{D}}(m))^{1/2\operatorname{ar}(\mathcal{B}_k)}\right)
\end{aligned}$$

$$\underset{\text{(Lemma 16)}}{=} \Omega\left(((\mathsf{d}_{\mathcal{K}_{(k+2)/2}}(m))^{1/2\,\mathrm{ar}(\mathcal{B}_k)}\right)$$
$$\underset{\text{(Theorem 7)}}{=} \Omega\left(m^{(k+2)/4r}/\log^{(3k+2)/2r}(m)\right)$$

Where $\mathfrak{C}$ is the class of $\sigma \cup \sigma_{B_k}$ structures $\mathcal{C}$ such that $\{P_a^{\mathcal{C}} \mid a \in B_k\}$ is a partition of the universe, $\mathfrak{H}$ the class of $V(\mathcal{G}^{\mathrm{id}}_{\mathcal{B}_k})$-partitioned graphs, $\mathfrak{H}'$ the class of $V(\mathcal{G}^{\mathrm{id}}_{k})$-partitioned graphs and $\mathfrak{D}$ the class of $V(\mathcal{K}^{\mathrm{id}}_{(k+2)/2})$-partitioned graphs. From the above we can conclude that (3) implies (1) in the statement of the theorem. Moreover, as discussed in Section 4, (1) implies (2) follows from [32] and (2) implies (3) trivially. ◀

7 Conclusion

Our main result characterises those bounded-arity classes of structures $\mathfrak{A}$ where the set of homomorphisms from $\mathcal{A} \in \mathfrak{A}$ to $\mathcal{B}$ can be succinctly represented. More precisely, the known upper bound of $O(\|\mathcal{A}\|^2\|\mathcal{B}\|^{\mathrm{tw}(\mathcal{A})+1})$ is matched by a corresponding lower bound of $\Omega(\|\mathcal{B}\|^{\mathrm{tw}(\mathcal{A})^\varepsilon})$, where $\mathrm{tw}(\mathcal{A})$ is the tree-width of $\mathcal{A}$ and $\varepsilon > 0$ is a constant depending on the excluded grid theorem and the arity of the signature. A future task would be to further close the gap between upper and lower bounds.

Another open question is to understand the representation complexity for all classes of structures $\mathfrak{A}$ (of unbounded arity). As mentioned in Section 4, a polynomial $O(\|\mathcal{A}\|^2 \cdot \|\mathcal{B}\|^{\mathrm{fhtw}(\mathcal{A})})$ upper bound was shown where $\mathrm{fhtw}(\mathcal{A})$ is the fractional hypertreewidth of $\mathcal{A}$ [32] and one might wonder whether this is tight. At least this is not the case in a parametrised setting, where a $f(\|\mathcal{A}\|)\|\mathcal{B}\|^w$ sized representation for some (not necessarily polynomial-time) computable f, is considered *tractable*. It is known that for structures $\mathcal{A}$ of *bounded submodular width* the homomorphism problem can be decomposed into a (not necessarily disjoint) union of $f(\|\mathcal{A}\|)$ instances of bounded fractional hypertreewidth [29, 6], leading to a d-representation of size $f(\|\mathcal{A}\|)\|\mathcal{B}\|^{\mathrm{subw}(\mathcal{A})}$ where $\mathrm{subw}(\mathcal{A})$ denotes the submodular width of $\mathcal{A}$, see Appendix A in the full version of this paper for details. Note that submodular width can be strictly smaller than fractional hypertreewidth [28]. For a more concrete example in this direction, the fractional hypertreewidth of $\mathcal{C}_4$ is 2, but one can show that $\mathrm{Hom}(\mathcal{C}_4, \mathcal{H})$ has deterministic d-representations of size $O(\|\mathcal{H}\|^{3/2})$ – almost matching the $O(\|\mathcal{H}\|^{3/2}/\log^7(\|\mathcal{H}\|))$ lower bound in Example 17. Note that while submodular width characterises the FPT-fragment of deciding the existence of homomorphisms on structures of unbounded arity [29], a tight characterisation for the parameterised counting problem is, despite some recent progress [23], still missing. In particular, it is not clear whether bounded submodular width implies tractable counting. We may face similar difficulties when studying the complexity of *deterministic* d-representations that allow efficient counting.

In the course of proving our main result we have developed tools and techniques for proving lower bounds on the size of d-representations, in particular using our k-clique lower bound as a starting point, defining an appropriate notion of reduction and showing that one can always get such a reduction if the "almost minor" relation holds. Whilst the proof of the clique lower bound in Section 5 exploits the specific nature of d-representations, we observe that much of the content of Section 6 can easily be used for other forms of representations. Since we now have understood the limitations of unrestricted d-representations, it would be good to know whether there are even more succinct representation formats that still allow efficient enumeration.

References

1 Noga Alon and Ravi B. Boppana. The monotone circuit complexity of boolean functions. *Comb.*, 7(1):1–22, 1987. doi:10.1007/BF02579196.

2 Antoine Amarilli, Pierre Bourhis, Louis Jachiet, and Stefan Mengel. A circuit-based approach to efficient enumeration. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 111:1–111:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.111.

3 Antoine Amarilli, Pierre Bourhis, Stefan Mengel, and Matthias Niewerth. Enumeration on trees with tractable combined complexity and efficient updates. In Dan Suciu, Sebastian Skritek, and Christoph Koch, editors, *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 – July 5, 2019*, pages 89–103. ACM, 2019. doi:10.1145/3294052.3319702.

4 Jérôme Amilhastre, Hélène Fargier, Alexandre Niveau, and Cédric Pralet. Compiling CSPs: A Complexity Map of (Non-Deterministic) Multivalued Decision Diagrams. *International Journal on Artificial Intelligence Tools*, 23(4), 2014. doi:10.1142/S021821301460015X.

5 Albert Atserias, Martin Grohe, and Dániel Marx. Size bounds and query plans for relational joins. *SIAM Journal on Computing*, 42(4):1737–1767, 2013. doi:10.1137/110859440.

6 Christoph Berkholz and Nicole Schweikardt. Constant delay enumeration with FPT-preprocessing for conjunctive queries of bounded submodular width. *arXiv preprint*, 2020. arXiv:2003.01075.

7 Markus Bläser, Balagopal Komarath, and Karteek Sreenivasaiah. Graph pattern polynomials. In Sumit Ganguly and Paritosh K. Pandya, editors, *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, December 11-13, 2018, Ahmedabad, India*, volume 122 of *LIPIcs*, pages 18:1–18:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.FSTTCS.2018.18.

8 Andrei A. Bulatov. The complexity of the counting constraint satisfaction problem. *J. ACM*, 60(5):34:1–34:41, 2013. doi:10.1145/2528400.

9 Andrei A. Bulatov. A dichotomy theorem for nonuniform CSPs. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2017)*, pages 319–330, 2017. doi:10.1109/FOCS.2017.37.

10 Andrei A. Bulatov, Victor Dalmau, Martin Grohe, and Daniel Marx. Enumerating Homomorphisms. In Susanne Albers and Jean-Yves Marion, editors, *26th International Symposium on Theoretical Aspects of Computer Science*, volume 3 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 231–242, Dagstuhl, Germany, 2009. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.STACS.2009.1838.

11 Chandra Chekuri and Julia Chuzhoy. Polynomial bounds for the grid-minor theorem. *J. ACM*, 63(5):40:1–40:65, 2016. doi:10.1145/2820609.

12 FRK Chung, P Erdős, and Joel Spencer. On the decomposition of graphs into complete bipartite subgraphs. In *Studies in pure mathematics*, pages 95–101. Springer, 1983.

13 Nadia Creignou and Jean-Jacques Hébrard. On generating all solutions of generalized satisfiability problems. *RAIRO Theor. Informatics Appl.*, 31(6):499–511, 1997. doi:10.1051/ita/1997310604991.

14 Víctor Dalmau and Peter Jonsson. The complexity of counting homomorphisms seen from the other side. *Theor. Comput. Sci.*, 329(1-3):315–323, 2004. doi:10.1016/j.tcs.2004.08.008.

15 Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *J. Artif. Intell. Res.*, 17:229–264, 2002. doi:10.1613/jair.989.

16 Christian Engels. Dichotomy theorems for homomorphism polynomials of graph classes. *J. Graph Algorithms Appl.*, 20(1):3–22, 2016. doi:10.7155/jgaa.00382.

17 Tomás Feder and Moshe Y Vardi. Monotone monadic snp and constraint satisfaction. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 612–622, 1993.

18 Herbert Fleischner, Egbert Mujuni, Daniël Paulusma, and Stefan Szeider. Covering graphs with few complete bipartite subgraphs. *Theoretical Computer Science*, 410(21-23):2045–2053, 2009.

19 Gianluigi Greco and Francesco Scarcello. Structural tractability of enumerating CSP solutions. *Constraints An Int. J.*, 18(1):38–74, 2013. doi:10.1007/s10601-012-9129-8.

20 Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1), March 2007. doi:10.1145/1206035.1206036.

21 Martin Grohe and Dániel Marx. Constraint solving via fractional edge covers. *ACM Transactions on Algorithms (TALG)*, 11(1):1–20, 2014.

22 Stasys Jukna and Alexander S Kulikov. On covering graphs by complete bipartite subgraphs. *Discrete Mathematics*, 309(10):3399–3403, 2009.

23 Mahmoud Abo Khamis, Ryan R. Curtin, Benjamin Moseley, Hung Q. Ngo, Xuanlong Nguyen, Dan Olteanu, and Maximilian Schleich. Functional aggregate queries with additive inequalities. *ACM Trans. Database Syst.*, 45(4), December 2020. doi:10.1145/3426865.

24 Balagopal Komarath, Anurag Pandey, and Chengot Sankaramenon Rahul. Monotone arithmetic complexity of graph homomorphism polynomials. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 83:1–83:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.83.

25 Frédéric Koriche, Jean-Marie Lagniez, Pierre Marquis, and Samuel Thomas. Compiling Constraint Networks into Multivalued Decomposable Decision Graphs. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 332–338, Buenos Aires, Argentina, 2015. AAAI Press. URL: http://dl.acm.org/citation.cfm?id=2832249.2832295.

26 Yuan Li, Alexander A. Razborov, and Benjamin Rossman. On the ac^0 complexity of subgraph isomorphism. *SIAM J. Comput.*, 46(3):936–971, 2017. doi:10.1137/14099721X.

27 Dániel Marx. Can you beat treewidth? In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 169–179. IEEE, 2007.

28 Dániel Marx. Tractable structures for constraint satisfaction with truth tables. *Theory of Computing Systems*, 48(3):444–464, 2011.

29 Dániel Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. *Journal of the ACM (JACM)*, 60(6):1–51, 2013.

30 Robert Mateescu and Rina Dechter. Compiling Constraint Networks into AND/OR Multivalued Decision Diagrams (AOMDDs). In *Principles and Practice of Constraint Programming – CP 2006*, Lecture Notes in Computer Science, pages 329–343. Springer, Berlin, Heidelberg, September 2006. doi:10.1007/11889205_25.

31 Dhruv Mubayi and György Turán. Finding bipartite subgraphs efficiently. *arXiv preprint*, 2009. arXiv:0905.2527.

32 Dan Olteanu and Jakub Závodný. Size bounds for factorised representations of query results. *ACM Transactions on Database Systems (TODS)*, 40(1):1–44, 2015.

33 Knot Pipatsrisawat and Adnan Darwiche. New compilation languages based on structured decomposability. In Dieter Fox and Carla P. Gomes, editors, *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 2008)*, pages 517–522. AAAI Press, 2008. URL: http://www.aaai.org/Library/AAAI/2008/aaai08-082.php.

34 Neil Robertson and P.D Seymour. Graph minors. v. excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41(1):92–114, 1986. doi:10.1016/0095-8956(86)90030-4.

35 Benjamin Rossman. On the constant-depth complexity of k-clique. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 721–730. ACM, 2008. doi:10.1145/1374376.1374480.

36 Benjamin Rossman. The monotone complexity of k-clique on random graphs. *SIAM J. Comput.*, 43(1):256–279, 2014. doi:10.1137/110839059.

37 Henning Schnoor and Ilka Schnoor. Enumerating all solutions for constraint satisfaction problems. In Wolfgang Thomas and Pascal Weil, editors, *STACS 2007, 24th Annual Symposium on Theoretical Aspects of Computer Science, Aachen, Germany, February 22-24, 2007, Proceedings*, volume 4393 of *Lecture Notes in Computer Science*, pages 694–705. Springer, 2007. doi:10.1007/978-3-540-70918-3_59.

38 Dmitriy Zhuk. A proof of the CSP dichotomy conjecture. *Journal of the ACM (JACM)*, 67(5):1–78, 2020.

Nominal Topology for Data Languages

Fabian Birkmann
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Stefan Milius
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Henning Urbat
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Abstract

We propose a novel topological perspective on data languages recognizable by orbit-finite nominal monoids. For this purpose, we introduce pro-orbit-finite nominal topological spaces. Assuming globally bounded support sizes, they coincide with nominal Stone spaces and are shown to be dually equivalent to a subcategory of nominal boolean algebras. Recognizable data languages are characterized as topologically clopen sets of pro-orbit-finite words. In addition, we explore the expressive power of pro-orbit-finite equations by establishing a nominal version of Reiterman's pseudovariety theorem.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Nominal sets, Stone duality, Profinite space, Data languages

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.114

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: https://arxiv.org/abs/2304.13337

Funding Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 470467389.

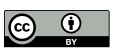
1 Introduction

While automata theory is largely concerned with formal languages over finite alphabets, the extension to *infinite alphabets* has been identified as a natural approach to modelling structures involving *data*, such as nonces [26], channel names [23], object identities [22], process identifiers [11], URLs [5], or values in XML documents [31]. For example, if $\mathbb{A}$ is a (countably infinite) set of data values, typical languages to consider might be

$$L_0 = \{\, vaaw \mid a \in \mathbb{A},\, v, w \in \mathbb{A}^* \,\} \qquad \text{(“some data value occurs twice in a row”), or}$$
$$L_1 = \{\, avaw \mid a \in \mathbb{A},\, v, w \in \mathbb{A}^* \,\} \qquad \text{(“the first data value occurs again”).}$$

Automata for data languages enrich finite automata with register mechanisms that allow to store data and test data values for equality (or more complex relations, e.g. order) [24, 31]. In a modern perspective first advocated by Bojańczyk, Klin, and Lasota [9], a convenient abstract framework for studying data languages is provided by the theory of *nominal sets* [36].

Despite extensive research in the past three decades, no universally acknowledged notion of *regular* data language has emerged so far. One reason is that automata models with data notoriously lack robustness, in that any alteration of their modus operandi (e.g. deterministic vs. nondeterministic, one-way vs. two-way) usually affects their expressive power. Moreover, machine-independent descriptions of classes of data languages in terms of algebra or model theory are hard to come by. However, there is one remarkable class of data languages that closely mirrors classical regular languages: data languages recognizable by orbit-finite



nominal monoids [7]. Originally introduced from a purely algebraic angle, recognizable data languages have subsequently been characterized in terms of *rigidly guarded MSO*$^\sim$, a fragment of monadic second-order logic with equality tests [13], *single-use register automata* [10] (both one-way and two-way), and *orbit-finite regular list functions* [10]. In addition, several landmark results from the algebraic theory of regular languages, namely the McNaughton-Papert-Schützenberger theorem [29, 40], the Krohn-Rhodes theorem [25], and Eilenberg's variety theorem [14] have been extended to recognizable data languages [7, 10, 13, 44].

In the present paper, we investigate recognizable data languages through the lens of *topology*, thereby providing a further bridge to classical regular languages. The topological approach to the latter is closely tied to the algebraic one, which regards regular languages as the languages recognizable by finite monoids. Its starting point is the construction of the topological space $\widehat{\Sigma^*}$ of *profinite words*. Informally, this space casts all information represented by regular languages over Σ and their recognizing monoids into a single mathematical object. Regular languages can then be characterized by purely topological means: they may be interpreted as precisely the clopen subsets of $\widehat{\Sigma^*}$, in such way that algebraic recognition by finite monoids becomes a continuous process. Properties of regular languages are often most conveniently classified in terms of the topological concept of *profinite equations*, that is, equations between profinite words; see [3, 4, 33] for a survey of profinite methods in automata theory. Moreover, since $\widehat{\Sigma^*}$ forms a Stone space, the power of *Stone duality* – the dual equivalence between Stone spaces and boolean algebras – becomes available. This allows for the use of duality-theoretic methods for the study of regular languages and their connection to logic and model theory, which in part even extend to *non-regular* languages [18–21, 35].

On a conceptual level, the topological view of regular languages rests on a single category-theoretic fact: Stone spaces admit a universal property. In fact, they arise from the category of finite sets as the free completion under codirected limits, a.k.a. its *Pro-completion*:

$$\mathbf{Stone} \simeq \mathrm{Pro}(\mathbf{Set}_\mathrm{f}). \tag{1.1}$$

In the world of data languages, the role of finite sets is taken over by orbit-finite nominal sets. This strongly suggests to base a topological approach on their free completion $\mathrm{Pro}(\mathbf{Nom}_\mathrm{of})$. However, this turns out to be infeasible: the category $\mathrm{Pro}(\mathbf{Nom}_\mathrm{of})$ is not concrete over nominal sets (Proposition 3.6), hence it cannot be described via any kind of nominal topological spaces. This is ultimately unsurprising given that the description (1.1) of Stone spaces as a free completion depends on the axiom of choice, which is well-known to fail in the topos of nominal sets. As a remedy, we impose *global bounds* on the support sizes of nominal sets, that is, we consider the categories $\mathbf{Nom}_k$ and $\mathbf{Nom}_{\mathrm{of},k}$ of (orbit-finite) nominal sets where every element has a support of size k, for some fixed natural number k. This restriction is natural from an automata-theoretic perspective, as it corresponds to imposing a bound k on the number of registers of automata, and it fixes exactly the issue making unrestricted nominal sets non-amenable (Lemma 3.9). Let us emphasize, however, that the category $\mathbf{Nom}_k$ is not proposed as a new foundation for names and variable binding; for instance, it generally fails to be a topos.

The first main contribution of our paper is a generalization of (1.1) to k-bounded nominal sets. For this purpose we introduce *nominal Stone spaces*, a suitable nominalization of the classical concept, and prove that k-bounded nominal Stone spaces form the Pro-completion of the category of k-bounded orbit-finite sets. We also derive a nominal version of Stone duality, which relates k-bounded nominal Stone spaces to *locally k-atomic orbit-finitely complete nominal boolean algebras*. Hence we establish the following equivalences of categories:

$$\mathbf{nC}_\mathrm{of}\mathbf{A}_{\mathrm{l}k}\mathbf{BA} \simeq^\mathrm{op} \mathbf{nStone}_k \simeq \mathrm{Pro}(\mathbf{Nom}_{\mathrm{of},k}).$$

The above equivalences are somewhat remarkable since even the category of k-bounded nominal sets does not feature choice. They hold because the presence of bounds allows us to reduce topological properties of nominal Stone spaces, most notably compactness, to their classical counterparts.

Building on the above topological foundations, which we regard to be of independent interest, we subsequently develop first steps of a topological theory of data languages. Specifically, we introduce nominal Stone spaces of (bounded) *pro-orbit-finite words* and prove their clopen subsets to correspond to data languages recognizable by bounded equivariant monoid morphisms, generalizing the topological characterization of classical regular languages (Theorem 5.14). Moreover, we investigate the expressivity of *pro-orbit-finite equations* and show that they model precisely classes of orbit-finite monoids closed under finite products, submonoids, and *multiplicatively support-reflecting quotients* (Theorem 6.8). This provides a nominal version of Reiterman's celebrated pseudovariety theorem [37] for finite monoids.

Related work. The perspective taken in our paper draws much of its inspiration from the recent categorical approach to algebraic recognition based on monads [8, 38, 42]. The importance of Pro-completions in algebraic language theory has been isolated in the work of Chen et al. [12] and Urbat et al. [42]. In the latter work the authors introduce *profinite monads* and present a general version of Eilenberg's variety theorem parametric in a given Stone-type duality. The theory developed there applies to algebraic base categories, but not to the category of nominal sets.

Our version of nominal Stone duality builds on the orbit-finite restriction of the duality between nominal sets and complete atomic nominal boolean algebras due to Petrişan [17]. It is fundamentally different from the nominal Stone duality proposed by Gabbay, Litak, and Petrişan [16], which relates *nominal Stone spaces with* И to *nominal boolean algebras with* И. The latter duality is not amenable for the theory of data languages; see Remark 3.17.

Reiterman's pseudovariety theorem has recently been generalized to the level of finite algebras for a monad [1, 12] and, in a more abstract disguise, finite objects in a category [30]. For nominal sets, varieties of algebras over binding signatures have been studied by Gabbay [16] and by Kurz and Petrişan [27], resulting in nominal Birkhoff-type theorems [6]. Urbat and Milius [44] characterize classes of orbit-finite monoids called *weak* pseudovarieties by sequences of nominal word equations. This gives a nominal generalization of the classical Eilenberg-Schützenberger theorem [15], which in fact is a special case of the general HSP theorem in [30]. Nominal pro-orbit-finite equations as introduced in the present paper are strictly more expressive than sequences of nominal word equations (Example 6.11), hence our nominal Reiterman theorem is not equivalent to the nominal Eilenberg-Schützenberger theorem. Moreover, we note that the nominal Reiterman theorem does not appear to be an instance of any of the abstract categorical frameworks mentioned above.

2 Preliminaries

We assume that readers are familiar with basic notions from category theory, e.g. functors, natural transformations, and (co)limits, and from point-set topology, e.g. metric and topological spaces, continuous maps, and compactness. In the following we recall some facts about Pro-completions, the key categorical concept underlying our topological approach to data languages. Moreover, we give a brief introduction to the theory of nominal sets [36].

Pro-completions. A small category I is *cofiltered* if (i) I is non-empty, (ii) for every pair of objects $i, j \in I$ there exists a span $i \leftarrow k \rightarrow j$, and (iii) for every pair of parallel arrows $f, g\colon j \to k$, there exists a morphism $h\colon i \to j$ such that $f \cdot h = g \cdot h$. Cofiltered preorders are

called *codirected*; thus a preorder I is codirected if $I \neq \emptyset$ and every pair $i, j \in I$ has a lower bound $k \leq i, j$. For instance, every meet-semilattice with bottom is codirected. A diagram $D\colon I \to \mathbf{C}$ in a category $\mathbf{C}$ is *cofiltered* if its index category I is cofiltered. A *cofiltered limit* is a limit of a cofiltered diagram. *Codirected limits* are defined analogously. The two concepts are closely related: a category has cofiltered limits iff it has codirected limits, and a functor preserves cofiltered limits iff it preserves codirected limits [2, Cor. 1.5]. The dual concept is that of a *filtered colimit* or a *directed colimit*, respectively.

▶ **Example 2.1.**

1. In the category **Set** of sets and functions, every filtered diagram $D\colon I \to \mathbf{Set}$ has a colimit cocone $c_i\colon D_i \to \operatorname{colim} D$ $(i \in I)$ given by $\operatorname{colim} D = \big(\coprod_{i\in I} D_i\big)/{\sim}$ and $c_i(x) = [x]_\sim$, where the equivalence relation $\sim$ on the coproduct (i.e. disjoint union) $\coprod_{i\in I} D_i$ relates $x \in D_i$ and $y \in D_j$ iff there exist morphisms $f\colon i \to k$ and $g\colon j \to k$ in I such that $Df(x) = Dg(y)$.
2. Every cofiltered diagram $D\colon I \to \mathbf{Set}$ has a limit whose cone $p_i\colon \lim D \to D_i$ $(i \in I)$ is given by the *compatible families* of D and projection maps:

$$\lim D = \{(x_i)_{i\in I} \mid x_i \in D_i \text{ and } Df(x_i) = x_j \text{ for all } f\colon i \to j \text{ in } I\} \quad \text{and} \quad p_j((x_i)_{i\in I}) = x_j.$$

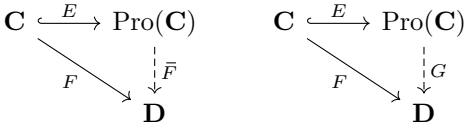
3. In the category **Top** of topological spaces and continuous maps, the limit cone of a cofiltered diagram $D\colon I \to \mathbf{Top}$ is formed by taking the limit in **Set** and equipping $\lim D$ with the *initial topology*, viz. the topology generated by the basic open sets $p_i^{-1}[U_i]$ for $i \in I$ and $U_i \subseteq D_i$ open.

An object C of a category $\mathbf{C}$ is *finitely copresentable* if the contravariant hom-functor $\mathbf{C}(-, C)\colon \mathbf{C}^{\mathrm{op}} \to \mathbf{Set}$ preserves directed colimits. In more elementary terms, this means that for every codirected diagram $D\colon I \to \mathbf{C}$ with limit cone $p_i\colon L \to D_i$ $(i \in I)$,

1. every morphism $f\colon L \to C$ factorizes as $f = g \circ p_i$ for some $i \in I$ and $g\colon D_i \to C$, and
2. the factorization is essentially unique: given another factorization $f = h \cdot p_i$, there exists $j \leq i$ such that $g \cdot D_{j,i} = h \cdot D_{j,i}$.

A *Pro-completion* of a small category $\mathbf{C}$ is a free completion under codirected (equivalently cofiltered) limits. It is given by a category $\mathrm{Pro}(\mathbf{C})$ with codirected limits together with a full embedding $E\colon \mathbf{C} \hookrightarrow \mathrm{Pro}(\mathbf{C})$ satisfying the following universal property:

1. every functor $F\colon \mathbf{C} \to \mathbf{D}$, where the category $\mathbf{D}$ has codirected limits, extends to a functor $\bar{F}\colon \mathrm{Pro}(\mathbf{C}) \to \mathbf{D}$ that preserves codirected limits and satisfies $F = \bar{F} \circ E$;
2. $\bar{F}$ is essentially unique: For every functor G that preserves codirected limits and satisfies $F = G \circ E$, there exists a natural isomorphism $\alpha\colon \bar{F} \cong G$ such that $\alpha E = \mathrm{id}_F$.



The universal property determines $\mathrm{Pro}(\mathbf{C})$ uniquely up to equivalence of categories. We note that every object EC $(C \in \mathbf{C})$ is finitely copresentable in $\mathrm{Pro}(\mathbf{C})$, see e.g. [1, Thm A.4]. The dual of Pro-completions are *Ind-completions*: free completions under *directed colimits*.

▶ **Example 2.2.** The Pro-completion $\mathrm{Pro}(\mathbf{Set}_{\mathrm{f}})$ of the category of finite sets is the full subcategory of **Top** given by *profinite spaces* (topological spaces that are codirected limits of finite discrete spaces). Profinite spaces are also known as *Stone spaces* or *boolean spaces* and can be characterized by topological properties: they are precisely compact Hausdorff spaces with a basis of clopen sets. This equivalent characterization depends on the axiom of

choice (or rather the ultrafilter theorem, a weak form of choice), as does *Stone duality*, the dual equivalence between the categories of Stone spaces and boolean algebras. The duality maps a Stone space to its boolean algebra of clopen sets, equipped with the set-theoretic boolean operations. Its inverse maps a boolean algebra the set of ultrafilters (equivalently, prime filters) on it, equipped with a suitable profinite topology.

Profinite words. The topological approach to classical regular languages is based on the space $\widehat{\Sigma^*}$ of *profinite words* over the alphabet Σ. This space is constructed as the codirected limit of all finite quotient monoids of Σ^*, the free monoid of finite words generated by Σ. Formally, let $\Sigma^*{\downarrow}\mathbf{Mon}_\mathrm{f}$ be the codirected poset of all surjective monoid morphisms $e\colon \Sigma^* \twoheadrightarrow M$, where M is a finite monoid; the order on $\Sigma^*{\downarrow}\mathbf{Mon}_\mathrm{f}$ is defined by $e \leq e'$ if $e' = e \cdot h$ for some h. Then $\widehat{\Sigma^*}$ is the limit of the diagram $D\colon \Sigma^*{\downarrow}\mathbf{Mon}_\mathrm{f} \to \mathrm{Pro}(\mathbf{Set}_\mathrm{f})$ sending $e\colon \Sigma^* \twoheadrightarrow M$ to the underlying set of M, regarded as a finite discrete topological space. The space $\widehat{\Sigma^*}$ is completely metrizable; in fact, it is the Cauchy completion of the metric space (Σ^*, d) where $d(v,w) = \sup\{\, 2^{-|M|} \mid M \text{ is a finite monoid separating } v, w \,\}$. Here a monoid M *separates* $v, w \in \Sigma^*$ if there exists a morphism $h\colon \Sigma^* \to M$ such that $h(v) \neq h(w)$. Regular languages over Σ correspond to clopen subsets of $\widehat{\Sigma^*}$, or equivalently to continuous maps $L\colon \widehat{\Sigma^*} \to 2$ into the discrete two-element space.

Nominal Sets. Fix a countable set $\mathbb{A}$ of *names*, and denote by $\mathrm{Perm}\,\mathbb{A}$ the group of finite permutations, i.e. bijections $\pi\colon \mathbb{A} \to \mathbb{A}$ fixing all but finitely many names. Given $S \subseteq \mathbb{A}$ write

$$\mathrm{Perm}_S\,\mathbb{A} = \{\pi \in \mathrm{Perm}\,\mathbb{A} \mid \pi(a) = a \text{ for all } a \in S\}$$

for the the subgroup of permutations fixing S. A $\mathrm{Perm}\,\mathbb{A}$*-set* is a set X with a group action, that is, an operation $\cdot\colon \mathrm{Perm}\,\mathbb{A} \times X \to X$ such that $\mathrm{id} \cdot x = x$ and $\pi \cdot (\sigma \cdot x) = (\pi \circ \sigma) \cdot x$ for every $x \in X$ and $\pi, \sigma \in \mathrm{Perm}\,\mathbb{A}$. The *trivial* group action on X is given by $\pi \cdot x = x$ for all $x \in X$ and $\pi \in \mathrm{Perm}\,\mathbb{A}$.

A subset $S \subseteq \mathbb{A}$ is a *support* of $x \in X$ if every permutation $\pi \in \mathrm{Perm}_S\,\mathbb{A}$ acts trivially on x, that is, $\pi \cdot x = x$. The idea is that x is some syntactic object (e.g. a word, a tree, or a λ-term) whose free variables are contained in S. A $\mathrm{Perm}\,\mathbb{A}$-set X is a *nominal set* if every element $x \in X$ has a finite support. This implies that every $x \in X$ has a *least* finite support, denoted by $\mathrm{supp}\,x \subseteq \mathbb{A}$.

For a nominal set X its *nominal powerset* $\mathcal{P}_\mathrm{fs}X \subseteq \mathcal{P}X$ consists of all subsets of $U \subseteq X$ which are finitely supported under the action $\pi \cdot U := \{\pi \cdot x \mid x \in U\}$. For example, for the nominal set $\mathbb{A}$ of names with the action $\pi \cdot a = \pi(a)$, its nominal powerset $\mathcal{P}_\mathrm{fs}\mathbb{A}$ consists of all finite and cofinite subsets of $\mathbb{A}$. A subset $U \subseteq X$ is *equivariant* if it has empty support. If there exists a finite subset $S \subseteq \mathbb{A}$ supporting every $x \in U$ then U is *uniformly finitely supported*, and S also supports U. Given a finite set $S \subseteq \mathbb{A}$ of names and a subset $U \subseteq X$, we define the S*-hull* of U by $\mathrm{hull}_S\,U = \{\pi \cdot x \mid x \in U, \pi \in \mathrm{Perm}_S\,\mathbb{A}\}$. This is the smallest S-supported subset of X containing U.

For finite $S \subseteq \mathbb{A}$ the S*-orbit* of an element $x \in X$ is the set $\mathrm{orb}_S\,x = \{\pi \cdot x \mid \pi \in \mathrm{Perm}_S\,\mathbb{A}\}$. The $\emptyset$-orbit of x is called its *orbit*, denoted $\mathrm{orb}\,x$. We write $\mathrm{orb}_S\,X = \{\mathrm{orb}_S\,x \mid x \in X\}$ for the set of all S-orbits of X, and $\mathrm{orb}\,X$ for the set of all orbits. The S-orbits form a partition of X. A finitely supported subset $Y \subseteq X$ is *orbit-finite* if it intersects only finitely many orbits of X. In particular, the nominal set X is *orbit-finite* if $\mathrm{orb}\,X$ is a finite set. This implies that for every finite subset $S \subseteq \mathbb{A}$ the set $\mathrm{orb}_S\,X$ is finite. Moreover, X contains only finitely many elements with support S.

▶ **Example 2.3.** The set $\mathbb{A}^*$ of finite words over $\mathbb{A}$ forms a nominal set with the group action $\pi \cdot (a_1 \cdots a_n) = \pi(a_1) \cdots \pi(a_n)$. The languages $L_0, L_1 \subseteq \mathbb{A}^*$ from the Introduction are equivariant subsets. Given a fixed name $a \in \mathbb{A}$, the subset $L_2 = \{awa \mid w \in \mathbb{A}^*\}$ is finitely supported with $\operatorname{supp} L_2 = \{a\}$. All the above sets have an infinite number of orbits. An example of an orbit-finite set is given by $\mathbb{A}^2 = \mathbb{A} \times \mathbb{A} \subseteq \mathbb{A}^*$; its two orbits are $\{aa \mid a \in \mathbb{A}\}$ and $\{ab \mid a \neq b \in \mathbb{A}\}$.

A map $f\colon X \to Y$ between nominal sets is *finitely supported* if there exists a finite set $S \subseteq \mathbb{A}$ such that $f(\pi \cdot x) = \pi \cdot f(x)$ for all $x \in X$ and $\pi \in \operatorname{Perm}_S \mathbb{A}$, and *equivariant* if it is supported by $S = \emptyset$. Equivariant maps satisfy $\operatorname{supp} f(x) \subseteq \operatorname{supp} x$ for all $x \in X$. Nominal sets and equivariant maps form a category **Nom**, with the full subcategory $\mathbf{Nom}_{\mathrm{of}}$ of orbit-finite nominal sets. The category **Nom** is complete and cocomplete. Colimits and finite limits are formed like in **Set**; general limits are formed by taking the limit in **Set** and restricting to finitely supported elements. The category $\mathbf{Nom}_{\mathrm{of}}$ is closed under finite limits and finite colimits in **Nom**. *Quotients* and *subobjects* in **Nom** are represented by surjective and injective equivariant maps. Every equivariant map f has an image factorization $f = m \cdot e$ with m injective and e surjective; we call e the *coimage* of f.

A nominal set is *strong* if for all $x \in X$ and $\pi \in \operatorname{Perm} \mathbb{A}$ one has $\pi \cdot x = x$ iff $\pi \in \operatorname{Perm}_S \mathbb{A}$, where $S = \operatorname{supp} x$. (Note that the "if" direction holds in every nominal set.) For example, the nominal set $\mathbb{A}^{\#n} = \{f\colon n \to \mathbb{A} \mid f \text{ injective}\}$ with pointwise action is strong and has a single orbit. Up to isomorphism, (orbit-finite) strong nominal sets are precisely (finite) coproducts of such sets.

3 Nominal Stone Spaces

In this section, we establish the topological foundations for our pro-orbit-finite approach to data languages. We start by recalling the basic definitions of nominal topology [17, 32].

▶ **Definition 3.1.**

1. *A* nominal topology *on a nominal set X is an equivariant subset $\mathcal{O}_X \subseteq \mathcal{P}_{fs}X$ closed under finitely supported union (that is, if $\mathcal{U} \subseteq \mathcal{O}_X$ is finitely supported then $\bigcup \mathcal{U} \in \mathcal{O}_X$) and finite intersection. Sets $U \in \mathcal{O}_X$ are called* open *and their complements* closed*; sets that are both open and closed are* clopen*. A nominal set X together with a nominal topology $\mathcal{O}_X$ is a* nominal topological space*. An equivariant map $f\colon X \to Y$ between nominal topological spaces is* continuous *if for every open set U of Y its preimage $f^{-1}[U]$ is an open set of X. Nominal topological spaces and continuous maps form the category* **nTop**.
2. *A* subbasis *of a nominal topological space $(X, \mathcal{O}_X)$ is an equivariant subset $\mathcal{B} \subseteq \mathcal{O}_X$ such that every open set of X is a finitely supported union of finite intersections of sets in $\mathcal{B}$. If additionally every finite intersection of sets in $\mathcal{B}$ is a finitely supported union of sets in $\mathcal{B}$, then $\mathcal{B}$ is called a* basis*. In this case, every open set of X is a finitely supported union of elements of $\mathcal{B}$.*

▶ **Example 3.2.**

1. A topological space may be viewed as a nominal topological space equipped with the trivial group action. Then every (open) subset has empty support and every union is finitely supported, so we recover the axioms of classical topology.
2. Every nominal set X equipped with the *discrete topology*, where all finitely supported subsets are open, is a nominal topological space. It has a basis given by all singleton sets.

3. A *nominal (pseudo-)metric space* is given by a nominal set X with a (pseudo-)metric[1] $d\colon X \times X \to \mathbb{R}$ which is equivariant as a function into the set $\mathbb{R}$, regarded as a nominal set with the trivial group action. As usual, the open ball around $x \in X$ with radius $r > 0$ is given by $B_r x = \{y \in X \mid d(x,y) < r\}$. Since $\pi \cdot B_r(x) = B_r(\pi \cdot x)$ for all $\pi \in \operatorname{Perm}\mathbb{A}$ and $x \in X$, every nominal (pseudo-)metric space carries a nominal topology whose basic opens are the open balls.

▶ Remark 3.3. Every nominal topological space induces two families of ordinary topological spaces, one by taking only opens with a certain support and the other by forming orbits. In more detail, let $S \subseteq \mathbb{A}$ be a finite set of names and let X be a nominal topological space with topology $\mathcal{O}$.

1. The underlying set of the nominal space X carries a classical topology $\mathcal{O}_S$ consisting of all S-supported open sets of $\mathcal{O}$. We denote the resulting topological space by $|X|_S$.
2. The set $\operatorname{orb}_S X$ of S-orbits can be equipped with the quotient topology $\mathcal{O}_{\operatorname{orb}_S}$ induced by the projection $X \twoheadrightarrow \operatorname{orb}_S X$ mapping each $x \in X$ to its S-orbit $\operatorname{orb}_S x$. In this topology, a set $O \subseteq \operatorname{orb}_S X$ of S-orbits is open iff its union $\bigcup O$ is open in X.

These constructions give rise to functors $|-|_S, \operatorname{orb}_S\colon \mathbf{nTop} \to \mathbf{Top}$. They allow us to switch between nominal and classical topology.

As noted in Example 2.2, the Pro-completion of the category $\mathbf{Set}_\mathrm{f}$ is the category of profinite spaces. One may expect that the Pro-completion of $\mathbf{Nom}_\mathrm{of}$ analogously consists of all *pro-orbit-finite* spaces, that is, nominal topological spaces that are codirected limits of orbit-finite discrete spaces. However, this fails due to a simple fact: while codirected limits of non-empty finite sets are always non-empty (which is a consequence of Tychonoff's theorem, thus the axiom of choice), codirected limits of non-empty orbit-finite nominal sets may be empty.

▶ Remark 3.4. Similar to **Top**, codirected limits in **nTop** are formed by taking the limit in **Nom** equipping it with the initial topology.

▶ **Example 3.5.** Consider the ω^{op}-chain $1 \leftarrow \mathbb{A} \leftarrow \mathbb{A}^{\#2} \leftarrow \mathbb{A}^{\#3} \leftarrow \cdots$ in $\mathbf{Nom}_\mathrm{of}$ with connecting maps omitting the last component. Its limit in **Set** (see Example 2.1) is given by $\mathbb{A}^{\#\omega}$, the set of all injective functions from ω to $\mathbb{A}$. Clearly no such function has finite support, thus the limit in **Nom** (and therefore also in **nTop**) is empty.

This entails that it is in fact impossible to characterize $\operatorname{Pro}(\mathbf{Nom}_\mathrm{of})$ by any sort of spaces. By definition of the free completion $\operatorname{Pro}(\mathbf{Nom}_\mathrm{of})$, the inclusion functor $I\colon \mathbf{Nom}_\mathrm{of} \hookrightarrow \mathbf{Nom}$ extends uniquely to a functor $\bar{I}\colon \operatorname{Pro}(\mathbf{Nom}_\mathrm{of}) \to \mathbf{Nom}$ preserving codirected limits. The analogous functor $\bar{I}\colon \operatorname{Pro}(\mathbf{Set}_\mathrm{f}) \to \mathbf{Set}$ is the forgetful functor of the category of profinite spaces. In contrast, we have

▶ **Proposition 3.6.** *The category* $\operatorname{Pro}(\mathbf{Nom}_\mathrm{of})$ *is* not *concrete: the functor* $\bar{I}$ *is not faithful.*

Proof. Consider the chain $1 \leftarrow \mathbb{A} \leftarrow \mathbb{A}^{\#2} \leftarrow \cdots$ of Example 3.5. Let $D\colon \omega^{\mathrm{op}} \to \mathbf{Nom}_\mathrm{of}$ denote the corresponding diagram, and let $E\colon \mathbf{Nom}_\mathrm{of} \hookrightarrow \operatorname{Pro}(\mathbf{Nom}_\mathrm{of})$ be the embedding. To prove that $\bar{I}$ is not faithful, let 2 be the two element nominal set. We show that $|\operatorname{Pro}(\mathbf{Nom}_\mathrm{of})(\lim ED, E2)| > |\mathbf{Nom}(\bar{I}(\lim ED), \bar{I}E2)|$. Indeed, we have

$$\begin{aligned}
\operatorname{Pro}(\mathbf{Nom}_\mathrm{of})(\lim_{n<\omega} ED_n, E2) &\cong \operatorname*{colim}_{n<\omega} \operatorname{Pro}(\mathbf{Nom}_\mathrm{of})(ED_n, E2) && E2 \text{ finitely copresentable}\\
&\cong \operatorname*{colim}_{n<\omega} \mathbf{Nom}_\mathrm{of}(D_n, 2) && E \text{ full embedding}\\
&\cong 2
\end{aligned}$$

[1] Recall that a pseudometric differs from a metric by not requiring $d(x,y) \neq 0$ for $x \neq y$.

because $\mathbf{Nom}_{\mathrm{of}}(D_0, 2) \cong 2$ and the two elements are not merged by the colimit injection. However,

$$\begin{aligned}
\mathbf{Nom}(\bar{I}(\lim_{n<\omega} ED_n), \bar{I}E2) &\cong \mathbf{Nom}(\lim_{n<\omega} \bar{I}ED_n, \bar{I}E2) && \bar{I} \text{ preserves codirected limits} \\
&\cong \mathbf{Nom}(\lim_{n<\omega} ID_n, 2) && I = \bar{I}E \\
&\cong \mathbf{Nom}(\emptyset, 2) && \text{Example 3.5} \\
&\cong 1. &&
\end{aligned}$$

◀

We thus restrict our focus to well-behaved subcategories of $\mathbf{Nom}_{\mathrm{of}}$. We choose these subcategories in such way that situations like in Example 3.5, where unrestricted accumulation of supports results in empty codirected limits, are avoided.

▶ **Definition 3.7.** *A nominal set X is* k-bounded*, for $k \in \mathbb{N}$, if $|\mathrm{supp}\, x| \leq k$ for every $x \in X$.*

For concrete categories $\mathbf{C}$ over $\mathbf{Nom}$ (or $\mathbf{Nom}_{\mathrm{of}}$) we denote by $\mathbf{C}_k$ the full subcategory of $\mathbf{C}$ whose underlying objects are k-bounded. For instance, $\mathbf{Nom}_k$ is the category of k-bounded nominal sets, and $\mathbf{nTop}_k$ is the category of k-bounded nominal topological spaces.

▶ Remark 3.8.

1. The full subcategories $\mathbf{Nom}_k \hookrightarrow \mathbf{Nom}$ and $\mathbf{Nom}_{\mathrm{of},k} \hookrightarrow \mathbf{Nom}_{\mathrm{of}}$ are coreflective [28, Section IV.3]: the coreflector (viz. the right adjoint of the inclusion functor) sends a nominal set X to its subset $X_k = \{x \in X \mid |\mathrm{supp}\, x| \leq k\}$. Hence $\mathbf{Nom}_k$ is complete: limits are formed by taking the limit in $\mathbf{Nom}$ and applying the coreflector. Analogously, $\mathbf{Nom}_{\mathrm{of},k}$ is finitely complete.
2. In contrast to $\mathbf{Nom}$, the category $\mathbf{Nom}_k$ generally fails to be a topos because it is not cartesian closed. For instance, the functor $\mathbb{A}^{\#2} \times (-)$ on $\mathbf{Nom}_2$ does not preserve coequalizers, hence it is not a left adjoint.
3. The category $\mathbf{Nom}$ is known to be equivalent to the category of pullback-preserving presheaves $\mathbb{I} \to \mathbf{Set}$, where $\mathbb{I}$ is the category of finite sets and injective functions [36, Theorem 6.8]. By inspecting the proof it is easy to see that this restricts to an equivalence between $\mathbf{Nom}_k$ and the category of *k-generated* pullback-preserving presheaves $\mathbb{I} \to \mathbf{Set}$. Here a presheaf $F \colon \mathbb{I} \to \mathbf{Set}$ is k-generated if for every finite set S and every $x \in FS$ there exists a set S' of cardinality at most k and an injective map $f \colon S' \to S$ such that $x \in Ff[FS']$.

With regard to codirected limits, the restriction to bounded nominal sets fixes the issue arising in Example 3.5:

▶ **Lemma 3.9.** *Codirected limits in $\mathbf{Nom}_k$ are formed at the level of $\mathbf{Set}$.*

We proceed to give a topological characterization of $\mathrm{Pro}(\mathbf{Nom}_{\mathrm{of},k})$ in terms of nominal Stone spaces, generalizing the corresponding result (1.1) for $\mathrm{Pro}(\mathbf{Set}_{\mathrm{f}})$. To this end, we introduce suitable nominalizations of the three characteristic properties of Stone spaces: compactness, Hausdorffness, and existence of a basis of clopens. The nominal version of compactness comes natural and is compatible with the functors $|-|_S$ and orb_S of Remark 3.3.

▶ **Definition 3.10.** *An* open cover *of a nominal topological space $(X, \mathcal{O})$ is a finitely supported set $\mathcal{C} \subseteq \mathcal{O}$ that covers X, i.e. $\bigcup \mathcal{C} = X$. A* subcover *of $\mathcal{C}$ is a finitely supported subset of $\mathcal{C}$ that also covers X. A nominal topological space X is* compact *if every open cover $\mathcal{C}$ of X has an orbit-finite subcover: there exist $U_1, \ldots, U_n \in \mathcal{C}$ such that $X = \bigcup_{i=1}^{n} \bigcup \mathrm{orb}\, U_i$.*

▶ **Lemma 3.11.** *For every nominal topological space X the following conditions are equivalent:*

1. *The space X is compact.*
2. *Every uniformly finitely supported open cover of X has a finite subcover.*
3. *For every finite set $S \subseteq \mathbb{A}$ the topological space $|X|_S$ is compact.*
4. *For every finite set $S \subseteq \mathbb{A}$ the topological space $\mathrm{orb}_S X$ is compact.*

The Hausdorff property is more subtle: rather than just separation of points, we require separation of S-orbits ("thick points") by disjoint S-supported open neighbourhoods.

▶ **Definition 3.12.** *A nominal topological space X is* (nominal) Hausdorff *if for every finite set $S \subseteq \mathbb{A}$ and every pair $x_1, x_2 \in X$ of points lying in different S-orbits, there exist disjoint S-supported open sets $U_1, U_2 \subseteq X$ such that $x_i \in U_i$ for $i = 1, 2$.*

Note that the nominal Hausdorff condition is clearly equivalent to being able to separate disjoint *S-orbits*: If $\mathrm{orb}_S\, x_1 \neq \mathrm{orb}_S\, x_2$, then any two disjoint open S-supported neighbourhoods U_1, U_2 of x_1, x_2 satisfy $\mathrm{orb}_S\, x_i \subseteq U_i$ for $i = 1, 2$. Note also that $\mathrm{orb}_S\, x = \{x\}$ whenever $\mathrm{supp}\, x \subseteq S$, hence the nominal Hausdorff condition implies the ordinary one. For bounded nominal compact Hausdorff spaces, we have a codirected Tychonoff theorem:

▶ **Proposition 3.13.** *For every codirected diagram of non-empty k-bounded nominal compact Hausdorff spaces, the limit in* **nTop** *is a non-empty k-bounded nominal compact Hausdorff space.*

Finally, having a basis of clopen sets is not sufficient in our setting. To see this, note that in an ordinary topological space X every clopen subset $C \subseteq X$ can be represented as $C = f^{-1}[A]$ for some continuous map $f\colon X \to Y$ into a finite discrete space Y and some subset $A \subseteq Y$. (In fact, one may always take $Y = 2$ and $A = \{1\}$.) This is no longer true in the nominal setting, see Remark 3.15 below. Therefore, in lieu of clopens we work with *representable* subsets:

▶ **Definition 3.14.** *A subset $R \subseteq X$ of a nominal space X is* representable *if there exists a continuous map $f\colon X \to Y$ into an orbit-finite discrete space Y such that $R = f^{-1}[A]$ for some $A \in \mathcal{P}_{fs} Y$.*

▶ Remark 3.15.

1. Every representable set is clopen, but the converse generally fails. To see this, consider the discrete space $X = \coprod_{n<\omega} \mathbb{A}^{\#n}$. We show that for fixed $a \in \mathbb{A}$ the (clopen) subset $R = \{x \mid a \in \mathrm{supp}\, x\} \subseteq X$ is not representable. Towards a contradiction suppose that R is represented by $f\colon X \to Y$ as $R = f^{-1}[A]$ for some $A \in \mathcal{P}_{\mathrm{fs}} Y$. Since Y is orbit-finite, we can choose m large enough such that there exists some $x \in \mathbb{A}^{\#m} \setminus R \subseteq X$ for which $\mathrm{supp}\, f(x) \subsetneq \mathrm{supp}\, x$. Choose a name $b \in \mathrm{supp}\, x \setminus \mathrm{supp}\, f(x)$. Then $a, b \notin \mathrm{supp}\, f(x)$, and so we have
$$f((a\ b) \cdot x) = (a\ b) \cdot f(x) = f(x).$$
Since $(a\ b) \cdot x \in R$, this shows $f(x) \in A$ and thus $x \in R$. This contradicts the above choice of x.
2. If a nominal space X has a basis of representable sets, then we may assume without loss of generality that the basic open sets are of the form $f^{-1}[y]$ for some $f\colon X \to Y$ and $y \in Y$, where Y is orbit-finite and discrete. Indeed, if $R = f^{-1}[A]$ for $A \in \mathcal{P}_{\mathrm{fs}} Y$, then $R = \bigcup_{y \in A} f^{-1}[y]$. Moreover, given representable sets $R_i = f_i^{-1}[y_i]$, $i = 1, 2$, the set $R_1 \cap R_2$ is equal to $\langle f_1, f_2 \rangle^{-1}[y_1, y_2]$ and therefore representable as well. Hence, to show that representable subsets form a basis it suffices to check whether every open set is a finitely supported union of subsets of the form $f^{-1}[y]$.

▶ **Definition 3.16.** *A* nominal Stone space *is a nominal compact Hausdorff space with a basis of representables. We let* **nStone** *denote the full subcategory of* **nTop** *given by nominal Stone spaces.*

▶ Remark 3.17. Nominal Stone spaces as per Definition 3.16 are conceptually very different from *nominal Stone spaces with* M, introduced by Gabbay et al. [17] as the dual of *nominal boolean algebras with* M. The latter are equipped with a *restriction operator* $\mathbf{n}$ tightly related to the freshness quantifier M of nominal sets, which enables a nominal version of the ultrafilter theorem and thus a represention of boolean algebras with M via spaces of ultrafilters. In nominal Stone spaces with M, the Hausdorff property is implicit (but would be analogous to that in standard topology), the basis is given by clopen rather than representable sets, and the notion of compactness (called $\mathbf{n}$-*compactness*) considers open covers closed under the operator $\mathbf{n}$, which are required to have a *finite* subcover. By this definition, the orbit-finite discrete space $\mathbb{A}$ fails to be compact (the $\mathbf{n}$-cover $\{\{a\} \mid a \in \mathbb{A}\} \cup \{\emptyset\}$ has no finite subcover). Hence, given that algebraic recognition is based on orbit-finite sets, nominal Stone spaces with M are not suitable for a topological interpretion of data languages.

▶ **Example 3.18.** Every orbit-finite nominal set can be viewed as a nominal Stone space equipped with the discrete topology. We thus regard $\mathbf{Nom}_{\mathrm{of}}$ as a full subcategory of **nStone**. Nontrivial examples of nominal Stone spaces are given by the spaces of pro-orbit-finite words introduced later.

Within the class of nominal Stone spaces, representable and clopen subsets coincide:

▶ **Lemma 3.19.** *If X is a nominal Stone space, then every clopen set $C \subseteq X$ is representable.*

The following theorem is the key result leading to our topological approach to data languages.

▶ **Theorem 3.20.** *For each $k \in \mathbb{N}$, the category of k-bounded nominal Stone spaces is the Pro-completion of the category of k-bounded orbit-finite nominal sets:*

$$\mathrm{Pro}(\mathbf{Nom}_{\mathrm{of},k}) = \mathbf{nStone}_k.$$

Moreover, k-bounded nominal Stone spaces are precisely the nominal topological spaces arising as codirected limits of k-bounded orbit-finite discrete spaces.

For $k = 0$, we recover the corresponding characterization of classical Stone spaces.

4 Nominal Stone Duality

Next, we give a dual characterization of (bounded) nominal Stone spaces. It builds on the known duality between nominal sets and complete atomic nominal boolean algebras due to Petrişan [32].

▶ **Definition 4.1.** *A* nominal boolean algebra *is a nominal set equipped with the structure of a boolean algebra such that all operations are equivariant. It is* (orbit-finitely) complete *if every (orbit-finite) finitely supported subset has a supremum. A* subalgebra *of an (orbit-finitely) complete nominal boolean algebra is an equivariant subset closed under boolean operations and the respective suprema. Let* $\mathbf{nC}_{\mathrm{of}}\mathbf{BA}$ *and* **nCBA** *denote the categories of (orbit-finitely) complete nominal boolean algebras; their morphisms are equivariant homomorphisms preserving (orbit-finite) suprema.*

▶ **Definition 4.2.** *An element $x \in B$ of a nominal boolean algebra is an* atom *if $x \neq \bot$ and $y < x$ implies $y = \bot$. The (equivariant) set of atoms of B is denoted* $\mathrm{At}\, B$. *The algebra B is* atomic *if every element is the supremum of all atoms below it; if additionally* $\mathrm{At}\, B \in \mathbf{Nom}_{\mathrm{of},k}$ *we call it* k-atomic. *If $A \subseteq B$ is a k-atomic subalgebra we write $A \leq_{\mathrm{of},k} B$. An algebra $B \in \mathbf{nCBA}$ is called* locally k-atomic *if every element of B is contained in some $A \leq_{\mathrm{of},k} B$. We denote by $\mathbf{nCA}_k\mathbf{BA} \subseteq \mathbf{nCBA}$ the full subcategory of all k-atomic complete nominal boolean algebras, and $\mathbf{nC}_{\mathrm{of}}\mathbf{A}_{\mathrm{l}k}\mathbf{BA} \subseteq \mathbf{nC}_{\mathrm{of}}\mathbf{BA}$ denotes the full subcategory of all locally k-atomic orbit-finitely complete nominal boolean algebras.*

▶ Remark 4.3.

1. Orbit-finite completeness is equivalent to the weaker condition that suprema of S-orbits exist for all finite subsets $S \subseteq \mathbb{A}$. In fact, every S-supported orbit-finite subset $X \subseteq B$ is a finite union $X = \bigcup_{i=1}^{n} \mathrm{orb}_S\, x_i$ of S-orbits, whence $\bigvee X = \bigvee_{i=1}^{n} \bigvee \mathrm{orb}_S\, x_i$.
2. Every k-atomic orbit-finitely complete nominal boolean algebra is complete: For every finitely supported subset $X \subseteq B$ we have $\bigvee X = \bigvee\{b \in \mathrm{At}(B) \mid \exists (x \in X).\ b \leq x\}$, which is a supremum of an orbit-finite subset.

▶ **Theorem 4.4.** *For each $k \in \mathbb{N}$, the category of locally k-atomic orbit-finitely complete nominal boolean algebras is the Ind-completion of the category of k-atomic complete nominal boolean algebras:*

$$\mathbf{nC}_{\mathrm{of}}\mathbf{A}_{\mathrm{l}k}\mathbf{BA} \simeq \mathrm{Ind}(\mathbf{nCA}_k\mathbf{BA}).$$

▶ **Theorem 4.5** (Nominal Stone Duality)**.** *For each $k \in \mathbb{N}$, the category of locally k-atomic orbit-finitely complete nominal boolean algebras is dual to the category of k-bounded nominal Stone spaces:*

$$\mathbf{nC}_{\mathrm{of}}\mathbf{A}_{\mathrm{l}k}\mathbf{BA} \simeq^{\mathrm{op}} \mathbf{nStone}_k.$$

Proof. The category **Nom** of nominal sets is dually equivalent to the category **nCABA** of complete atomic nominal boolean algebras [32]. The duality sends a nominal set X to the boolean algebra $\mathcal{P}_{\mathrm{fs}}X$, equippped with the set-theoretic boolean structure. Conversely, a complete atomic nominal boolean algebra B is mapped to the nominal set $\mathrm{At}(B)$ of its atoms, and an **nCABA**-morphism $h\colon C \to B$ to the equivariant map $\mathrm{At}(B) \to \mathrm{At}(C)$ sending $b \in \mathrm{At}(B)$ to the unique $c \in \mathrm{At}(C)$ such that $c \leq h(b)$. For every $k \in \mathbb{N}$ the duality clearly restricts to one between k-bounded orbit-finite nominal sets and k-atomic complete nominal boolean algebras. Thus Theorem 4.4 and Theorem 3.20 yield

$$\mathbf{nC}_{\mathrm{of}}\mathbf{A}_{\mathrm{l}k}\mathbf{BA} \simeq \mathrm{Ind}(\mathbf{nCA}_k\mathbf{BA}) \simeq^{\mathrm{op}} \mathrm{Pro}(\mathbf{nCA}_k\mathbf{BA}^{\mathrm{op}}) \simeq \mathrm{Pro}(\mathbf{Nom}_{\mathrm{of},k}) \simeq \mathbf{nStone}_k. \quad \blacktriangleleft$$

▶ Remark 4.6. We give an explicit description of the dual equivalence of Theorem 4.5.

1. In the direction $\mathbf{nStone}_k \to \mathbf{nC}_{\mathrm{of}}\mathbf{A}_{\mathrm{l}k}\mathbf{BA}$ it maps a k-bounded nominal Stone space X to the nominal boolean algebra $\mathrm{Clo}(X)$ of clopens (or representables, see Lemma 3.19). A continuous map $f\colon X \to Y$ is mapped to the homomorphism $f^{-1}\colon \mathrm{Clo}(Y) \to \mathrm{Clo}(X)$ taking preimages.
2. The direction $\mathbf{nC}_{\mathrm{of}}\mathbf{A}_{\mathrm{l}k}\mathbf{BA} \to \mathbf{nStone}_k$ requires some terminology. A finitely supported subset $F \subseteq B$ of an algebra $B \in \mathbf{nC}_{\mathrm{of}}\mathbf{A}_{\mathrm{l}k}\mathbf{BA}$ is a *nominal orbit-finitely complete prime filter* if (i) $F \neq \emptyset$, (ii) F is upwards closed ($x \in F \wedge x \leq y \Rightarrow y \in F$), (iii) F is downwards directed ($x, y \in F \Rightarrow x \wedge y \in F$), and (iv) for every finitely supported k-bounded orbit-finite subset $X \subseteq B$ such that $\bigvee X \in F$, one has $X \cap F \neq \emptyset$. The equivalence now maps $B \in \mathbf{nC}_{\mathrm{of}}\mathbf{A}_{\mathrm{l}k}\mathbf{BA}$ to the space $\mathcal{F}_{\mathrm{np}}(B)$ of nominal orbit-finitely complete prime

filters of B, whose topology is generated by the basic open sets $\{F \in \mathcal{F}_{\mathrm{np}}(B) \mid b \in F\}$ for $b \in B$. A morphism $h\colon B \to C$ of $\mathbf{nC}_{\mathrm{of}}\mathbf{A}_{\mathrm{l}k}\mathbf{BA}$ is mapped to the continuous map $h^{-1}\colon \mathcal{F}_{\mathrm{np}}(C) \to \mathcal{F}_{\mathrm{np}}(B)$ taking preimages.

In Theorem 4.5 we made the support bound k explicit, but we can also leave it implicit. A nominal Stone space is *bounded* if it lies in $\mathbf{nStone}_k$ for some natural number k; similarly, a *locally bounded atomic orbit-finitely complete nominal boolean algebras* is an element of $\mathbf{nC}_{\mathrm{of}}\mathbf{A}_{\mathrm{l}k}\mathbf{BA}$ for some k.

▶ **Corollary 4.7.** *The category of locally bounded atomic orbit-finitely complete nominal boolean algebras is dual to the category of bounded nominal Stone spaces.*

▶ Remark 4.8. For $k = 0$ we recover the classical Stone duality between boolean algebras and Stone spaces. Indeed, 0-bounded nominal Stone spaces are precisely Stone spaces, and locally 0-atomic orbit-finitely complete nominal boolean algebras are precisely boolean algebras.

5 Pro-Orbit-Finite Words

In this section, we generalize the topological characterization of regular languages to data languages recognizable by orbit-finite nominal monoids [7, 10, 13].

▶ **Definition 5.1.** *A* nominal monoid M *is a monoid object in* $\mathbf{Nom}$*, that is, it is given by nominal set M equipped with an equivariant associative multiplication $M \times M \to M$ and an equivariant unit $1 \in M$. Nominal monoids and equivariant monoid homomorphisms form a category* $\mathbf{nMon}$*.*

As for ordinary monoids, the *free monoid* generated by $\Sigma \in \mathbf{Nom}$ is the nominal set Σ^* of finite words (with pointwise group action); its multipliation is concatenation and its unit the empty word.

▶ Remark 5.2. We emphasize the difference between k-bounded nominal monoids – nominal monoids whose carrier is k-bounded – and monoid objects in $\mathbf{Nom}_k$, which are partial nominal monoids where the product $x \cdot y$ is defined iff $|\operatorname{supp} x \cup \operatorname{supp} y| \leq k$.

▶ **Definition 5.3.** *A* data language *over $\Sigma \in \mathbf{Nom}_{\mathrm{of}}$ is a finitely supported subset $L \subseteq \Sigma^*$. It is* recognizable *if there exists an equivariant monoid morphism $h\colon \Sigma^* \to M$ with M orbit-finite and a finitely supported subset $P \subseteq M$ such that $L = h^{-1}[P]$. In this case, we say that the morphism h* recognizes *L.*

For example, the equivariant language L_0 from the Introduction is recognizable, while the language L_1 is not recognizable.

▶ Remark 5.4.

1. The morphism h can be taken to be surjective; otherwise, take its coimage.
2. Via characteristic functions, data languages correspond precisely to finitely supported maps $L\colon \Sigma^* \to 2$, where 2 is the two-element nominal set. Recognizablity then states that L factorizes through some equivariant monoid morphism with orbit-finite codomain.

Recall from Section 2 that the Stone space $\widehat{\Sigma^*}$ of profinite words over a finite alphabet Σ is constructed as the limit in $\mathbf{Stone} \simeq \mathrm{Pro}(\mathbf{Set}_{\mathrm{f}})$ of all finite quotient monoids of Σ^*. The obvious generalization to a *nominal* alphabet $\Sigma \in \mathbf{Nom}_{\mathrm{of}}$, which constructs the limit of all orbit-finite quotient monoids in $\mathrm{Pro}(\mathbf{Nom}_{\mathrm{of}})$, is unlikely to yield a useful object since this category is not concrete (Proposition 3.6); in fact, it is futile from a language-theoretic

perspective, cf. Remark 5.15. Instead, our results of Section 3 suggest to restrict the diagram scheme to $\Sigma^*{\downarrow}\mathbf{nMon}_{\mathrm{of},k}$, the poset of k-bounded orbit-finite quotient monoids (where $e \leq e'$ iff e' factorizes through e), and take the limit in $\mathrm{Pro}(\mathbf{Nom}_{\mathrm{of},k}) = \mathbf{nStone}_k$. However, this diagram is not codirected, so its limit may not be a nominal Stone space. We again focus on well-behaved (i.e., codirected), subcategories by introducing *support bounds*.

▶ **Definition 5.5.** *A* support bound *is a map $s\colon \Sigma^* \to \mathcal{P}\mathbb{A}$ such that $s[\Sigma^*] \subseteq \mathcal{P}_k\mathbb{A}$ for some $k \in \mathbb{N}$, where $\mathcal{P}_k\mathbb{A} = \{S \subseteq \mathbb{A} \mid |S| \leq k\}$. We usually identify s with its codomain restrictions to $\mathcal{P}_k\mathbb{A}$ for sufficiently large k. A morphism $h\colon \Sigma^* \to M$ of nominal monoids is* s-bounded *if $\operatorname{supp} h(w) \subseteq s(w)$ for all $w \in \Sigma^*$; we write $h\colon \Sigma^* \to_s M$. We denote by $\Sigma^*{\downarrow}_s\mathbf{nMon}_{\mathrm{of},k}$ the subposet of $\Sigma^*{\downarrow}\mathbf{nMon}_{\mathrm{of},k}$ given by s-bounded quotient monoids.*

▶ **Lemma 5.6.** *For every support bound s, the poset $\Sigma^*{\downarrow}_s\mathbf{nMon}_{\mathrm{of},k}$ is codirected.*

Proof. Let $h\colon \Sigma^* \to_s M_h$ and $h'\colon \Sigma^* \to_s M_{h'}$ be two s-bounded quotients in $\Sigma^*{\downarrow}_s\mathbf{nMon}_{\mathrm{of},k}$. Form the coimage $k\colon \Sigma^* \twoheadrightarrow M$ of their pairing $\langle h, h'\rangle\colon \Sigma^* \to M_h \times M_{h'}$. Then for all $w \in \Sigma^*$

$$\operatorname{supp} k(w) = \operatorname{supp}(h(w), h'(w)) = \operatorname{supp} h(w) \cup \operatorname{supp} h'(w) \subseteq s(w).$$

Hence, k is a lower bound for h, h' in the poset $\Sigma^*{\downarrow}_s\mathbf{nMon}_{\mathrm{of},k}$. ◀

▶ **Definition 5.7.** *For an orbit-finite nominal set Σ and a support bound $s\colon \Sigma^* \to \mathcal{P}_k\mathbb{A}$ we define the nominal Stone space $\widehat{\Sigma^*_s}$ to be the limit of the codirected diagram*

$$D\colon \Sigma^*{\downarrow}_s\mathbf{nMon}_{\mathrm{of},k} \to \mathbf{nStone}_k, \qquad (e\colon \Sigma^* \twoheadrightarrow_s M) \mapsto |M|,$$

where $|M|$ is the nominal set underlying M, regarded as a discrete nominal topological space. The elements of $\widehat{\Sigma^*_s}$ are called the (s-bounded) pro-orbit-finite words over Σ. *We denote by $\hat{e}\colon \widehat{\Sigma^*_s} \to M$ the limit projection associated to $e\colon \Sigma^* \twoheadrightarrow_s M$ in $\Sigma^*{\downarrow}_s\mathbf{nMon}_{\mathrm{of},k}$.*

▶ Remark 5.8.

1. One may equivalently define $\widehat{\Sigma^*_s}$ as the limit of the larger cofiltered diagram D' given by

   $$D'\colon \Sigma^*{\downarrow}_s\mathbf{nMon}_{\mathrm{of},k} \to \mathbf{nStone}_k, \qquad (e\colon \Sigma^* \to_s M) \mapsto |M|,$$

   where $\Sigma^*{\downarrow}_s\mathbf{nMon}_{\mathrm{of},k}$ is the category of all equivariant s-bounded monoid morphisms $h\colon \Sigma^* \to_s M$ with k-bounded orbit-finite codomain; a morphism from h to $h'\colon \Sigma^* \to_s M'$ is an equivariant monoid morphism $k\colon M \to M'$ such that $h' = k \cdot h$. In fact, the inclusion $\Sigma^*{\downarrow}_s\mathbf{nMon}_{\mathrm{of},k} \hookrightarrow \Sigma^*{\downarrow}_s\mathbf{nMon}_{\mathrm{of},k}$ is an initial functor, hence the limits of D and D' coincide. Since the limit of D' is formed as in **Set** (Lemma 3.9), the space $\widehat{\Sigma^*_s}$ is carried by the nominal set of compatible families $(x_h)_h$ of D', and the limit projection $\hat{h}$ associated to $h\colon \Sigma^* \to_s M$ is given by $(x_h)_h \mapsto x_h$.
2. The forgetful functor $V\colon \mathbf{nStone}_k \to \mathbf{Nom}_k$ and the inclusion $I\colon \mathbf{Nom}_k \to \mathbf{Nom}$ both preserve codirected limits. The morphisms $\Sigma^*{\downarrow}_s\mathbf{nMon}_{\mathrm{of},k}$ viewed as equivariant functions form a cone for the diagram IVD', so there exists a unique equivariant map $\eta\colon \Sigma^* \to IV\widehat{\Sigma^*_s}$ such that

   $$h = \big(\Sigma^* \dashrightarrow^{\eta} IV\widehat{\Sigma^*_s} \xrightarrow{IV\hat{h}} IVM\big) \qquad \text{for all } h \in \Sigma^*{\downarrow}_s\mathbf{nMon}_{\mathrm{of},k}.$$

   In more explicit terms, the map η is given by $\eta(w) = (h(w))_h$ for $w \in \Sigma^*$. For simplicity we omit I and V and write $\eta\colon \Sigma^* \to \widehat{\Sigma^*_s}$. The image of η forms a dense subset of $\widehat{\Sigma^*_s}$. We note that η is generally not injective since we restrict to a subdiagram $\Sigma^*{\downarrow}_s\mathbf{nMon}_{\mathrm{of},k}$ of the diagram $\Sigma^*{\downarrow}\mathbf{nMon}_{\mathrm{of}}$,

3. The space $\widehat{\Sigma^*_s}$ is a nominal monoid with product $\hat{h}(x \cdot y) = \hat{h}(x) \cdot \hat{h}(y)$ and unit $\eta(\varepsilon)$, with ε the empty word. Since the multiplication is readily seen to be continuous, $\widehat{\Sigma^*_s}$ can be regarded as an object of $\mathbf{Mon}(\mathbf{nStone})$, the category of nominal Stone spaces equipped with a continuous monoid structure and continuous equivariant monoid morphisms.

Now recall from Section 2 that the space $\widehat{\Sigma^*}$ can be constructed as the metric completion of Σ^*, where the metric measures the size of separating monoids. We now investigate to what extent the metric approach applies to the nominal setting, using nominal (pseudo-)metrics; see Example 3.2.

▶ **Definition 5.9.** *Let s be a support bound on Σ^*. We say that a nominal monoid M s-*separates *$v, w \in \Sigma^*$ if there exists an s-bounded equivariant monoid morphism $h\colon \Sigma^* \to_s M$ such that $h(v) \neq h(w)$. We define a nominal pseudometric d_s on Σ^* by setting*

$$d_s(v,w) = \sup\{\, 2^{-|\operatorname{orb} M|} \mid \text{the orbit-finite nominal monoid } M \text{ } s\text{-separates } v, w \,\}.$$

We let Σ^*/d_s denote the corresponding nominal metric space, obtained as a quotient space of the pseudometric space (Σ^*, d_s) by identifying v, w if $d_s(v,w) = 0$.

▶ Remark 5.10. In contrast to the classical case, d_s is generally not a metric: there may exist words $v \neq w$ which are not s-separated by any orbit-finite nominal monoids. For example, if $\Sigma = \mathbb{A}$ and $s(a_1 \cdots a_n) = a_1$ for $a_1, \ldots, a_n \in \Sigma$, then for every s-bounded h and distinct names $a, b, c \in \mathbb{A}$ we have $h(ab) = h((b\ c) \cdot ac) = (b\ c) \cdot h(ac) = h(ac)$ since $b, c \notin s(ac) \supseteq \operatorname{supp} h(ac)$. Therefore, the additional metrization process is required.

For the next lemma we need some terminology. A nominal metric space is *complete* if every finitely supported Cauchy sequence has a limit. A nominal topological space is *completely metrizable* if its topology is induced by a complete metric. A subset $D \subseteq X$ of a nominal metric space is *(topologically) dense* if every open neighbourhood of a point $x \in X$ contains an element of D.

▶ Remark 5.11. In contrast to classical metric spaces, density is not equivalent to *sequential density* (every point $x \in X$ is a limit of a finitely supported sequence in D). To see this, consider the space $\mathbb{A}^\omega$ of finitely supported infinite words with the prefix metric, that is, $d(v,w) = 2^{-n}$ if n is the length of the longest common prefix of v, w. Let $D \subseteq X$ be the equivariant subset given by

$$D = \{\, x \in \mathbb{A}^\omega \mid |\operatorname{supp} x| \geq 2 \text{ and } |\operatorname{supp} x| \geq |\operatorname{initialblock}(x)| \,\},$$

where $\operatorname{initialblock}(x)$ is the longest prefix of x of the form a^n ($a \in \mathbb{A}$). The set D is dense, but not sequentially dense: $a^\omega \in \mathbb{A}^\omega$ is not the limit of any finitely supported sequence in D.

▶ **Lemma 5.12.**

1. *The space $\widehat{\Sigma^*_s}$ is completely metrizable via the complete nominal metric*

$$\hat{d}_s(x,y) = \sup\{\, 2^{-|\operatorname{orb} M|} \mid \exists (h\colon \Sigma^* \to_s M)\colon \hat{h}(x) \neq \hat{h}(y) \,\}. \tag{5.1}$$

2. *The canonical map η (Remark 5.8) yields a dense isometry $\eta\colon (\Sigma^*, d_s) \to (\widehat{\Sigma^*_s}, \hat{d}_s)$.*

▶ Remark 5.13. In classical topology, it would now be clear that $\widehat{\Sigma^*_s}$ is the metric completion of the metric space Σ^*/d_s, i.e. it satisfies the universal property that every uniformly continuous map from Σ^*/d_s to a complete metric space has a unique uniformly continuous extension to $\widehat{\Sigma^*_s}$. However, this rests on the coincidence of topological and sequential density, which fails over nominal sets as seen in Remark 5.11. We therefore conjecture that $\widehat{\Sigma^*_s}$ is not the nominal metric completion of Σ^*/d_s.

By using support bounds, we obtain a topological perspective on recognizable data languages. Let $\mathrm{Rec}_s\,\Sigma$ denote the set of data languages recognized by s-bounded equivariant monoid morphisms.

▶ **Theorem 5.14.** *For every support bound $s\colon \Sigma^* \to \mathcal{P}_k\mathbb{A}$, the k-bounded nominal Stone space $\widehat{\Sigma^*_s}$ of s-bounded pro-orbit-finite words is dual to the locally k-atomic orbit-finitely complete boolean algebra $\mathrm{Rec}_s(\Sigma^*)$ of s-recognizable languages. In particular, we have the isomorphism*

$$\mathrm{Rec}_s(\Sigma^*) \;\cong\; \mathrm{Clo}(\widehat{\Sigma^*_s}) \qquad \textit{in} \qquad \mathbf{nC}_{\mathrm{of}}\mathbf{A}_{\mathrm{l}k}\mathbf{BA}.$$

Proof (Sketch). The isomorphism is illustrated by the two diagrams below:

$$\begin{array}{ccccccc} L & = h^{-1}[P] & \subseteq & \Sigma^* & \xrightarrow{h} & M & \supseteq\ P \\ \downarrow & & & {\scriptstyle\eta}\downarrow & \nearrow_{\hat h} & & \\ \overline{\eta[L]} & = \hat h^{-1}[P] & \subseteq & \widehat{\Sigma^*_s} & & & \end{array} \qquad \begin{array}{ccccccc} \eta^{-1}[C] & = h^{-1}[P] & \subseteq & \Sigma^* & \xrightarrow{h} & M & \supseteq\ P = p^{-1}[U] \\ \updownarrow & & & {\scriptstyle\eta}\downarrow & \nearrow_{\hat h} & \downarrow{\scriptstyle\exists p} & \\ C & = f^{-1}[U] & \subseteq & \widehat{\Sigma^*_s} & \xrightarrow{f} & Y & \supseteq\ U \end{array}$$

In more detail, if $L \subseteq \Sigma^*$ is s-recognizable, say $L = h^{-1}[P]$ for an s-bounded morphism h, then its corresponding clopen is the topological closure $\overline{\eta[L]} = \hat h^{-1}[P]$ represented by the continuous extension $\hat h$. Conversely, every clopen $C \subseteq \widehat{\Sigma^*_s}$ restricts to an s-recognizable language $\eta^{-1}[C] \subseteq \Sigma^*$. We get s-recognizability of $\eta^{-1}[C]$ by factorizing a representation $f\colon \widehat{\Sigma^*_s} \to Y$ of C through a limit projection $\hat h$ as $f = p\cdot\hat h$, using that Y is finitely copresentable. Thus h recognizes $\eta^{-1}[C]$. ◀

▶ Remark 5.15. In the proof of Theorem 5.14, finite copresentability of orbit-finite sets is crucial to recover recognizable languages from representable subsets, highlighting the importance of working in the Pro-completion $\mathrm{Pro}(\mathbf{Nom}_{\mathrm{of},k}) = \mathbf{nStone}_k$. In a naive approach one might instead want to consider the limit of the diagram $D\colon \Sigma^*{\downarrow}\mathbf{nMon}_{\mathrm{of}} \to \mathbf{nTop}$ of *all* equivariant morphisms from Σ^* to orbit-finite monoids. The resulting space $\widehat{\Sigma^*}$ is still a nominal Hausdorff space with a basis of representables, but it generally fails to be compact, and its representable subsets do not correspond to recognizable data languages. To see this, consider the space $\widehat{\mathbb{A}^*}$ and the orbit-finite nominal monoids $\mathbb{A}^{\leq n}$ (words of length at most n) with multiplication cutting off after n letters. We denote by $h_n\colon \mathbb{A}^* \to \mathbb{A}^{\leq n}$ and $p_{k,n}\colon \mathbb{A}^{\leq k} \twoheadrightarrow \mathbb{A}^{\leq n}$, $n \leq k$, the equivariant monoid morphisms given by projection to the first n letters. For every compatible family $x = (x_h) \in \widehat{\mathbb{A}^*}$ its subfamily $(x_{h_n})_{n\in\mathbb{N}}$ corresponds to a (possibly infinite) word over $\mathbb{A}$ with finite support. Hence there exists a largest natural number $N = N(x)$ such that $|\operatorname{supp} x_{h_N}| = N$. The subsets $C_n = \{x \in \widehat{\mathbb{A}^*} \mid N(x) = n\}$, $n \in \mathbb{N}$, are equivariant clopens since $C_n = \hat h_n^{-1}[\mathbb{A}^{\#n}] \cap \hat h_{n+1}^{-1}[\mathbb{A}^{\leq n+1} \setminus \mathbb{A}^{\#(n+1)}]$. Thus each C_n is representable (by a continuous map into the two-element discrete space), non-empty (since $\eta(w) = (h(w))_h \in C_n$ for every word $w \in \mathbb{A}^{\#n} \subseteq \mathbb{A}^*$ of pairwise distinct letters), and pairwise disjoint. Hence they form a cover of $\widehat{\mathbb{A}^*}$ that admits no orbit-finite (equivalently, finite) subcover, showing that $\widehat{\mathbb{A}^*}$ is not compact. Moreover, the sets $C_M = \bigcup_{m\in M} C_m$, where $M \subseteq \mathbb{N}$, are equivariant clopens (hence representable) and pairwise distinct. Thus $\widehat{\mathbb{A}^*}$ has uncountably many clopens. On the other hand, there exist only countably many recognizable languages over $\mathbb{A}$ (using that, up to isomorphism, there exist only countably many orbit-finite sets [36, Thm. 5.13] and thus countably many orbit-finite nominal monoids), showing that there is no bijective correspondence between representable sets in $\widehat{\mathbb{A}^*}$ and recognizable data languages over $\mathbb{A}$.

6 A Nominal Reiterman Theorem

As an application of pro-orbit-finite methods, we present a nominal extension of Reiterman's classical pseudovariety theorem [37]. The latter characterizes classes of finite algebras presentable by profinite equations as precisely those closed under finite products, subalgebras, and homomorphic images. This result has been generalized to first-order structures [34] and, recently, to abstract categories [1, 30]. A key insight for the categorical perspective is that equations should be formed over projective objects. (Recall that an object X in a category is projective w.r.t. a class $\mathcal{E}$ of morphisms if for all cospans $X \xrightarrow{f} Y \xleftarrow{e} Z$ with $e \in \mathcal{E}$ there exists a factorization of f through e.) In **Nom**, one takes strong nominal sets, which are projective with respect to support-reflecting quotients (see Definition 6.1.2). For spaces of pro-orbit-finite words we have the support bound as an additional constraint, which makes the situation more complex: In a cospan $\widehat{\Sigma_s^*} \xrightarrow{\hat{h}} N \xleftarrow{e} M$ with e support-reflecting, no s-bounded factorization of $\hat{h}$ through e may exist. Surprisingly, there nonetheless exists a suitable type of quotients for nominal monoids, called *MSR quotients*, which is *independent* of the support bound s.

▶ **Definition 6.1.** *A surjective equivariant morphism $e\colon M \twoheadrightarrow N$ of nominal monoids is*

1. support-preserving *if* $\operatorname{supp} e(x) = \operatorname{supp} x$ *for every* $x \in X$*;*
2. support-reflecting *if for every* $y \in Y$ *there exists* $x \in e^{-1}[y]$ *such that* $\operatorname{supp} x = \operatorname{supp} y$*;*
3. multiplicatively support-reflecting *(*MSR *for short) if there exists a nominal submonoid* $M' \subseteq M$ *such that the domain restriction* $e|_{M'}\colon M' \to N$ *of* e *is surjective and support-preserving.*

▶ Remark 6.2. Note that a surjective morphism e is support-reflecting iff it restricts to a support-preserving surjection $e|_{M'}$ for some equivariant subset $M' \subseteq M$. For MSR morphisms one additionally requires that M' may be chosen to form a submonoid. So the implications

$$\text{support-preserving} \implies \text{multiplicatively support-reflecting} \implies \text{support-reflecting}$$

hold, but none of the two converses holds in general; for the first one consider the morphism $\mathbb{A}^* \twoheadrightarrow 1$ into the trivial monoid, and for the second one see Example 6.11.

▶ **Proposition 6.3.** *A surjective equivariant morphism $e\colon M \twoheadrightarrow N$ between orbit-finite nominal monoids is MSR iff all the monoids $\widehat{\Sigma_s^*}$ (where $\Sigma \in \mathbf{Nom}_{\mathrm{of}}$ is strong and $s\colon \Sigma^* \to \mathcal{P}\mathbb{A}$ is a support bound) are projective with respect to e in $\mathbf{Mon}(\mathbf{nStone})$, with M and N regarded as discrete spaces.*

▶ **Definition 6.4.** *An* MSR-pseudovariety of nominal monoids *is a class $\mathcal{V} \subseteq \mathbf{nMon}_{\mathrm{of}}$ of orbit-finite nominal monoids closed under*

1. *finite products: if $M_1, \dots, M_n \in \mathcal{V}$, $n \in \mathbb{N}$, then $M_1 \times \cdots \times M_n \in \mathcal{V}$;*
2. *submonoids: if $M \in \mathcal{V}$ and $N \subseteq M$ is a nominal submonoid, then $N \in \mathcal{V}$:*
3. *MSR quotients: if $M \in \mathcal{V}$ and $e\colon M \twoheadrightarrow N$ is an MSR quotient, then $N \in \mathcal{V}$.*

▶ **Definition 6.5.** *Let $s\colon \Sigma^* \to \mathcal{P}\mathbb{A}$ be a support bound. A* morphic pro-orbit-finite equation, *or* morphic proequation *for short, is a surjective* **nStone***-morphism $\varphi\colon \widehat{\Sigma_s^*} \twoheadrightarrow E$. An orbit-finite monoid M* satisfies *φ if for every s-bounded morphism $h\colon \Sigma^* \to M$, the limit projection $\hat{h}\colon \widehat{\Sigma_s^*} \to M$ factorizes through φ in $\mathbf{nStone}_k$, for some $k \in \mathbb{N}$ such that $M \in \mathbf{Nom}_{\mathrm{of},k}$ and s corestricts to $\mathcal{P}_k\mathbb{A}$:*

$$\hat{h} = \big(\widehat{\Sigma_s^*} \xrightarrow{\varphi} E \overset{\exists}{\dashrightarrow} M\big).$$

For a set $\mathcal{T}$ of morphic proequations, taken over possibly different $\widehat{\Sigma_s^*}$, we denote by $\mathcal{V}(\mathcal{T})$ the class of orbit-finite monoids satisfying all proequations in $\mathcal{T}$. A class $\mathcal{V}$ of orbit-finite monoids is presentable by morphic proequations *if $\mathcal{V} = \mathcal{V}(\mathcal{T})$ for some set $\mathcal{T}$ of morphic proequations.*

Note that proequations use support bounds, while the definition of an MSR-pseudovariety does not.

▶ **Theorem 6.6** (Nominal Reiterman). *A class of orbit-finite nominal monoids is an MSR-pseudovariety iff it is presentable by morphic proequations.*

The main technical observations for the proof are that (i) every orbit-finite set is k-bounded for some k, hence finitely copresentable in $\mathbf{nStone}_k$, and (ii) there are "enough" proequations in the sense that every orbit-finite nominal monoid is a quotient of some $\widehat{\Sigma_s^*}$. The quotient is not necessarily MSR, which entails that abstract pseudovariety theorems [1, 30] do not apply to our present setting.

We also give a syntactic version of our nominal Reiterman theorem, which uses explicit proequations in lieu of morphic proequations.

▶ **Definition 6.7.** *An* explicit proequation *is a pair $(x, y) \in \widehat{\Sigma_s^*} \times \widehat{\Sigma_s^*}$ for some strong $\Sigma \in \mathbf{Nom}_{\mathrm{of}}$ and some support bound s, denoted by $x = y$. An orbit-finite monoid M* satisfies *the explicit proequation $x = y$ if*

$$\hat{h}(x) = \hat{h}(y) \qquad \text{for every } s\text{-bounded equivariant monoid morphism } h\colon \Sigma^* \to M.$$

(Here choose a common support size bound k for M and s, so that $\hat{h}$ lies in $\mathbf{nStone}_k$.)

▶ **Theorem 6.8** (Explicit Nominal Reiterman). *A class of orbit-finite nominal monoids is an MSR-pseudovariety iff it is presentable by explicit proequations.*

▶ **Example 6.9.** Recall that in a finite monoid M every element m has a unique idempotent power, denoted by m^ω. This holds analogously for orbit-finite nominal monoids M [7, Theorem 5.1]: one has $m^\omega = m^{(n\cdot k!)!}$ where n is the number of orbits M and k is the maximum support size. (The number $n \cdot k!$ is an upper bound on the number of elements of M with any given finite support [36, Thm. 5.13], hence on the cardinality of the set $\{m^i : i \in \mathbb{N}\}$.) The nominal monoid M is *aperiodic* if $m^\omega \cdot m = m^\omega$ for all $m \in M$. Languages recognizable by aperiodic orbit-finite monoids are captured precisely by first-order logic on data words [7, 13]. One readily verifies that the class of aperiodic orbit-finite monoids forms an MSR-pseudovariety; in fact, it is closed under all quotients. To present it by pro-orbit-finite equations, note that for every $x \in \widehat{\Sigma_s^*}$ the family $x^\omega = (\hat{h}(x)^\omega)_h$ is again compatible, hence $x^\omega \in \widehat{\Sigma_s^*}$. If $s\colon \Sigma^* \to \mathcal{P}_k\mathbb{A}$ and $h\colon \Sigma^* \to_s M$ is an s-bounded equivariant monoid morphism such that M has at most n orbits, then $\hat{h}(x^\omega) = \hat{h}(x)^\omega = \hat{h}(x)^{(n\cdot k!)!} = \hat{h}(x^{(n\cdot k!)!})$, hence $\hat{d}_s(x^\omega, x^{(n\cdot k!)!}) < 2^{-n}$ in the metric (5.1) on $\widehat{\Sigma_s^*}$. This shows that x^ω is the limit of the sequence $(x^{(n\cdot k!)!})_{n\in\mathbb{N}}$ in $\widehat{\Sigma_s^*}$, and moreover that the pseudovariety of aperiodic orbit-finite monoids is presented by the explicit proequations $x^\omega \cdot x = x^\omega$, where $x \in \widehat{\Sigma_s^*}$ and $s\colon \Sigma^* \to \mathcal{P}_k\mathbb{A}$ ranges over all support bounds on strong orbit-finite alphabets. Restricting to $k = 0$, we recover the well-known description of aperiodic finite monoids by the (single) profinite equation $x^\omega \cdot x = x^\omega$.

▶ Remark 6.10. **1.** Pseudovarieties of finite monoids admit an alternative equational characterization based on sequences of word equations rather than profinite equations. A *word equation* is a pair $(v, w) \in \Sigma^* \times \Sigma^*$ of words over some finite alphabet Σ, denoted $v = w$; it is *satisfied* by a monoid M if $h(v) = h(w)$ for every monoid morphism $h\colon \Sigma^* \to M$.

More generally, a sequence $(v_0 = w_0, v_1 = w_1, \ldots)$ of word equations, taken over possibly different finite alphabets, is *eventually satisfied* by M if it satisfies all but finitely many of the equations. As shown by Eilenberg and Schützenberger [15], a class of finite monoids forms a pseudovariety iff it is presentable by a (single) sequence of word equations.

2. Recently, a nominal version of the Eilenberg-Schützenberger theorem by Urbat and Milius [44]. They consider nominal word equations (defined as above, where Σ is now a strong orbit-finite nominal set) and show that sequences of nominal word equations present precisely *weak pseudovarieties*, i.e. classes of orbit-finite nominal monoids closed under finite products, submonoids, and support-reflecting quotients. Clearly every MSR-pseudovariety is weak, but the converse does not hold; hence over nominal sets, sequences of word equations and pro-orbit-finite equations are of different expressivity. The example below illustrates one source of additional expressivity of pro-orbit-finite equations: The support bound s can control how the support changes during multiplication, which is not expressible by sequences of word equations.

▶ **Example 6.11.** An example of an MSR-pseudovariety that is not a weak pseudovariety is given by the class $\mathcal{V}$ of all orbit-finite nominal monoids M such that

$$\forall (m, n \in M)\colon \quad \operatorname{supp}(mn) = \emptyset \quad \Longleftrightarrow \quad \operatorname{supp}(m, n) = \emptyset. \tag{6.1}$$

(Note that $\operatorname{supp}(m, n) = \operatorname{supp} m \cup \operatorname{supp} n$ and that "$\Leftarrow$" always holds by equivariance of the monoid multiplication.) It is not difficult to prove that $\mathcal{V}$ is an MSR-pseudovariety. To show that $\mathcal{V}$ is not a weak pseudovariety, we construct a support-reflecting quotient under which $\mathcal{V}$ is not closed. The nominal set $\overline{1} + \overline{\mathbb{A}} = \{\overline{1}\} + \{\overline{a} \mid a \in \mathbb{A}\}$ forms a nominal monoid with multiplication given by projection on the first component and unit $\overline{1}$. We extend the multiplication to the nominal set $M = 1 + \mathbb{A} + \overline{1} + \overline{\mathbb{A}}$ by letting 1 be the unit and setting $x \cdot y = \overline{x} \cdot \overline{y}$ whenever $x, y \neq 1$; here overlining is idempotent ($\overline{\overline{x}} := \overline{x}$). This makes the multiplication associative and equivariant. Thus, M is a nominal monoid. Now let $N = 1 + \mathbb{A} + 0 = \{1\} + \mathbb{A} + \{0\}$ be the nominal monoid with multiplication $x \cdot y = 0$ for $x, y \neq 1$. Thus 0 is an absorbing element. Letting $\mathrm{const}_0\colon \overline{1} + \overline{\mathbb{A}} \to 0$ denote the constant map, we have the equivariant surjective map

$$e = \mathrm{id}_{1+\mathbb{A}} + \mathrm{const}_0\colon M = (1 + \mathbb{A}) + (\overline{1} + \overline{\mathbb{A}}) \twoheadrightarrow (1 + \mathbb{A}) + 0 = N.$$

Note that e is a monoid morphism: it maps 1 to 1 and if $x, y \neq 1$ then $e(x), e(y) \neq 1$ and hence $e(x \cdot y) = e(\overline{x} \cdot \overline{y}) = 0 = e(x) \cdot e(y)$. The quotient e is support-reflecting, but it is not MSR: the subset $1 + \mathbb{A} + \overline{1} \subseteq M$ of support-preserving elements does not form a submonoid of M. Finally, clearly M satisfies (6.1) while N does not.

7 Conclusion and Future Work

We have introduced topological methods to the theory of data languages, and also explored some of their subtleties and limitations. Following the spirit of Marshall Stone's slogan *"always topologize"*, the core insight of our paper may be summarized as:

Data languages topologize for bounded supports.

In fact, by restricting to support-bounded orbit-finite nominal sets and analyzing their Pro-completion, we have shown that fundamental results from profinite topology (notably Stone duality and the equivalence between profinite spaces and Stone spaces) generalize to the pro-orbit-finite world. These results are of independent interest; in particular, they are

potentially applicable to data languages recognizable by all kinds of orbit-finite structures. For the case of monoids, we derived a topological interpretation of recognizable data languages via clopen sets of pro-orbit-finite words, as well as a nominal version of Reiterman's pseudovariety theorem characterizing the expressive power of pro-orbit-finite equations.

The foundations laid in the present paper open up a number of promising directions for future research. One first goal is to develop a fully fledged duality theory for data languages along the lines of the work of Gehrke et al. [18] on classical regular languages, based on an *extended nominal Stone duality* between pro-orbit-finite monoids and nominal boolean algebras with operators.

Regarding specific applications, we aim to analyze further classes of orbit-finite monoids in terms of pro-orbit-finite equations, following the lines of Example 6.9, in order to classify the corresponding data languages. One natural candidate is the class of $\mathcal{J}$-trivial monoids, with the vision of a nominal version of Simon's theorem [41] relating $\mathcal{J}$-triviality to existential first-order logic on data words.

Finally, we aim to extend our topological theory of recognizable data languages, and the corresponding nominal Reiterman theorem, to algebraic structures beyond orbit-finite monoids. Potential instances include algebras for a signature Σ, which serve as recognizers for data tree languages, infinitary structures such as nominal ω-semigroups [45], modeling languages of infinite data words, and algebraic structures with binders, which we expect to bear interesting connections to data languages with binders and their automata models [39,43].

References

1 Jiří Adámek, Liang-Ting Chen, Stefan Milius, and Henning Urbat. Reiterman's theorem on finite algebras for a monad. *ACM Trans. Comput. Log.*, 22(4):23:1–23:48, 2021.

2 Jiří Adámek and Jiří Rosický. *Locally Presentable and Accessible Categories*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1994.

3 Jorge Almeida. Profinite semigroups and applications. In *Structural Theory of Automata, Semigroups, and Universal Algebra*, pages 1–45. Springer Netherlands, 2005.

4 Jorge Almeida and Alfredo Costa. Profinite topologies. In Jean-Éric Pin, editor, *Handbook of Automata Theory*, pages 615–652. European Mathematical Society Publishing House, Zürich, Switzerland, 2021.

5 Michał Bielecki, Jan Hidders, Jan Paredaens, Jerzy Tyszkiewicz, and Jan Van den Bussche. Navigating with a browser. In *ICALP 2002*, volume 2380 of *LNCS*, pages 764–775. Springer, 2002.

6 Garrett Birkhoff. On the Structure of Abstract Algebras. *Mathematical Proceedings of the Cambridge Philosophical Society*, 31(4):433–454, 1935.

7 Mikołaj Bojańczyk. Nominal monoids. *Theory Comput. Syst.*, 53(2):194–222, 2013. doi:10.1007/s00224-013-9464-1.

8 Mikołaj Bojańczyk. Recognisable languages over monads. In *DLT 2015*, volume 9168 of *LNCS*, pages 1–13. Springer, 2015.

9 Mikołaj Bojańczyk, Bartek Klin, and Sławomir Lasota. Automata theory in nominal sets. *Log. Methods Comput. Sci.*, 10(3), 2014. doi:10.2168/LMCS-10(3:4)2014.

10 Mikołaj Bojańczyk and Rafał Stefański. Single-use automata and transducers for infinite alphabets. In *ICALP 2020*, volume 168 of *LIPIcs*, pages 113:1–113:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.113.

11 Benedikt Bollig, Peter Habermehl, Martin Leucker, and Benjamin Monmege. A robust class of data languages and an application to learning. *Log. Meth. Comput. Sci.*, 10(4:19):23pp., 2014. doi:10.2168/LMCS-10(4:19)2014.

12 Liang-Ting Chen, Jiří Adámek, Stefan Milius, and Henning Urbat. Profinite monads, profinite equations, and Reiterman's theorem. In *FOSSACS 2019*, volume 9634 of *LNCS*, pages 531–547. Springer, 2016. doi:10.1007/978-3-662-49630-5_31.

13 Thomas Colcombet, Clemens Ley, and Gabriele Puppis. Logics with rigidly guarded data tests. *Log. Methods Comput. Sci.*, 11(3), 2015.

14 Samuel Eilenberg. *Automata, Languages, and Machines*. Elsevier Science, 1974.

15 Samuel Eilenberg and Marcel-Paul Schützenberger. On pseudovarieties. *Advances Math.*, 10:413–418, 1976.

16 Murdoch James Gabbay. Nominal algebra and the HSP theorem. *J. Log. Comput.*, 19(2):341–367, 2009. doi:10.1093/logcom/exn055.

17 Murdoch James Gabbay, Tadeusz Litak, and Daniela Petrisan. Stone Duality for Nominal Boolean Algebras with И. In *CALCO 2011*, volume 6859 of *LNCS*, pages 192–207. Springer, 2011. doi:10.1007/978-3-642-22944-2_14.

18 Mai Gehrke, Serge Grigorieff, and Jean-Eric Pin. Duality and equational theory of regular languages. In *ICALP 2008*, volume 5126 of *LNCS*, pages 246–257. Springer, 2008.

19 Mai Gehrke, Serge Grigorieff, and Jean-Eric Pin. A topological approach to recognition. In *ICALP 2010*, volume 6199 of *LNCS*, pages 151–162. Springer, 2010.

20 Mai Gehrke, Daniela Petrişan, and Luca Reggio. The Schützenberger product for syntactic spaces. In *ICALP 2016*, volume 55 of *LIPIcs*, pages 112:1–112:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

21 Mai Gehrke, Daniela Petrişan, and Luca Reggio. Quantifiers on languages and codensity monads. In *LICS 2017*, pages 1–12. IEEE Computer Society, 2017.

22 Radu Grigore, Dino Distefano, Rasmus Petersen, and Nikos Tzevelekos. Runtime verification based on register automata. In *TACAS 2013*, volume 7795 of *LNCS*, pages 260–276. Springer, 2013. doi:10.1007/978-3-642-36742-7_19.

23 Matthew Hennessy. A fully abstract denotational semantics for the pi-calculus. *Theoret. Comput. Sci.*, 278:53–89, 2002. doi:10.1016/S0304-3975(00)00331-5.

24 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994. doi:10.1016/0304-3975(94)90242-9.

25 Kenneth Krohn and John Rhodes. Algebraic theory of machines. I. Prime decomposition theorem for finite semigroups and machines. *Trans. Am. Math. Soc.*, 116:450–464, 1965.

26 Klaas Kürtz, Ralf Küsters, and Thomas Wilke. Selecting theories and nonce generation for recursive protocols. In *FSME 2007*, pages 61–70. ACM, 2007.

27 Alexander Kurz and Daniela Petrisan. On universal algebra over nominal sets. *Math. Struct. Comput. Sci.*, 20(2):285–318, 2010. doi:10.1017/S0960129509990399.

28 Saunders MacLane. *Categories for the Working Mathematician*. Springer-Verlag, 1971.

29 R. McNaughton and S. Papert. *Counter-free Automata*. M.I.T. Press research monographs. M.I.T. Press, 1971.

30 Stefan Milius and Henning Urbat. Equational axiomatization of algebras with structure. In *FOSSACS 2019*, volume 11425 of *LNCS*, pages 400–417. Springer, 2019. doi:10.1007/978-3-030-17127-8_23.

31 Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Log.*, 5(3):403–435, 2004. doi:10.1145/1013560.1013562.

32 Daniela Petrişan. *Investigations into Algebra and Topology over Nominal Sets*. PhD thesis, University of Leicester, 2012.

33 Jean-Eric Pin. Profinite methods in automata theory. In *STACS 2009*, volume 3 of *LIPIcs*, pages 31–50. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2009. doi:10.4230/LIPIcs.STACS.2009.1856.

34 Jean-Eric Pin and Pascal Weil. A Reiterman theorem for pseudovarieties of finite first-order structures. *Algebra Universalis*, 35(4):577–595, 1996.

35 Nicholas Pippenger. Regular languages and Stone duality. *Theory Comput. Syst.*, 30(2):121–134, 1997. doi:10.1007/s002240000045.

36 Andrew M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*. Cambridge University Press, 2013.

37 Jan Reiterman. The Birkhoff theorem for finite algebras. *Algebra Universalis*, 14(1):1–10, 1982.

38 Julian Salamanca. Unveiling eilenberg-type correspondences: Birkhoff's theorem for (finite) algebras + duality. *CoRR*, 2017. arXiv:1702.02822.

39 Lutz Schröder, Dexter Kozen, Stefan Milius, and Thorsten Wißmann. Nominal automata with name binding. In *FOSSACS 2017*, pages 124–142, 2017.

40 Marcel Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965.

41 Imre Simon. Piecewise testable events. In H. Brakhage, editor, *Automata Theory and Formal Languages*, pages 214–222. Springer, 1975.

42 Henning Urbat, Jiří Adámek, Liang-Ting Chen, and Stefan Milius. Eilenberg theorems for free. In *MFCS 2017*, volume 83 of *LIPIcs*, pages 43:1–43:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.MFCS.2017.43.

43 Henning Urbat, Daniel Hausmann, Stefan Milius, and Lutz Schröder. Nominal Büchi automata with name allocation. In *CONCUR 2021*, pages 4:1–4:16, 2021. doi:10.4230/LIPIcs.CONCUR.2021.4.

44 Henning Urbat and Stefan Milius. Varieties of data languages. In *ICALP 2019*, volume 132 of *LIPIcs*, pages 130:1–130:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.130.

45 Thomas Wilke. An Eilenberg theorem for infinity-languages. In *ICALP 1991*, volume 510 of *LNCS*, pages 588–599. Springer, 1991.

Population Protocols with Unordered Data

Michael Blondin
Department of Computer Science, Université de Sherbrooke, Canada

François Ladouceur
Department of Computer Science, Université de Sherbrooke, Canada

Abstract

Population protocols form a well-established model of computation of passively mobile anonymous agents with constant-size memory. It is well known that population protocols compute Presburger-definable predicates, such as absolute majority and counting predicates. In this work, we initiate the study of population protocols operating over arbitrarily large data domains. More precisely, we introduce *population protocols with unordered data* as a formalism to reason about anonymous crowd computing over unordered sequences of data. We first show that it is possible to determine whether an unordered sequence from an infinite data domain has a datum with absolute majority. We then establish the expressive power of the "immediate observation" restriction of our model, namely where, in each interaction, an agent observes another agent who is unaware of the interaction.

2012 ACM Subject Classification Theory of computation → Distributed computing models; Theory of computation → Automata over infinite objects

Keywords and phrases Population protocols, unordered data, colored Petri nets

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.115

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `https://arxiv.org/abs/2305.00872`

Funding *Michael Blondin*: Supported by a Discovery Grant from the Natural Sciences and Engineering Research Council of Canada (NSERC), and by the Fonds de recherche du Québec – Nature et technologies (FRQNT).
François Ladouceur: Supported by a scholarship from the Natural Sciences and Engineering Research Council of Canada (NSERC), and by the Fondation J.A. De Sève.

Acknowledgements We thank Manuel Lafond for his ideas and feedback in the early phase of our research. We further thank the anonymous reviewers for their comments and insightful suggestions.

1 Introduction

Context. Population protocols form a well-established model of computation of passively mobile anonymous agents with constant-size memory [1]. Population protocols allow, e.g., for the formal analysis of chemical reaction networks and networks of mobile sensors (see [23] for a review article on population protocols and more generally on dynamic networks).

In a population protocol, anonymous agents hold a mutable state from a finite set. They collectively seek to evaluate a predicate on the initial global state of the population. At each discrete moment, a scheduler picks two agents who jointly update their respective states according to their current states. Such a scheduler is assumed to be "fair" (or, equivalently, to pick pairs of agents uniformly at random). Let us illustrate the model with a classical protocol for the aboslute majority predicate. Consider a population of ℓ (anonymous) agents, each initialized with either Y or N, that seek to compute whether the number of Y exceeds the number of N, i.e., to collectively evaluate the predicate $\varphi(\#Y, \#N) \coloneqq (\#Y > \#N)$. For example, a population of $\ell = 5$ agents may be initialized to $\{\!\{Y, N, Y, Y, N\}\!\}$. An update of two agents occurs according to these four rules:

strong to weak	*propagation of winning side*	*tiebreaker*
$\{\!\{Y, N\}\!\} \to \{\!\{n, n\}\!\}$	$\{\!\{Y, n\}\!\} \to \{\!\{Y, y\}\!\}$ $\{\!\{N, y\}\!\} \to \{\!\{N, n\}\!\}$	$\{\!\{y, n\}\!\} \to \{\!\{n, n\}\!\}$

A possible execution from the aforementioned population is $\{\!\{Y, N, Y, Y, N\}\!\} \to \{\!\{Y, N, Y, n, n\}\!\} \to \{\!\{Y, n, n, n, n\}\!\} \to \{\!\{Y, y, n, n, n\}\!\} \to \cdots \to \{\!\{Y, y, y, y, y\}\!\}$.

Agents in states $\{Y, y\}$ believe that the output of φ should be *true*, while agents in $\{N, n\}$ believe that it should be *false*. Thus, in the above execution, a lasting *true*-consensus has been reached by the population (although no agent is locally certain of it).

It is well known that population protocols compute precisely the predicates definable in Presburger arithmetic, namely first-order logic over the naturals with addition and order. This was first shown through convex geometry [2], and reproven using the theory of vector addition systems [15]. For example, this means that, given voting options $\{1, \ldots, k\}$, there is a population protocol that determines whether some option i has an absolute majority, i.e., whether more than $\ell/2$ of the ℓ agents initially hold a common $i \in \{1, \ldots, k\}$.

Since k must be stored in the state-space, such a majority protocol can only handle a fixed number of voting options. As rules also depend on k, this means that a whole population would need to be reconfigured in order to handle a larger k, e.g. if new voting options are made available. This is conceptually impractical in the context of flocks of anonymous mobile agents. Instead, we propose that the input of an agent can be modeled elegantly as drawn from an infinite set $\mathbb{D}$, with rules independent from $\mathbb{D}$.

Contribution. In this work, we initiate the study of population protocols operating over arbitrarily large domains. More precisely, we propose a more general model where each agent carries a read-only datum from an infinite domain $\mathbb{D}$ together with a mutable state from a finite set. In this setting, a population can, e.g., seek to determine whether there is an absolute majority datum. For example, if $\mathbb{D} \coloneqq \{1, 2, 3, \ldots\}$, then the population initialized with $\{\!\{(1, x), (1, x), (2, x), (3, x), (1, x)\}\!\}$ should reach a lasting *true*-consensus, while it should reach a lasting *false*-consensus from $\{\!\{(1, x), (1, x), (2, x), (3, x), (2, x)\}\!\}$.

As in the standard model, a fair scheduler picks a pair of agents. An interaction occurs according to a rule of the form $\{\!\{p, q\}\!\} \xrightarrow{d \sim e} \{\!\{p', q'\}\!\}$, where $d, e \in \mathbb{D}$ are the data values of the two agents, and where $\sim \in \{=, \neq\}$ compares them. As for states, we assume that arbitrarily many agents may be initialized with the same datum; and that agents can only compare data through (in)equality. So, $\mathbb{D}$ is not a set of (unique) identifiers and hence agents remain anonymous as in the standard model.

To illustrate our proposed model of computation, we first show that it can compute the absolute majority predicate. This means that a *single* protocol can handle *any* number of options in an absolute majority vote. From the perspective of distributed computing, this provides a framework to reason about anonymous crowd computing over unordered sequences of data. From the standpoint of computer-aided verification, this opens the possibility of formally analyzing single protocols (e.g. modeled as colored Petri nets) rather than resorting to parameterized verification, which is particularly difficult in the context of counter systems.

As a stepping stone towards pinpointing the expressive power of *population protocols with unordered data*, we then characterize *immediate observation* protocols. In this well-studied restriction, rules have the form $\{\!\{p, q\}\!\} \xrightarrow{d \sim e} \{\!\{p, q'\}\!\}$, i.e. an agent updates its state by observing another agent (who is unaware of it). In standard population protocols, this class is known to compute exactly predicates from $\mathbf{COUNT}_*$ [2]. The latter is the Boolean

closure of predicates of the form $\#q \geq c$, where $\#q$ counts the number of agents in state q, and $c \in \mathbb{N}$ is a constant. In our case, we show that immediate observation protocols compute exactly *interval predicates*, which are Boolean combinations of such *simple interval predicates*:

$$\exists \text{ pairwise distinct } d_1, d_2, \ldots, d_n \in \mathbb{D} : \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{m} \#(d_i, q_j) \in T(i,j), \tag{1}$$

where $\#(d_i, q_j)$ counts agents in state q_j with datum d_i, and each $T(i,j) \subseteq \mathbb{N}$ is an interval.

In order to show that immediate observation protocols do not compute more than interval predicates, we exploit the fact that (finitely supported) data vectors are well-quasi-ordered. While our approach is inspired by [2], it is trickier to simultaneously deal with the several sources of unboundedness: number of data values, number of agents with a given datum, and number of agents with a given state. As a byproduct, we show that the absolute majority predicate cannot be computed by immediate observation protocols.

To show the other direction, i.e. that interval predicates are computable by immediate observation protocols, we describe a protocol for simple interval predicates. In contrast with the standard setting, we need to implement existential quantification. This is achieved by a data leader election and a global leader election. We call the latter elected agent the "controller". Its purpose is to handle the bookkeeping of data leaders choosing their role in (1). A correction mechanism is carefully implemented so that the population only reaches a *true*-consensus upon locking a correct assignment to the existential quantification.

Related work. It has been observed by the verification and concurrency communities that population protocols can be recast as Petri nets. In particular, this has enabled the automatic formal analysis of population protocols [15, 7] and the discovery of bounds on their state complexity [12, 6]. Our inspiration comes from the other direction: we introduce protocols with data by drawing from the recent attention to colored Petri nets [17, 19, 18]. Our model corresponds to unordered data Petri nets where the color and number of tokens is invariant.

Population protocols for computing majority and plurality have been extensively studied (e.g., see [14, 4, 5, 3] for recent results). To the best of our knowledge, the closest work is [16], where the authors propose space-efficient *families* of deterministic protocols for variants of the majority problem including plurality consensus. They consider the k voting options as "colors" specified by $\lceil \log k \rceil$ bits stored within the agents.

Other incomparable models of distributed systems with some sort of data include broadcast networks of register automata [13], distributed register automata [9], and distributed memory automata [10]. Such formalisms, inspired by register automata [20], allow identities, control structures and alternative communication mechanisms; none allowed in population protocols.

Paper organization. Section 2 provides basic definitions and introduces our model. In Section 3, we present a protocol that computes the absolute majority predicate. Section 4 establishes the expressive power of immediate observation protocols. We conclude in Section 5. Note that most proofs appear in the appendix of the full version.

2 Preliminaries

We write $\mathbb{N}$ and $[a..b]$ to respectively denote sets $\{0, 1, 2, \ldots\}$ and $\{a, a+1, \ldots, b\}$. The *support* of a multiset $\boldsymbol{m}$ over E is $\text{act}(\boldsymbol{m}) \coloneqq \{e \in E : \boldsymbol{m}(e) > 0\}$ (We use the notation $\text{act}(\boldsymbol{m})$ rather than $\text{supp}(\boldsymbol{m})$ as we will later refer to "active states".) We write $\mathbb{N}^E$ to denote the set of multisets over E with finite support. The empty multiset, denoted $\mathbf{0}$, is such that

$\mathbf{0}(e) = 0$ for all $e \in E$. Let $\boldsymbol{m}, \boldsymbol{m}' \in \mathbb{N}^E$. We write $\boldsymbol{m} \leq \boldsymbol{m}'$ iff $\boldsymbol{m}(e) \leq \boldsymbol{m}'(e)$ for all $e \in E$. We define $\boldsymbol{m} + \boldsymbol{m}'$ as the multiset such that $(\boldsymbol{m} + \boldsymbol{m}')(e) \coloneqq \boldsymbol{m}(e) + \boldsymbol{m}'(e)$ for all $e \in E$. The difference, denoted $\boldsymbol{m} - \boldsymbol{m}'$, is defined similarly provided that $\boldsymbol{m} \geq \boldsymbol{m}'$.

2.1 Population protocols with unordered data

A *population protocol with unordered data*, over an infinite domain $\mathbb{D}$ equipped with equality, is a tuple (Q, δ, I, O) where

- Q is a finite set of elements called *states*,
- $\delta \subseteq Q^2 \times \{=, \neq\} \times Q^2$ is the set of *transitions*,
- $I \subseteq Q$ is the set of *initial states*, and
- $O \colon Q \to \{\mathit{false}, \mathit{true}\}$ is the *output* function.

We refer to an element of $\mathbb{D}$ as a *datum* or as a *color*. We will implicitly assume throughout the paper that δ contains $((p, q), \sim, (p, q))$ for all $p, q \in Q$ and $\sim \in \{=, \neq\}$.

A *form* $\boldsymbol{f}$ is an element from $\mathbb{N}^Q$. We denote the set of all forms by $\mathbb{F}$. Given $Q' \subseteq Q$, let $\boldsymbol{f}(Q') \coloneqq \sum_{q \in Q'} \boldsymbol{f}(q)$. A *configuration* is a mapping $\mathbf{C}$ from $\mathbb{D}$ to $\mathbb{F}$ such that $\mathrm{supp}(\mathbf{C}) \coloneqq \{d \in \mathbb{D} : \mathbf{C}(d) \neq \mathbf{0}\}$ is finite, and $\sum_{d \in \mathbb{D}, q \in Q} \mathbf{C}(d)(q) \geq 2$. We often write $\mathbf{C}(d)(q)$ as $\mathbf{C}(d, q)$. Informally, the latter denotes the number of agents with datum d in state q. We extend this notation to subsets of states: $\mathbf{C}(d, Q') \coloneqq \mathbf{C}(d)(Q')$. We naturally extend $+$, $-$ and $\leq$ to $\mathbb{D} \to \mathbb{F}$, e.g. $\mathbf{C} + \mathbf{C}'$ is such that $(\mathbf{C} + \mathbf{C}')(d) \coloneqq \mathbf{C}(d) + \mathbf{C}'(d)$ for all $d \in \mathbb{D}$.

Let $\mathbf{C}$ be a configuration. We define the *active states* as the set $\mathrm{act}(\mathbf{C}) \coloneqq \{q \in Q : \mathbf{C}(d, q) > 0 \text{ for some } d \in \mathbb{D}\}$. We say that $\mathbf{C}$ is *initial* if $\mathrm{act}(\mathbf{C}) \subseteq I$. Given $Q' \subseteq Q$, let $|\mathbf{C}|_{Q'} \coloneqq \sum_{d \in \mathbb{D}} \mathbf{C}(d, Q')$ and $|\mathbf{C}| \coloneqq |\mathbf{C}|_Q$. The *output* of $\mathbf{C}$ is defined by $O(\mathbf{C}) \coloneqq b$ if $O(q) = b$ for every $q \in \mathrm{act}(\mathbf{C})$; and by $O(\mathbf{C}) = \bot$ otherwise. Informally, $O(\mathbf{C})$ indicates whether all agents agree on some output b.

▶ **Example 1.** Let $\mathbb{D} \coloneqq \{\bullet, \blacksquare, \blacklozenge, \ldots\}$ and $Q \coloneqq \{p, q\}$. Let $\boldsymbol{f} \coloneqq \{\!\{p, p, q\}\!\}$ and $\boldsymbol{f}' \coloneqq \{\!\{q\}\!\}$. Let $\mathbf{C} \coloneqq \{\bullet \mapsto \boldsymbol{f}, \blacksquare \mapsto \boldsymbol{f}', \blacklozenge \mapsto \mathbf{0}, \ldots\}$. We have $\mathbf{C}(\bullet, p) = 2$, $\mathbf{C}(\bullet, q) = \mathbf{C}(\blacksquare, q) = 1$, $\mathbf{C}(\blacksquare, p) = \mathbf{C}(\blacklozenge, p) = \mathbf{C}(\blacklozenge, q) = 0$ and $|\mathbf{C}|_{\{q\}} = 2$. Configuration $\mathbf{C}$ represents a population of four agents carrying an immutable datum and a mutable state: $\{\!\{(\bullet, p), (\bullet, p), (\bullet, q), (\blacksquare, q)\}\!\}$. ◀

For the sake of brevity, given a form $\boldsymbol{f}$, let $\boldsymbol{f}_d \colon \mathbb{D} \to \mathbb{F}$ be defined by $\boldsymbol{f}_d(d) \coloneqq \boldsymbol{f}$ and $\boldsymbol{f}_d(d') \coloneqq \mathbf{0}$ for every $d' \neq d$. Furthermore, given a state q, let $\boldsymbol{q}_d \colon \mathbb{D} \to \mathbb{F}$ be defined by $\boldsymbol{q}_d(d)(q) \coloneqq 1$ and $\boldsymbol{q}_d(d')(q') \coloneqq 0$ for every $(d', q') \neq (d, q)$.

Let $\mathbf{C}$ be a configuration and let $t = ((p, q), \sim, (p', q')) \in \delta$. We say that transition t is *enabled* in $\mathbf{C}$ if there exist $d, e \in \mathbb{D}$ such that $d \sim e$, $\mathbf{C} \geq \boldsymbol{p}_d + \boldsymbol{q}_e$. If the latter holds, then t can be used to obtain the configuration $\mathbf{C}' \coloneqq \mathbf{C} - (\boldsymbol{p}_d + \boldsymbol{q}_e) + (\boldsymbol{p}'_d + \boldsymbol{q}'_e)$, which we denote $\mathbf{C} \xrightarrow{t} \mathbf{C}'$. We write $\mathbf{C} \to \mathbf{C}'$ to denote that $\mathbf{C} \xrightarrow{t} \mathbf{C}'$ holds for some $t \in \delta$. We further define $\xrightarrow{*}$ as the reflexive-transitive closure of $\to$.

▶ **Example 2.** Let $O(p) \coloneqq \mathit{false}$, $O(q) \coloneqq \mathit{true}$ and $t \coloneqq ((p, q), =, (q, q))$. Using the notation of Example 1 to represent configurations, we have:

$$\{\!\{(\bullet, p), (\bullet, p), (\bullet, q), (\blacksquare, q)\}\!\} \xrightarrow{t} \{\!\{(\bullet, p), (\bullet, q), (\bullet, q), (\blacksquare, q)\}\!\} \xrightarrow{t} \{\!\{(\bullet, q), (\bullet, q), (\bullet, q), (\blacksquare, q)\}\!\}.$$

Let $\mathbf{C}$, $\mathbf{C}'$ and $\mathbf{C}''$ denote the three configurations above. We have $O(\mathbf{C}) = O(\mathbf{C}') = \bot$ and $O(\mathbf{C}'') = \mathit{true}$. Moreover, transition t is not enabled in $\mathbf{C}''$ as no datum $d \in \mathbb{D}$ satisfies $\mathbf{C}''(d, p) \geq 1$ and $\mathbf{C}''(d, q) \geq 1$. So, the agents have "converged to a *true*-consensus". ◀

An *execution* is an infinite sequence of configurations $\mathbf{C}_0\mathbf{C}_1\cdots$ such that $\mathbf{C}_0 \to \mathbf{C}_1 \to \cdots$. We say that such an execution *converges* to output $b \in \{false, true\}$ if there exists $\tau \in \mathbb{N}$ such that $O(\mathbf{C}_\tau) = O(\mathbf{C}_{\tau+1}) = \cdots = b$. An execution $\mathbf{C}_0\mathbf{C}_1\cdots$ is *fair* if, for every configuration $\mathbf{C}'$, it is the case that $|\{i \in \mathbb{N} : \mathbf{C}_i \xrightarrow{*} \mathbf{C}'\}| = \infty$ implies $|\{i \in \mathbb{N} : \mathbf{C}_i = \mathbf{C}'\}| = \infty$. In words, fairness states that if $\mathbf{C}'$ is reachable infinitely often, then it appears infinitely often along the execution. Informally, this means that some "progress" cannot be avoided forever.

Let Σ be a nonempty finite set. An *input* is some $\mathbf{M} \in \mathbb{N}^{\mathbb{D}\times\Sigma}$ with $\sum_{d\in\mathbb{D},\sigma\in\Sigma} \mathbf{M}(d,\sigma) \geq 2$. An input $\mathbf{M}$ is translated, via a bijective *input mapping* $\iota\colon \Sigma \to I$, into the initial configuration $\iota(\mathbf{M}) \coloneqq \sum_{d\in\mathbb{D},\sigma\in\Sigma} \sum_{j=1}^{\mathbf{M}(d,\sigma)} \boldsymbol{\iota(\sigma)}_d$. We say that a protocol *computes* a predicate φ if, for every input $\mathbf{M}$, every fair execution starting in $\iota(\mathbf{M})$ converges to output $\varphi(\mathbf{M})$. By abuse of notation, we sometimes write $\varphi(\mathbf{C}_0)$ for $\varphi(\iota^{-1}(\mathbf{C}_0))$.

▶ **Example 3.** Let $\mathbb{D} \coloneqq \{\bullet, \blacksquare, \blacklozenge, \ldots\}$, $\Sigma \coloneqq \{x_1, \ldots, x_4\}$, $I \coloneqq \{q_1, \ldots, q_4\}$ and $\iota(x_i) \coloneqq q_i$. The input $\mathbf{M} \coloneqq \{\!\{(\bullet, x_1), (\bullet, x_1), (\bullet, x_2), (\bullet, x_2), (\bullet, x_2), (\bullet, x_4), (\blacksquare, x_1), (\blacksquare, x_3)\}\!\}$ yields the initial configuration $\iota(\mathbf{M}) = \{\bullet \mapsto \{\!\{q_1, q_1, q_2, q_2, q_2, q_4\}\!\}, \blacksquare \mapsto \{\!\{q_1, q_3\}\!\}, \blacklozenge \mapsto \mathbf{0}, \ldots\}$.

Observe that, as for standard protocols, the set of predicates computed by population protocols with unordered data is closed under Boolean operations. Given a protocol that computes φ, we obtain a protocol that computes $\neg\varphi$ by changing the value of $O(q)$ to $\neg O(q)$ for all $q \in Q$. Given predicates ψ_1 and ψ_2, respectively computed by protocols $(Q_1, \delta_1, I_1, O_1)$ and $(Q_2, \delta_2, I_2, O_2)$, it is easy to obtain a protocol computing $\psi = \psi_1 \wedge \psi_2$ by having both protocols run in parallel. This is achieved by defining $(Q \coloneqq Q_1 \times Q_2, \delta, I \coloneqq I_1 \times I_2, O)$ where

- δ contains $(((p_1, p_2), (q_1, q_2)), \sim, ((p'_1, p_2), (q'_1, q_2)))$ for every $((p_1, q_1), \sim, (p'_1, q'_1)) \in \delta_1$;
- δ contains $(((p_1, p_2), (q_1, q_2)), \sim, ((p_1, p'_2), (q_1, q'_2)))$ for every $((p_2, q_2), \sim, (p'_2, q'_2)) \in \delta_2$;
- $O(q_1, q_2) = O_1(q_1) \wedge O_2(q_2)$.

3 A protocol for the majority predicate

Let $\Sigma \coloneqq \{x\}$. In this section, we present a protocol for the absolute majority predicate defined as $\varphi_{\text{maj}}(\mathbf{M}) \coloneqq \exists d \in \mathbb{D} : \mathbf{M}(d, x) > \sum_{d'\neq d} \mathbf{M}(d', x)$. Since each input pair has the form (d, x) with $d \in \mathbb{D}$, we omit the "dummy element" x in the informal presentation of the protocol. Note that for the sake of brevity, we use the term majority instead of absolute majority for the remainder of this paper.

Our protocol is not unlike the classical (sequential) Boyer–Moore algorithm [11]: we seek to elect a color as the majority candidate, and then check whether it indeed has the majority. It is intended to work in stages. In the *pairing stage*, each unpaired agent seeks to form a pair with an unpaired agent of a distinct color. For example, if the initial population is $\{\!\{\bullet, \bullet, \bullet, \bullet, \blacksquare, \blacksquare, \blacklozenge\}\!\}$, then we (non-deterministically) end up with either of these two pairings:

paired agents	*agents left unpaired*
$\{\!\{\bullet\!-\!\blacksquare, \bullet\!-\!\blacksquare, \bullet\!-\!\blacklozenge\}\!\}$	$\{\!\{\bullet\}\!\}$
$\{\!\{\bullet\!-\!\blacksquare, \blacksquare\!-\!\blacklozenge\}\!\}$	$\{\!\{\bullet, \bullet, \bullet\}\!\}$

The agents left unpaired must all have the same color d, e.g. "$\bullet$" in the above example. Moreover, if the population has a majority color, then it must be d.

Since the agents are anonymous and have a finite memory, they cannot actually remember with whom they have been paired. Thus, once a candidate color has been elected, e.g. "$\bullet$" in the above example, there is a *grouping stage*. In the latter, unpaired agents indicate to agents of their color that they are part of the candidate majority group. This is done by internally storing the value "Y", which stands for "Yes". Similarly, unpaired agents indicate to agents of a distinct color that they are part of the candidate minority group using "N". Once this is over, the *majority stage* takes place using the classical protocol from the introduction.

Two issues arise from this idealized description. First, the protocol is intended to work in stages, but they may occur concurrently due to their distributed nature. For this reason, we add a correction mechanism:

- If an unpaired agent of the candidate majority color d finds a paired agent of color d (resp. $d' \neq d$) with "N" (resp. "Y"), then it flips it to "Y" (resp. "N");
- If an unpaired agent of the candidate majority color d finds a paired agent of color d (resp. $d' \neq d$) with either "n" or "y", then it flips it to "$\overline{Y}$" (resp. "$\overline{N}$").

The intermediate value $\overline{Y}$ (resp. $\overline{N}$) must be reverted to Y (resp. N) by finding an agent that has initially played role Y (resp. N) and is then reset to its original value.

The second issue has to do with the fact that, in even-size populations, all agents may get paired. In that case, no unpaired agent is left to group the agents. To address this, each agent carries an "even bit" to indicate its belief on whether some unpaired agent remains.

3.1 States

The set of states is defined as $Q \coloneqq \{\mathit{false}, \mathit{true}\}^3 \times \{Y, N, \overline{Y}, \overline{N}, y, n\}$. To ease the reader's understanding, we manipulate states with four "macros". Each macro has a set of possible values; each state is a combination of values for the different macros.

name	*values for $q \in Q$*	*value for $q \in I$*
$\mathrm{pair}(q)$	$\{\mathit{false}, \mathit{true}\}$	*false*
$\mathrm{grp}(q)$	$\{\mathit{false}, \mathit{true}\}$	*true*

name	*values for $q \in Q$*	*value for $q \in I$*
$\mathrm{even}(q)$	$\{\mathit{false}, \mathit{true}\}$	*false*
$\mathrm{maj}(q)$	$\{Y, N, \overline{Y}, \overline{N}, y, n\}$	Y

The input mapping is defined by $\iota(x) \coloneqq q_I$, where q_I is the unique state of I. Informally, $\mathrm{pair}(q)$ indicates whether the agent has been paired; $\mathrm{grp}(q)$ indicates whether the agent belongs to the candidate majority group; $\mathrm{maj}(q)$ is the current value of the majority computation; and $\mathrm{even}(q)$ is the even bit.

3.2 Transitions and stages

We describe the protocol by introducing rules corresponding to each stage. Note that a rule is a structure on which transitions can be based; therefore, a single rule can yield multiple transitions of the same nature. For convenience, some lemmas are stated before they can actually be proven, as they require the full set of transitions to be defined first. Proofs in the appendix take into account the complete list of transitions.

As the set of transitions for the protocol is lengthy, we present it using a "precondition-update" notation where for any two agents in state $p, q \in Q$, respectively with colors $d_1, d_2 \in \mathbb{D}$, a single transition whose preconditions on p, q and d_1, d_2 are met is used. The result of such an interaction is the agent initially in state p updating its state to p', where p' is identical to p except for the specified macros; and likewise for q. To help the readability, the precondition and update of states p and q are on distinct lines in the forthcoming tables.

3.2.1 Pairing stage

The first rule is used for the pairing stage whose main goal is to match as many agents as possible with agents of a different color:

rule	*state precondition*	*color precondition*	*state update*
(1)	$\neg\mathrm{pair}(p)$ $\neg\mathrm{pair}(q)$	$d_1 \neq d_2$	$\mathrm{pair}(p') \wedge \mathrm{even}(p')$ $\mathrm{pair}(q') \wedge \mathrm{even}(q')$

This rule gives rise to the following lemmas concerning the end of the pairing stage and the nature of unpaired agents, if they exist. For the remainder of the section, let us fix a fair execution $\mathbf{C}_0\mathbf{C}_1\cdots$ where $\mathbf{C}_0$ is initial. Moreover, let $P \coloneqq \{q \in Q : \text{pair}(q)\}$ and $U \coloneqq Q \setminus P$.

▶ **Lemma 4.** *There exists $\tau \in \mathbb{N}$ such that $|\mathbf{C}_\tau|_U = |\mathbf{C}_{\tau+1}|_U = \cdots$. Furthermore, for every $i \geq \tau$, all unpaired agents of $\mathbf{C}_i$ share the same color, i.e. the set $\{d \in \mathbb{D} : \mathbf{C}_i(d, U) > 0\}$ is either empty or a singleton.*

Let α denote the minimal threshold τ given by Lemma 4, which is informally the "end of the pairing stage".

▶ **Lemma 5.** *Let $i \in \mathbb{N}$. If $\varphi_{maj}(\mathbf{C}_0)$ and d is the majority color, then $\mathbf{C}_i(d, U) > 0$.*

3.2.2 Grouping stage

The next set of transitions seeks to correctly set each agent's group, representing its status in the computation of the majority. An agent is either part of the candidate majority group (*true*), or part of the candidate minority group (*false*). Note that this group (and its related majority computing value) are irrelevant if there are no unpaired agents in $\mathbf{C}_\alpha$; this special case is handled using the even bit, which is ignored for now.

rule	*state precondition*	*color precondition*	*state update*
(2)	$\neg\text{pair}(p)$ $\text{pair}(q) \wedge \quad \text{grp}(q) \wedge \text{maj}(q) = Y$	$d_1 \neq d_2$	*none* $\neg\text{grp}(q') \wedge \text{maj}(q') = N$
(3)	$\neg\text{pair}(p)$ $\text{pair}(q) \wedge \neg\text{grp}(q) \wedge \text{maj}(q) = N$	$d_1 = d_2$	*none* $\text{grp}(q') \wedge \text{maj}(q') = Y$
(4)	$\neg\text{pair}(p)$ $\text{pair}(q) \wedge \quad \text{grp}(q) \wedge \text{maj}(q) \in \{y, n\}$	$d_1 \neq d_2$	*none* $\text{maj}(q') = \overline{N}$
(5)	$\neg\text{pair}(p)$ $\text{pair}(q) \wedge \neg\text{grp}(q) \wedge \text{maj}(q) \in \{y, n\}$	$d_1 = d_2$	*none* $\text{maj}(q') = \overline{Y}$

The forthcoming rules below are part of a two-rule combination whose aim is to rectify an error in grouping assignments. It allows agents who engaged in the computation within the candidate minority group (resp. majority group) who encountered a currently valid majority candidate of their color (resp. a different color) to reset their value to Y (resp. N) and their group to *true* (resp. *false*) by finding another agent, also engaged, to do the same. This, along with the rules described in the next subsection, ensures that the invariant below holds.

Let $Q_a \coloneqq \{q \in Q : \text{maj}(q) = a\}$, $Q_M \coloneqq \{q \in Q : \text{grp}(q)\}$ and $Q_m \coloneqq Q \setminus Q_M$.

▶ **Lemma 6.** *For every $i \in \mathbb{N}$, it is the case that $|\mathbf{C}_i|_{Q_Y} - |\mathbf{C}_i|_{Q_N} = |\mathbf{C}_i|_{Q_M} - |\mathbf{C}_i|_{Q_m}$.*

rule	*state precondition*	*color precondition*	*state update*
(6)	$\text{grp}(p) \wedge \text{maj}(p) = \overline{N}$ $\neg\text{grp}(q) \wedge \text{maj}(q) \in \{y, n\}$	*none*	$\neg\text{grp}(p') \wedge \text{maj}(p') = N$ $\text{maj}(q') = N$
(7)	$\neg\text{grp}(p) \wedge \text{maj}(p) = \overline{Y}$ $\text{grp}(q) \wedge \text{maj}(q) \in \{y, n\}$	*none*	$\text{grp}(p') \wedge \text{maj}(p') = Y$ $\text{maj}(q') = Y$
(8)	$\text{grp}(p) \wedge \text{maj}(p) = \overline{N}$ $\neg\text{grp}(q) \wedge \text{maj}(q) = \overline{Y}$	*none*	$\neg\text{grp}(p') \wedge \text{maj}(p') = n$ $\text{grp}(q') \wedge \text{maj}(q') = n$

We give the following example to help illustrate the necessity of the intermediate states $\overline{Y}, \overline{N}$ in the context of the suggested protocol.

▶ **Example 7.** Let us first consider a possible execution from the initial population $\{\!\{\bullet, \bullet, \blacksquare, \blacksquare, \blacklozenge\}\!\}$, for which there is no majority datum. Observe that since the number of agents is odd, in any execution, there will be a datum with an unpaired agent after the pairing stage. Assume, for the sake of our demonstration, that this datum is blue ($\blacklozenge$). Entering the grouping stage, this blue agent will eventually let the other agents know that they are not part of the majority candidate group and, at some point, rule (11) will occur, leading to all agents permanently with $\mathrm{maj}(q) \in \{N, n\}$. This is summarized in these three snapshots (where the even bit is omitted for the sake of clarity):

input	pair	grp	maj
$\bullet$	×	✓	Y
$\bullet$	×	✓	Y
$\blacksquare$	×	✓	Y
$\blacksquare$	×	✓	Y
$\blacklozenge$	×	✓	Y

$\xrightarrow{*}$

input	pair	grp	maj
$\bullet$	✓	×	N
$\bullet$	✓	×	N
$\blacksquare$	✓	×	N
$\blacksquare$	✓	×	N
$\blacklozenge$	×	✓	Y

$\rightarrow$

input	pair	grp	maj
$\bullet$	✓	×	N
$\bullet$	✓	×	n
$\blacksquare$	✓	×	N
$\blacksquare$	✓	×	N
$\blacklozenge$	×	✓	n

Now, consider a population where a majority datum does indeed exist: $\{\!\{\bullet, \bullet, \bullet, \bullet, \blacksquare, \blacksquare, \blacklozenge\}\!\}$. Note that this population is strictly greater than the previous population. Therefore, we can promptly obtain a configuration similar to the one described above, where two more agents of datum red ($\bullet$) have yet to participate in the computation. Since the computation must output *true*, the consensus on $\{N, n\}$ initiated by the blue agent has to be reverted.

In this case, after the final pairing is done via an interaction between the blue agent ($\blacklozenge$) and one of the newly introduced red agents ($\bullet$), the error handling first works through rule (4) or (5): the unpaired red agent ($\bullet$) notifies the blue agent ($\blacklozenge$) that its group is incorrect by setting its computation value to $\overline{N}$ and similarly, it notifies all red agents ($\bullet$) who had previously participated in the (now incorrect) majority stage to switch their computation value to $\overline{Y}$. This is summarized in these three snapshots:

input	pair	grp	maj
$\bullet$	✓	×	N
$\bullet$	✓	×	n
$\blacksquare$	✓	×	N
$\blacksquare$	✓	×	N
$\blacklozenge$	✓	✓	n
$\bullet$	✓	✓	Y
$\bullet$	×	✓	Y

$\ldots \xrightarrow{*}$

input	pair	grp	maj
$\bullet$	✓	✓	Y
$\bullet$	✓	×	$\overline{Y}$
$\blacksquare$	✓	×	N
$\blacksquare$	✓	×	N
$\blacklozenge$	✓	✓	$\overline{N}$
$\bullet$	✓	✓	Y
$\bullet$	×	✓	Y

$\rightarrow$

input	pair	grp	maj
$\bullet$	✓	✓	Y
$\bullet$	✓	✓	n
$\blacksquare$	✓	×	N
$\blacksquare$	✓	×	N
$\blacklozenge$	✓	×	n
$\bullet$	✓	✓	Y
$\bullet$	×	✓	Y

This inevitably leads each incorrectly grouped agent to rectify its group bit as well as its computation value, accordingly, through rules (6), (7) or (8). We then have a configuration for which the grouping stage is over and where either the majority stage is not yet initiated, or it has been correctly initiated with the right majority candidate. ◀

The following lemmas show that the grouping stage eventually ends if there are unpaired agents in $\mathbf{C}_\alpha$. Moreover, they show that the majority candidate color d eventually propagates the majority group to agents of color d, and the minority group to agents of color $d' \neq d$.

▶ **Lemma 8.** *Let E be the set of states engaged in the majority computation, i.e. $E \coloneqq \{q \in Q : \mathrm{maj}(q) \in \{y, n, \overline{Y}, \overline{N}\}\}$. Let $E_M \coloneqq E \cap Q_M$ and $E_m \coloneqq E \cap Q_m$. For every $i \in \mathbb{N}$, the following holds: $|\mathbf{C}_i|_{E_M} = |\mathbf{C}_i|_{E_m}$.*

▶ **Lemma 9.** *Let $d \in \mathbb{D}$. If $\mathbf{C}_\alpha(d, U) > 0$, then there exists some $\tau \geq \alpha$ such that, for all $i \geq \tau$, $d' \in \mathbb{D}$ and $q \in \mathrm{act}(\mathbf{C}_i(d'))$, the following holds: $\mathrm{grp}(q) = (d' = d)$.*

3.2.3 Majority stage

The last set of transitions emulates a standard population protocol for the majority predicate. Populations of even size without a majority give rise to a case requiring careful handling. Indeed, for such a population the pairing stage may leave no unmatched agent. Therefore, we give the following rules to fix this specific issue.

rule	*state precondition*	*color precondition*	*state update*
(9)	$\neg\mathrm{pair}(p)$ $\mathrm{even}(q)$	*none*	*none* $\neg\mathrm{even}(q')$
(10)	$\mathrm{pair}(p) \wedge \mathrm{even}(p)$ $\mathrm{pair}(q) \wedge \neg\mathrm{even}(q)$	*none*	*none* $\mathrm{even}(q')$

▶ **Lemma 10.** *There exists $\tau \geq \alpha$ such that for every $i \geq \tau$ and $q \in \mathrm{act}(\mathbf{C}_i)$, it is the case that $\mathrm{even}(q)$ holds iff $|\mathbf{C}_i|_U = 0$.*

For other populations, a unique candidate color for the majority exists following the pairing stage. For the predicate to be *true*, this candidate must have more agents than all of the other colors combined. This is validated (or invalidated) through the following rules.

rule	*state pre.*	*col. pre.*	*state update*
(11)	$\mathrm{maj}(p) = Y$ $\mathrm{maj}(q) = N$	*none*	$\mathrm{maj}(p') = n$ $\mathrm{maj}(q') = n$
(12)	$\mathrm{maj}(p) = Y$ $\mathrm{maj}(q) = n$	*none*	*none* $\mathrm{maj}(q') = y$

rule	*state pre.*	*col. pre.*	*state update*
(13)	$\mathrm{maj}(p) = N$ $\mathrm{maj}(q) = y$	*none*	*none* $\mathrm{maj}(q') = n$
(14)	$\mathrm{maj}(p) = n$ $\mathrm{maj}(q) = y$	*none*	*none* $\mathrm{maj}(q') = n$

▶ **Lemma 11.** *If $\varphi_{maj}(\mathbf{C}_0)$ holds, then there exists $\tau \geq \alpha$ such that for every $i \geq \tau$ and $q \in \mathrm{act}(\mathbf{C}_i)$, it is the case that $\mathrm{maj}(q) \in \{Y, y\}$ and $\neg\mathrm{even}(q)$ hold.*

▶ **Lemma 12.** *If $\neg\varphi_{maj}(\mathbf{C}_0)$ holds, then there exists $\tau \geq \alpha$ such that either:*

- *$\mathrm{even}(q)$ holds for every $i \geq \tau$ and $q \in \mathrm{act}(\mathbf{C}_i)$; or*
- *$\mathrm{maj}(q) \in \{N, n\}$ holds for every $i \geq \tau$ and $q \in \mathrm{act}(\mathbf{C}_i)$.*

We define the output of a given state $q \in Q$ as $O(q) \coloneqq (\mathrm{maj}(q) \in \{Y, y\} \wedge \neg\mathrm{even}(q))$. The correctness of the protocol follows immediately from Lemmas 11 and 12:

▶ **Corollary 13.** *There exists $\tau \in \mathbb{N}$ such that $O(\mathbf{C}_\tau) = O(\mathbf{C}_{\tau+1}) = \cdots = \varphi_{maj}(\mathbf{C}_0)$.*

4 Immediate observation protocols

We say that a population protocol is *immediate observation (IO)* if each of its transitions has the form $((p, q), \sim, (p, q'))$, i.e. only one agent can update its state by "observing" the other agent. There is no restriction on $\sim$, but one can also imagine the datum to be observed.

In this section, we characterize the expressive power of immediate observation protocols. First, we establish properties of IO protocols regarding truncations, thereby allowing us to prove that the majority predicate is not computable. Then, we show that IO protocols do not compute more than interval predicates. Finally, we show that every interval predicate can be computed by an IO protocol. Before proceeding, let us define interval predicates.

Let $\dot{\exists} d_1, d_2, \ldots, d_n$ denote a disjoint existential quantification, i.e. it indicates that $d_i \neq d_j$ for all $i, j \in [1..n]$ such that $i \neq j$. A *simple interval predicate*, interpreted over inputs from $\mathbb{N}^{\mathbb{D}\times\Sigma}$, where $\Sigma = \{x_1, \ldots, x_m\}$, is a predicate of the form

$$\psi(\mathbf{M}) = \dot{\exists} d_1, d_2, \ldots, d_n \in \mathbb{D} : \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{m} \mathbf{M}(d_i, x_j) \in T(i, j), \tag{2}$$

where $m, n \in \mathbb{N}_{>0}$, each $T(i, j) \subseteq \mathbb{N}$ is a nonempty interval, and for every $i \in [1..n]$, there exists $j \in [1..m]$ such that $0 \notin T(i, j)$. An *interval predicate* is a Boolean combination of simple interval predicates.

4.1 State and form truncations

Given configurations $\mathbf{C}, \mathbf{C}'$, we write $\mathbf{C} \sqsubseteq \mathbf{C}'$ if there exists an injection $\rho\colon \mathbb{D} \to \mathbb{D}$ such that $\mathbf{C}(d) \leq \mathbf{C}'(\rho(d))$ for every $d \in \mathbb{D}$. We write $\mathbf{C} \equiv \mathbf{C}'$ if $\mathbf{C} \sqsubseteq \mathbf{C}'$ and $\mathbf{C}' \sqsubseteq \mathbf{C}$. We say that a subset of configurations X is *upward closed* if $\mathbf{C} \in X$ and $\mathbf{C} \sqsubseteq \mathbf{C}'$ implies $\mathbf{C}' \in X$. We say that a set B is a *basis* of an upward closed set X if $X = \{\mathbf{C}' : \mathbf{C} \sqsubseteq \mathbf{C}' \text{ for some } \mathbf{C} \in B\}$.

A configuration $\mathbf{C}$ is said *unstable* if either $O(\mathbf{C}) = \bot$ or there exists $\mathbf{C}'$ such that $\mathbf{C} \xrightarrow{*} \mathbf{C}'$ with $O(\mathbf{C}) \neq O(\mathbf{C}')$. Let $\mathcal{U}$ denote the set of unstable configurations, and let $\mathcal{S}_b \coloneqq \{\mathbf{C} : \mathbf{C} \notin \mathcal{U}, O(\mathbf{C}) = b\}$ denote the set of stable configurations with output b. As in the case of standard protocols (without data) [1], it is simple to see that $\mathcal{U}$ is upward closed. Moreover, since $\sqsubseteq$ is a well-quasi-order, it follows that $\mathcal{U}$ has a finite basis.

This allows us to extend the notion of truncations from [1]. A *state truncation* to $k \geq 1$ of some form $\boldsymbol{f}$, denoted by $\tau_k(\boldsymbol{f})$, is the form such that $\tau_k(\boldsymbol{f})(q) \coloneqq \min(\boldsymbol{f}(q), k)$ for all $q \in Q$. The concept of state truncations is also extended to configurations: $\tau_k(\mathbf{C})$ is the configuration such that $\tau_k(\mathbf{C})(d) \coloneqq \tau_k(\mathbf{C}(d))$ for all $d \in \mathbb{D}$. From a sufficiently large threshold, the stability and output of a configuration remain unchanged under state truncations:

▶ **Lemma 14.** *Let ψ be a predicate computed by a population protocol with unordered data. Let S_b be the set of stable configurations with output b of the protocol. There exists $k \geq 1$ such that, for all $b \in \{0, 1\}$, we have $\mathbf{C} \in \mathcal{S}_b$ iff $\tau_k(\mathbf{C}) \in \mathcal{S}_b$.*

Given a configuration $\mathbf{C}$ and a form $\boldsymbol{f}$, let $\#_{\boldsymbol{f}}(\mathbf{C}) \coloneqq |\{d \in \mathbb{D} : \mathbf{C}(d) = \boldsymbol{f}\}|$. Due to the nature of immediate observation protocols, it is always possible to take a form $\boldsymbol{f}$ of color d present in an configuration $\mathbf{C}$, duplicate $\boldsymbol{f}$ with a fresh color d', and have the latter mimic the behaviour of the former.

▶ **Lemma 15.** *Let $\mathbf{C}$ and $\mathbf{C}'$ be configurations such that $\mathbf{C} \xrightarrow{*} \mathbf{C}'$. For every $d \in \operatorname{supp}(\mathbf{C})$ and $d' \in \mathbb{D} \setminus \operatorname{supp}(\mathbf{C})$, it is the case that $\mathbf{C} + (\mathbf{C}(d))_{d'} \xrightarrow{*} \mathbf{C}' + (\mathbf{C}'(d))_{d'}$.*

Combined with the fact that $\mathcal{U}$ has a finite basis, this allows to show that from some threshold, duplicating forms with fresh colors does not change the output of the population.

▶ **Lemma 16.** *Let ψ be a predicate computed by a population protocol with unordered data. Let $\boldsymbol{f}$ be a form with $\mathrm{act}(\boldsymbol{f}) \subseteq I$. There exists $h(\boldsymbol{f}) \in \mathbb{N}$ such that, for all initial configuration $\mathbf{C}_0$ and $d \in \mathbb{D} \setminus \mathrm{supp}(\mathbf{C}_0)$ with $\#_{\boldsymbol{f}}(\mathbf{C}_0) \geq h(\boldsymbol{f})$, it is the case that $\psi(\mathbf{C}_0 + \boldsymbol{f}_d) = \psi(\mathbf{C}_0)$.*

The *form truncation* of a configuration $\mathbf{C}$, denoted $\sigma(\mathbf{C})$, is an (arbitrary) configuration such that $\sigma(\mathbf{C}) \sqsubseteq \mathbf{C}$ and $\#_{\boldsymbol{f}}(\sigma(\mathbf{C})) = \min(\#_{\boldsymbol{f}}(\mathbf{C}), h(\boldsymbol{f}))$ for every form $\boldsymbol{f}$, where $h(\boldsymbol{f})$ is given by Lemma 16. By Lemma 16, $\psi(\mathbf{C}_0)$ holds iff $\psi(\sigma(\mathbf{C}_0))$ holds. Moreover, Lemma 16 allows us to show that IO protocols are less expressive than the general model.

▶ **Proposition 17.** *No IO population protocol computes the majority predicate φ_{maj}.*

Proof. For the sake of contradiction, suppose that some IO protocol computes φ_{maj}. Let q_I be the unique initial state and let $\boldsymbol{f} \coloneqq \{\!\{q_I\}\!\}$. Let $h(\boldsymbol{f})$ be given by Lemma 16. Let $\mathbf{C}_0$ be an initial configuration such that

- $\mathbf{C}_0(d) = \sum_{i=1}^{h(\boldsymbol{f})+1} \boldsymbol{f}$ holds for a unique datum $d \in \mathbb{D}$, and
- $\mathbf{C}_0(d') = \boldsymbol{f}$ holds for exactly $h(\boldsymbol{f})$ other data $d' \in \{d_1, d_2, \ldots, d_{h(\boldsymbol{f})}\}$.

We have $\varphi_{\mathrm{maj}}(\mathbf{C}_0) = \mathit{true}$, since d has $h(\boldsymbol{f})+1$ agents in a population of $2 \cdot h(\boldsymbol{f})+1$ agents. Let $\mathbf{C}_0'$ be the initial configuration obtained from $\mathbf{C}_0$ by adding a datum $d^* \notin \mathrm{supp}(\mathbf{C}_0)$ such that $\mathbf{C}_0'(d^*) = \boldsymbol{f}$. By Lemma 16, $\varphi_{\mathrm{maj}}(\mathbf{C}_0') = \varphi_{\mathrm{maj}}(\mathbf{C}_0) = \mathit{true}$. However, datum d no longer has a majority in $\mathbf{C}_0'$, which is a contradiction. ◀

4.2 Predicates computed by IO protocols are interval predicates

▶ **Theorem 18.** *Let (Q, δ, I, O) be an immediate observation protocol with unordered data that computes a predicate ψ. The predicate ψ can be expressed as an interval predicate.*

Proof. We will express ψ as a finite Boolean combination of simple interval predicates.

Let h be the mapping given by Lemma 16. Let $\mathbb{T} \coloneqq \{\mathbf{C} : \psi(\mathbf{C}) = \mathit{true}\}$ and $\mathbb{T}_1 \coloneqq \{\mathbf{C} \in \mathbb{T} : \mathbf{C}(d, q) \leq k \text{ for all } d \in \mathbb{D}, q \in Q\}$. From Lemma 14, we learn that state truncations do not change the stability of a configuration. So, $\psi(\mathbf{C})$ holds iff $\bigvee_{\mathbf{C}' \in \mathbb{T}_1} \tau_k(\mathbf{C}) = \mathbf{C}'$ holds. Let $\mathbb{T}_2 \coloneqq \{\mathbf{C} \in \mathbb{T}_1 : \#_{\boldsymbol{f}}(\mathbf{C}) \leq h(\boldsymbol{f}) \text{ for all } \boldsymbol{f} \in \mathbb{F}\}$. It follows from Lemma 16 that $\psi(\mathbf{C})$ holds iff $\bigvee_{\mathbf{C}' \in \mathbb{T}_2} \sigma(\tau_k(\mathbf{C})) = \mathbf{C}'$ holds.

The latter is an infinite disjunction. Let us make it finite. Observe that if $\mathbf{C} \xrightarrow{*} \mathbf{C}'$ and $\overline{\mathbf{C}} \equiv \mathbf{C}$ hold, then there exists $\overline{\mathbf{C}'} \equiv \mathbf{C}'$ such that $\overline{\mathbf{C}} \xrightarrow{*} \overline{\mathbf{C}'}$. Moreover, note that equivalent configurations have the same output as they share the same active states. Indeed, $\mathbf{C} \equiv \overline{\mathbf{C}}$ iff $\bigwedge_{\boldsymbol{f} \in \mathbb{F}} \#_{\boldsymbol{f}}(\mathbf{C}) = \#_{\boldsymbol{f}}(\overline{\mathbf{C}})$. Hence, for every initial configuration $\mathbf{C} \equiv \overline{\mathbf{C}}$, we have $\psi(\mathbf{C}) = \psi(\overline{\mathbf{C}})$. Let $\mathbb{T}_2/\!\equiv$ be the set of all equivalence classes of $\equiv$ on $\mathbb{T}_2$, and let $\mathbb{T}_3$ be a set that contains one representative configuration per equivalence class of $\mathbb{T}_2/\!\equiv$. It is readily seen that $\psi(\mathbf{C})$ holds iff $\bigvee_{\mathbf{C}' \in \mathbb{T}_3} \sigma(\tau_k(\mathbf{C})) \equiv \mathbf{C}'$ holds.

Let us argue that $\mathbb{T}_3$ is finite. Let $\mathbb{F}_k \coloneqq \{\boldsymbol{f} \neq \mathbf{0} : \boldsymbol{f}(q) \leq k \text{ for all } q \in Q\}$. For every configuration $\mathbf{C} \in \mathbb{T}_1$, each form $\boldsymbol{f}$ with $\#_{\boldsymbol{f}}(\mathbf{C}) > 0$ belongs to $\mathbb{F}_k$. As $\mathbb{T}_2 \subseteq \mathbb{T}_1$, this also holds for configurations of $\mathbb{T}_2$. Given $\mathbf{C} \in \mathbb{T}_2$, we have $\#_{\boldsymbol{f}}(\mathbf{C}) \leq h(\boldsymbol{f})$ for all $\boldsymbol{f} \in \mathbb{F}_k$, and $\#_{\boldsymbol{f}}(\mathbf{C}) = 0$ for all $\boldsymbol{f} \notin \mathbb{F}_k$. Thus, as $\mathbb{F}_k$ is finite, we conclude that $\mathbb{T}_3$ is finite.

Let us now exploit our observations to express ψ as an interval predicate. Let us fix some $\mathbf{C}' \in \mathbb{T}_3$. It suffices to explain how to express "$\sigma(\tau_k(\mathbf{C})) \equiv \mathbf{C}'$". Indeed, as $\mathbb{T}_3$ is finite, we can conclude by taking the finite disjunction $\bigvee_{\mathbf{C}' \in \mathbb{T}_3} \sigma(\tau_k(\mathbf{C})) \equiv \mathbf{C}'$.

For every form $\boldsymbol{f} \in \mathbb{F}_k$, let $\mathrm{lt}(\boldsymbol{f}) \coloneqq \{q \in Q : \boldsymbol{f}(q) < k\}$, $\mathrm{eq}(\boldsymbol{f}) \coloneqq \{q \in Q : \boldsymbol{f}(q) = k\}$ and

$$\varphi_{\boldsymbol{f},d}(\mathbf{C}) \coloneqq \bigwedge_{q \in \mathrm{lt}(\boldsymbol{f})} (\mathbf{C}(d, q) = \boldsymbol{f}(q)) \wedge \bigwedge_{q \in \mathrm{eq}(\boldsymbol{f})} (\mathbf{C}(d, q) \geq \boldsymbol{f}(q)).$$

Observe that $\varphi_{\boldsymbol{f},d}(\mathbf{C})$ holds iff $\tau_k(\mathbf{C})(d) = \boldsymbol{f}$.

For every $\boldsymbol{f} \in \mathbb{F}_k$ such that $\#_{\boldsymbol{f}}(\mathbf{C}') < h(\boldsymbol{f})$, we define this formula, where $n \coloneqq \#_{\boldsymbol{f}}(\mathbf{C}')$:

$$\psi_{\boldsymbol{f}}(\mathbf{C}) \coloneqq \dot{\exists} d_1, d_2, \ldots, d_n \in \mathbb{D} : \bigwedge_{i=1}^{n} \varphi_{\boldsymbol{f},d_i}(\mathbf{C}) \wedge \neg \dot{\exists} d_1, d_2, \ldots, d_{n+1} \in \mathbb{D} : \bigwedge_{i=1}^{n+1} \varphi_{\boldsymbol{f},d_i}(\mathbf{C}).$$

For every $\boldsymbol{f} \in \mathbb{F}_k$ such that $\#_{\boldsymbol{f}}(\mathbf{C}') = h(\boldsymbol{f})$, we define this formula, where $n \coloneqq \#_{\boldsymbol{f}}(\mathbf{C}')$:

$$\psi_{\boldsymbol{f}}(\mathbf{C}) \coloneqq \dot{\exists} d_1, d_2, \ldots, d_n \in \mathbb{D} : \bigwedge_{i=1}^{n} \varphi_{\boldsymbol{f},d_i}(\mathbf{C}).$$

Observe that $\psi_{\boldsymbol{f}}$ is either a simple interval predicate or a Boolean combination of two simple interval predicates. Note that $\psi_{\boldsymbol{f}}(\mathbf{C})$ holds iff $\#_{\boldsymbol{f}}(\sigma(\tau_k(\mathbf{C}))) = \#_{\boldsymbol{f}}(\mathbf{C}')$ holds. This means that $\bigwedge_{\boldsymbol{f} \in \mathbb{F}_k} \psi_{\boldsymbol{f}}(\mathbf{C})$ holds iff $\sigma(\tau_k(\mathbf{C})) \equiv \mathbf{C}'$ holds, and hence we are done. ◀

4.3 An IO protocol for simple interval predicates

As Boolean combinations can be implemented (see end of Section 2.1), it suffices to describe a protocol for a simple interval predicate of the form (2). We refer to each $i \in [1..n]$ as a *role*. In the forthcoming set of states Q, we associate to each $q \in Q$ an element $\mathrm{elem}(q) \in [1..m]$. Each agent's element is set through the input; e.g. an agent mapped from symbol $(\bullet, x_1)$ is initially in a state q such that $\mathrm{elem}(q) = 1$. Let $Q_j \coloneqq \{q \in Q : \mathrm{elem}(q) = j\}$. For any two configurations $\mathbf{C}_a$ and $\mathbf{C}_b$ of an execution, any datum $d \in \mathbb{D}$ and any element $j \in [1..m]$, the invariant $\mathbf{C}_a(d, Q_j) = \mathbf{C}_b(d, Q_j)$ holds. We say that $d \in \mathbb{D}$ *matches* role i in configuration $\mathbf{C}$ if $\mathbf{C}(d, Q_j) \in T(i, j)$ holds for all $j \in [1..m]$. Let $r \coloneqq \max(r_1, \ldots, r_n) + 1$, where

$$r_i \coloneqq \max(\{\min T(i,j) : j \in [1..m]\} \cup \{\max T(i,j) : j \in [1..m], \sup T(i,j) < \infty\}).$$

Agents will not need to count beyond value r to decide whether a role is matched.

As for the majority protocol, our simple interval protocol works in stages, each one being necessary to ensure properties and invariants for the subsequent stages. In the *election stage*, a unique controller for the population and a single leader per datum of the support are selected; the former seeks to distribute a set of roles to the latter.

All agents contribute to the tallying of their immutable element j through the *counting stage*. This is done using the "tower method" described in [2], whereby two agents of the same datum, element *and* value meet and allow one of the two agents to increment its value. The maximal value computed in that manner is subsequently communicated to the (unique) datum leader.

Once the leaders carry correct counts for each element of their respective datum, they undertake roles that they match in the *distribution stage*. These roles can be swapped for other roles (as long as requirements are met) through a process of interrogating the controller. The controller is constantly notified of selected roles and updates its list of tasks accordingly.

If a fully assigned task list is obtained by the controller, it spreads a *true*-output throughout the population in what we call the *output propagation stage*. If that is not possible, leaders are in a consistent state of trial-and-error for their role assignments, ultimately failing to completely fill the task list, leaving the controller free to propagate its *false*-output.

▶ **Example 19.** Consider $n = m = 2$ with $T(1,1) \coloneqq [2..\infty)$, $T(1,2) \coloneqq [0..4]$, $T(2,1) \coloneqq \mathbb{N}$, $T(2,2) \coloneqq [1..\infty)$. Let $\mathbf{M} \coloneqq \{\!\{(\bullet, x_1), (\bullet, x_1), (\bullet, x_1), (\bullet, x_2), (\blacksquare, x_1), (\blacklozenge, x_2)\}\!\}$. Note that $r = 5$. Datum "$\bullet$" could match roles 1 and 2, "$\blacksquare$" cannot match any role, and "$\blacklozenge$" could match role 2.

After executing the protocol for a while, we may end up with the configuration illustrated in the table below. The third, fourth, fifth and sixth agents contain the correct value for their datum and element: $\mathbf{M}(\bullet, x_1) = 3$ and $\mathbf{M}(\bullet, x_2) = \mathbf{M}(\blacksquare, x_1) = \mathbf{M}(\blacklozenge, x_2) = 1$. The second agent has been elected controller. The last three agents have been elected their respective datum's leader and have collected the correct counts for each element. Either the $\bullet$-leader or the $\blacklozenge$-leader (possibly both) has notified the controller that they play role 2.

input	val	lead	ctrl	role	count of $[\#x_1, \#x_2]$	task list for [role 1, role 2]
$(\bullet, x_1)$	1					
$(\bullet, x_1)$	2		✔			[✘, ✔]
$(\bullet, x_1)$	**3**					
$(\bullet, x_2)$	**1**	✔		2	[3, 1]	
$(\blacksquare, x_1)$	**1**	✔			[1, 0]	
$(\blacklozenge, x_2)$	**1**	✔		2	[0, 1]	

The $\bullet$-leader may change its mind and decide to play role 1 after noticing the controller does not have its task 1 assigned. This switches its role to -2. Once the $\bullet$-leader notifies the controller, its role is set to 0 and (in doubt) the controller considers that role 2 is not assigned anymore. The $\bullet$-leader then changes its role to 1. Eventually the $\bullet$-leader and $\blacklozenge$-leader notify the controller that roles 1 and 2 are taken. This is summarized in these three snapshots:

input ⋯	role ⋯	task
$(\bullet, x_1)$		
$(\bullet, x_1)$		[✘, ✘]
$(\bullet, x_1)$		
$(\bullet, x_2)$	-2	
$(\blacksquare, x_1)$		
$(\blacklozenge, x_2)$	2	

⋯ role ⋯	task
	[✘, ✘]
0	
2	

⋯ role ⋯	task
	[✔, ✔]
1	
2	

◀

Note that while we rely on stages to describe our protocol, the distributed nature of the model implies that some stages may interfere with others. Therefore, we present here a list of potential problems and the way our protocol fixes them.

- While leader election is straightforward, role assignment for leaders can happen at any time before the actual leader is elected. This could lead to the controller being notified of a role assignment for which no current leader is assigned. Thus, when an agent loses its leadership status, it reverts its role to a negative value, meaning it will have to inform the controller of the change before returning to a passive value.
- A leader may take a role before having the correct counts. We provide a reset mechanism through which the leader falls into a "negative role". This forces it to then contact the controller and rectify the situation.
- A leader may have previously taken a role before realizing it does not actually meet the requirements. The leader is then forced to convey its mistake to the controller. But the controller it notifies may not ultimately be the population's controller. Therefore, after losing the controller status, an agent has to go to a negative controller state, meaning it must reset the controller's tasks before reverting to a passive value.
- There may be many leaders with the same role. To prevent deadlocks, we allow a leader to self-reassign to a new role if it notices the controller does not have the task filled.

4.3.1 States

The set of states is defined as $Q \coloneqq \{\mathit{false}, \mathit{true}\}^{n+2} \times [1..m] \times [1..r]^{m+1} \times [-n..n] \times \{-1, 0, 1\}$. For the sake of readability, we specify and manipulate states with these macros:

name	*values for* $q \in Q$	*values for* $q \in I$
$\mathrm{elem}(q)$	$[1..m]$	$j \in [1..m]$
$\mathrm{val}(q)$	$[1..r]$	1
$\mathrm{out}(q)$	$\{\mathit{false}, \mathit{true}\}$	*false*
$\mathrm{lead}(q)$	$\{\mathit{false}, \mathit{true}\}$	*true*
$\mathrm{role}(q)$	$[-n..n]$	0
$\mathrm{count}_\ell(q)$	$[1..r]$	1 if $\ell = j$, 0 otherwise
$\mathrm{ctrl}(q)$	$\{-1, 0, 1\}$	1
$\mathrm{task}_i(q)$	$\{\mathit{false}, \mathit{true}\}$	*false*

The input mapping is defined by $\iota(x_j) \coloneqq p_j$, where p_j is the unique state of I with $\mathrm{elem}(p_j) = j$. Informally, $\mathrm{elem}(q) = j$ indicates that the agent holds the j-th element; $\mathrm{val}(q)$ is the current tally of element $\mathrm{elem}(q)$ for the datum of the agent; $\mathrm{lead}(q)$ and $\mathrm{ctrl}(q)$ respectively indicate whether an agent is a datum leader or a controller; $\mathrm{role}(q)$ indicates the role for a leader; $\mathrm{count}_j(q)$ allows a datum leader to maintain the highest count currently witnessed for element j; $\mathrm{task}_i(q)$ allows the controller to maintain a list of the currently matched roles; and $\mathrm{out}(q)$ is the current belief of an agent on the output of the protocol.

Note that the rules presented in this section are used to succinctly describe transitions. A single rule may induce several transitions. Furthermore, for the sake of brevity, we mark rules allowing *mirror transitions* with an asterisk ($*$) next to the rule number. Mirror transitions are transitions in which an agent may observe its own state and react accordingly. Thus, a $*$-rule generating transitions whose precondition formula is $A(p) \wedge B(q)$ also generates a transition whose precondition is $A(q) \wedge B(q)$, effectively making state p the state of any "dummy agent". Note that q is still the only state to be updated to q'.

4.3.2 Leader and controller election

The first two rules are meant to elect a unique leader per datum present in the population, and a unique global controller for the whole population. For the remainder of the section, let us fix a fair execution $\mathbf{C}_0\mathbf{C}_1\cdots$ where $\mathbf{C}_0$ is initial.

rule	*state precondition*	*color precondition*	*state update*
(1)	$\mathrm{lead}(p)$ $\mathrm{lead}(q)$	$d_1 = d_2$	$\mathrm{role}(q') = -\lvert\mathrm{role}(q)\rvert$ $\neg\mathrm{lead}(q')$
(2)	$\mathrm{ctrl}(p) = 1$ $\mathrm{ctrl}(q) = 1$	*none*	$\mathrm{ctrl}(q') = -1$

Note that rule (1) guarantees that the agent losing leadership has its role set to a non-positive value. Similarly, rule (2) pushes the non-controller into a temporary intermediate state for its controller value, i.e. -1. Let $Q_L \coloneqq \{q \in Q : \mathrm{lead}(q)\}$ and $Q_C \coloneqq \{q \in Q : \mathrm{ctrl}(q) = 1\}$. The following lemma identifies the end of both elections.

▶ **Lemma 20.** *There exists* $\tau \in \mathbb{N}$ *such that* $|\mathbf{C}_\tau|_{Q_C} = |\mathbf{C}_{\tau+1}|_{Q_C} = \cdots = 1$, *and* $|\mathbf{C}_\tau(d)|_{Q_L} = |\mathbf{C}_{\tau+1}(d)|_{Q_L} = \cdots = 1$ *for every* $d \in \mathbb{D}$.

Let α denote the minimal value τ given by Lemma 20, which we refer to as the end of the election stage.

4.3.3 Element count by datum

The next rules allow to count how many agents with a common datum hold the same element. This count is ultimately communicated to the datum leader. Given $d \in \mathbb{D}$ and $\tau \in \mathbb{N}$, we say that a state $q \in Q$ is (d, j)-*valid* if $\mathrm{elem}(q) = j$ and $\mathrm{val}(q) = \min(\mathbf{C}_\tau(d, Q_j), r)$.

rule	*state precondition*	*color precondition*	*state update*
(3)	$\mathrm{elem}(p) = \mathrm{elem}(q)$ $\mathrm{val}(q) = \mathrm{val}(p) < r$	$d_1 = d_2$	$\mathrm{val}(q') = \mathrm{val}(q) + 1$
(4)*	$\mathrm{count}_{\mathrm{elem}(p)}(q) < \mathrm{val}(p)$ $\mathrm{lead}(q)$	$d_1 = d_2$	$\mathrm{count}_{\mathrm{elem}(p)}(q') = \mathrm{val}(p)$ `if` $(\mathrm{role}(q) > 0 \wedge$ $\mathrm{val}(p) \notin T(\mathrm{role}(q), \mathrm{elem}(p)))$: $\mathrm{role}(q') = -\mathrm{role}(q)$

Observe another correction mechanism; rule (4) guarantees that a leader with an assigned role $i > 0$ verifies that it can still assume role i after updating its count. The following lemmas explain that the correct counts are eventually provided to each datum leader.

▶ **Lemma 21.** *There exists $\tau \in \mathbb{N}$ such that, for every $\tau' \geq \tau$, $d \in \mathrm{supp}(\mathbf{C}_{\tau'})$ and $j \in [1..m]$, if $\mathbf{C}_0(d, Q_j) > 0$, then $\mathbf{C}_{\tau'}(d, q) > 0$ holds for some (d, j)-valid state q.*

▶ **Lemma 22.** *There exists $\tau \geq \alpha$ such that, for every $\tau' \geq \tau$, $d \in \mathrm{supp}(\mathbf{C}_{\tau'})$, $j \in [1..m]$ and $q \in \mathrm{act}(\mathbf{C}_{\tau'}(d)) \cap Q_L$, it is the case that $\mathrm{count}_j(q) = \min(\mathbf{C}_{\tau'}(d, Q_j), r)$.*

Let τ' and τ'' denote the minimal values τ given by Lemmas 21 and 22. From now on, let $\beta \coloneqq \max(\tau', \tau'')$.

4.3.4 Role distribution and task tracking

The following rules assign roles to leaders and allow leaders to reset their roles when possible, therefore preventing deadlocks. In rule (5), variable i can take any value from $[1..n]$.

rule	*state precondition*	*color precondition*	*state update*
(5)	$\mathrm{lead}(q)$ $\mathrm{role}(q) = 0$ $\bigwedge_{j \in [1..m]} \mathrm{count}_j(q) \in T(i, j)$	*none*	$\mathrm{role}(q') = i$
(6)*	$\mathrm{ctrl}(p)$ $\mathrm{lead}(q)$ $\mathrm{role}(q) = i > 0$ $\bigvee_{i' \in [1..n] \setminus \{i\}} (\neg \mathrm{task}_{i'}(p) \wedge$ $\bigwedge_{j \in [1..m]} \mathrm{count}_j(q) \in T(i', j))$	*none*	$\mathrm{role}(q') = -i$

This induces the following result, informally meaning that if a leader has taken a role, then it currently *believes* it can fill this role.

▶ **Lemma 23.** *For every $\tau \in \mathbb{N}$, $j \in [1..m]$ and $q \in \mathrm{act}(\mathbf{C}_\tau) \cap Q_L$ such that $\mathrm{role}(q) > 0$, it is the case that $\mathrm{count}_j(q) \in T(\mathrm{role}(q), j)$.*

These rules allow to update the controller's task list and reset roles when needed:

rule	*state precondition*	*color precondition*	*state update*
(7)*	$\text{role}(p) \neq 0$ $\text{ctrl}(q) = 1$	*none*	$\text{task}_{\lvert\text{role}(p)\rvert}(q') = (\text{role}(p) > 0)$
(8)*	$\text{ctrl}(p) = 1$ $\text{role}(q) < 0$ $\neg\text{task}_{\lvert\text{role}(q)\rvert}(p)$	*none*	$\text{role}(q') = 0$

To illustrate how rules (5) through (8) operate, we give the following example.

▶ **Example 24.** Recall Example 19, introduced earlier. Consider the configuration of its first snapshot. While we initially gave intuitions on how role reassignment might happen from this specific configuration, we give here a deeper analysis of the important configurations involved in this process.

input	val	lead	ctrl	role	count of $[\#x_1, \#x_2]$	task list for [role 1, role 2]
$(\bullet, x_1)$	1					
$(\bullet, x_1)$	2		✔			[✘, ✔]
$(\bullet, x_1)$	**3**					
$(\bullet, x_2)$	**1**	✔		2	$[3, 1]$	
$(\blacksquare, x_1)$	**1**	✔			$[1, 0]$	
$(\blacklozenge, x_2)$	**1**	✔		2	$[0, 1]$	

In the above, the $\bullet$-leader currently believes (rightly so) that it can fill roles 1 and 2. Observe that the controller has task 2 assigned. However, its task 1 is still unassigned. Therefore, rule (6) allows the $\bullet$-leader to initiate its reassignment by setting its role to -2 through an interaction with the controller. This leads to the following configuration:

input	val	lead	ctrl	role	count of $[\#x_1, \#x_2]$	task list for [role 1, role 2]
$(\bullet, x_1)$	1					
$(\bullet, x_1)$	2		✔			[✘, ✔]
$(\bullet, x_1)$	**3**					
$(\bullet, x_2)$	**1**	✔		-2	$[3, 1]$	
$(\blacksquare, x_1)$	**1**	✔			$[1, 0]$	
$(\blacklozenge, x_2)$	**1**	✔		2	$[0, 1]$	

Since its role is set to -2, the $\bullet$-leader now seeks to inform the controller that it should unassign role 2 from its task list. This is achieved on their next meeting through rule (7). We then have this next configuration:

input	val	lead	ctrl	role	count of $[\#x_1, \#x_2]$	task list for [role 1, role 2]
$(\bullet, x_1)$	1					
$(\bullet, x_1)$	2		✔			[✘, ✘]
$(\bullet, x_1)$	**3**					
$(\bullet, x_2)$	**1**	✔		-2	$[3, 1]$	
$(\blacksquare, x_1)$	**1**	✔			$[1, 0]$	
$(\blacklozenge, x_2)$	**1**	✔		2	$[0, 1]$	

Note that this does not mean that no leader currently has its role set to 2; indeed, the ◆-leader still has its role set to 2. Let us now assume that, immediately after reaching this configuration, the ●-leader and the controller meet again. Since the ●-leader observes that the controller no longer has its task 2 assigned, it can assume that either it unassigned it, some other leader did, or it was never assigned. In any case, it can safely reset its role to 0 through rule (8), giving us the following configuration:

input	val	lead	ctrl	role	count of $[\#x_1, \#x_2]$	task list for [role 1, role 2]
$(●, x_1)$	1					
$(●, x_1)$	2		✔			[✘, ✘]
$(●, x_1)$	**3**					
$(●, x_2)$	**1**	✔		0	$[3, 1]$	
$(■, x_1)$	**1**	✔			$[1, 0]$	
$(◆, x_2)$	**1**	✔		2	$[0, 1]$	

Observe that the ◆-leader could have met the controller before the ●-leader, thereby reassigning role 2 in the controller's task list and undoing the ●-leader's work. This would only delay the ●-leader's role resetting; through fairness, it would not endlessly prevent it.

From this last configuration, since the ●-leader's role is set to 0, it is now free to take any role it can fill through rule (5). Let us assume, for the sake of brevity, that it takes on role 1:

input	val	lead	ctrl	role	count of $[\#x_1, \#x_2]$	task list for [role 1, role 2]
$(●, x_1)$	1					
$(●, x_1)$	2		✔			[✘, ✘]
$(●, x_1)$	**3**					
$(●, x_2)$	**1**	✔		1	$[3, 1]$	
$(■, x_1)$	**1**	✔			$[1, 0]$	
$(◆, x_2)$	**1**	✔		2	$[0, 1]$	

Suppose the ◆-leader meets the controller before the ●-leader. Then, rule (7) assigns task 2 in the controller's task list. The ●-leader can no longer reset its role through rule (6) because the controller has task 2 already assigned. Therefore, when the ●-leader eventually meets the controller again, it finally assigns task 1 to its task list via rule (7).

input	val	lead	ctrl	role	count of $[\#x_1, \#x_2]$	task list for [role 1, role 2]
$(●, x_1)$	1					
$(●, x_1)$	2		✔			[✔, ✔]
$(●, x_1)$	**3**					
$(●, x_2)$	**1**	✔		1	$[3, 1]$	
$(■, x_1)$	**1**	✔			$[1, 0]$	
$(◆, x_2)$	**1**	✔		2	$[0, 1]$	

◀

The following lemmas show that at some point in the execution, a configuration is reached where agents who are neither leaders nor controllers no longer interact with other agents.

▶ **Lemma 25.** *There exists some* $\tau \in \mathbb{N}$ *such that for every* $\tau' \geq \tau$ *and* $q \in \text{act}(\mathbf{C}_{\tau'}) \setminus Q_L$*, it is the case that* $\text{role}(q) = 0$.

rule	*state precondition*	*color precondition*	*state update*
(9)	$\mathrm{ctrl}(p) = -1$ $\mathrm{ctrl}(q) = 1$	*none*	$\bigwedge_{i\in[1..m]} \neg\mathrm{task}_i(q')$
(10)	$\mathrm{ctrl}(p) = 1$ $\mathrm{ctrl}(q) = -1$ $\bigwedge_{i\in[1..m]} \neg\mathrm{task}_i(p)$	*none*	$\mathrm{ctrl}(q') = 0$

▶ **Lemma 26.** *There exists $\tau \geq \alpha$ such that for every $\tau' \geq \tau$ and $q \in \mathrm{act}(\mathbf{C}_{\tau'})$, it is the case that $\mathrm{ctrl}(q) \in \{0, 1\}$.*

Let τ' and τ'' denote the minimal values τ given by Lemmas 25 and 26. From now on, let $\gamma \coloneqq \max(\beta, \tau', \tau'')$. Informally, this delimits the configuration where there are no negative controllers, therefore preventing recurring resets of the controller's tasks through rule (9). Finally, the following lemma argues that past $\mathbf{C}_\gamma$, a controller can only have a task set to *true* if some leader is currently assuming the corresponding role (whether positive or negative).

▶ **Lemma 27.** *For every $\tau \geq \gamma$, $i \in [1..n]$ and $q \in \mathrm{act}(\mathbf{C}_\tau) \cap Q_C$ such that $\mathrm{task}_i(q)$ holds, there exists $q' \in \mathrm{act}(\mathbf{C}_\tau)$ such that $|\mathrm{role}(q')| = i$.*

4.3.5 Output propagation

The last rule allows the controller to communicate to the other agents whether it currently has its task list completely assigned or not. Note that the output of a state q is precisely the value of $\mathrm{out}(q)$, i.e. $O(q) \coloneqq \mathrm{out}(q)$.

rule	*state precondition*	*color precondition*	*state update*
(11)*	$\mathrm{ctrl}(p)$	*none*	$\mathrm{out}(q') = \bigwedge_{i=1}^{n} \mathrm{task}_i(p)$

▶ **Lemma 28.** *It is the case that $\psi(\mathbf{C}_0)$ holds iff there exists some $\tau \geq \gamma$ such that for every $\tau' \geq \tau$, there exists $q \in \mathrm{act}(\mathbf{C}_{\tau'}) \cap Q_C$ such that $\bigwedge_{i\in[1..n]} \mathrm{task}_i(q)$ holds.*

▶ **Corollary 29.** *There exists $\tau \in \mathbb{N}$ such that $O(\mathbf{C}_\tau) = O(\mathbf{C}_{\tau+1}) = \cdots = \psi(\mathbf{C}_0)$.*

5 Conclusion

In this article, we introduced population protocols with unordered data; we presented such a protocol that computes majority over an infinite data domain; and we established the expressive power of immediate observation protocols: they compute interval predicates.

This work initiates the study of population protocols operating over arbitrarily large domains. Hence, this opens the door to numerous exciting questions, e.g. on space-efficient and time-efficient protocols. In particular, the expressive power of our model remains open.

There exist results on logics over data multisets (e.g., see [22, 24]). In particular, the author of [22] provides a decidable logic reminiscent of Presburger arithmetic. It appears plausible that population protocols with unordered data compute (perhaps precisely) this logic. While we are fairly confident that remainder and threshold predicates with respect to the data counts can be computed in our model, the existential quantification, arising in the (non-ambiguous) solved forms of [22], seems more challenging to implement than the one of simple interval predicates.

Our model further relates to logic and automata on data words: inputs of a protocol with data can be seen as data words where Q is the alphabet and $\mathbb{D}$ is the data domain. Importantly, these data words are commutative, i.e., permutations do not change acceptance. For example, the logic $\mathsf{FO}^2(+1, \sim, <)$ of [8] allows to specify non-commutative properties such as "there is a block of a's followed by a block of b's". In this respect, this logic is too "strong". It is also too "weak" as it cannot express "for each datum, the number of a's is even". For this same reason, $\mathsf{EMSO}^2(+1, \sim)$, and equivalently weak data automata [21], is too "weak". The logic $\mathsf{EMSO}^2_{\#}(+1, \sim)$, and equivalently commutative data automata [25], can express the latter, but, again, the successor relation allows to express non-commutative properties on letters. Thus, while models related to data words have been studied and could influence research on the complete characterization of the expressive power of our model, we have yet to directly connect them to our model.

References

1. Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, 2006. doi:10.1007/s00446-005-0138-3.
2. Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007. doi:10.1007/s00446-007-0040-2.
3. Gregor Bankhamer, Petra Berenbrink, Felix Biermeier, Robert Elsässer, Hamed Hosseinpour, Dominik Kaaser, and Peter Kling. Population protocols for exact plurality consensus: How a small chance of failure helps to eliminate insignificant opinions. In *Proc. 41st ACM Symposium on Principles of Distributed Computing (PODC)*, pages 224–234, 2022. doi:10.1145/3519270.3538447.
4. Petra Berenbrink, Felix Biermeier, Christopher Hahn, and Dominik Kaaser. Loosely-stabilizing phase clocks and the adaptive majority problem. In *Proc. 1st Symposium on Algorithmic Foundations of Dynamic Networks (SAND)*, pages 7:1–7:17, 2022.
5. Petra Berenbrink, Robert Elsässer, Tom Friedetzky, Dominik Kaaser, Peter Kling, and Tomasz Radzik. A population protocol for exact majority with $O(\log^{5/3} n)$ stabilization time and $\Theta(\log n)$ states. In *Proc. 32nd International Symposium on Distributed Computing (DISC)*, pages 10:1–10:18, 2018. doi:10.4230/LIPIcs.DISC.2018.10.
6. Michael Blondin, Javier Esparza, Blaise Genest, Martin Helfrich, and Stefan Jaax. Succinct population protocols for Presburger arithmetic. In *Proc. 37th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 40:1–40:15, 2020. doi:10.4230/LIPIcs.STACS.2020.40.
7. Michael Blondin, Javier Esparza, Stefan Jaax, and Philipp J. Meyer. Towards efficient verification of population protocols. *Formal Methods in System Design (FMSD)*, 57(3):305–342, 2021. doi:10.1007/s10703-021-00367-3.
8. Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data words. *ACM Transactions on Computational Logic (TOCL)*, 12(4):27:1–27:26, 2011. doi:10.1145/1970398.1970403.
9. Benedikt Bollig, Patricia Bouyer, and Fabian Reiter. Identifiers in registers – describing network algorithms with logic. In *Proc. 22nd International Conference on Foundations of Software Science and Computation Structures (FoSSaCS)*, pages 115–132, 2019. doi:10.1007/978-3-030-17127-8_7.
10. Benedikt Bollig, Fedor Ryabinin, and Arnaud Sangnier. Reachability in distributed memory automata. In *Proc. 29th EACSL Annual Conference on Computer Science Logic (CSL)*, pages 13:1–13:16, 2021. doi:10.4230/LIPIcs.CSL.2021.13.
11. Robert S. Boyer and J. Strother Moore. Mjrty: A fast majority vote algorithm. In *Automated Reasoning: Essays in Honor of Woody Bledsoe*, 1991.

12 Philipp Czerner, Roland Guttenberg, Martin Helfrich, and Javier Esparza. Fast and succinct population protocols for Presburger arithmetic. In *Proc. 1st Symposium on Algorithmic Foundations of Dynamic Networks (SAND)*, pages 11:1–11:17, 2022. doi:10.4230/LIPIcs.SAND.2022.11.

13 Giorgio Delzanno, Arnaud Sangnier, and Riccardo Traverso. Adding data registers to parameterized networks with broadcast. *Fundamenta Informaticae*, 143(3-4):287–316, 2016. doi:10.3233/FI-2016-1315.

14 David Doty, Mahsa Eftekhari, Leszek Gasieniec, Eric E. Severson, Przemyslaw Uznanski, and Grzegorz Stachowiak. A time and space optimal stable population protocol solving exact majority. In *Proc. 62nd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1044–1055, 2021. doi:10.1109/FOCS52979.2021.00104.

15 Javier Esparza, Pierre Ganty, Jérôme Leroux, and Rupak Majumdar. Verification of population protocols. *Acta Informatica*, 54(2):191–215, 2017. doi:10.1007/s00236-016-0272-3.

16 Leszek Gasieniec, David D. Hamilton, Russell Martin, Paul G. Spirakis, and Grzegorz Stachowiak. Deterministic population protocols for exact majority and plurality. In *Proc. 20th International Conference on Principles of Distributed Systems (OPODIS)*, pages 14:1–14:14, 2016. doi:10.4230/LIPIcs.OPODIS.2016.14.

17 Utkarsh Gupta, Preey Shah, S. Akshay, and Piotr Hofman. Continuous reachability for unordered data Petri nets is in PTime. In *Proc. 22nd International Conference on Foundations of Software Science and Computation Structures (FoSSaCS)*, pages 260–276, 2019. doi:10.1007/978-3-030-17127-8_15.

18 Piotr Hofman, Slawomir Lasota, Ranko Lazic, Jérôme Leroux, Sylvain Schmitz, and Patrick Totzke. Coverability trees for Petri nets with unordered data. In *Proc. 19th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS)*, pages 445–461, 2016. doi:10.1007/978-3-662-49630-5_26.

19 Piotr Hofman, Jérôme Leroux, and Patrick Totzke. Linear combinations of unordered data vectors. In *Proc. 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–11, 2017. doi:10.1109/LICS.2017.8005065.

20 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theoretical Computer Science*, 134(2):329–363, 1994. doi:10.1016/0304-3975(94)90242-9.

21 Ahmet Kara, Thomas Schwentick, and Tony Tan. Feasible automata for two-variable logic with successor on data words. In *Proc. 6th International Conference on Language and Automata Theory and Applications (LATA)*, volume 7183, pages 351–362, 2012. doi:10.1007/978-3-642-28332-1_30.

22 Denis Lugiez. Multitree automata that count. *Theoretical Computer Science*, 333(1-2):225–263, 2005. doi:10.1016/j.tcs.2004.10.023.

23 Othon Michail and Paul G. Spirakis. Elements of the theory of dynamic networks. *Communications of the ACM*, 61(2):72, 2018. doi:10.1145/3156693.

24 Ruzica Piskac and Viktor Kuncak. Decision procedures for multisets with cardinality constraints. In *Proc. 9th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI)*, pages 218–232, 2008. doi:10.1007/978-3-540-78163-9_20.

25 Zhilin Wu. Commutative data automata. In *Proc. 26th International Workshop/21st Annual Conference of the EACSL on Computer Science Logic (CSL)*, volume 16, pages 528–542, 2012. doi:10.4230/LIPIcs.CSL.2012.528.

Network Satisfaction Problems Solved by k-Consistency

Manuel Bodirsky
Institut für Algebra, TU Dresden, Germany

Simon Knäuer
Institut für Algebra, TU Dresden, Germany

Abstract

We show that the problem of deciding for a given finite relation algebra $\mathbf{A}$ whether the network satisfaction problem for $\mathbf{A}$ can be solved by the k-consistency procedure, for some $k \in \mathbb{N}$, is undecidable. For the important class of finite relation algebras $\mathbf{A}$ with a normal representation, however, the decidability of this problem remains open. We show that if $\mathbf{A}$ is symmetric and has a flexible atom, then the question whether NSP($\mathbf{A}$) can be solved by k-consistency, for some $k \in \mathbb{N}$, is decidable (even in polynomial time in the number of atoms of $\mathbf{A}$). This result follows from a more general sufficient condition for the correctness of the k-consistency procedure for finite symmetric relation algebras. In our proof we make use of a result of Alexandr Kazda about finite binary conservative structures.

2012 ACM Subject Classification Theory of computation → Finite Model Theory; Theory of computation → Problems, reductions and completeness; Theory of computation → Complexity theory and logic

Keywords and phrases Constraint Satisfaction, Computational Complexity, Relation Algebras, Network Satisfaction, Qualitative Reasoning, k-Consistency, Datalog

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.116

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `https://arxiv.org/abs/2304.12871` [13]

Funding *Manuel Bodirsky*: The author has received funding from the European Union (ERC Grant NO. 681988, CSP-Infinity and ERC Synergy Grant No. 101071674, POCOCOP).
Simon Knäuer: The author was supported by DFG Graduiertenkolleg 1763 (QuantLA).

1 Introduction

Many computational problems in qualitative temporal and spatial reasoning can be phrased as *network satisfaction problems (NSPs)* for *finite relation algebras*. Such a network consists of a finite set of nodes, and a labelling of pairs of nodes by elements of the relation algebra. In applications, such a network models some partial (and potentially inconsistent) knowledge that we have about some temporal or spatial configuration. The computational task is to replace the labels by *atoms* of the relation algebra such that the resulting network has an embedding into a representation of the relation algebra. In applications, this embedding provides a witness that the input configuration is consistent (a formal definition of relation algebras, representations, and the network satisfaction problem can be found in Section 2.1). The computational complexity of the network satisfaction problem depends on the fixed finite relation algebra, and is of central interest in the mentioned application areas. Relation algebras have been studied since the 40's with famous contributions of Tarski [41], Lyndon [34], McKenzie [37, 38], and many others, with renewed interest since the 90s [7, 11, 22, 25–27, 30].

One of the most prominent algorithms for solving NSPs in polynomial time is the so-called *path consistency procedure*. The path consistency procedure has a natural generalisation to the *k-consistency procedure*, for some fixed $k \geq 3$. Such consistency algorithms have a number of advantages: e.g., they run in polynomial time, and they are *one-sided correct*, i.e., if they reject an instance, then we can be sure that the instance is unsatisfiable. Because of these properties, consistency algorithms can be used to prune the search space in exhaustive approaches that are used if the network consistency problem is NP-complete. The question for what temporal and spatial reasoning problems the k-consistency procedure provides a necessary and sufficient condition for satisfiability is among the most important research problems in the area [9, 40]. The analogous problem for so-called *constraint satisfaction problems (CSPs)* was posed by Feder and Vardi [23] and has been solved for finite-domain CSPs by Barto and Kozik [5]. Their result also shows that for a given finite-domain template, the question whether the corresponding CSP can be solved by the k-consistency procedure can be decided in polynomial time.

In contrast, we show that there is no algorithm that decides for a given finite relation algebra $\mathbf{A}$ whether NSP($\mathbf{A}$) can be solved by the k-consistency procedure, for some $k \in \mathbb{N}$. The question is also undecidable for every fixed $k \geq 3$; in particular, there is no algorithm that decides whether NSP($\mathfrak{A}$) can be solved by the path consistency procedure. Our proof relies on results of Hirsch [29] and Hirsch and Hodkinson [25]. The proof also shows that Hirsch's *Really Big Complexity Problem (RBCP; [27])* is undecidable. The RBCP asks for a description of those finite relation algebras $\mathbf{A}$ whose NSP can be solved in polynomial time.

Many of the classic examples of relation algebras that are used in temporal and spatial reasoning, such as the point algebra, Allen's Interval Algebra, RCC5, RCC8, have so-called *normal representations*, which are representations that are particularly well-behaved from a model theory perspective [7, 9, 27]. The importance of normal representations combined with our negative results for general finite relation algebras prompts the question whether solvability of the NSP by the k-consistency procedure can at least be characterised for relation algebras $\mathbf{A}$ with a normal representation. Our main result is a sufficient condition that implies that NSP($\mathbf{A}$) can be solved by the k-consistency procedure (Theorem 30). The condition can be checked algorithmically for a given $\mathbf{A}$. Moreover, for symmetric relation algebras with a flexible atom, which form a large subclass of the class of relation algebras with a normal representation, our condition provides a necessary and sufficient criterion for solvability by k-consistency (Theorem 39). We prove that the NSP for every symmetric relation algebra with a flexible atom that cannot be solved by the k-consistency procedure is already NP-complete. Finally, for symmetric relation algebras with a flexible atom our tractability condition can even be checked in polynomial time for a given relation algebra $\mathbf{A}$ (Theorem 42).

In our proof, we exploit a connection between the NSP for relation algebras $\mathbf{A}$ with a normal representation and finite-domain constraint satisfaction problems. In a next step, this allows us to use strong results for CSPs over finite domains. There are similarities between the fact that the set of relations of a representation of $\mathbf{A}$ is closed under taking unions on the one hand, and so-called *conservative finite-domain CSPs* [3, 16–18] on the other hand; in a conservative CSP the set of allowed constraints in instances of the CSP contains all unary relations. The complexity of conservative CSPs has been classified long before the solution of the Feder-Vardi Dichotomy Conjecture [19, 23, 42, 43]. Moreover, there are particularly elegant descriptions of when a finite-domain conservative CSP can be solved by the k-consistency procedure for some $k \in \mathbb{N}$ (see, e.g., Theorem 2.17 in [17]). Our approach is to turn the

similarities into a formal correspondence so that we can use these results for finite-domain conservative CSPs to prove that k-consistency solves NSP($\mathbf{A}$). A key ingredient here is a contribution of Kazda [31] about conservative *binary* CSPs.

All the missing proofs and details can be found in an extended version of this article [13].

2 Preliminaries

A *signature* τ is a set of function or relation symbols each of which has an associated finite *arity* $k \in \mathbb{N}$. A τ-structure $\mathfrak{A}$ consists of a set A together with a function $f^{\mathfrak{A}} \colon A^k \to A$ for every function symbol $f \in \tau$ of arity k and a relation $R^{\mathfrak{A}} \subseteq A^k$ for every relation symbol $R \in \tau$ of arity k. The set A is called the *domain of* $\mathfrak{A}$. Let $\mathfrak{A}$ and $\mathfrak{B}$ be τ-structures. The *(direct) product* $\mathfrak{C} = \mathfrak{A} \times \mathfrak{B}$ is the τ-structure where

- $A \times B$ is the domain of $\mathfrak{C}$;
- for every relation symbol Q of arity $n \in \mathbb{N}$ and every tuple $((a_1, b_1), \dots, (a_n, b_n)) \in (A \times B)^n$, we have that $((a_1, b_1), \dots, (a_n, b_n)) \in Q^{\mathfrak{C}}$ if and only if $(a_1, \dots, a_n) \in Q^{\mathfrak{A}}$ and $(b_1, \dots, b_n) \in Q^{\mathfrak{B}}$;
- for every function symbol Q of arity $n \in \mathbb{N}$ and every tuple $((a_1, b_1), \dots, (a_n, b_n)) \in (A \times B)^n$, we have that

$$Q^{\mathfrak{C}}((a_1, b_1), \dots, (a_n, b_n)) := (Q^{\mathfrak{A}}(a_1, \dots, a_n), Q^{\mathfrak{B}}(b_1, \dots, b_n)).$$

We denote the (direct) product $\mathfrak{A} \times \mathfrak{A}$ by $\mathfrak{A}^2$. The k-fold product $\mathfrak{A} \times \cdots \times \mathfrak{A}$ is defined analogously and denoted by $\mathfrak{A}^k$. Structures with a signature that only contains function symbols are called *algebras* and structures with purely relational signature are called *relational structures*. Since we do not deal with signatures of mixed type in this article, we will use the term structure for relational structures only.

2.1 Relation Algebras

Relation algebras are particular algebras; in this section we recall their definition and state some of their basic properties. We introduce *proper* relation algebras, move on to abstract relation algebras, and finally define representations of relation algebras. For an introduction to relation algebras we recommend the textbook by Maddux [36]. Proper relation algebras are algebras whose domain is a set of binary relations over a common domain, and which are equipped with certain operations on binary relations.

▶ **Definition 1.** *Let D be a set and $\mathcal{R}$ a set of binary relations over D such that $(\mathcal{R}; \cup, {}^{-}, 0, 1, \mathrm{Id}, {}^{\smile}, \circ)$ is an algebra with operations defined as follows:*

1. $0 := \emptyset$,
2. $1 := \bigcup \mathcal{R}$,
3. $\mathrm{Id} := \{(x, x) \mid x \in D\}$,
4. $a \cup b := \{(x, y) \mid (x, y) \in a \vee (x, y) \in b\}$,
5. $\bar{a} := 1 \setminus a$,
6. $a^{\smile} := \{(x, y) \mid (y, x) \in a\}$,
7. $a \circ b := \{(x, z) \mid \exists y \in D : (x, y) \in a \text{ and } (y, z) \in b\}$,

for $a, b \in \mathcal{R}$. Then $(\mathcal{R}; \cup, {}^{-}, 0, 1, \mathrm{Id}, {}^{\smile}, \circ)$ is called a proper relation algebra.

The class of all proper relation algebras is denoted by PA. Abstract relation algebras are a generalisation of proper relation algebras where the domain does not need to be a set of binary relations.

▶ **Definition 2.** *An* (abstract) relation algebra $\mathbf{A}$ *is an algebra with domain* A *and signature* $\{\cup, \bar{}, 0, 1, \mathrm{Id}, \breve{}, \circ\}$ *such that*

1. *the structure* $(A; \cup, \cap, \bar{}, 0, 1)$*, with* $\cap$ *defined by* $x \cap y := \overline{(\bar{x} \cup \bar{y})}$*, is a Boolean algebra,*
2. $\circ$ *is an associative binary operation on* A*, called* composition*,*
3. *for all* $a, b, c, \in A$*:* $(a \cup b) \circ c = (a \circ c) \cup (b \circ c)$*,*
4. *for all* $a \in A$*:* $a \circ \mathrm{Id} = a$*,*
5. *for all* $a \in A$*:* $\breve{\breve{a}} = a$*,*
6. *for all* $a, b \in A$*:* $\breve{x} = \breve{a} \cup \breve{b}$ *where* $x := a \cup b$*,*
7. *for all* $a, b \in A$*:* $\breve{x} = \breve{b} \circ \breve{a}$ *where* $x := a \circ b$*,*
8. *for all* $a, b, c \in A$*:* $\bar{b} \cup (\breve{a} \circ (\overline{(a \circ b)})) = \bar{b}$*.*

We denote the class of all relation algebras by RA. Let $\mathbf{A} = (A; \cup, \bar{}, 0, 1, \mathrm{Id}, \breve{}, \circ)$ be a relation algebra. By definition, $(A; \cup, \cap, \bar{}, 0, 1)$ is a Boolean algebra and therefore induces a partial order $\leq$ on A, which is defined by $x \leq y :\Leftrightarrow x \cup y = y$. Note that for proper relation algebras this ordering coincides with the set-inclusion order. The minimal elements of this order in $A \setminus \{0\}$ are called *atoms*. The set of atoms of $\mathbf{A}$ is denoted by A_0. Note that for the finite Boolean algebra $(A; \cup, \cap, \bar{}, 0, 1)$ each element $a \in A$ can be uniquely represented as the union $\cup$ (or "join") of elements from a subset of A_0. We will often use this fact and directly denote elements of the relation algebra $\mathbf{A}$ by subsets of A_0.

By item 3. in Definition 2 the values of the composition operation $\circ$ in $\mathbf{A}$ are completely determined by the values of $\circ$ on A_0. This means that for a finite relation algebra the operation $\circ$ can be represented by a multiplication table for the atoms A_0.

An algebra with signature $\tau = \{\cup, \bar{}, 0, 1, \mathrm{Id}, \breve{}, \circ\}$ with corresponding arities 2, 1, 0, 0, 0, 1, and 2 that is isomorphic to some proper relation algebra is called *representable*. The class of representable relation algebras is denoted by RRA. Since every proper relation algebra and therefore also every representable relation algebra satisfies the axioms from the previous definition we have PA $\subseteq$ RRA $\subseteq$ RA. A classical result of Lyndon [34] states that there exist finite relation algebras $\mathbf{A} \in$ RA that are not representable; so the inclusions above are proper. If a relation algebra $\mathbf{A}$ is representable then the isomorphism to a proper relation algebra is usually called the *representation of* $\mathbf{A}$.

We will be interested in the model-theoretic behavior of sets of relations which form the domain of a proper relation algebra, and therefore consider relational structures whose relations are precisely the relations of a proper relation algebra. If the set of relations of a relational structure $\mathfrak{B}$ forms a proper relation algebra which is a representation of some abstract relation algebra $\mathbf{A}$, then it will be convenient to also call $\mathfrak{B}$ a representation of $\mathbf{A}$.

▶ **Definition 3.** *Let* $\mathbf{A} \in$ RA*. A* representation of $\mathbf{A}$ *is a relational structure* $\mathfrak{B}$ *such that*

- $\mathfrak{B}$ *is an* A*-structure, i.e., the elements of* A *are binary relation symbols of* $\mathfrak{B}$*;*
- *The map* $a \mapsto a^{\mathfrak{B}}$ *is an isomorphism between the abstract relation algebra* $\mathbf{A}$ *and the proper relation algebra* $(\mathcal{R}; \cup, \bar{}, 0, 1, \mathrm{Id}, \breve{}, \circ)$ *with domain* $\mathcal{R} := \{a^{\mathfrak{B}} \mid a \in A\}$*.*

Recall that the set of atoms of a relation algebra $\mathbf{A} = (A; \cup, \bar{}, 0, 1, \mathrm{Id}, \breve{}, \circ)$ is denoted by A_0. The following definitions are crucial for this article.

▶ **Definition 4.** *A tuple* $(x, y, z) \in (A_0)^3$ *is called an* allowed triple (of $\mathbf{A}$) *if* $z \leq x \circ y$*. Otherwise,* (x, y, z) *is called a* forbidden triple (of $\mathbf{A}$)*; in this case* $\overline{z} \cup \overline{x \circ y} = 1$*. We say that a relational* A*-structure* $\mathfrak{B}$ induces a forbidden triple (from $\mathbf{A}$) *if there exist* $b_1, b_2, b_3 \in B$ *and* $(x, y, z) \in (A_0)^3$ *such that* $x(b_1, b_2), y(b_2, b_3)$ *and* $z(b_1, b_3)$ *hold in* $\mathfrak{B}$ *and* (x, y, z) *is a forbidden triple of* $\mathbf{A}$*.*

Note that a representation of $\mathbf{A}$ by definition does not induce a forbidden triple. A relation $R \subseteq A^3$ is called *totally symmetric* if for every bijection $\pi\colon \{1,2,3\} \to \{1,2,3\}$ we have

$$(a_1, a_2, a_3) \in R \;\Rightarrow\; (a_{\pi(1)}, a_{\pi(2)}, a_{\pi(3)}) \in R.$$

The following is an immediate consequence of the definition of allowed triples.

▶ Remark 5. The set of allowed triples of a symmetric relation algebra $\mathbf{A}$ is totally symmetric.

2.2 The Network Satisfaction Problem

In this section we present computational decision problems associated with relation algebras. We first introduce the inputs to these decision problems, so-called $\mathbf{A}$-networks.

▶ **Definition 6.** *Let $\mathbf{A}$ be a relation algebra. An* $\mathbf{A}$-network $(V; f)$ *is a finite set V together with a partial function $f\colon E \subseteq V^2 \to A$, where E is the domain of f. An $\mathbf{A}$-network $(V; f)$ is* satisfiable in a representation $\mathfrak{B}$ *of $\mathbf{A}$ if there exists an assignment $s\colon V \to B$ such that for all $(x, y) \in E$ the following holds:*

$$(s(x), s(y)) \in f(x, y)^{\mathfrak{B}}.$$

An $\mathbf{A}$-network $(V; f)$ is satisfiable *if there exists a representation $\mathfrak{B}$ of $\mathbf{A}$ such that $(V; f)$ is satisfiable in $\mathfrak{B}$.*

With these notions we can define the network satisfaction problem.

▶ **Definition 7.** *The* (general) network satisfaction problem *for a finite relation algebra $\mathbf{A}$, denoted by* $\mathrm{NSP}(\mathbf{A})$*, is the problem of deciding whether a given $\mathbf{A}$-network is satisfiable.*

In the following we assume that for an $\mathbf{A}$-network $(V; f)$ it holds that $f(V^2) \subseteq A \setminus \{0\}$. Otherwise, $(V; f)$ is not satisfiable. Note that every $\mathbf{A}$-network $(V; f)$ can be viewed as an A-structure $\mathfrak{C}$ on the domain V: for all $x, y \in V$ in the domain of f and $a \in A$ the relation $a^{\mathfrak{C}}(x, y)$ holds if and only if $f(x, y) = a$.

It is well-known that for relation algebras $\mathbf{A}_1$ and $\mathbf{A}_2$ the direct product $\mathbf{A}_1 \times \mathbf{A}_2$ is also a relation algebra (see, e.g., [30]). We will see in Lemma 9 that the direct product of representable relation algebras is also a representable relation algebra.

▶ **Definition 8.** *Let $\mathbf{A}_1$ and $\mathbf{A}_2$ be representable relation algebras. Let $\mathfrak{B}_1$ and $\mathfrak{B}_2$ be representations of $\mathbf{A}_1$ and $\mathbf{A}_2$ with disjoint domains. Then the* union representation *of the direct product $\mathbf{A}_1 \times \mathbf{A}_2$ is the $(A_1 \times A_2)$-structure $\mathfrak{B}_1 \uplus \mathfrak{B}_2$ on the domain $B_1 \uplus B_2$ with the following definition for all $(a_1, a_2) \in A_1 \times A_2$:*

$$(a_1, a_2)^{\mathfrak{B}_1 \uplus \mathfrak{B}_2} := a_1^{\mathfrak{B}_1} \cup a_2^{\mathfrak{B}_2}.$$

The following well-known lemma establishes a connection between products of relation algebras and union representations (see, e.g., Lemma 7 in [21] or Lemma 3.7 in [30]); it states that union representations are indeed representations. A proof of the lemma can be found, for example, in the extended version of this article [13].

▶ **Lemma 9.** *Let $\mathbf{A}_1$ and $\mathbf{A}_2$ be relation algebras. Then the following holds:*

1. *If $\mathfrak{B}_1$ and $\mathfrak{B}_2$ are representations of $\mathbf{A}_1$ and $\mathbf{A}_2$ with disjoint domains, then $\mathfrak{B}_1 \uplus \mathfrak{B}_2$ is a representation of $\mathbf{A}_1 \times \mathbf{A}_2$.*
2. *If $\mathfrak{B}$ is a representation of $\mathbf{A}_1 \times \mathbf{A}_2$, then there exist representations $\mathfrak{B}_1$ and $\mathfrak{B}_2$ of $\mathbf{A}_1$ and $\mathbf{A}_2$ such that $\mathfrak{B}$ is isomorphic to $\mathfrak{B}_1 \uplus \mathfrak{B}_2$.*

The following result uses Lemma 9 to obtain reductions between different network satisfaction problems. A similar statement can be found in Lemma 7 from [21], however there the assumption on representability of the relation algebras $\mathbf{A}$ and $\mathbf{B}$ is missing. Note that without this assumption the statement is not longer true. Consider relation algebras $\mathbf{A}$ and $\mathbf{B}$ such that NSP($\mathbf{A}$) is undecidable and $\mathbf{B}$ does not have a representation. Then $\mathbf{A} \times \mathbf{B}$ does also not have a representation (see Lemma 9) and hence NSP($\mathbf{A} \times \mathbf{B}$) is trivial. We observe that the undecidable problem NSP($\mathbf{A}$) cannot have a polynomial-time reduction to the trivial problem NSP($\mathbf{A} \times \mathbf{B}$).

▶ **Lemma 10.** *Let* $\mathbf{A}, \mathbf{B} \in$ RRA *be finite. Then there exists a polynomial-time reduction from* NSP($\mathbf{A}$) *to* NSP($\mathbf{A} \times \mathbf{B}$).

Proof. Consider the following polynomial-time reduction from NSP($\mathbf{A}$) to NSP($\mathbf{A} \times \mathbf{B}$). We map a given $\mathbf{A}$-network $(V; f)$ to the $(\mathbf{A} \times \mathbf{B})$-network $(V; f')$ where f' is defined by $f'(x, y) := (f(x, y), 0)$. This reduction can be computed in polynomial time.

▷ Claim 1. If $(V; f)$ is satisfiable then $(V; f')$ is also satisfiable. Let $\mathfrak{A}$ be a representation of $\mathbf{A}$ in which $(V; f)$ is satisfiable and let $\mathfrak{B}$ be an arbitrary representation of $\mathbf{B}$. By Lemma 9, the structure $\mathfrak{A} \uplus \mathfrak{B}$ is a representation of $\mathbf{A} \times \mathbf{B}$. Moreover, the definition of union representations (Definition 8) yields that the $(\mathbf{A} \times \mathbf{B})$-network $(V; f')$ is satisfiable in $\mathfrak{A} \uplus \mathfrak{B}$.

▷ Claim 2. If $(V; f')$ is satisfiable then $(V; f)$ is satisfiable. Assume that $(V; f')$ is satisfiable in some representation $\mathfrak{C}$ of $\mathbf{A} \times \mathbf{B}$. By item 2 in Lemma 9 we get that $\mathfrak{C}$ is isomorphic to $\mathfrak{A} \uplus \mathfrak{B}$, where $\mathfrak{A}$ and $\mathfrak{B}$ are representations of $\mathbf{A}$ and $\mathbf{B}$. It again follows from the definition of union representations that $(V; f)$ is satisfiable in the representation $\mathfrak{A}$ of $\mathbf{A}$.

This shows the correctness of the polynomial-time reduction from NSP($\mathbf{A}$) to NSP($\mathbf{A} \times \mathbf{B}$) and finishes the proof. ◀

2.3 Normal Representations and Constraint Satisfaction Problems

We consider a subclass of RRA introduced by Hirsch in 1996. For relation algebras $\mathbf{A}$ from this class, NSP($\mathbf{A}$) corresponds naturally to a constraint satisfaction problem. In the following let $\mathbf{A}$ be in RRA. We call an $\mathbf{A}$-network $(V; f)$ *closed* (transitively closed in the work by Hirsch [28]) if f is total and for all $x, y, z \in V$ it holds that

- $f(x, x) \leq \mathrm{Id}$,
- $f(x, y) = \breve{a}$ for $a = f(y, x)$,
- $f(x, z) \leq f(x, y) \circ f(y, z)$.

It is called *atomic* if the range of f only contains atoms from $\mathbf{A}$.

▶ **Definition 11** (from [27]). *Let* $\mathfrak{B}$ *be a representation of* $\mathbf{A}$*. Then* $\mathfrak{B}$ *is called*

- fully universal*, if every atomic closed* $\mathbf{A}$*-network is satisfiable in* $\mathfrak{B}$*;*
- square*, if* $1^{\mathfrak{B}} = B^2$*;*
- homogeneous*, if for every isomorphism between finite substructures of* $\mathfrak{B}$ *there exists an automorphism of* $\mathfrak{B}$ *that extends this isomorphism;*
- normal*, if it is fully universal, square and homogeneous.*

We now investigate the connection between NSP($\mathbf{A}$) for a finite relation algebra with a normal representation $\mathfrak{B}$ and constraint satisfaction problems. Let τ be a finite relational signature and let $\mathfrak{B}$ be a (finite or infinite) τ-structure. Then the *constraint satisfaction problem for* $\mathfrak{B}$, denoted by CSP($\mathfrak{B}$), is the computational problem of deciding whether a finite input structure $\mathfrak{A}$ has a homomorphism to $\mathfrak{B}$. The structure $\mathfrak{B}$ is called the template of CSP($\mathfrak{B}$).

$\circ$	Id	$<$	$>$
Id	Id	$<$	$>$
$<$	$<$	$<$	1
$>$	$>$	1	$>$

Figure 1 Multiplication table of the point algebra $\mathbf{P}$.

Consider the following translation which associates to each $\mathbf{A}$-network $(V; f)$ an A-structure $\mathfrak{C}$ as follows: the set V is the domain of $\mathfrak{C}$ and $(x, y) \in C^2$ is in a relation $a^{\mathfrak{C}}$ if and only if (x, y) is in the domain of f and $f(x, y) = a$ holds. For the other direction let $\mathfrak{C}$ be an A-structure with domain C and consider the $\mathbf{A}$-network $(C; f)$ with the following definition: for every $x, y \in C$, if (x, y) does not appear in any relation of $\mathfrak{C}$ we leave $f(x, y)$ undefined, otherwise let $a_1(x, y), \ldots, a_n(x, y)$ be all atomic formulas that hold in $\mathfrak{C}$. We compute in $\mathbf{A}$ the element $a := a_1 \cap \cdots \cap a_n$ and define $f(x, y) := a$.

The following theorem is based on the natural 1-to-1 correspondence between $\mathbf{A}$-networks and A-structures; it subsumes the connection between network satisfaction problems and constraint satisfaction problems.

▶ **Proposition 12** (Proposition 1.3.16 in [6], see also [7, 9]). *Let $\mathbf{A} \in$ RRA be finite. Then the following holds:*

1. *$\mathbf{A}$ has a representation $\mathfrak{B}$ such that* NSP($\mathbf{A}$) *and* CSP($\mathfrak{B}$) *are the same problem up to the translation between $\mathbf{A}$-networks and A-structures.*
2. *If $\mathbf{A}$ has a normal representation $\mathfrak{B}$ the problems* NSP($\mathbf{A}$) *and* CSP($\mathfrak{B}$) *are the same up to the translation between $\mathbf{A}$-networks and A-structures.*

Usually, normal representations of relation algebras are infinite relational structures. This means that the transfer from NSPs to CSPs from Proposition 12 results in CSPs over infinite templates, as in the following example.

▶ **Example 13.** Consider the *point algebra* $\mathbf{P}$. The set of atoms of $\mathbf{P}$ is $P_0 = \{\mathrm{Id}, <, >\}$. The composition operation $\circ$ on the atoms is given by the multiplication table in Figure 1. The table completely determines the composition operation $\circ$ on all elements of $\mathbf{P}$. Note that the structure $\mathfrak{P} := (\mathbb{Q};\ \emptyset, <, >, =, \leq, \geq, \neq, \mathbb{Q}^2)$ is the normal representation of $\mathbf{P}$ and therefore NSP($\mathbf{P}$) and CSP($\mathfrak{P}$) are the same problems up to the translation between networks and structures.

2.4 The Universal-Algebraic Approach

In this section we give a brief introduction to the the universal-algebraic approach to CSPs.

2.4.1 Polymorphisms

Let τ be a finite relational signature. A *polymorphism* of a τ-structure $\mathfrak{B}$ is a homomorphism f from $\mathfrak{B}^k$ to $\mathfrak{B}$, for some $k \in \mathbb{N}$ called the *arity of* f. We write Pol($\mathfrak{B}$) for the set of all polymorphisms of $\mathfrak{B}$. The set of polymorphisms is closed under composition, i.e., for all n-ary $f \in$ Pol($\mathfrak{B}$) and s-ary $g_1, \ldots, g_n \in$ Pol($\mathfrak{B}$) it holds that $f(g_1, \ldots, g_n) \in$ Pol($\mathfrak{B}$), where $f(g_1, \ldots, g_n)$ is a homomorphism from $\mathfrak{B}^s$ to $\mathfrak{B}$ defined as follows

$$f(g_1, \ldots, g_n)(x_1, \ldots, x_s) := f(g_1(x_1, \ldots, x_s), \ldots, g_n(x_1, \ldots, x_s)).$$

If $r_1, \dots, r_n \in B^k$ and $f\colon B^n \to B$ an n-ary operation, then we write $f(r_1, \dots, r_n)$ for the k-tuple obtained by applying f component-wise to the tuples $r_1, \dots, r_n$. We say that $f\colon B^n \to B$ *preserves* a k-ary relation $R \subseteq B^k$ if for all $r_1, \dots, r_n \in R$ it holds that $f(r_1, \dots, r_n) \in R$. We want to remark that the polymorphisms of $\mathfrak{B}$ are precisely those operations that preserve all relations from $\mathfrak{B}$.

A first-order τ-formula $\varphi(x_1, \dots, x_n)$ is called *primitive positive (pp)* if it has the form

$$\exists x_{n+1}, \dots, x_m(\varphi_1 \wedge \cdots \wedge \varphi_s)$$

where $\varphi_1, \dots, \varphi_s$ are atomic τ-formulas, i.e., formulas of the form $R(y_1, \dots, y_l)$ for $R \in \tau$ and $y_i \in \{x_1, \dots, x_m\}$, of the form $y = y'$ for $y, y' \in \{x_1, \dots, x_m\}$, or of the form $\bot$. We say that a relation R is *primitively positively definable over* $\mathfrak{A}$ if there exists a primitive positive τ-formula $\varphi(x_1, \dots, x_n)$ such that R is definable over $\mathfrak{A}$ by $\varphi(x_1, \dots, x_n)$.

▶ **Proposition 14** ([15, 24]). *Let $\mathfrak{B}$ be a τ-structure with a finite domain. Then the set of primitive positive definable relations in $\mathfrak{B}$ is exactly the set of relations preserved by* $\mathrm{Pol}(\mathfrak{B})$.

2.4.2 Atom Structures

In this section we introduce for every finite $\mathbf{A} \in \mathrm{RA}$ an associated finite structure, called the *atom structure* of $\mathbf{A}$. If $\mathbf{A}$ has a fully universal representation, then there exists a polynomial-time reduction from $\mathrm{NSP}(\mathbf{A})$ to the finite-domain constraint satisfaction problem $\mathrm{CSP}(\mathfrak{A}_0)$ (Proposition 16). Hence, this reduction provides polynomial-time algorithms to solve NSPs, whenever the CSP of the associated atom structure can be solved in polynomial-time. For a discussion of the atom structure and related objects we recommend Section 4 in [12].

▶ **Definition 15.** *The* atom structure of $\mathbf{A} \in \mathrm{RA}$ *is the finite relational structure $\mathfrak{A}_0$ with domain A_0 and the following relations:*

- *for every $x \in A$ the unary relation $x^{\mathfrak{A}_0} := \{a \in A_0 \mid a \leq x\}$,*
- *the binary relation $E^{\mathfrak{A}_0} := \{(a_1, a_2) \in A_0^2 \mid \breve{a_1} = a_2\}$,*
- *the ternary relation $R^{\mathfrak{A}_0} := \{(a_1, a_2, a_3) \in A_0^3 \mid a_3 \leq a_1 \circ a_2\}$.*

Note that $\mathfrak{A}_0$ has all subsets of A_0 as unary relations and that the relation $R^{\mathfrak{A}_0}$ consists of the allowed triples of $\mathbf{A} \in \mathrm{RRA}$. We say that an operation *preserves the allowed triples* if it preserves the relation $R^{\mathfrak{A}_0}$.

▶ **Proposition 16** ([11, 12]). *Let $\mathfrak{B}$ be a fully universal representation of a finite $\mathbf{A} \in \mathrm{RRA}$. Then there is a polynomial-time reduction from* $\mathrm{CSP}(\mathfrak{B})$ *to* $\mathrm{CSP}(\mathfrak{A}_0)$.

2.4.3 Conservative Clones

Let $\mathfrak{B}$ be a finite τ-structure. An operation $f\colon B^n \to B$ is called *conservative* if for all $x_1, \dots, x_n \in B$ it holds that $f(x_1, \dots, x_n) \in \{x_1, \dots, x_n\}$. The operation clone $\mathrm{Pol}(\mathfrak{B})$ is *conservative* if every $f \in \mathrm{Pol}(\mathfrak{B})$ is conservative. We call a relational structure $\mathfrak{B}$ *conservative* if $\mathrm{Pol}(\mathfrak{B})$ is conservative.

▶ Remark 17. Let $\mathfrak{A}_0$ be the atom structure of a finite relation algebra $\mathbf{A}$. Every $f \in \mathrm{Pol}(\mathfrak{A}_0)$ preserves all subsets of A_0, and is therefore conservative. Hence, $\mathrm{Pol}(\mathfrak{A}_0)$ is conservative.

This remark justifies our interest in the computational complexity of certain CSPs where the template has conservative polymorphisms. Their complexity can be studied via universal algebraic methods as we will see in the following. An operation $f\colon B^3 \to B$ is called

- a *majority operation* if $\forall x, y \in B. f(x, x, y) = f(x, y, x) = f(y, x, x) = x$;
- a *minority operation* if $\forall x, y \in B. f(x, x, y) = f(x, y, x) = f(y, x, x) = y$.

An operation $f\colon B^n \to B$, for $n \geq 2$, is called

- a *cyclic operation* if $\forall x_1, \ldots, x_n \in B. f(x_1, \ldots, x_n) = f(x_n, x_1, \ldots, x_{n-1})$;
- a *weak near-unanimity operation* if

$$\forall x, y \in B. f(x, \ldots, x, y) = f(x, \ldots, x, y, x) = \ldots = f(y, x, \ldots, x);$$

- a *Siggers operation* if $n = 6$ and $\forall x, y \in B. f(x, x, y, y, z, z) = f(y, z, x, z, x, y)$.

The following terminology was introduced by Bulatov and has proven to be extremely powerful, especially in the context of conservative clones.

▶ **Definition 18** ([16, 17]). *A pair $(a, b) \in B^2$ is called a* semilattice edge *if there exists $f \in \mathrm{Pol}(\mathfrak{B})$ of arity two such that $f(a, b) = b = f(b, a) = f(b, b)$ and $f(a, a) = a$. A two-element set $\{a, b\} \subseteq B$* has a semilattice edge *if (a, b) or (b, a) is a semilattice edge.*

A two-element subset $\{a, b\}$ of B is called a majority edge *if neither (a, b) nor (b, a) is a semilattice edge and there exists an $f \in \mathrm{Pol}(\mathfrak{B})$ of arity three whose restriction to $\{a, b\}$ is a majority operation.*

A two-element subset $\{a, b\}$ of B is called an affine edge *if it is not a majority edge, if neither (a, b) nor (b, a) is a semilattice edge, and there exists an $f \in \mathrm{Pol}(\mathfrak{B})$ of arity three whose restriction to $\{a, b\}$ is a minority operation.*

If $S \subseteq B$ and $(a, b) \in S^2$ is a semilattice edge then we say that (a, b) *is a semilattice edge on S*. Similarly, if $\{a, b\} \subseteq S$ is a majority edge (affine edge) then we say that $\{a, b\}$ is a *majority edge on S* (*affine edge on S*).

The main result about conservative finite structures and their CSPs is the following dichotomy, first proved by Bulatov, 14 years before the proof of the Feder-Vardi conjecture.

▶ **Theorem 19** ([16]; see also [3, 17, 18]). *Let $\mathfrak{B}$ be a finite structure with a finite relational signature such that $\mathrm{Pol}(\mathfrak{B})$ is conservative. Then precisely one of the following holds:*

1. *$\mathrm{Pol}(\mathfrak{B})$ contains a Siggers operation; in this case, $\mathrm{CSP}(\mathfrak{B})$ is in P.*
2. *There exist distinct $a, b \in B$ such that for every $f \in \mathrm{Pol}(\mathfrak{B})^{(n)}$ the restriction of f to $\{a, b\}^n$ is a projection. In this case, $\mathrm{CSP}(\mathfrak{B})$ is NP-complete.*

Note that this means that $\mathrm{Pol}(\mathfrak{B})$ contains a Siggers operation if and only if for all two elements $a, b \in B$ the set $\{a, b\}$ is a majority edge, an affine edge, or there is a semilattice edge on $\{a, b\}$.

2.5 The k-Consistency Procedure

We present in the following the k-consistency procedure. It was introduced in [2] for finite structures and extended to infinite structures in several equivalent ways, for example in terms of Datalog programs, existential pebble games, and finite variable logics [8]. Also see [39] for recent results about the power of k-consistency for infinite-domain CSPs.

Let τ be a finite relational signature and let $k, l \in \mathbb{N}$ with $k < l$ and let $\mathfrak{B}$ be a fixed τ-structures with finitely many orbits of l-tuples. We define $\mathfrak{B}'$ to be the expansion of $\mathfrak{B}$ by all orbits of n-tuples for every $n \leq l$. We denote the extended signature of $\mathfrak{B}'$ by τ'. Let $\mathfrak{A}$ be an arbitrary finite τ-structure. A *partial l-decoration of* $\mathfrak{A}$ is a set g of atomic τ'-formulas such that

1. the variables of the formulas from g are a subset of A and denoted by $\mathrm{Var}(g)$,
2. $|\mathrm{Var}(g)| \leq l$,
3. the τ-formulas in g hold in $\mathfrak{A}$, where variables are interpreted as domain elements of $\mathfrak{A}$,
4. the conjunction over all formulas in g is satisfiable in $\mathfrak{B}'$.

A partial l-decoration g of $\mathfrak{A}$ is called *maximal* if there exists no partial l-decoration h of $\mathfrak{A}$ with $\mathrm{Var}(g) = \mathrm{Var}(h)$ such that $g \subsetneq h$. We denote the set of maximal partial l-decorations of $\mathfrak{A}$ by $\mathcal{R}^l_{\mathfrak{A}}$. Note that a fixed finite set of at most l variables, there are only finitely many partial l-decorations of $\mathfrak{A}$, because $\mathfrak{B}$ has by assumption finitely many orbits of l-tuples. Since this set is constant and can be precomputed, the set $\mathcal{R}^l_{\mathfrak{A}}$ can be computed efficiently. Then the (k,l)-*consistency procedure for* $\mathfrak{B}$ is the following algorithm.

Algorithm 1 (k,l)-consistency procedure for $\mathfrak{B}$.

Input : A finite τ-structure $\mathfrak{A}$.
1 **compute** $\mathcal{H} := \mathcal{R}^l_{\mathfrak{A}}$.
2 **repeat**
3 For every $f \in \mathcal{H}$ with $\mathrm{Var}(f) \leq k$ and every $U \subseteq A$ with $|U| \leq l - k$, if there does not exist $g \in \mathcal{H}$ with $f \subseteq g$ and $U \subseteq \mathrm{Dom}(g)$, then remove f from $\mathcal{H}$.
4 **until** $\mathcal{H}$ *does not change*
5 **if** $\mathcal{H}$ *is empty* **then**
6 **return** Reject.
7 **else**
8 **return** Accept.

Since $\mathcal{R}^l_{\mathfrak{A}}$ is of polynomial size (in the size of A) and the (k,l)-consistency procedure removes in step 3. at least one element from $\mathcal{R}^l_{\mathfrak{A}}$ the algorithm has a polynomial run time. The $(k,k+1)$-consistency procedure is also called k-*consistency procedure*. The $(2,3)$-consistency procedure is called *path consistency procedure*.[1]

▶ **Definition 20.** *Let $\mathfrak{B}$ be a relation τ-structure as defined before. Then the* (k,l)-consistency procedure for $\mathfrak{B}$ solves $\mathrm{CSP}(\mathfrak{B})$ *if the satisfiable instances of* $\mathrm{CSP}(\mathfrak{B})$ *are precisely the accepted instances of the (k,l)-consistency procedure.*

▶ Remark 21. Let $\mathbf{A}$ be a relation algebra with a normal representation $\mathfrak{B}$. We will in the following say that the k-consistency procedure solves $\mathrm{NSP}(\mathbf{A})$ if it solves $\mathrm{CSP}(\mathfrak{B})$. This definition is justified by the correspondence of NSPs and CSPs from Theorem 12.

▶ **Theorem 22** ([33])**.** *Let $\mathfrak{B}$ be a finite τ-structure. Then the following are equivalent:*
1. *There exist $k \in \mathbb{N}$ such that the k-consistency procedure solves* $\mathrm{CSP}(\mathfrak{B})$.
2. *$\mathfrak{B}$ has a 3-ary weak near-unanimity polymorphism f and a 4-ary weak near-unanimity polymorphism g such that: $\forall x, y, z \in B.\ f(y,x,x) = g(y,x,x,x)$.*

Let $\mathfrak{A}_0$ be the atom structure of a relation algebra $\mathbf{A}$ with a normal representation $\mathfrak{B}$. We finish this section by connecting the solvability of $\mathrm{CSP}(\mathfrak{A}_0)$ by k-consistency (or its characterization in terms of polymorphims from the previous proposition) with the solvability of $\mathrm{CSP}(\mathfrak{B})$ by k-consistency. By Remark 21 this gives a criterion for the solvability of $\mathrm{NSP}(\mathbf{A})$ by the k-consistency procedure.

The following theorem is from [39] building on ideas from [14]. We present it here in a specific formulation that already incorporates a correspondence between polymorphisms of the atom structure and canonical operations. For more details see [11, 12].

[1] Some authors also call it the *strong path consistency algorithm*, because some forms of the definition of the path consistency procedure are only equivalent to our definition of the path consistency procedure if $\mathfrak{B}$ has a transitive automorphism group.

▶ **Theorem 23** ([39]). *Let* $\mathfrak{B}$ *be a normal representation of a finite relation algebra* $\mathbf{A}$ *and* $\mathfrak{A}_0$ *the atom structure* $\mathbf{A}$*. If* $\mathrm{Pol}(\mathfrak{A}_0)$ *contains a* 3*-ary weak near-unanimity polymorphism* f *and a* 4*-ary weak near-unanimity polymorphism* g *such that* $\forall x, y, z \in B.\ f(y, x, x) = g(y, x, x, x)$*, then* $\mathrm{NSP}(\mathbf{A})$ *is solved by the* $(4, 6)$*-consistency algorithm.*

3 The Undecidability of RBCP, CON, and PC

In order to view RBCP as a decision problem, we need the following definitions. Let FRA be the set of all relation algebras $\mathbf{A}$ with domain $\mathcal{P}(\{1, \ldots, n\})$.

▶ **Definition 24** (RBCP). *We define the following subsets of* FRA*:*

- RBCP *denotes the set such that* $\mathrm{NSP}(\mathbf{A})$ *is in P.*
- RBCP^c *denotes* $\mathrm{FRA} \setminus \mathrm{RBCP}$*.*
- CON *denotes the set such that* $\mathrm{NSP}(\mathbf{A})$ *is solved by* k*-consistency for some* $k \in \mathbb{N}$*.*
- PC *denotes the set such that* $\mathrm{NSP}(\mathbf{A})$ *is solved by path consistency.*

The following theorem is our first result. Note that this can be seen as a negative answer to Hirsch's Really Big Complexity Problem [27].

▶ **Theorem 25.** RBCP *is undecidable,* CON *is undecidable, and* PC *is undecidable.*

In our undecidability proofs we reduce from the following well-known undecidable problem for relation algebras [25].

▶ **Definition 26** (Rep). *Let* Rep *be the computational problem of deciding for a given* $\mathbf{A} \in \mathrm{FRA}$ *whether* $\mathbf{A}$ *has a representation.*

In our proof we also use the fact that there exists $\mathbf{U} \in \mathrm{FRA}$ such that $\mathrm{NSP}(\mathbf{U})$ is undecidable [29]. Note that $\mathbf{U} \in \mathrm{Rep}$ since the network satisfaction problem for non-representable relation algebras is trivial and therefore decidable.

Proof of Theorem 25. We reduce the problem Rep to RBCP^c. Consider the following reduction $f\colon \mathrm{FRA} \to \mathrm{FRA}$. For a given $\mathbf{A} \in \mathrm{FRA}$, we define $f(\mathbf{A}) := \mathbf{A} \times \mathbf{U}$.

▷ Claim 1. If $\mathbf{A} \in \mathrm{Rep}$ then $f(\mathbf{A}) \in \mathrm{RBCP}^c$. If $\mathbf{A}$ is representable, then $\mathbf{A} \times \mathbf{U}$ is representable by the first part of Lemma 9. Then there is a polynomial-time reduction from $\mathrm{NSP}(\mathbf{U})$ to $\mathrm{NSP}(\mathbf{A} \times \mathbf{U})$ by Lemma 10. This shows that $\mathrm{NSP}(\mathbf{A} \times \mathbf{U})$ is undecidable, and hence $f(\mathbf{A})$ is in RBCP^c.

▷ Claim 2. If $\mathbf{A} \in \mathrm{FRA} \setminus \mathrm{Rep}$ then $f(\mathbf{A}) \in \mathrm{RBCP}$. If $\mathbf{A}$ is not representable, then $\mathbf{A} \times \mathbf{U}$ is not representable by the second part of Lemma 9, and hence $\mathrm{NSP}(\mathbf{A} \times \mathbf{U})$ is trivial and in P, and therefore in RBCP.

Clearly, f is computable (even in polynomial time). Since Rep is undecidable [25], this shows that RBCP^c, and hence RBCP, is undecidable as well. The proof for CON and PC is analogous; all we need is the fact that $\mathrm{NSP}(\mathbf{U}) \notin \mathrm{CON}$ and $\mathrm{NSP}(\mathbf{U}) \notin \mathrm{PC}$. ◀

4 Tractability via k-Consistency

We provide in this section a criterion that ensures solvability of NSPs by the k-consistency procedure (Theorem 30). A relation algebra $\mathbf{A}$ is called *symmetric* if all its elements are symmetric, i.e., $\breve{a} = a$ for every $a \in A$. We will see in the following that the assumption on $\mathbf{A}$ to be symmetric will simplify the atom structure A_0 of $\mathbf{A}$, which has some advantages in the upcoming arguments.

▶ **Definition 27.** *Let* $\mathbf{A}$ *be a finite symmetric relation algebra with set of atoms* A_0*. We say that* $\mathbf{A}$ *admits a* Siggers behavior *if there exists an operation* $s\colon A_0^6 \to A_0$ *such that*
1. *s preserves the allowed triples of* $\mathbf{A}$*,*
2. $\forall x_1, \dots, x_6 \in A_0.\ s(x_1, \dots, x_6) \in \{x_1, \dots, x_6\}$*,*
3. *s satisfies the Siggers identity:* $\forall x, y, z \in A_0.\ s(x, x, y, y, z, z) = s(y, z, x, z, x, y)$*.*

▶ Remark 28. We mention that if $\mathbf{A}$ has a normal representation $\mathfrak{B}$, then $\mathbf{A}$ admits a Siggers behavior if and only if $\mathfrak{B}$ has a pseudo-Siggers polymorphism which is canonical with respect to $\mathrm{Aut}(\mathfrak{B})$; see [14].

We say that a finite symmetric relation algebra $\mathbf{A}$ *has all* 1*-cycles* if for every $a \in A_0$ the triple (a, a, a) is allowed. Details on the notion of cycles from the relation algebra perspective can be found in [36]. The relevance of the existence of 1-cycles for constraint satisfaction comes from the following observation.

▶ **Lemma 29.** *Let* $\mathbf{A}$ *be a finite symmetric relation algebra with a normal representation* $\mathfrak{B}$ *that has a binary injective polymorphism. Then* $\mathbf{A}$ *has all* 1*-cycles.*

Proof. Let i be a binary injective polymorphism of $\mathfrak{B}$ and let $a \in A_0$ be arbitrary. Consider $x_1, x_2, y_1, y_2 \in B$ such that $a^{\mathfrak{B}}(x_1, x_2)$ and $a^{\mathfrak{B}}(y_1, y_2)$. The application of i on the tuples (x_1, x_1, x_2) and (y_1, y_2, y_2) results in a substructure of $\mathfrak{B}$ that witnesses that (a, a, a) is an allowed triple. ◀

▶ **Theorem 30.** *Let* $\mathbf{A}$ *be a finite symmetric relation algebra with a normal representation* $\mathfrak{B}$*. Suppose that the following holds:*
1. $\mathbf{A}$ *has all* 1*-cycles.*
2. $\mathbf{A}$ *admits a Siggers behavior.*

Then the $\mathrm{NSP}(\mathbf{A})$ *can be solved by the* $(4, 6)$*-consistency procedure.*

We will outline the proof of Theorem 30 and cite some results from the literature that we will use. Assume that $\mathbf{A}$ is a finite symmetric relation algebra that satisfies the assumptions of Theorem 30. Since $\mathbf{A}$ admits a Siggers behavior there exists an operation $s\colon A_0^6 \to A_0$ that is by 1. and 2. in Definition 27 a polymorphism of the atom structure $\mathfrak{A}_0$ (see Paragraph 2.4.2). By Remark 17, $\mathrm{Pol}(\mathfrak{A}_0)$ is a conservative operation clone. Recall the notion of semilattice, majority, and affine edges for conservative clones (cf. Definition 18). Since s is by 3. a Siggers operation, Theorem 19 implies that every edge in $\mathfrak{A}_0$ is semilattice, majority, or affine.

Our goal is to show that there are no affine edges in $\mathfrak{A}_0$, since this implies that there exists $k \in \mathbb{N}$ such that $\mathrm{CSP}(\mathfrak{A}_0)$ can be solved by k-consistency [17]. We present this fact here via the characterization of (k, l)-consistency in terms of weak near-unanimity polymorphisms from Theorem 22.

▶ **Proposition 31** (cf. Corollary 3.2 in [31])**.** *Let* $\mathfrak{A}_0$ *be a finite conservative relational structure with a Siggers polymorphism and no affine edge. Then* $\mathfrak{A}_0$ *has a* 3*-ary weak near-unanimity polymorphism* f *and a* 4*-ary weak near-unanimity polymorphism* g *such that*

$$\forall x, y, z \in B.\ f(y, x, x) = g(y, x, x, x).$$

Note that the existence of the weak near-unanimity polymorphisms from Proposition 31 would finish the proof of Theorem 30, because Theorem 23 implies that in this case $\mathrm{NSP}(\mathbf{A})$ can be solved by the $(4, 6)$-consistency procedure. We therefore want to prove that there are no affine edges in $\mathfrak{A}_0$. We start by analyzing the different types of edges in the atom structure $\mathfrak{A}_0$ and obtain results about their appearance (see Section 4.1).

Fortunately, there is the following result by Alexandr Kazda about binary structures with a conservative polymophism clone. A *binary structure* is a structure where all relations have arity at most two.

▶ **Theorem 32** (Theorem 4.5 in [31]). *If $\mathfrak{A}$ is a finite binary conservative relational structure with a Siggers polymorphism, then $\mathfrak{A}$ has no affine edges.*

Notice that we cannot simply apply this theorem to the atom structure $\mathfrak{A}_0$, since the maximal arity of its relations is three. We circumvent this obstacle by defining for $\mathfrak{A}_0$ a closely related binary structure $\mathfrak{A}_0^{\mathrm{b}}$, which we call the "binarisation of $\mathfrak{A}_0$":

▶ **Definition 33.** *We denote by $\mathfrak{A}_0^{\mathrm{b}}$ the structure with domain A_0 and the following relations:*
- *a unary relation U_S for each subset S of A_0;*
- *for every $a \in A_0$ the binary relation $R_a := \{(x, y) \in A_0^2 \mid (a, x, y) \in R\}$;*
- *a relation for every union of relations of the form R_a.*

In the next step we investigate how $\mathrm{Pol}(\mathfrak{A}_0)$ and $\mathrm{Pol}(\mathfrak{A}_0^{\mathrm{b}})$ relate to each other. It follows from these observations that $\mathfrak{A}_0^{\mathrm{b}}$ does not have an affine edge. In other words, it only has semilattice and majority edges. The crucial step in our proof is to transfer a witness of this fact to $\mathfrak{A}_0$ and conclude that also $\mathfrak{A}_0$ has no affine edge. The detailed proofs can be found in the extended version of the article [13] and in the PhD thesis of the second author [32].

4.1 Results about the Atom Structure

In this section we present some of our findings about the atom structure. We obtain conditions on the atom structure (namely the ternary relation R) that imply the (non-)existence of semilatice edges in the atom structure. As we explained in the previous section, these results are the starting point for our proof of Theorem 30.

For the sake of notation, we make some global assumptions for this section. Let $\mathbf{A}$ be a finite relation algebra that satisfies the assumptions from Theorem 30. We denote by $\mathfrak{A}_0$ the atom structure of $\mathbf{A}$ (Definition 15). Since $\mathbf{A}$ is a symmetric relation algebra, the relation $R^{\mathfrak{A}_0}$ is totally symmetric. Furthermore, we can drop the binary relation $E^{\mathfrak{A}_0}$, since it consists only of loops and does not change the set of polymorphisms. Let $s \in \mathrm{Pol}(\mathfrak{A}_0)$ be the Siggers operation that exists by the assumptions in Theorem 30. This implies by Theorem 19 for every $a, b \in A_0$ that the set $\{a, b\}$ is a majority edge or an affine edge, or that there is a semilattice edge on $\{a, b\}$. The different types of edges are witnessed by certain operations that we get from Proposition 3.1.in [17]: there exist a binary operation $f \in \mathrm{Pol}(\mathfrak{A}_0)$ and ternary operations $g, h \in \mathrm{Pol}(\mathfrak{A}_0)$ such that for every two element subset C of A_0,
- $f|_C$ is a semilattice operation if C has a semilattice edge, and $f|_C(x, y) = x$ otherwise;
- $g|_C$ is a majority operation if C is a majority edge, $g|_C(x, y, z) = x$ if C is affine and $g|_C(x, y, z) = f|_C(f|_C(x, y), z)$ if C has a semilattice edge;
- $h|_C$ is a minority operation if C is an affine edge, $h|_C(x, y, z) = x$ if C is majority and $h|_B(x, y, z) = f|_C(f|_C(x, y), z)$ if C has a semilattice edge.

We will fix these operations and introduce the following terminology. A tuple $(a, b) \in A_0$ is called *f-sl* if $f(a, b) = b = f(b, a)$ holds. Next, we prove several important properties of the relation R: that it must contain certain triples (Lemma 34), that it must not contain certain other triples (Lemma 35), and that it is affected by the presence of semilattice edges in $\mathbf{A}_0$ (Lemma 36 and Lemma 37).

▶ **Lemma 34.** *The relation R of the atom structure $\mathfrak{A}_0$ has the following properties:*
- *for all $a \in A_0$ we have $(a, a, a) \in R$.*
- *for all $a, b \in A_0$ we have $(a, a, b) \in R$ or $(a, b, b) \in R$;*

Proof. The first item follows from the assumption that $\mathbf{A}$ has all 1-cycles.

For the second item observe that $\{a, \mathrm{Id}\}$ cannot be a majority edge. Otherwise,

$$g((a,a,\mathrm{Id}),(\mathrm{Id},a,a),(\mathrm{Id},\mathrm{Id},\mathrm{Id})) = (\mathrm{Id},a,\mathrm{Id}) \in R$$

is a contradiction to the properties of Id. Furthermore, (a, Id) cannot be f-sl, since

$$f((a,a,\mathrm{Id}),(\mathrm{Id},a,a)) = (\mathrm{Id},a,\mathrm{Id}) \in R.$$

This is again a contradiction. Since these observations also hold for b instead of a we have the following case distinction.

1. (Id, a) is f-sl and (Id, b) is f-sl. It follows that $f((a,a,\mathrm{Id}),(\mathrm{Id},b,b)) \in \{(a,a,b),(a,b,b)\}$. Since f preserves R, $(a,a,\mathrm{Id}) \in R$, and $(\mathrm{Id},b,b) \in R$ we get that $f((a,a,\mathrm{Id}),(\mathrm{Id},b,b)) \in R$. This implies that $(a,a,b) \in R$ or $(a,b,b) \in R$.
2. (Id, a) is f-sl and $\{b, \mathrm{Id}\}$ is affine. By the definition of f we get $f((b,b,\mathrm{Id}),(\mathrm{Id},a,a)) \in \{(b,a,a),(b,b,a)\}$. By the same argument as in Case 1 we get that $(a,a,b) \in R$ or $(a,b,b) \in R$.
3. (Id, b) is f-sl and $\{a, \mathrm{Id}\}$ is affine. This case is analogous to Case 2.
4. $\{a, \mathrm{Id}\}$ is affine and $\{b, \mathrm{Id}\}$ is affine. Observe that

   $$g((a,\mathrm{Id},a),(\mathrm{Id},b,b),(\mathrm{Id},\mathrm{Id},\mathrm{Id})) \in \{(a,b,a),(a,b,b)\},$$

   since $g(a,b,\mathrm{Id}) \in \{a,b,\mathrm{Id}\}$ and the triple (a,b,Id) is forbidden. As in the cases before it follows that $(a,a,b) \in R$ or $(a,b,b) \in R$.

This concludes the proof of the second item. ◀

▶ **Lemma 35.** *Let $a,b,c \in A_0$ be such that $(a,b,c) \notin R$ and $|\{a,b,c\}| = 3$. Then there are $x,y \in \{a,b,c\}$ such that $(x,x,y) \notin R$.*

Proof. We first suppose that there is a semilattice edge on $\{a,b,c\}$. Without loss of generality we assume that (a,b) is f-sl. If $f(c,a) = c$ then $(a,a,c) \notin R$ or $(b,a,a) \notin R$ because otherwise

$$f((a,a,c),(b,a,a)) = (b,a,c) \in R$$

contradicting our assumption. If $f(c,a) = a$ then $(b,c,c) \notin R$ or $(a,a,c) \notin R$ because otherwise

$$f((b,c,c),(a,a,c)) = (b,a,c) \in R$$

which is again a contradiction. Hence, in all the cases we found $x,y \in \{a,b,c\}$ such that $(x,x,y) \notin R$ and are done. In the following we therefore assume that there is no semilattice edge on $\{a,b,c\}$.

Next we suppose that there is an affine edge on $\{a,b,c\}$. Without loss of generality we assume that $\{a,b\}$ is an affine edge. Since there are no semilattice edges on $\{a,b,c\}$ we distinguish the following two cases:

1. $\{a,c\}$ is an affine edge. In this case $(c,a,a) \notin R$ or $(a,b,a) \notin R$ because otherwise

   $$h((c,a,a),(a,a,a),(a,b,a)) = (c,b,a) \in R.$$

2. $\{a,c\}$ is a majority edge. In this case $(a,a,c) \notin R$ or $(a,b,a) \notin R$ or $(b,b,c) \notin R$, because otherwise

   $$h((a,a,c),(a,b,a),(b,b,c)) = (b,a,c) \in R.$$

In both cases we again found $x, y \in \{a, b, c\}$ such that $(x, x, y) \notin R$ and are done. We therefore suppose in the following that there are no affine edges on $\{a, b, c\}$. Hence, all edges on $\{a, b, c\}$ are majority edges. Then $(a, a, c) \notin R$ or $(a, b, a) \notin R$ or $(b, b, c) \notin R$ because otherwise

$$g((a, a, c), (a, b, a), (b, b, c)) = (a, b, c) \in R.$$

Thus, also in this case we found $x, y \in \{a, b, c\}$ such that $(x, x, y) \notin R$. ◀

The next lemma states that the edge type on $\{a, b\}$ is predetermined whenever a triple (a, a, b) is not in R.

▶ **Lemma 36.** *Let $a, b \in A_0$ be such that $(a, a, b) \notin R$. Then (a, b) is a semilattice edge in $\mathfrak{A}_0$ but (b, a) is not.*

Proof. By Lemma 34 we know that $(a, b, b) \in R$, $(a, a, a) \in R$, and $(b, b, b) \in R$. Assume for contradiction that $\{a, b\}$ is a majority edge. Then

$$g((a, a, a), (a, b, b), (b, b, a)) = (a, b, a)$$

which contradicts the fact that g preserves R. Assume next that $\{a, b\}$ is an affine edge. Then

$$h((a, b, b), (b, a, b), (b, b, b)) = (a, a, b)$$

which again contradicts that h preserves R. Finally, if (b, a) is a semilattice edge then

$$f((a, b, b), (b, a, b)) = (a, a, b)$$

which contradicts the assumption that f preserves R. If follows that (a, b) is the only semilattice edge on $\{a, b\}$ and therefore $f(a, b) = b = f(b, a)$ holds. ◀

▶ **Lemma 37.** *Let $a, a', b, c \in A_0$ be such that $(a, b, c) \notin R$, $(a, a, b) \notin R$, and $(a', b, c) \in R$. Then (a', a) is not a semilattice edge.*

Proof. Assume for contradiction (a', a) is a semilattice edge, i.e., there exists $p \in \mathrm{Pol}(\mathfrak{A}_0)$ with $p(a, a') = a = p(a', a)$. Note that by Lemma 34 it follows that $(a, a, a) \in R$ and $(a, b, b) \in R$.

▷ Claim 1. $p(b, a) = a$ implies $p(a, b) = b$. This follows immediately, since otherwise $p((a, b, b), (b, a, b)) = (a, a, b) \in R$ is a contradiction.

▷ Claim 2. $(a, a, c) \notin R$. We assume the opposite and consider the only two possible cases for $p(b, a)$.

1. $p(b, a) = b$: We get a contradiction by $p((a', b, c), (a, a, c)) = (a, b, c) \in R$.
2. $p(b, a) = a$: By Claim 1 we know that $p(a, b) = b$ follows. Then $p((a, a, c), (a', b, c)) = (a, b, c) \in R$ contrary to our assumptions.

▷ Claim 3. $p(c, a) = a$ implies $p(a, c) = c$. Lemma 34 together with Claim 2 implies that $(a, c, c) \in R$. Now Claim 3 follows immediately, since otherwise $p((a, c, c), (c, a, c)) = (a, a, c) \in R$, which contradicts Claim 2.

We finally make a case distinction for all possible values of p on (b, a) and (c, a).

1. $p(b, a) = b$ and $p(c, a) = c$: We get a contradiction by $p((a', b, c), (a, a, a)) = (a, b, c) \in R$.
2. $p(b, a) = b$ and $p(c, a) = a$: We get a contradiction by $p((a', b, c), (a, a, a)) = (a, b, a) \in R$.
3. $p(b, a) = a$ and $p(c, a) = c$: $p((a', b, c), (a, a, a)) = (a, a, c) \in R$ contradicts Claim 2.
4. $p(b, a) = a$ and $p(c, a) = a$: By Claim 1 we get $p(a, b) = b$ and by Claim 3 we get $p(a, c) = c$. This yields a contradiction by $p((a, a, a), (a', b, c)) = (a, b, c) \in R$. ◀

5 k-Consistency and Symmetric Flexible-Atom Algebras

We apply our result from Section 4 to the class of finite symmetric relation algebras with a flexible atom and obtain a k-consistency versus NP-complete complexity dichotomy.

A finite relation algebra $\mathbf{A}$ is called *integral* if the element Id is an atom of $\mathbf{A}$, i.e., $\mathrm{Id} \in A_0$. We define flexible atoms for integral relation algebras only. For a discussion about integrality and flexible atoms consider Section 3 in [12].

▶ **Definition 38.** *Let* $\mathbf{A} \in \mathrm{RA}$ *be finite and integral. An atom* $s \in A_0$ *is called* flexible *if for all* $a, b \in A \setminus \{\mathrm{Id}\}$ *it holds that* $s \leq a \circ b$.

Relation algebras with a flexible atom have been studied intensively in the context of the *flexible atoms conjecture* [1, 35]. It can be shown easily that finite relation algebras with a flexible atom have a normal representation [11, 12]. In [12] the authors obtained a P versus NP-complete complexity dichotomy for NSPs of finite symmetric relation algebras with a flexible atom (assuming $\mathrm{P} \neq \mathrm{NP}$). In the following we strengthen this result and prove that every problem in this class can be solved by k-consistency for some $k \in \mathbb{N}$ or is NP-complete (without any complexity-theoretic assumptions).

We combine Theorem 30 with the main result of [12] to obtain the following characterization for NSPs of finite symmetric relation algebras with a flexible atom that are solved by the $(4, 6)$-consistency procedure. Note that the difference of Theorem 39 and the related result in [12] is the algorithm that solves the problems in P.

▶ **Theorem 39.** *Let* $\mathbf{A}$ *be a finite symmetric integral relation algebra with a flexible atom. Then the following are equivalent:*

- $\mathbf{A}$ *admits a Siggers behavior.*
- $\mathrm{NSP}(\mathbf{A})$ *can be solved by the* $(4, 6)$*-consistency procedure.*

Proof. Every finite symmetric relation algebra $\mathbf{A}$ with a flexible atom has a normal representation $\mathfrak{B}$ by Proposition 3.5 in [12].

If the first item holds it follows from Proposition 6.1. in [12] that $\mathfrak{B}$ has a binary injective polymorphism. By Lemma 29 the relation algebra $\mathbf{A}$ has all 1-cycles. We apply Theorem 30 and get that the second item in Theorem 39 holds.

We prove the converse implication by showing the contraposition. Assume that the first item is not satisfied. Then Theorem 9.1 in [12] implies that there exists a polynomial-time reduction from $\mathrm{CSP}(K_3)$ to $\mathrm{NSP}(\mathbf{A})$ which preserves solvability by the (k, l)-consistency procedure. The problem $\mathrm{CSP}(K_3)$ is the 3-colorability problem which is known (e.g., by [4]) to be not solvable by the (k, l)-consistency procedure for every $k, l \in \mathbb{N}$. Hence $\mathrm{NSP}(\mathbf{A})$ cannot be solved by the $(4, 6)$-consistency procedure. ◀

As a consequence of Theorem 39 we obtain the following strengthening of the complexity dichotomy NSPs of finite symmetric integral relation algebra with a flexible atom [12].

▶ **Corollary 40** (Complexity Dichotomy)**.** *Let* $\mathbf{A}$ *be a finite symmetric integral relation algebra with a flexible atom. Then* $\mathrm{NSP}(\mathbf{A})$ *can be solved by the* $(4, 6)$*-consistency procedure, or it is NP-complete.*

Proof. Suppose that the first condition in Theorem 39 holds. Then Theorem 39 implies that $\mathrm{NSP}(\mathbf{A})$ can be solved by the $(4, 6)$-consistency procedure. If the first condition in Theorem 39 is not satisfied it follows from Theorem 9.1. in [12] that $\mathrm{NSP}(\mathbf{A})$ is NP-complete. ◀

6 The Complexity of the Meta Problem

In this section we study the computational complexity of deciding for a given finite symmetric relation algebra $\mathbf{A}$ with a flexible atom whether the k-consistency algorithm solves $\mathrm{NSP}(\mathbf{A})$. We show that this problem is decidable in polynomial time even if $\mathbf{A}$ is given by the restriction of its composition table to the atoms of $\mathbf{A}$: note that this determines a symmetric relation algebra uniquely, and that this is an (exponentially) more succinct representation of $\mathbf{A}$ compared to explicitly storing the full composition table.

▶ **Definition 41** (Meta Problem). *We define* Meta *to be the following computational problem.*
Input: *the composition table of a finite symmetric relation algebra* $\mathbf{A}$ *restricted to* A_0.
Question: *is there a* $k \in \mathbb{N}$ *such that* k*-consistency solves* $\mathrm{NSP}(\mathbf{A})$*?*

Our proof of Theorem 25 shows that Meta is undecidable as well.

▶ **Theorem 42.** Meta *can be decided in polynomial time if the input is restricted to finite symmetric integral relation algebras* $\mathbf{A}$ *with a flexible atom.*

Proof. By Theorem 39 it suffices to test the existence of an operation $f\colon A_0^6 \to A_0$ which satisfies conditions 1.-3. in this theorem. The three conditions can clearly be checked in polynomial time, so we already know that Meta is in NP.

Note that the search for f may be phrased as an instance of $\mathrm{CSP}(\mathfrak{A}_0)$ with $|A|^6$ variables. Using the fact that the k-consistency procedure is one-sided correct even in the case that $\mathrm{CSP}(\mathfrak{A}_0)$ is NP-hard (i.e., if the procedure rejects a given instance of $\mathrm{CSP}(\mathfrak{A}_0)$, then the instance is always unsatisfiable), we may use a standard self-reducibility argument (see, e.g., [20]) to obtain a polynomial-time algorithm for finding f. ◀

7 Conclusion and Open Questions

The question whether the network satisfaction problem for a given finite relation algebra can be solved by the famous k-consistency procedure is undecidable. Our proof of this fact heavily relies on prior work of Hirsch [29] and of Hirsch and Hodkinson [25] and shows that almost any question about the network satisfaction problem for finite relation algebras is undecidable.

However, if we further restrict the class of finite relation algebras, one may obtain strong classification results. We have demonstrated this for the class of finite symmetric integral relation algebras with a flexible atom (Theorem 40); the complexity of deciding whether the conditions in our classification result hold drops from undecidable to P (Theorem 42). One of the remaining open problems is a characterisation of the power of k-consistency for the larger class of all finite relation algebras with a normal representation.

Our main result (Theorem 30) is a sufficient condition for the applicability of the k-consistency procedure; the condition does not require the existence of a flexible atom but applies more generally to finite symmetric relation algebras $\mathbf{A}$ with a normal representation. Our condition consists of two parts: the first is the existence of all 1-cycles in $\mathbf{A}$, the second is that $\mathbf{A}$ admits a Siggers behavior. We conjecture that dropping the first part of the condition leads to a necessary and sufficient condition for solvability by the k-consistency procedure.

▶ **Conjecture 43.** *A finite symmetric relation algebra* $\mathbf{A}$ *with a normal representation admits a Siggers behavior if and only if* $\mathrm{NSP}(\mathbf{A})$ *can be solved by the* k*-consistency procedure for some* $k \in \mathbb{N}$.

$\circ$	Id	E	N
Id	Id	E	N
E	E	Id	N
N	N	N	1

Figure 2 Multiplication table of the relation algebra $\mathbf{C}$.

Note that this conjecture generalises Theorem 39. Both directions of the conjecture are open. However, the forward direction of the conjecture is true if $\mathbf{A}$ has a normal representation with a primitive automorphism group: in this case, it is known that a Siggers behavior implies the existence of all 1-cycles [10], and hence the claim follows from our main result (Theorem 39). The following example shows a finite symmetric relation algebra $\mathbf{A}$ which does not have all 1-cycles and an imprimitive normal representation, but still NSP($\mathbf{A}$) can be solved by the k-consistency procedure for some $k \in \mathbb{N}$.

▶ **Example 44.** Theorem 30 is a sufficient condition for the NSP of a relation algebra $\mathbf{A}$ to be solved by the k-consistency procedure for some $k \in \mathbb{N}$. However, there exists a finite symmetric relation algebra $\mathbf{C}$ such that NSP($\mathbf{C}$) is solved by the 2-consistency procedure, but we cannot prove this by the methods used to obtain Theorem 30. Consider the relation algebra $\mathbf{C}$ with atoms $\{\mathrm{Id}, E, N\}$ and the multiplication table in Figure 2. This relation algebra has a normal representation, namely the expansion of the infinite disjoint union of the clique K_2 by all first-order definable binary relations. We denote this structure by $\overline{\omega K_2}$. One can observe that CSP($\overline{\omega K_2}$) and therefore also the NSP of the relation algebra can be solved by the $(2,3)$-consistency algorithm (for details see [32]).

The relation algebra $\mathbf{C}$ does not have all 1-cycles and therefore does not fall into the scope of Theorem 30. In fact, our proof of Theorem does not work for $\mathbf{C}$, because the CSP of the atom structure $\mathfrak{C}_0$ of $\mathbf{C}$ cannot be solved by the k-consistency procedure for some $k \in \mathbb{N}$. Hence, the reduction of NSP($\mathbf{C}$) to CSP($\mathfrak{C}_0$) (incorporated in Theorem 23) does not imply that NSP($\mathbf{C}$) can be solved by k-consistency procedure for some $k \in \mathbb{N}$.

The following problems are still open and are relevant for resolving Conjecture 43.

- Show Conjecture 43 if the normal representation of $\mathbf{A}$ has a primitive automorphism group.
- Characterise the power of the k-consistency procedure for the NSP of finite relation algebras with a normal representation whose automorphism group is imprimitive. In this case, there is a non-trivial definable equivalence relation. It is already known that if this equivalence relation has finitely many classes, then the NSP is NP-complete and the k-consistency procedure does not solve the NSP [10]. Similarly, the NSP is NP-complete if there are equivalence classes of finite size larger than two. It therefore remains to study the case of infinitely many two-element classes, and with infinitely many infinite classes. In both cases we wish to reduce the classification to the situation with a primitive automorphism group.

Finally, we ask whether it is true that if $\mathbf{A}$ is a finite symmetric relation algebra with a flexible atom and NSP($\mathbf{A}$) can be solved by the k-consistency procedure for some k, then it can also be solved by the $(2,3)$-consistency procedure? In other words, can we improve $(4,6)$ in Corollary 40 to $(2,3)$?

References

1 Jeremy F. Alm, Roger D. Maddux, and Jacob Manske. Chromatic graphs, Ramsey numbers and the flexible atom conjecture. *The Electronic Journal of Combinatorics*, 15(1), March 2008. doi:10.37236/773.

2 Albert Atserias, Andrei A. Bulatov, and Víctor Dalmau. On the power of k-consistency. In *ICALP*, pages 279–290, 2007.

3 Libor Barto. The dichotomy for conservative constraint satisfaction problems revisited. In *Proceedings of the Symposium on Logic in Computer Science (LICS)*, Toronto, Canada, 2011.

4 Libor Barto and Marcin Kozik. Constraint satisfaction problems of bounded width. In *Proceedings of Symposium on Foundations of Computer Science (FOCS)*, pages 595–603, 2009.

5 Libor Barto and Marcin Kozik. Constraint satisfaction problems solvable by local consistency methods. *Journal of the ACM*, 61(1):3:1–3:19, 2014.

6 Manuel Bodirsky. Complexity classification in infinite-domain constraint satisfaction. Mémoire d'habilitation à diriger des recherches, Université Diderot – Paris 7. Available at arXiv:1201.0856v8, 2012.

7 Manuel Bodirsky. Finite relation algebras with normal representations. In *Relational and Algebraic Methods in Computer Science – 17th International Conference, RAMiCS 2018, Groningen, The Netherlands, October 29 – November 1, 2018, Proceedings*, pages 3–17, 2018.

8 Manuel Bodirsky and Víctor Dalmau. Datalog and constraint satisfaction with infinite templates. *Journal on Computer and System Sciences*, 79:79–100, 2013. A preliminary version appeared in the proceedings of the Symposium on Theoretical Aspects of Computer Science (STACS'05).

9 Manuel Bodirsky and Peter Jonsson. A model-theoretic view on qualitative constraint reasoning. *Journal of Artificial Intelligence Research*, 58:339–385, 2017.

10 Manuel Bodirsky and Simon Knäuer. Hardness of network satisfaction for relation algebras with normal representations. In *Relational and Algebraic Methods in Computer Science*, pages 31–46. Springer International Publishing, 2020. doi:10.1007/978-3-030-43520-2_3.

11 Manuel Bodirsky and Simon Knäuer. Network satisfaction for symmetric relation algebras with a flexible atom. In *Proceedings of AAAI*, 2021. Preprint https://arxiv.org/abs/2008.11943.

12 Manuel Bodirsky and Simon Knäuer. The complexity of network satisfaction problems for symmetric relation algebras with a flexible atom. *Journal of Artificial Intelligence Research*, 75:1701–1744, December 2022. doi:10.1613/jair.1.14195.

13 Manuel Bodirsky and Simon Knäuer. Network satisfaction problems solved by k-consistency, 2023.

14 Manuel Bodirsky and Antoine Mottet. A dichotomy for first-order reducts of unary structures. *Logical Methods in Computer Science*, 14(2), 2018. doi:10.23638/LMCS-14(2:13)2018.

15 V. G. Bodnarčuk, L. A. Kalužnin, V. N. Kotov, and B. A. Romov. Galois theory for Post algebras, part I and II. *Cybernetics*, 5:243–539, 1969.

16 Andrei A. Bulatov. Tractable conservative constraint satisfaction problems. In *Proceedings of the Symposium on Logic in Computer Science (LICS)*, pages 321–330, Ottawa, Canada, 2003.

17 Andrei A. Bulatov. Complexity of conservative constraint satisfaction problems. *ACM Trans. Comput. Logic*, 12(4), July 2011. doi:10.1145/1970398.1970400.

18 Andrei A. Bulatov. Conservative constraint satisfaction re-revisited. *Journal Computer and System Sciences*, 82(2):347–356, 2016. ArXiv:1408.3690. doi:10.1016/j.jcss.2015.07.004.

19 Andrei A. Bulatov. A dichotomy theorem for nonuniform CSPs. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17*, pages 319–330, 2017.

20 Hubie Chen and Benoît Larose. Asking the metaquestions in constraint tractability. *TOCT*, 9(3):11:1–11:27, 2017.

21 Matteo Cristiani and Robin Hirsch. The complexity of the constraint satisfaction problem for small relation algebras. *Artificial Intelligence Journal*, 156:177–196, 2004.

22 Ivo Düntsch. Relation algebras and their application in temporal and spatial reasoning. *Artificial Intelligence Review*, 23:315–357, 2005.

23 Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory. *SIAM Journal on Computing*, 28:57–104, 1999.

24 David Geiger. Closed systems of functions and predicates. *Pacific Journal of Mathematics*, 27:95–100, 1968.

25 R. Hirsch and I. Hodkinson. Representability is not decidable for finite relation algebras. *Transactions of the American Mathematical Society*, 353(4):1387–1401), 2001.

26 R. Hirsch and I. Hodkinson. Strongly representable atom structures of relation algebras. *Transactions of the American Mathematical Society*, 130(6):1819–1831), 2001.

27 Robin Hirsch. Relation algebras of intervals. *Artificial Intelligence Journal*, 83:1–29, 1996.

28 Robin Hirsch. Expressive power and complexity in algebraic logic. *Journal of Logic and Computation*, 7(3):309–351, 1997.

29 Robin Hirsch. A finite relation algebra with undecidable network satisfaction problem. *Logic Journal of the IGPL*, 7(4):547–554, 1999.

30 Robin Hirsch and Ian Hodkinson. *Relation Algebras by Games*. North Holland, 2002.

31 Alexandr Kazda. CSP for binary conservative relational structures. *Algebra universalis*, 75(1):75–84, December 2015. doi:10.1007/s00012-015-0358-8.

32 Simon Knäuer. *Constraint Network Satisfaction for Finite Relation Algebras*. PhD thesis, Technische Universität Dresden, 2023.

33 Marcin Kozik, Andrei Krokhin, Matt Valeriote, and Ross Willard. Characterizations of several Maltsev conditions. *Algebra universalis*, 73(3):205–224, 2015. doi:10.1007/s00012-015-0327-2.

34 R. Lyndon. The representation of relational algebras. *Annals of Mathematics*, 51(3):707–729, 1950.

35 Roger D. Maddux. A perspective on the theory of relation algebras. *Algebra Universalis*, 31(3):456–465, September 1994. doi:10.1007/bf01221799.

36 Roger D. Maddux. *Relation Algebras: Volume 150*. Studies in logic and the foundations of mathematics. Elsevier Science, London, England, May 2006.

37 Ralph McKenzie. *The representation of relation algebras*. PhD thesis, University of Colorado at Boulder, 1966.

38 Ralph McKenzie. Representations of integral relation algebras. *Michigan Mathematical Journal*, 17(3):279–287, 1970. doi:10.1307/mmj/1029000477.

39 Antoine Mottet, Tomás Nagy, Michael Pinsker, and Michal Wrona. Smooth approximations and relational width collapses. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 138:1–138:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.138.

40 Jochen Renz and Bernhard Nebel. Qualitative spatial reasoning using constraint calculi. In M. Aiello, I. Pratt-Hartmann, and J. van Benthem, editors, *Handbook of Spatial Logics*, pages 161–215. Springer Verlag, Berlin, 2007.

41 Alfred Tarski. Representation problems for relation algebras. *Bulletin of the AMS*, 54(80), 1948.

42 Dmitriy Zhuk. A proof of the CSP dichotomy conjecture. *J. ACM*, 67(5):30:1–30:78, 2020. doi:10.1145/3402029.

43 Dmitriy N. Zhuk. A proof of CSP dichotomy conjecture. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17*, pages 331–342, 2017. https://arxiv.org/abs/1704.01914.

Algebraic Recognition of Regular Functions

Mikołaj Bojańczyk
Institute of Informatics, University of Warsaw, Poland

Lê Thành Dũng (Tito) Nguyễn
Laboratoire de l'informatique du parallélisme (LIP), École normale supérieure de Lyon, France

Abstract

We consider regular string-to-string functions, i.e. functions that are recognized by copyless streaming string transducers, or any of their equivalent models, such as deterministic two-way automata. We give yet another characterization, which is very succinct: finiteness-preserving functors from the category of semigroups to itself, together with a certain output function that is a natural transformation.

2012 ACM Subject Classification Theory of computation → Transducers

Keywords and phrases string transducers, semigroups, category theory

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.117

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version with Appendices*: `https://hal.science/hal-03985883`

Funding *Lê Thành Dũng (Tito) Nguyễn*: Supported by the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR).

Acknowledgements We thank the reviewers for helpful suggestions on the presentation of the paper.

1 Introduction

This paper is about the regular string-to-string functions (see e.g. [17]). This is a fundamental class of functions; it is one of the standard generalizations of regular languages to produce string outputs (instead of merely accepting or rejecting inputs), covering examples such as

string reversal: $123 \mapsto 321$ duplication: $123 \mapsto 123123$

It has many equivalent descriptions, including deterministic two-way automata [22, Note 4], copyless streaming string transducers (SST) [1, Section 3] (or the earlier and very similar single-use restricted macro tree transducers [14, Section 5]), MSO transductions [13, Theorem 13], combinators [4, Section 2], a functional programming language [8, Section 6], λ-calculus with linear types [15, Theorem 3] (see also [19, Claim 6.2] and [18, Theorem 1.2.3]), decompositions *à la* Krohn–Rhodes [10, Theorem 18, item 4], etc.

The number of equivalent characterizations clearly indicates that the class of regular functions is important and worth studying. However, from a mathematical point of view, a disappointing phenomenon is that each of the known descriptions uses syntax that is more complicated than one could wish for.[1] These complications are perhaps minor annoyances, and the corresponding models are undeniably useful. Nevertheless, it would be desirable to have a model with a short and abstract definition, similar to the definition of recognizability of regular languages by finite semigroups.

[1] For example, the definition of a two-way automaton requires a discussion of endmarkers and what happens when the automaton loops. In an MSO transduction, an unwiedly copying mechanism is necessary. In an SST, one needs to be careful about bounding the copies among registers. The calculi of [4, 8] both have a long list of primitives. Similar remarks apply to the other formalisms.

This paper proposes such an abstract model. We prove that the regular string-to-string functions are exactly those that can be obtained by composing two functions

$$\Sigma^* \xrightarrow{\text{some semigroup homomorphism}} \mathsf{F}(\Gamma^*) \xrightarrow{\text{out}_{\Gamma^*}} \Gamma^*$$

where F is a *functor* from the category of semigroups to itself that maps finite semigroups to finite semigroups, and the output function out_{Γ^*} – not necessarily a homomorphism – is part of a family $\text{out}_A \colon \mathsf{F}A \to A$ that is natural in the semigroup A.

We use the name transducer semigroup for the model implicit in this description, i.e. a semigroup-to-semigroup functor F together with a natural transformation for producing outputs. One of the surprising features of this model is the fact that linear growth of the output size, which is one of the salient properties of the regular string-to-string functions, does not seem to be a trivial consequence of the definition.

2 Transducer semigroups and warm-up theorems

In this section, we define the model that is introduced in this paper, namely transducer semigroups. The purpose of this model is to recognize *string-to-string* functions, which are defined to be functions of type $\Sigma^* \to \Gamma^*$, for some finite alphabets Σ and Γ. Some results will work in the slightly more general case where the domain or codomain is a more general semigroup, but we focus on the string-to-string case for the sake of concreteness.

The model is defined using terminology based on category theory. However, we do not assume that the reader has a background in category theory, beyond the two most basic notions of category and functor. Recall that a *category* consists of objects with morphisms between them, such that the morphisms can be composed and each object has an identity morphism to itself. In this paper, we will be working mainly with two categories:

Sets. The objects are sets, the morphisms are functions between them.

Semigroups. The objects are semigroups, the morphisms are semigroup homomorphisms.

To transform categories, we use functors. Recall that a *functor* between two categories consists of two maps: one that assigns to each object A in the source category an object in the target category, and another one that assigns to each morphism $f \colon A \to B$ a morphism $\mathsf{F}f \colon \mathsf{F}A \to \mathsf{F}B$. These maps need to be consistent with composition of morphisms, and the identity must go to the identity.

▶ **Example 2.1.** The *forgetful functor* from the category of semigroups to the category of sets maps a semigroup to its underlying set, and a semigroup homomorphism to the corresponding function on sets. It is an example of a semigroup-to-set functor.

▶ **Example 2.2.** These constructions can be seen as semigroup-to-semigroup functors:

Tuples. This functor maps a semigroup A to its square $A \times A$, with the semigroup operation defined coordinate-wise. The functor extends to morphisms in the expected way. This functor also makes sense for higher powers, including infinite powers, such as A^ω.

Opposite. This functor maps a semigroup A to the semigroup where the underlying set is the same, but multiplication is reversed, i.e. the product of a and b in the new semigroup is the product b and a in the old semigroup. Morphisms are not changed by the functor: they retain the homomorphism property despite the change in the multiplication operation.

Lists. This functor maps a semigroup A to the free monoid A^* that consists of lists of elements of A equipped with concatenation. (When A is finite, it can be regarded as an alphabet, in which case we shall also call these lists "strings" or "words".) On morphisms, the functor is defined element-wise (or letter-wise). A similar construction would make sense as a set-to-semigroup functor.

Non-empty lists. A variant of the previous example, which sends a semigroup A to the free semigroup A^+ that consists of non-empty lists of elements in A.

Powerset. This (covariant) powerset functor maps a semigroup A to the powerset semigroup $\mathsf{P}A$, whose underlying set is the family of all subsets of A, endowed with the operation

$$(A_1, A_2) \quad \mapsto \quad \{a_1 a_2 \mid a_1 \in A_1 \text{ and } a_2 \in A_2\}.$$

We now present the central definition of this paper.

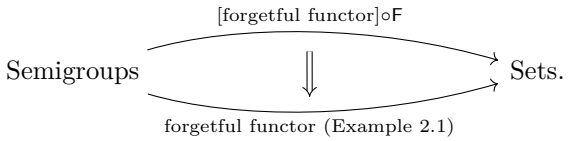
▶ **Definition 2.3** (Transducer semigroup). *A* transducer semigroup *consists of*

- *a semigroup-to-semigroup functor* F*,*
- *together with an* output mechanism *which associates to each semigroup* A *a function of type* $\mathsf{F}A \to A$ *called the* output function for A*,*

such that for every homomorphism $h\colon A \to B$*, the diagram below commutes:*

$$\begin{array}{ccc} \mathsf{F}A & \xrightarrow{\mathsf{F}h} & \mathsf{F}B \\ \downarrow{\scriptstyle \text{output function for } A} & & \downarrow{\scriptstyle \text{output function for } B} \\ A & \xrightarrow[h]{} & B \end{array}$$

In the language of category theory, a *natural transformation* between two semigroup-to-set functors G and K is a family of functions $f_A\colon \mathsf{G}A \to \mathsf{K}A$ such that $f_B \circ \mathsf{G}h = \mathsf{K}h \circ f_A$ for every semigroup homomorphism $h\colon A \to B$. So in Definition 2.3, the diagram says that the output mechanism is a natural transformation of type



Note that in a transducer semigroup, the output functions *are not necessarily homomorphisms*, which is why the forgetful semigroup-to-set functor appears above. This is important in view of the purpose of transducer semigroups, which is to define functions between semigroups, as explained in the following definition; asking out_B to be a homomorphism would severly restrict the functions that can be recognized.

▶ **Definition 2.4.** *We say that a function* $f\colon A \to B$ *between semigroups, not necessarily a homomorphism, is* recognized *by a transducer semigroup* (F, out) *if it can be decomposed as*

$$A \xrightarrow{h} \mathsf{F}B \xrightarrow{\text{out}_B} B \qquad \text{for some semigroup homomorphism } h.$$

The definition discusses functions between arbitrary semigroups, with no assumption on F, but we will mainly care about the special case – treated in Section 3 – where:

- the function is string-to-string[2] ($f\colon \Sigma^* \to \Gamma^*$), i.e. both the input and output semigroups are finitely generated free monoids;
- the functor F is *finiteness-preserving*, i.e. it maps finite semigroups to finite semigroups.

This special case will correspond to the regular string functions. Some minor results that do not assume F is finiteness-preserving are presented in Section 2.2: we characterize all functions (Theorem 2.10) and "recognizability reflecting" string-to-string functions (Theorem 2.13).

[2] Although this case involves monoids, which are a special case of semigroups, the use of a semigroup homomorphism that is not necessarily a monoid homomorphism is required to recognize functions such that $f(\varepsilon) \neq \varepsilon$. Furthermore, it will be useful in the proofs to work in the category of semigroups, rather than the category of monoids.

2.1 Examples and intuitions

▶ **Example 2.5.** Consider the transducer semigroup in which the functor is the identity, and the output mechanism is also the identity. The functions of type $A \to B$ that are recognized by this transducer semigroup are exactly the semigroup homomorphisms from A to B.

▶ **Example 2.6.** Consider the transducer semigroup in which

- the functor is the "opposite semigroup" functor from Example 2.2;
- the output function maps $a \in \mathsf{F}A$, seen as an element in A, to $aa \in A$.

The functions of type $A \to B$ that are recognized by this transducer semigroup are exactly those of the form $a \mapsto h(a)h(a)$ where $h\colon A \to B$ is some "anti-homomorphism", i.e. satisfies $h(a_1a_2) = h(a_2)h(a_1)$ for all $a_1, a_2 \in A$. In particular, if h is the string reversal function $\texttt{rev}$ on the free monoid Σ^*, which is also a semigroup, then we get the "reverse then duplicate" function that maps a string w over the alphabet Σ to $\texttt{rev}(w) \cdot \texttt{rev}(w)$.

▶ **Example 2.7.** We present here a transducer semigroup that recognizes the squaring function $w \in \Sigma^* \mapsto w^{|w|} \in \Sigma^*$ (illustrated by $\texttt{123} \mapsto \texttt{123123123}$) for any alphabet Σ.

The functor maps A to $A \times \mathbb{N}$, with the semigroup structure defined componentwise ($\mathbb{N} = \{0, 1, \dots\}$ is equipped with addition), and making the morphisms act on the left component. The output mechanism $A \times \mathbb{N} \to A$ is defined below:

$$\text{for } n \geq 1,\ (a, n) \mapsto a^n \qquad \underbrace{(a, 0) \mapsto a}_{\text{we handle this case separately because } a^0 \text{ does not make sense in an arbitrary semigroup}}$$

▶ **Example 2.8.** Our last example function is

$$w \in \{a, b, c\}^* \quad \mapsto \quad (\text{longest } c\text{-free suffix of } w) \cdot (\text{longest } c\text{-free prefix of } w) \in \{a, b\}^*$$

This can be recognized using the functor $\mathsf{F}A = A + A^2$, equipped with a suitable semigroup operation that makes the following map $h\colon \{a, b, c\}^* \to \mathsf{F}(\{a, b\}^*)$ a homomorphism:

$$h(w) = w \text{ for } w \in \{a, b\}^* \qquad h(uc \dots cv) = (u, v) \text{ for } u, v \in \{a, b\}^*$$

The output mechanism of the transducer semigroup sends any element $a \in A$ – seen as belonging to the left summand of $A + A^2$ – to aa, and (b, c) to cb.

▶ Remark 2.9. Consider a transducer semigroup with functor F. For a semigroup S, we may often think of an element of $\mathsf{F}S$ as a data structure that contains elements of S (such as a pair or a list, cf. Example 2.2). Then naturality of the output mechanism expresses that, being defined "uniformly in S", it cannot "look inside"[3] those elements of S (but it can combine them using the semigroup operation). In other words, the control flow may depend only on the part that is "independent of S" – and the condition that F is finiteness preserving (satisfied by all our examples except Example 2.7) somehow means that this part is finite.

2.2 Two simple characterizations

All functions. Our first theorem concerns transducer semigroups without any restrictions.

▶ **Theorem 2.10.** *Every function between semigroups is recognized by a transducer semigroup.*

[3] This is similar to the "generic" or "polymorphic" function definitions supported by many statically typed programming languages. The corresponding notion in type theory is *parametric polymorphism*, and it is closely related to naturality, see the introduction to [16].

Proof. We prove a slightly stronger result: for any semigroup A, there exists a transducer semigroup that recognizes all functions from A to other semigroups. The functor is

$$\mathsf{F}B = A \times (\text{set of all functions of type } A \to B, \text{ not necessarily recognizable}).$$

The semigroup operation in $\mathsf{F}B$ is defined as follows: on the first coordinate, we inherit the semigroup operation from A, while on the second coordinate, we use the trivial *left zero* semigroup structure, in which the product of two functions is simply the first one (this is a trivial way of equipping every set with a semigroup structure). The functor is defined on morphisms as in the tuple construction from Example 2.2: the first coordinate, corresponding to A, is not changed, and the second coordinate, corresponding to the set of functions, is transformed coordinate-wise, when viewed as a tuple indexed by A. This is easily seen to be a functor. The output mechanism, which is easily seen to be natural, is function application i.e. $(a, f) \mapsto f(a)$. Every function $f\colon A \to B$ is recognized by this transducer semigroup, with the appropriate homomorphism is $a \in A \mapsto (a, f)$. ◀

Recognizability reflecting functions. We now characterize functions with the property that inverse images of recognizable languages are also recognizable. We use a slightly more general setup, where instead of languages we use functions into arbitrary sets (languages can be seen as the case of functions into $\{\text{yes}, \text{no}\}$).

▶ **Definition 2.11.** *We say that a function from a semigroup A to some set X is* recognizable *if it factors through some semigroup homomorphism from A to a* finite *semigroup.*

In the rest of the paper, we shall sometimes speak of recognizable functions with infinite codomain, but note that the range of a recognizable function is always finite.

A function $f\colon B \to A$ between semigroups is called *recognizability reflecting* if for every recognizable function $g\colon A \to X$, the composition $g \circ f$ is recognizable.

▶ **Example 2.12.** Consider the semigroup $(\mathbb{N}, +)$ of natural numbers with addition, which is isomorphic to the free monoid a^*. In this semigroup, the recognizable functions are ultimately periodic colourings of numbers. A corollary is that every recognizable function gives the same answer to all factorials $\{1!, 2!, \ldots\}$, with finitely many exceptions. Take any function $f\colon \mathbb{N} \to \mathbb{N}$ such that (a) every output number arises from at most finitely many input numbers; (b) every output number is a factorial. The composition of f with any recognizable function will give the same answer to all numbers with finitely many exceptions, thus being also recognizable.

In the above example, a function with conditions (a) and (b) can be chosen in uncountably many ways, even if we require that it has linear growth. Therefore, there are too many recognizability reflecting functions (even just from $\{a\}^*$ to itself) to allow a machine model, or some other effective syntax. The following result gives a non-effective syntax.

▶ **Theorem 2.13.** *The following conditions are equivalent for a string-to-string function:*

1. *it is recognizability reflecting.*
2. *it is recognized by a transducer semigroup such that for every finite semigroup A, the corresponding output function of type $\mathsf{F}A \to A$ is recognizable.*

▶ **Example 2.14.** For any finite semigroup A, the map $(a, n) \in A \times (\mathbb{N} \setminus \{0\}) \mapsto a^n \in A$ is recognizable. This is because, since $a^{|A|!}$ is idempotent for every $a \in A$, this map factors through a homomorphism into the semigroup $A \times ((\mathbb{N} \setminus \{0\})/\sim)$ where

$$n \sim m \quad \iff \quad n = m \quad \vee \quad \big(n, m \geq |A|! \ \wedge \ n \equiv m \bmod |A|!\big)$$

is a congruence of finite index. Extending this slightly to handle the case $(a, 0) \mapsto a$, one can show that the output mechanism out_A of Example 2.7 is recognizable whenever A is finite. Therefore, the squaring function is recognizability reflecting.

3 The regular functions

The two straightforward constructions in Theorems 2.10 and 2.13 amount to little more than symbol pushing. In this section, we present a more substantial characterization, which is the main result of this paper. This characterization concerns finiteness-preserving functors. This is a strengthening of the condition from Theorem 2.13: if the functor F in a transducer semigroup is finiteness-preserving, then for every finite semigroup A, the output function $\mathsf{F}A \to A$ will be recognizable, since all functions from a finite semigroup are trivially recognizable. However, the condition is strictly stronger, as witnessed by Example 2.7, which is recognizability reflecting (cf. Example 2.14) but not finiteness preserving. As we will see, the stronger condition will characterize exactly the regular string-to-string functions.

The following counterexample illustrates the non-trivial interaction between naturality of the output mechanism and the requirement that the functor is finiteness preserving.

▶ **Example 3.1.** Consider the powerset functor P from Example 2.2. It is finiteness preserving, since the powerset of a finite semigroup is also finite. One could imagine that using powersets, one could construct a transducer semigroup that recognizes functions that are not regular, e.g. because they have exponential growth (unlike regular functions, which have linear growth). It turns out that this is impossible, because there is no possible output mechanism, i.e. no natural transformation of type $\mathsf{P}A \to A$, as we explain below.

The issue is that the naturality condition disallows choosing elements from a subset. To see why, consider a semigroup A with two elements, with the trivial left zero semigroup structure. For this semigroup, the output mechanism of type $\mathsf{P}A \to A$ would need to choose some element $a \in A$ when given as input the full set $A \in \mathsf{P}A$. However, none of the two choices is right, because swapping the two elements of A is an automorphism of the semigroup A, which maps the full set to itself, but does not map any element to itself.

We now state the main theorem of this paper.

▶ **Theorem 3.2.** *The following conditions are equivalent for every string-to-string function:*

1. *it is a regular string-to-string function;*
2. *it is recognized by a transducer semigroup in which the functor is finiteness preserving;*
3. *it is recognized by a transducer semigroup in which the functor* F maps the singleton semigroup 1 to a finite semigroup*:* $|\mathsf{F}1| < \infty$.

(Note that (2) ⇒ (3) is immediate.) Here is the plan for the rest of this section:
Section 3.1 gives a formal definition of regular functions
Section 3.2 proves the easy implication in the theorem, namely (1) ⇒ (2)
Section 3.3 proves the hard implication in the theorem, namely (3) ⇒ (1)

Before continuing, we remark on one advantage of the characterization in item (2), namely a straightforward proof of closure under composition. In contrast, for some (but not all) existing models defining regular string-to-string functions, composition requires a non-trivial construction – examples include two-way transducers [11, Theorem 2] or copyless streaming string transducers [2, Theorem 1].

▶ **Proposition 3.3.** *Functions recognized by finiteness-preserving transducer semigroups are closed under composition.*

Proof. This is because finiteness-preserving functors are closed under composition, natural families of output functions are also closed under composition, and naturality means by definition that the output functions "commute" in a suitable sense with functors.

More precisely, consider the following diagram:

$$\begin{array}{ccccc} \Sigma^* & \xrightarrow{h} & \mathsf{F}(\Gamma^*) & \xrightarrow{\mathrm{out}_{\Gamma^*}} & \Gamma^* \\ & & \downarrow{\scriptstyle \mathsf{F}h'} & & \downarrow{\scriptstyle h'} \\ & & \mathsf{FF}'(\Pi^*) & \xrightarrow{\mathrm{out}_{\mathsf{F}'(\Pi^*)}} & \mathsf{F}'(\Pi^*) \xrightarrow{\mathrm{out}'_{(\Pi^*)}} \Pi^* \end{array}$$

where the square commutes because the output mechanism is natural. The upper path describes the composition of two functions recognized by the transducer semigroups $(\mathsf{F}, \mathrm{out})$ and $(\mathsf{F}', \mathrm{out}')$, while the lower path describes a function recognized by $(\mathsf{FF}', \mathrm{out}_{\mathsf{F}'(-)} \circ \mathrm{out}')$. ◀

3.1 Defininition of streaming string transducers

In this section, we formally describe the regular functions, using a model based on streaming string transducers (SST). This model, like our proof of Theorem 3.2, covers a slightly more general case, namely string-to-semigroup functions instead of only string-to-string functions. These are functions of type $\Sigma^* \to A$ where Σ is a finite alphabet and A is an arbitrary semigroup. The purpose of this generalization is to make notation more transparent, since the fact that the output semigroup consists of strings will not play any role in our proof.

The model uses *registers* to store elements of the output semigroup. We begin by describing notation for registers and their updates. Suppose that R is a finite set of register names, and A is a semigroup called the *output semigroup*. We consider two sets

$$\textit{register valuations}\colon (R \to A) \qquad \textit{register updates}\colon (R \to (A + R)^+)$$

Below we show two examples of register updates, presented as assignments, using two registers X, Y and the semigroup $A = a^*$. (The right-hand sides are the values in $(A + R)^+$.)

$$\underbrace{\begin{array}{lcl} X & := & aYaXaaa \\ Y & := & XaaXaa \end{array}}_{\text{copyful}} \qquad \underbrace{\begin{array}{lcl} X & := & aaYaaXaaa \\ Y & := & aaa \end{array}}_{\text{copyless}}$$

The crucial property is being *copyless* – a register update is called copyless if every register name appears in at most one right-hand side of the update, and in that right-hand side it appears at most once. The main operation on these sets is *application*: a register update u can be applied to a register valuation v, giving a new register valuation vu.

In our model of streaming string tranducers, the registers will be updated by a stream of register updates that is produced by a rational function, defined as follows. Intuitively speaking, a rational function corresponds to an automaton that produces one output letter for each input position, with the output letter depending on regular properties of the input position within the input string. More formally:

▶ **Definition 3.4.** *A* rational function *of type* $\Sigma^* \to X^*$ *– where* Σ *is a finite alphabet but* X *can be any set – is a length-preserving*[4] *function with the following property: for some family*[5]

$$f_a \colon \underbrace{\Sigma^* \times \Sigma^*}_{\text{equipped with componentwise multiplication}} \to \Gamma \quad \text{for } a \in \Sigma \text{ of recognizable functions,}$$

for every input $a_1 \dots a_n$ *and* $i \in \{1, \dots, n\}$*, the* i*-th output letter is* $f_{a_i}(a_1 \dots a_{i-1},\ a_{i+1} \dots a_n)$.

Note that the range of a rational function with codomain X^* may contain only finitely many "letters" from X, so it can always be seen as a string function over finite alphabets.

Having defined register updates and rational functions, we are ready to introduce the machine model used in this paper as the reference definition of *regular functions*.

▶ **Definition 3.5.** *The syntax of a* streaming string transducer *(SST) is given by:*

- *A finite* input alphabet Σ *and an* output semigroup A.
- *A finite set* R *of* register names. *All register valuations and updates below use* R *and* A.
- *A designated* initial register valuation, *and a* final output pattern *in* R^+ *(that does not need to be copyless, though adding this restriction would not affect the expressive power).*
- *An* update oracle, *which is a rational function of type* $\Sigma^* \to (\textit{copyless register updates})^*$.

The semantics of the SST is a function of type $\Sigma^* \to A$ defined as follows. When given an input string, the SST begins in the designated initial register valuation. Next, it applies all updates produced by the update oracle, in left-to-right order. Finally, the output of the SST is obtained by combining the last register values according to the final output pattern.

▶ **Example 3.6.** We define an SST that computes the function of Example 2.8. It has two registers X and Y, whose initial valuation is $X = Y = \varepsilon$, and the final output pattern is YX. The update associated to an input letter $\ell \in \{a, b, c\}$ at position i is:

- if the position i is part of the longest c-free prefix, then $X := X\ell$, otherwise $X := X$;
- if the position i is part of the longest c-free suffix, then $Y := Y\ell$, otherwise $Y := Y$.

This sequence of updates can be produced by a rational function generated by a family of functions $(f_\ell)_{\ell \in \{a,b,c\}}$ that are recognized by $\mathbb{B}^2$, where $\mathbb{B}$ is the monoid of booleans with conjunction (rephrase the conditions as "there is no c to the left (resp. right) of i").

In a rational function, the label of the i-th output position is allowed to depend on letters of the input string that are on both sides of the i-th input position; this corresponds to regular lookaround in a streaming string transducer. Therefore, the model described above is easily seen to be equivalent to copyless SSTs with regular lookaround, which are one of the equivalent models defining the regular string-to-string functions, see [3, Section IV.C].

3.2 From a regular function to a transducer semigroup

Having defined the transducer model, we prove the easy implication in Theorem 3.2. It is apparent from Definition 3.5 that every regular function can be decomposed as a rational function followed by a function computed by a streaming string transducer whose i-th register update depends only on the i-th input letter – let us call that a *local* SST. Thanks to closure under composition (Proposition 3.3), we only need to handle these two special cases: we show that finiteness-preserving transducer semigroups recognize

- all rational functions in Section 3.2.1;
- and all local streaming string transducers in Section 3.2.2.

[4] Often in the literature, rational functions are not required to be length-preserving, see e.g. [21, p. 525], but in this paper, we only need the length-preserving case.

[5] The family $(f_a)_{a \in \Sigma}$ is very close to what is called an *(Eilenberg) bimachine* in the literature.

3.2.1 Recognizing rational functions by transducer semigroups

Consider a rational function, generated by the family $(f_a)_{a\in\Sigma}$ of recognizable functions of type $\Sigma^* \times \Sigma^* \to \Gamma$. By definition of recognizability, each f_a decomposes into

$$\Sigma^* \times \Sigma^* \xrightarrow{h_a} B_a \xrightarrow{g_a} \Gamma \qquad \text{where } h_a \text{ is a semigroup homomorphism and } B_a \text{ is finite.}$$

One can check that every f_a then factors through a monoid morphism to the finite monoid

$$\prod_{a\in\Sigma} h_a(\Sigma^* \times \Sigma^*)$$

Thus, without loss of generality, we may assume for the rest of the proof that all of the above semigroups B_a are equal to a common finite monoid B and that each semigroup homomorphism h_a is in fact a monoid morphism.

For any semigroup A, we let[6] $\mathsf{F}A = B \times (B \to A) \times B$, endowed with the following semigroup operation:

$$(\ell_1, \varphi_1, r_1) \cdot (\ell_2, \varphi_2, r_2) = \Big(\ell_1\ell_2,\ \big(b \mapsto \varphi_1(br_2) \cdot \varphi_2(\ell_1 b)\big),\ r_1 r_2\Big).$$

The construction F is extended to morphisms by considering $B \to A$ as the set of B-indexed tuples (cf. Example 2.2) of elements of A. To get a transducer semigroup, we take the output mechanism to be $(\ell, \varphi, r) \mapsto \varphi(e)$ where $e \in B$ is the neutral element.

Our rational function is then recognized by the unique monoid homomorphism of type $\Sigma^* \to \mathsf{F}(\Gamma^*)$ (indeed, F preserves monoids) which maps $a \in \Sigma$ to $\big(h_a(a, \varepsilon), g_a, h_a(\varepsilon, a)\big)$.

3.2.2 From a local SST to a transducer semigroup

Suppose now that a string-to-semigroup function $f\colon \Sigma^* \to A$ is computed by some local streaming string transducer. In the proof below, when referring to register valuations and register updates, we refer to those that use the registers and output semigroup of the fixed transducer. We say that a register update is in *normal form* if, in every right-hand side, one cannot find two consecutive letters from the semigroup A. Any register update can be *normalized*, i.e. converted into one that is in normal form, by using the semigroup operation to merge consecutive elements of the output semigroup in the right-hand sides. Here is an example, which uses three registers X, Y, Z and the semigroup $A = (\{0, 1\}, \cdot)$:

$$\underbrace{\begin{array}{lcl} X & := & 01Y1111X111 \\ Y & := & 01011 \end{array}}_{\text{not in normal form}} \xrightarrow{\text{normalization}} \underbrace{\begin{array}{lcl} X & := & 0Y1X1 \\ Y & := & 0 \end{array}}_{\text{in normal form}}$$

The register updates before and after normalization act in the same way on register valuations. If an update is copyless and in normal form, then the combined length of all right-hand sides is at most three times the number of registers. Therefore, if a semigroup A is finite, then the set of copyless register updates in normal form, call it $\mathsf{U}A$, is also finite. (However, there are infinitely many copyful register updates even when A is finite.) This set $\mathsf{U}A$ can be equipped with a composition operation

$$u_1, u_2 \in \mathsf{U}A \quad \mapsto \quad u_1 u_2 \in \mathsf{U}A,$$

[6] A construction similar in spirit to the classical *two-sided semidirect product* [20, §6].

which is defined in the same way as applying a register update to a register valuation, except that we normalize at the end. This composition operation is associative, and compatible with applying register updates to register valuations, in the sense that $(vu_1)u_2 = v(u_1u_2)$ holds for every valuation v and all updates u_1 and u_2. Therefore, $A \mapsto \mathsf{U}A$ is a finiteness-preserving semigroup-to-semigroup functor (with the natural extension to morphisms, where the homomorphism is applied to every semigroup element in a right-hand side).

The functor U described above is almost but not quite the functor that will be used in the transducer semigroup that we will define to prove the easy implication in Theorem 3.2. That functor F will also take into account the initial register valuation:

$$\mathsf{F}A = \mathsf{U}A \times \underbrace{(R \to A)}_{\text{endowed with the trivial left zero semigroup structure}} \quad \text{with componentwise multiplication \& action on morphisms}$$

Given $(u, v) \in \mathsf{F}A$, the output mechanism in the transducer semigroup applies the register update u to the register valuation v, and then multiplies together the register values given by the resulting valuation vu according to the final output pattern. Using this, we can recognize f via the homomorphism that sends each input letter to:

- the register update that this letter determines (our SST being local) in the first component;
- the designated initial register valuation in the second component.

3.3 From a transducer semigroup to a regular function

We now turn to the difficult implication (3) $\Rightarrow$ (1) in Theorem 3.2. The proof is presented in a way which, if sometimes slightly verbose, makes it easier to see how it can be adapted to other algebraic structures instead of semigroups (such as forest algebras, cf. Section 4).

3.3.1 Polynomial functors and functorial streaming string transducers

The assumption of the implication uses an abstract model (transducer semigroups), while the conclusion uses a concrete operational model (streaming string transducers). To bridge the gap, we use an intermediate model, similar to SSTs, but a bit more abstract. The abstraction arises by using polynomial functors instead of registers, as described below.

▶ **Definition 3.7.** *By* polynomial functor*, we mean a semigroup-to-set functor of the form*

$$A \quad \mapsto \quad \coprod_{q \in Q} A^{\textit{dimension of } q},$$

where Q is some possibly infinite set, whose elements are called components*, with each component having an associated* dimension *in $\mathbb{N}$. The symbol $\coprod$ stands for disjoint union of sets. This functor does not take into account the semigroup structure of the input semigroup, since the output is seen only as a set. On morphisms, the functor works in the expected way, i.e. coordinate-wise.*

A *finite polynomial functor* is a polynomial functor with finitely many components – for example, $A \mapsto A^2 + A^2 + A$. The notion of finite polynomial functor can be seen as a mild generalization of the construction which maps a semigroup A to the set A^R of register valuations for some fixed finite set R of register names. In the generalization, we allow a variable number of registers, depending on some finite information (the component).

Having defined a more abstract notion of "register valuations", namely finite polynomial functors, we now define a more abstract notion of "register updates". The first condition for such updates is that they do not look inside the register contents; this condition is captured by naturality (as discussed in Remark 2.9).

▶ **Example 3.8.** Consider the polynomial functors (where 1 represents the singleton set A^0)

$$\mathsf{F}A = A^* = 1 + A^1 + A^2 + \cdots \qquad \mathsf{G}A = A + 1.$$

An example of a natural transformation between these two functors is the function which maps a nonempty list in A^* to the product of its elements, and which maps the empty list to the unique element of 1. A non-example is the function that maps a list $[a_1, \ldots, a_n] \in A^*$ to the leftmost element a_i that is an idempotent in the semigroup, and returns 1 if such an element does not exist. The reason why the non-example is not natural is that a semigroup homomorphism can map a non-idempotent to an idempotent.

Apart from naturality, we will want our register updates to be copyless. For the purposes of the following definition, let us call a tuple of numbers in $\mathbb{N}^k$ a "sub-unit" if it belongs to $\{0,1\}^k$ and at most one coordinate is equal to 1 – or, as an edge case, if $k = 0$. For a polynomial functor F, a sub-unit of $\mathsf{F}\mathbb{N} = \sum_q \mathbb{N}^{\dim(q)}$ is a sub-unit of any of the $\mathbb{N}^{\dim(q)}$.

▶ **Definition 3.9** (Copyless natural transformation). *A natural transformation between two polynomial functors* F *and* G *is called* copyless *if when instantiated to the semigroup*[7] $(\mathbb{N}, +)$*, the corresponding function of type* $\mathsf{F}\mathbb{N} \to \mathsf{G}\mathbb{N}$ *maps sub-units to sub-units.*

It will be convenient to speak of *natural functions* $f \colon \mathsf{F}A \to \mathsf{G}A$, where F and G are semigroup-to-set functors and A is a fixed semigroup, to refer to functions that can be extended to natural transformations $(f_B \colon \mathsf{F}B \to \mathsf{G}B)_{B \text{ semigroup}}$, with $f = f_A$. *Copyless natural functions* between instantiations of polynomial functors are defined analogously.

Having defined functions that are natural and copyless, we now describe the more abstract model of SSTs used in our proof. The main difference is that instead of register valuations and updates given by some finite set of register names, we have two abstract polynomial functors, one of them finite polynomial, together with an explicitly given application function. We also allow the computation to be initialized and finalized in a more liberal way, that may depend on a regular property of the input.

▶ **Definition 3.10.** *The syntax of a* functorial streaming string transducer *is given by:*

- *A finite* input alphabet Σ *and an* output semigroup A*.*
- *A finite polynomial functor* R*, called the* register functor*, and a (not necessarily finite) polynomial functor* U *called the* update functor*.*
- *A copyless natural function of type* $\mathsf{R}A \times \mathsf{U}A \to \mathsf{R}A$*, called* application*.*
- *An* initial function $\Sigma^* \to \mathsf{R}A$ *which is recognizable (and therefore has finite range).*
- *A polynomial* final data functor K*, a* final data function $\Sigma^* \to \mathsf{K}A$ *which is recognizable, and a* final output function *of type* $\mathsf{R}A \times \mathsf{K}A \to A$ *which is a natural function (but not necessarily copyless).*
- *An* update oracle*, which is a rational function of type* $\Sigma^* \to (\mathsf{U}A)^*$*.*

Analogously to Definition 3.5, the functorial SST computes the function $\Sigma^* \to A$ obtained by the following composition, where the first map bundles together the initial function, the update oracle and the final data function:

$$\Sigma^* \to \mathsf{R}A \times (\mathsf{U}A)^* \times \mathsf{K}A \xrightarrow{(\textit{apply} \text{ updates successively}) \times \mathrm{id}_{\mathsf{K}A}} \mathsf{R}A \times \mathsf{K}A \xrightarrow{\text{final output function}} A$$

In the appendix, we prove that this model is no more expressive than usual copyless SSTs.

[7] The choice of the semigroup $(\mathbb{N}, +)$ in the Definition 3.9 is not particularly important. For example, the same notion of copylessness would arise if instead of $(\mathbb{N}, +)$, we used the semigroup $\{0, 1, 2\}$ with addition up to threshold 2. In the appendix, we present a more syntactic characterization of copyless natural transformations as part of our proof of Lemma 3.11.

▶ **Lemma 3.11.** *Definitions 3.5 and 3.10 characterize the same string-to-semigroup functions.*

3.3.2 Coproducts and views

Apart from the more abstract transducer model from Definition 3.10, the other ingredient used in the proof of the hard implication in Theorem 3.2 will be coproducts of semigroups, and some basic operations on them, as described in this section.

The *coproduct*[8] of two semigroups A and B, denoted by $A \oplus B$, is the semigroup whose elements are nonempty words over an alphabet that is the disjoint union of A and B, restricted to words that are *alternating* in the sense that two consecutive letters cannot belong to the same semigroup. The semigroup operation is defined in the expected way. We draw elements of a coproduct using coloured boxes, with the following picture showing the product operation in the coproduct of two copies, red and blue, of the semigroup $\{a, b\}^+$:

$$(\boxed{aba} \cdot \boxed{b} \cdot \boxed{b} \cdot \boxed{aa}) \cdot (\boxed{abba} \cdot \boxed{aabb}) = \boxed{aba} \cdot \boxed{b} \cdot \boxed{b} \cdot \boxed{aaabba} \cdot \boxed{aabb}.$$

A coproduct can involve more than two semigroups; in the pictures this would correspond to more colours, subject to the condition that consecutive boxes have different colours.

▶ Remark 3.12. The copyless register updates $u : R \to (A + R)^+$ of ordinary SSTs that are in normal form (cf. Section 3.2) can be seen as maps $R \to A \oplus \bigoplus_{X \in R} \{X\}^+$.

We write 1 for the semigroup that has one element. This semigroup is unique up to isomorphism and it is a *terminal object* in the category of semigroups, which means that it admits a unique homomorphism from every other semigroup A. This unique homomorphism will be denoted by $! \colon A \to 1$. (It has no connection with the factorial function on numbers.)

Consider the semigroup-to-set functors defined by (the underlying set of) a coproduct of several copies of their argument with several copies of 1, such as $A \mapsto A \oplus A \oplus A \oplus 1 \oplus 1$. In our proof, it will be useful to see them as polynomial functors, even though strictly speaking they are not defined as sums of products. This identification is allowed by the following observation (stated for $A \oplus 1$ for convenience, but the same idea applies in general).

▶ **Proposition 3.13.** *There is a family of bijections, natural in the semigroup A, between*

$$A \oplus 1 \quad \text{and} \quad \coprod_{q \in 1 \oplus 1} A^{\text{dimension of } q},$$

where the dimension of q is the number of times that the first copy of 1 appears in q.

Idea. Given $x \in A \oplus 1$, we apply $! \colon A \to 1$ to the elements of A in x to determine the component q of the polynomial functor that contains the image of x by the left-to-right bijection. This operation, a special case of what is called the shape below, forgets those elements of A appearing in x, so we record them in a tuple living in $A^{\dim(q)}$. For example, $\boxed{aba} \cdot \boxed{1} \cdot \boxed{aa} \cdot \boxed{1}$ is sent to the tuple (aba, aa) in the component $\boxed{1} \cdot \boxed{1} \cdot \boxed{1} \cdot \boxed{1}$. ◀

The crucial property of semigroups that will be used in our proof is Lemma 3.14 below, which says that an element of a coproduct can be reconstructed based on certain partial information. This information is described using the following operations.

[8] The name *coproduct* is used because of the following universal property: if $f \colon A \to C$ and $g \colon B \to C$ are two semigroup homomorphisms, then there is a unique homomorphism $A \oplus B \to C$ that coincides with f (resp. g) on the subsemigroup consisting of words with a single letter from A (resp. B).

1. **Merging**. Consider a coproduct $A_1 \oplus \cdots \oplus A_n$, such that the same semigroup A appears on all coordinates from a subset $I \subseteq \{1, \ldots, n\}$, and possibly on other coordinates as well. Define *merging the parts from* I to be the function of type

$$A_1 \oplus \cdots \oplus A_n \to A \oplus \bigoplus_{i \notin I} A_i$$

that is defined in the expected way, and explained in the following picture. In the picture, merging is applied to a coproduct of three copies of the semigroup $\{a, b\}^+$, indicated using colours red, black and blue, and the merged coordinates are red and blue:

$$\boxed{aba} \cdot \boxed{b} \cdot \boxed{aa} \cdot \boxed{b} \cdot \boxed{aa} \cdot \boxed{abba} \cdot \boxed{b} \quad \mapsto \quad \underbrace{\boxed{abab} \cdot \boxed{aa} \cdot \boxed{baaabba}}_{\text{the merge of red and blue is drawn in violet}} \cdot \boxed{b}.$$

2. **Shape.** Define the *shape operation* to be the function of type

$$A_1 \oplus \cdots \oplus A_n \to 1 \oplus \cdots \oplus 1$$

obtained by applying ! on every coordinate. The shape says how many alternating blocks there are, and which semigroups they come from, as explained in the following picture:

$$\boxed{aba} \cdot \boxed{b} \cdot \boxed{aa} \cdot \boxed{b} \cdot \boxed{aa} \cdot \boxed{abba} \cdot \boxed{b} \quad \mapsto \quad \boxed{1} \cdot \boxed{1} \cdot \boxed{1} \cdot \boxed{1} \cdot \boxed{1} \cdot \boxed{1} \cdot \boxed{1}.$$

3. **Views.** The final operation is the i-th *view*

$$A_1 \oplus \cdots \oplus A_n \to 1 \oplus A_i.$$

This operation applies ! to all coordinates other than i, and then it merges all those coordinates. Here is a picture, in which we take the view of the blue coordinate:

$$\boxed{aba} \cdot \boxed{b} \cdot \boxed{aa} \cdot \boxed{b} \cdot \boxed{aa} \cdot \boxed{abba} \cdot \boxed{b} \quad \mapsto \quad \boxed{aba} \cdot \boxed{1} \cdot \boxed{aa} \cdot \boxed{1}.$$

The key observation is that an element of a coproduct is fully determined from its shape and views, as stated in the following lemma. It seems to contain the essential property of semigroups that makes the construction work. We expect our theorem to also be true for other algebraic structures for which the lemma is true; however, the lemma seems to fail in certain settings. Concrete examples will be discussed in the conclusion (Section 4).

▶ **Lemma 3.14.** *Let* $A_1, \ldots, A_n$ *be semigroups. The* deconstruction *function of type*

$$A_1 \oplus \cdots \oplus A_n \longrightarrow (1 \oplus A_1) \times \cdots \times (1 \oplus A_n) \times (1 \oplus \cdots \oplus 1),$$

which is obtained by combining the views for all $i \in \{1, \ldots, n\}$ *and the shape, is injective.*

We prove this by exhibiting an explicit partial left inverse: a *reconstruction* function of type

$$(1 \oplus A_1) \times \cdots \times (1 \oplus A_n) \times (1 \oplus \cdots \oplus 1) \longrightarrow (A_1 \oplus \cdots \oplus A_n) + 1$$

such that deconstruction followed by reconstruction maps every element of $A_1 \oplus \cdots \oplus A_n$ to itself. The idea is to start with the shape and replace the entries from 1 with the elements appearing in the views in the right order. Rather than a formal definition, we illustrate this on an example (in the 3 views, we omit the boxes around the 1s to avoid visual cluttering):

$$\begin{array}{ccccccccccccc} \boxed{aba} & & 1 & & & \boxed{aa} & & 1 & & & & & \\ 1 & \boxed{b} & 1 & \boxed{b} & 1 & \boxed{abba} & 1 & & & & & & \\ & 1 & & \boxed{aa} & & & 1 & & & \boxed{b} & & & \\ \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & & \boxed{1} & & & \boxed{1} & & & \end{array} \quad \mapsto \quad \boxed{aba} \cdot \boxed{b} \cdot \boxed{aa} \cdot \boxed{b} \cdot \boxed{aa} \cdot \boxed{abba} \cdot \boxed{b}$$

Besides proving Lemma 3.14, this reconstruction function also enjoys the following property, which can be seen from the definition and Proposition 3.13.

▶ **Proposition 3.15.** *When each A_i is either A or 1, reconstruction can be seen as a copyless natural function between polynomial functors in A.*

3.3.3 Factorized output

Now, consider some transducer semigroup, with the functor being F, and fix a string-to-semigroup function $f\colon \Sigma^* \to A$ that decomposes as some homomorphism $h\colon \Sigma^* \to \mathsf{F}A$ followed by the output function of type $\mathsf{F}A \to A$.

For semigroups $A_1, \ldots, A_n$, define the *vectorial output function* to be the function of type

$$\mathsf{F}A_1 \times \cdots \times \mathsf{F}A_n \longrightarrow A_1 \oplus \cdots \oplus A_n$$

that is obtained by the composition of three functions described below (where *co-projection* is the function $A_i \to A_1 \oplus \cdots \oplus A_n$ that outputs a singleton list containing its input):

$$\begin{array}{c} \mathsf{F}A_1 \times \cdots \times \mathsf{F}A_n \\ \downarrow{\scriptstyle \mathsf{F}(\text{co-projection}) \times \cdots \times \mathsf{F}(\text{co-projection})} \\ \mathsf{F}(A_1 \oplus \cdots \oplus A_n) \times \cdots \times \mathsf{F}(A_1 \oplus \cdots \oplus A_n) \\ \downarrow{\scriptstyle \text{semigroup operation}} \\ \mathsf{F}(A_1 \oplus \cdots \oplus A_n) \\ \downarrow{\scriptstyle \text{output mechanism for } A_1 \oplus \cdots \oplus A_n} \\ A_1 \oplus \cdots \oplus A_n. \end{array}$$

To illustrate the definitions in this section, we use a *running example* with the transducer semigroup for the "reverse then duplicate" function from Example 2.6. The functor F sends a semigroup A to the opposite semigroup (cf. Example 2.2), and the output mechanism is $a \mapsto aa$. Our example function on $\{a, b\}^*$ is obtained by composing the string reversal homomorphism $\{a, b\}^* \to \mathsf{F}(\{a, b\}^*)$ with the output function. Here is an example of the vectorial output function (for now, the homomorphism plays no role):

$$(1, abbb) \in \mathsf{F}1 \times \mathsf{F}(\{a, b\}^*) \qquad \mapsto \qquad \boxed{abbb}\,\boxed{1}\,\boxed{abbb}\,\boxed{1} \in 1 \oplus \{a, b\}^*.$$

The vectorial output function is natural in all of its arguments, which means that for all semigroup homomorphisms $h_1, \ldots, h_n$, the diagram below commutes:

$$\begin{array}{ccc} \mathsf{F}A_1 \times \cdots \times \mathsf{F}A_n & \xrightarrow{\text{vectorial output function}} & A_1 \oplus \cdots \oplus A_n \\ \downarrow{\scriptstyle \mathsf{F}h_1 \times \cdots \times \mathsf{F}h_n} & & \downarrow{\scriptstyle h_1 \oplus \cdots \oplus h_n} \\ \mathsf{F}B_1 \times \cdots \times \mathsf{F}B_n & \xrightarrow{\text{vectorial output function}} & B_1 \oplus \cdots \oplus B_n \end{array}$$

This is because each of the three steps in the definition of the vectorial output function is itself a natural transformation, and natural transformations compose. Naturality of the first two steps is easy to check, while for the last step we use the assumption that the (non-vectorial) output function is natural.

Let us return to our function $f = \mathrm{out}_A \circ h$ recognized by our transducer semigroup $(\mathsf{F}, \mathrm{out})$. For strings $w_1, \ldots, w_n \in \Sigma^*$, define the corresponding *factorized output* to be the result of first applying the semigroup homomorphism $h : \Sigma^* \to \mathsf{F}A$ to all the strings, and then applying the vectorial output function; we denote it by

$$\langle w_1 | \cdots | w_n \rangle \in \underbrace{A \oplus \cdots \oplus A}_{n \text{ times}},$$

Here is the factorized output illustrated on our running example (we use colours to distinguish which of the three parts of the input is used):

$$\langle abb|\varepsilon|baaba\rangle = \boxed{abaab}\ \boxed{\varepsilon}\ \boxed{bba}\ \boxed{abaab}\ \boxed{\varepsilon}\ \boxed{bba} \in \{a,b\}^* \oplus \{a,b\}^* \oplus \{a,b\}^*.$$

As we can see above, when the output semigroup is a free monoid, the factorized output morally tells us "which part of the output string comes from which part in the input string".

▶ Remark 3.16. This is similar to the idea of *origin semantics* [5] of regular functions (see also [17, Section 5]). Indeed, our definition of factorized output is inspired by a similar tool of the same name that appears in the study of origin semantics [5, Section 2].

We also use a similar notation but with some input strings underlined, e.g. the input could be $\langle \underline{abb}|\varepsilon|baaba\rangle$ with an underline for the first red part. In the underlined case, before applying the vectorial output function, we apply h to the non-underlined strings and $(\mathsf{F}! \circ h)\colon \Sigma^* \to \mathsf{F}1$ to the underlined strings. In our running example, we have

$$\langle \underline{abb}|\varepsilon|baaba\rangle = \boxed{abaab}\ \boxed{\varepsilon}\ \boxed{1}\ \boxed{abaab}\ \boxed{\varepsilon}\ \boxed{1} \in 1 \oplus \{a,b\}^* \oplus \{a,b\}^*.$$

3.3.4 Proof of (3) ⇒ (1) in Theorem 3.2

We have now collected all necessary ingredients to prove this hard direction of the equivalence. Therefore, our goal is now to show that the function $f\colon \Sigma^* \to A$ that we have previously fixed is computed by some functorial streaming string transducer as in Definition 3.10, *assuming that* $\mathsf{F}1$ *is finite*.

The idea is that we want the functorial SST to maintain the following *invariant*:

> after processing the first i letters in an input string $a_1 \cdots a_n$,
> the register valuation is equal to the factorized output $\langle a_1 \cdots a_i | \underline{a_{i+1} \cdots a_n}\rangle$.

This way, after processing all input letters, the last valuation $\langle a_1 \cdots a_n|\underline{\varepsilon}\rangle$ is very close to the output; indeed, if we see A as a 1-ary coproduct, then $f(a_1 \dots a_n) = \langle a_1 \dots a_n\rangle \in A$.

The naive choice for the register functor is then $\mathsf{R}' : A \mapsto A \oplus 1$, since $\langle w|\underline{v}\rangle \in A \oplus 1$ for all $w, v \in \Sigma^*$ by definition. However, while R' can be seen as a polynomial semigroup-to-set functor, whose set of components is $1 \oplus 1$ (cf. Proposition 3.13), it is not *finite* polynomial (the set $1 \oplus 1$ is infinite). That said, we have by naturality of vectorial output:[9]

▷ Claim 3.17. The component for $\langle w|\underline{v}\rangle \in \mathsf{R}'A$ is $\langle \underline{w}|\underline{v}\rangle \in 1 \oplus 1$.

This index is determined by definition by the values of $(\mathsf{F}! \circ h)\colon \Sigma^* \to \mathsf{F}1$ on w and v, where $h\colon \Sigma^* \to \mathsf{F}A$ is the homomorphism used to recognize f. *Since* $\mathsf{F}1$ *is finite,* the $\langle w|\underline{v}\rangle$ for w, v

[9] Let us give some details. From Proposition 3.13, we see that the component in $1 \oplus 1$ of an element in $A \oplus 1$ is obtained by applying $! \oplus 1$. So it suffices to show that

$$\langle \underline{w}|\underline{v}\rangle = (! \oplus 1)(\langle w|\underline{v}\rangle)$$

Expanding the definitions, our goal can be rewritten as

$$\text{vectorial output of } (\mathsf{F}! \circ h(w), \mathsf{F}! \circ h(v)) = (! \oplus 1)\big(\text{vectorial output of } (h(w), \mathsf{F}! \circ h(v))\big)$$

This is a direct consequence of the naturality of the vectorial output function, that can be expressed as follows: for every semigroup homomorphism $g\colon A \to B$,

$$(\text{vectorial output for } B) \circ (\mathsf{F}g \times \mathrm{id}_1) = (g \oplus 1) \circ \text{vectorial output for } A$$

(take $g = !$ and apply both sides of the equality to $(h(w), \mathsf{F}! \circ h(v))$ to get the desired Claim 3.17).

ranging over Σ^* live in finitely many components. We take our register functor $\mathsf{R}A \subset \mathsf{R}'A$ to be the finite polynomial functor consisting of these "useful" components, plus the unique component that does not use A (it will serve as a "*null value*").

To design the register updates, the key is the following lemma. It shall be proved later using the machinery of views on coproducts that we have introduced for this very purpose.

▶ **Lemma 3.18.** *There are two copyless natural functions*

$$\delta\colon (A \oplus 1) \times (1 \oplus A \oplus 1) \to (A \oplus 1) + 1 \qquad \kappa\colon (A \oplus 1) \times (1 \oplus A) \to A + 1$$

such that, for every pair of strings $w, v \in \Sigma^*$ and every letter $a \in \Sigma$,

$$\langle wa|\underline{v}\rangle = \delta(\langle w|\underline{av}\rangle, \langle \underline{w}|a|\underline{v}\rangle) \qquad f(w) = \langle w\rangle = \kappa(\langle w|\underline{\varepsilon}\rangle, \langle \underline{w}|\varepsilon\rangle)$$

Again, to make "copyless natural" meaningful in this context, we invoke Proposition 3.13 to see δ and κ as functions between polynomial functors in A.

This leads us to use the update functor $\mathsf{U} : A \mapsto 1 \oplus A \oplus 1$ and to define the application of updates to registers, of type $\mathsf{R}A \times \mathsf{U}A \to \mathsf{R}A$, to be δ followed by the map $(A \oplus 1) + 1 \to \mathsf{R}A$ which sends the components of $A \oplus 1$ that are in $\mathsf{R}A$ to themselves, and everything else to the "null value". As an direct consequence of the lemma, the desired invariant holds using

- the initial function $w \mapsto \langle \varepsilon|\underline{w}\rangle$,
- and the update oracle $a_1 \dots a_n \mapsto \langle \underline{\varepsilon}|a_1|\underline{a_2 \dots a_n}\rangle \dots \langle \underline{a_1 \dots a_{n-1}}|a_n|\underline{\varepsilon}\rangle$.

To fit Definition 3.10, we have to check that the initial function is recognizable and that the update oracle is a rational function; by definition, the latter amounts to saying that for any $a \in \Sigma$, the function $(w, v) \in (\Sigma^*)^2 \mapsto \langle \underline{w}|a|\underline{v}\rangle$ is recognizable. According to the definition of factorized output, the initial function factors through the semigroup homomorphism $\mathsf{F}! \circ h$, *whose codomain* $\mathsf{F}1$ *is finite*; therefore, the initial function is recognizable. The other recognizability condition holds for a similar reason.

To finish building our functorial streaming string transducer, we use the function κ from Lemma 3.18. Thanks to our invariant and to the equation concerning κ, it is immediate that the following choices lead to a functorial SST that indeed computes f. We take:

- the final data functor $\mathsf{K} : A \mapsto (1 \oplus A) \times A$,
- the final data function $w \in \Sigma^* \mapsto (\langle \underline{w}|\varepsilon\rangle$, some arbitrary fixed value in A) – once again, it is recognizable because $\mathsf{F}1$ is finite,
- and the final output function $\mathsf{R}A \times \mathsf{K}A \to A$ that proceeds as follows: first, it applies κ to get some value in $(A + 1) \times A$; if the left half of the pair is in A, it returns it; otherwise, it returns the right half.

This being done, let us discharge our only remaining subgoal.

Proof of Lemma 3.18. We cover here the part concerning δ; for κ, the arguments are similar and a bit simpler. We use the following claim, which is proved using mechanical diagram chasing (as detailed in the appendix). Recall that the merging, shape and view operations were introduced just before Lemma 3.14.

▷ Claim 3.19. $\langle wa|\underline{v}\rangle$ is obtained from $\langle w|a|\underline{v}\rangle$ by merging the first two parts in $A \oplus A \oplus 1$.

The above claim shows that the factorized output $\langle wa|\underline{v}\rangle$ is obtained from $\langle w|a|\underline{v}\rangle$ by a copyless natural function. In turn, $\langle w|a|\underline{v}\rangle$ is the image by the reconstruction function – which is copyless natural (Proposition 3.15) – of the following four items (the equalities below are proved similarly to Claims 3.17 and 3.19):

1. First view of $\langle w|a|\underline{v}\rangle$, which is equal to $\langle w|\underline{av}\rangle$ – this is the first argument which is passed, in the lemma statement, to the function δ that we want to define.

2. Second view of $\langle w|a|\underline{v}\rangle$, which is obtained by merging the first and third parts in $\langle \underline{w}|a|\underline{v}\rangle$.
3. Third view of $\langle w|a|\underline{v}\rangle$, which is equal to $\langle \underline{wa}|\underline{v}\rangle$.
4. Shape of $\langle w|a|\underline{v}\rangle$, which is equal to $\langle \underline{w}|\underline{a}|\underline{v}\rangle$.

To complete the proof, it remains to justify that the last three items above can be collectively obtained from the second argument given to δ, namely $\langle \underline{w}|a|\underline{v}\rangle$, by applying some copyless natural function. Each item is obtained separately by applying a natural function. Furthermore, the second item is obtained in a copyless way, while the last two items do not use A at all, and therefore they are obtained in a copyless way for trivial reasons, even when combined with the second item. ◀

4 Conclusions

In this paper, we have exhibited a concise algebraic characterization of the regular string-to-string functions, in the style of the definition of regular languages using recognizability by finite semigroups. To perform this extension from languages to functions, we have relied on the basic concepts of category theory: categories, functors, natural transformations.

It should be noted that our use of categories is quite different in spirit from many of the works that take a categorical perspective on automata-theoretic results – see for instance [12], whose introduction points to many further references. In such works, the correspondence between concrete automata models and their rephrasing as suitable (co)algebras or functors tends to be straightforward, with the technical focus lying elsewhere (typically, in generalizing constructions such as determinisation or minimisation). On the contrary, we define a truly new transducer model whose equivalence with the preexisting copyless streaming string transducers requires a non-trivial proof.

An advantage of our characterization of the regular string functions is that, as one would expect from an abstract result, it lends itself to generalizations.

Semigroup-to-semigroup functions. The notion of recognition by a finiteness-preserving transducer semigroup makes sense for functions between arbitrary semigroups. Furthermore, such functions are closed under composition (the proof of Proposition 3.3 works as it is). To check their robustness, it would be desirable to have a more concrete, machine-like model capturing the same function class; possibly a variant of streaming string transducers where the underlying finite automaton is morally "replaced" by a finite semigroup.

More string functions. Another direction is characterizing other classes of string-to-string functions, such as the rational functions or the polyregular functions [7]. In this paper, we have discovered that, somewhat mysteriously, combining two conditions – naturality and preserving finiteness – characterizes exactly the regular functions, which have linear growth. Perhaps there is some way of tweaking the definitions to describe, say, some class with polynomial growth. For instance, the squaring function (Example 2.7) seems to be recognized by a mixed-variance functor $A \mapsto (A \to A) \times A$ with a dinatural output mechanism.

Functions on other free algebras. The definition of a transducer semigroup can applied to other algebras, and not just semigroups. This may be done by taking some monad T and considering functions that can be decomposed, for some endofunctor F of the category of Eilenberg-Moore algebras for the monad T and some natural transformation out, as

$$\mathsf{T}\Sigma \xrightarrow{\text{some } \mathsf{T}\text{-algebra homomorphism}} \mathsf{FT}\Gamma \xrightarrow{\text{out}_{\mathsf{T}\Gamma}} \mathsf{T}\Gamma.$$

An example of this approach is forest algebras [6, Section 5], which are algebras for describing trees. Preliminary work shows that, in the case of forest algebras, the suitable version of Theorem 3.2 also holds, i.e. the finiteness-preserving functors lead to a characterization of the standard notion of regular tree-to-tree functions, namely MSO transductions (see [14, 9]). We believe that these results apply even further, namely for graphs of bounded treewidth, modeled using suitable monads [6, Section 6]. The crucial property is that Lemma 3.14, about reconstructing a coproduct from its views, holds for other monads than just the nonempty list monad for semigroups. Unfortunately, this lemma fails for some monads, such as the monad of formal linear combinations of strings that corresponds to weighted automata. In the future, we intend to conduct a more systematic investigation of the extent to which the characterizations from this paper can be generalized to other algebraic structures.

References

1 Rajeev Alur and Pavol Černý. Expressiveness of streaming string transducers. In Kamal Lodaya and Meena Mahajan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*, volume 8 of *LIPIcs*, pages 1–12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2010. doi:10.4230/LIPIcs.FSTTCS.2010.1.

2 Rajeev Alur, Taylor Dohmen, and Ashutosh Trivedi. Composing copyless streaming string transducers, 2022. arXiv:2209.05448.

3 Rajeev Alur, Emmanuel Filiot, and Ashutosh Trivedi. Regular transformations of infinite strings. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pages 65–74. IEEE Computer Society, 2012. doi:10.1109/LICS.2012.18.

4 Rajeev Alur, Adam Freilich, and Mukund Raghothaman. Regular combinators for string transformations. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) – CSL-LICS '14*, pages 1–10, Vienna, Austria, 2014. ACM Press. doi:10.1145/2603088.2603151.

5 Mikołaj Bojańczyk. Transducers with origin information. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 26–37. Springer, 2014. doi:10.1007/978-3-662-43951-7_3.

6 Mikołaj Bojańczyk. Languages recognised by finite semigroups, and their generalisations to objects such as trees and graphs, with an emphasis on definability in monadic second-order logic, 2020. arXiv:2008.11635.

7 Mikołaj Bojańczyk. Transducers of polynomial growth. In Christel Baier and Dana Fisman, editors, *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2–5, 2022*, pages 1:1–1:27. ACM, 2022. doi:10.1145/3531130.3533326.

8 Mikołaj Bojańczyk, Laure Daviaud, and Shankara Narayanan Krishna. Regular and First-Order List Functions. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science – LICS '18*, pages 125–134, Oxford, United Kingdom, 2018. ACM Press. doi:10.1145/3209108.3209163.

9 Mikołaj Bojańczyk and Amina Doumane. First-order tree-to-tree functions, 2020. Corrected version with erratum of a LICS 2020 paper. arXiv:2002.09307v2.

10 Mikołaj Bojańczyk and Rafał Stefański. Single-use automata and transducers for infinite alphabets. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 113:1–113:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.113.

11 Michal Chytil and Vojtech Jákl. Serial composition of 2-way finite-state transducers and simple programs on strings. In Arto Salomaa and Magnus Steinby, editors, *Automata, Languages and Programming, Fourth Colloquium, University of Turku, Finland, July 18-22, 1977, Proceedings*, volume 52 of *Lecture Notes in Computer Science*, pages 135–147. Springer, 1977. doi:10.1007/3-540-08342-1_11.

12 Thomas Colcombet and Daniela Petrişan. Automata Minimization: a Functorial Approach. *Logical Methods in Computer Science*, 16(1), March 2020. doi:10.23638/LMCS-16(1:32)2020.

13 Joost Engelfriet and Hendrik Jan Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Transactions on Computational Logic*, 2(2):216–254, April 2001. doi:10.1145/371316.371512.

14 Joost Engelfriet and Sebastian Maneth. Macro Tree Transducers, Attribute Grammars, and MSO Definable Tree Translations. *Information and Computation*, 154(1):34–91, October 1999. doi:10.1006/inco.1999.2807.

15 Paul Gallot, Aurélien Lemay, and Sylvain Salvati. Linear high-order deterministic tree transducers with regular look-ahead. In Javier Esparza and Daniel Král', editors, *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170 of *LIPIcs*, pages 38:1–38:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.MFCS.2020.38.

16 Claudio Hermida, Uday S. Reddy, and Edmund P. Robinson. Logical Relations and Parametricity – A Reynolds Programme for Category Theory and Programming Languages. In John Power and Cai Wingfield, editors, *Proceedings of the Workshop on Algebra, Coalgebra and Topology, WACT 2013, Bath, UK, March 1, 2013*, volume 303 of *Electronic Notes in Theoretical Computer Science*, pages 149–180. Elsevier, 2013. doi:10.1016/j.entcs.2014.02.008.

17 Anca Muscholl and Gabriele Puppis. The Many Facets of String Transducers. In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*, volume 126 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:21. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.STACS.2019.2.

18 Lê Thành Dũng Nguyễn. *Implicit automata in linear logic and categorical transducer theory.* PhD thesis, Université Paris XIII (Sorbonne Paris Nord), December 2021. URL: https://theses.hal.science/tel-04132636.

19 Lê Thành Dũng Nguyễn and Cécilia Pradic. Implicit automata in typed λ-calculi I: aperiodicity in a non-commutative logic. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 135:1–135:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.135.

20 John Rhodes and Bret Tilson. The kernel of monoid morphisms. *Journal of Pure and Applied Algebra*, 62(3):227–268, 1989. doi:10.1016/0022-4049(89)90137-0.

21 Jacques Sakarovitch. *Elements of Automata Theory.* Cambridge University Press, 2009. Translated by Reuben Thomas. doi:10.1017/CBO9781139195218.

22 John C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM Journal of Research and Development*, 3(2):198–200, April 1959. doi:10.1147/rd.32.0198.

How to Play Optimally for Regular Objectives?

Patricia Bouyer
Université Paris-Saclay, CNRS, ENS Paris-Saclay, Laboratoire Méthodes Formelles, 91190, Gif-sur-Yvette, France

Nathanaël Fijalkow
CNRS, LaBRI and Université de Bordeaux, France
University of Warsaw, Poland

Mickael Randour
F.R.S.-FNRS & UMONS – Université de Mons, Belgium

Pierre Vandenhove
F.R.S.-FNRS & UMONS – Université de Mons, Belgium
Université Paris-Saclay, CNRS, ENS Paris-Saclay, Laboratoire Méthodes Formelles, 91190, Gif-sur-Yvette, France

Abstract

This paper studies two-player zero-sum games played on graphs and makes contributions toward the following question: given an objective, how much memory is required to play optimally for that objective? We study regular objectives, where the goal of one of the two players is that eventually the sequence of colors along the play belongs to some regular language of finite words. We obtain different characterizations of the chromatic memory requirements for such objectives for both players, from which we derive complexity-theoretic statements: deciding whether there exist small memory structures sufficient to play optimally is NP-complete for both players. Some of our characterization results apply to a more general class of objectives: topologically closed and topologically open sets.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases two-player games on graphs, strategy complexity, regular languages, finite-memory strategies, NP-completeness

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.118

Category Track B: Automata, Logic, Semantics, and Theory of Programming

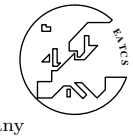
Related Version *Full Version*: `https://arxiv.org/abs/2210.09703` [4]

Funding This work has been partially supported by the ANR Project MAVeriQ (ANR-20-CE25-0012) and by the Fonds de la Recherche Scientifique – FNRS under Grant n° T.0188.23 (PDR ControlleRS). Mickael Randour is an F.R.S.-FNRS Research Associate and a member of the TRAIL Institute. Pierre Vandenhove is an F.R.S.-FNRS Research Fellow.

Acknowledgements We thank Antonio Casares and Igor Walukiewicz for valuable discussions about this article.

1 Introduction

Games on graphs is a fundamental model in theoretical computer science for modeling systems involving competing agents. Its applications include model-checking, program verification and synthesis, control theory, and reactive synthesis: in all cases, the system specification is turned into a winning objective for a player and the goal is to construct a winning strategy. Some central results in the field state that for some objectives, there exist memoryless optimal strategies, meaning not requiring any memory. For instance, the celebrated memoryless determinacy result for (infinite) parity games is a key ingredient in the modern proof of decidability of monadic second-order logic over infinite trees by Gurevich and Harrington [16].



Memory requirements. However for many objectives, some memory is required; a central question is therefore, stated informally:

Given an objective, how much memory is required to play optimally for this objective?

The first answers to this question, at the dawn of the study of games, were memory requirements for concrete objectives, such as Rabin objectives [22]. The work of Dziembowski, Jurdziński, and Walukiewicz [13] gave a computable characterization of memory requirements for the whole class of Muller objectives. This triggered the following long-term research goal: characterizing the memory requirements for ω-regular objectives.

Regular objectives. Many results have been obtained toward this research goal; we refer to the related works section in Section 3 for further details. The most pressing open question in that direction is regular objectives, meaning the special case of ω-regular objectives concerned with finite duration: in this setting, the objective is induced by a regular language over finite words and the goal of one of the players is that *eventually* the sequence of colors along the play belongs to this language. We call these *regular reachability objectives*. The opponent's objective is then to ensure that the sequence of colors *never* belongs to the language, describing *regular safety objectives*.

A first observation is that for such a regular (reachability or safety) objective, a deterministic finite automaton recognizing the regular language provides an upper bound on the memory requirements of both players. Indeed, playing with the extra information from the automaton reduces the game to a standard reachability or safety game, for which no further memory is required to make optimal decisions. Yet, as we will see, structures smaller than the minimal automaton recognizing the language may suffice for the players.

Chromatic memory. One of the many contributions of Kopczyński [18] in the study of memory for games on graphs is the notion of chromatic memory. In this model, the memory states are updated only using the sequence of colors seen along a play, and in particular do not depend on the graph itself (as opposed to chaotic memory, which may use information from the graph in its updates). Kopczyński conjectured [18] that for ω-regular objectives, chromatic and chaotic memory requirements coincide; unfortunately, this does not hold, as recently proved by Casares [8] (i.e., there are objectives for which the number of memory states required to play optimally in all arenas differs depending on the memory model). In our study, we will see another counterexample using regular objectives.

Contributions. We study the chromatic memory requirements of both regular reachability and regular safety objectives. For both cases, we give a combinatorial characterization of the memory structures sufficient to play optimally in all arenas (of any cardinality). As a by-product of the characterization we obtain complexity-theoretic statements: given as input a deterministic finite automaton representing the objective,

- deciding whether a memory structure suffices to play optimally in all arenas can be done in polynomial time;
- deciding the existence of a sufficient memory structure with a given number of states is NP-complete.

From our characterizations it also follows that for both regular reachability and safety objectives, chromatic and chaotic memory requirements do not coincide.

We also discuss when relevant the extension of our results to the more general class of topologically open and topologically closed objectives (called respectively *general reachability objectives* and *general safety objectives* for consistency in what follows), which include the regular reachability and regular safety objectives.

Implementation. In order to test ideas and conjectures, we have implemented algorithms that automatically build a memory structure with a minimal number of states, both for regular reachability and regular safety objectives. These algorithms are based on the theoretical analysis from this paper. Our implementation[1] uses SAT solvers provided by the Python package PySAT [17].

Structure of the paper. All required definitions are provided in Section 2. Section 3 includes an in-depth discussion of related works and a technical overview of the results and proofs: Section 3.1 for safety objectives, Section 3.2 for reachability objectives, and Section 3.3 for computational complexity results. Due to length constraints, only proof sketches are provided in this version of the article; complete proofs are available in the full version [4]. In particular, proofs for safety objectives are available in [4, Section 4], for reachability objectives in [4, Section 5], and for complexity-theoretic statements in [4, Section 6].

2 Preliminaries

Let C be a non-empty alphabet of *colors*.

Arenas. We study zero-sum turn-based games on graphs with two players, called $\mathcal{P}_1$ and $\mathcal{P}_2$. Players play on *arenas*, which are tuples $\mathcal{A} = (V, V_1, V_2, E)$ where V is a non-empty set of *vertices* such that $V = V_1 \uplus V_2$ (disjoint union) and $E \subseteq V \times C \times V$ is a *set of colored edges*. If $e = (v_1, c, v_2) \in E$, we write $\mathsf{in}(e) = v_1$, $\mathsf{col}(e) = c$, and $\mathsf{out}(e) = v_2$. Vertices in V_1 are controlled by $\mathcal{P}_1$ and vertices in V_2 are controlled by $\mathcal{P}_2$. An arena is *finite* if it has finitely many vertices and edges, and is *finitely branching* if for all $v \in V$, there are finitely many edges $e \in E$ such that $\mathsf{in}(e) = v$. Unless otherwise specified, we consider arenas of any cardinality. An arena $\mathcal{A} = (V, V_1, V_2, E)$ is a *one-player arena of* $\mathcal{P}_1$ (resp. *of* $\mathcal{P}_2$) if $V_2 = \emptyset$ (resp. $V_1 = \emptyset$).

A *history* on arena $\mathcal{A} = (V, V_1, V_2, E)$ is a finite sequence $\gamma = e_1 \dots e_n \in E^*$ such that for i, $1 \leq i \leq n-1$, we have $\mathsf{out}(e_i) = \mathsf{in}(e_{i+1})$. We write $\mathsf{out}(\gamma)$ for $\mathsf{out}(e_n)$. For convenience, we assume that for all $v \in V$, there is a distinct empty history λ_v such that $\mathsf{out}(\lambda_v) = v$. For $i \in \{1, 2\}$, we write $\mathsf{Hists}_i(\mathcal{A})$ for the set of histories γ on $\mathcal{A}$ such that $\mathsf{out}(\gamma) \in V_i$. A *play* on arena $\mathcal{A}$ is an infinite sequence $\pi = e_1 e_2 \dots \in E^\omega$ such that for $i \geq 1$, $\mathsf{out}(e_i) = \mathsf{in}(e_{i+1})$; play π is *from* v if $\mathsf{in}(e_1) = v$. If $\pi = e_1 e_2 \dots \in E^\omega$ is a play (resp. $\gamma = e_1 \dots e_n \in E^*$ is a history), we write $\mathsf{col}^\omega(\pi)$ (resp. $\mathsf{col}^*(\gamma)$) for the infinite sequence $\mathsf{col}(e_1)\mathsf{col}(e_2) \dots \in C^\omega$ (resp. the finite sequence $\mathsf{col}(e_1) \dots \mathsf{col}(e_n) \in C^*$).

Objectives. *Objectives* are subsets $W \subseteq C^\omega$. Given an objective W, we write $\overline{W} = C^\omega \setminus W$ for its complement. We focus on two types of objectives, both derived from a set $A \subseteq C^*$:

- the *general reachability objective derived from* A, denoted $\mathsf{Reach}(A)$, is the objective $\bigcup_{w \in A} wC^\omega$ of infinite words that have (at least) one finite prefix in A.
- the *general safety objective derived from* A, denoted $\mathsf{Safe}(A)$, is the objective $\overline{\bigcup_{w \in A} wC^\omega}$ of infinite words that have no finite prefix in A. We have $\mathsf{Safe}(A) = \overline{\mathsf{Reach}(A)}$.

General reachability and safety objectives are respectively the *topologically open* and *topologically closed sets*, at the first level of the Borel hierarchy. When A is a regular language, we call $\mathsf{Reach}(A)$ a *regular reachability objective* and $\mathsf{Safe}(A)$ a *regular safety objective*. We

[1] Our implementation is available at `https://github.com/pvdhove/regularMemoryRequirements`.

call an objective *regular* if it is a regular reachability or a regular safety objective. Our characterizations apply to regular reachability and safety objectives, but we sometimes discuss when we may generalize our results to the general case. For computational complexity questions, we restrict our focus to regular reachability and safety objectives so that an objective can be finitely represented as an automaton. The objectives that we consider are therefore very simple both in terms of their algebraic representation (using automata representing languages of finite words) and in terms of their topology (they are at the first level of the Borel hierarchy).

A *game* is a tuple $\mathcal{G} = (\mathcal{A}, W)$ where $\mathcal{A}$ is an arena and W is an objective.

Automata. A *deterministic automaton* is a tuple $\mathcal{D} = (Q, C, q_{\mathsf{init}}, \delta, F)$ where Q is a possibly infinite set of *states*, C is a non-empty *alphabet* (usually the set of colors), $q_{\mathsf{init}} \in Q$ is an *initial state*, $\delta \colon Q \times C \to Q$ is a (complete, deterministic) *update function*, and $F \subseteq Q$ is a set of *final states*. All automata in this work are deterministic, so we sometimes omit the word *deterministic*. Automaton $\mathcal{D}$ is *finite* if Q is finite. We write $\delta^* \colon M \times C^* \to M$ for the natural extension of δ to sequences of colors. The *language recognized by* $\mathcal{D}$, denoted $\mathcal{L}(\mathcal{D})$, is the set of finite words $w \in C^*$ such that $\delta^*(q_{\mathsf{init}}, w) \in F$. For $q_1, q_2 \in Q$, we write $\Pi^{\mathcal{D}}_{q_1,q_2}$ for the language of words $w \in C^*$ such that $\delta^*(q_1, w) = q_2$. We drop the superscript $\mathcal{D}$ if the automaton considered is clear in the context. We denote the empty word by ε.

Continuations. For an objective $W \subseteq C^\omega$ and $w \in C^*$, we define the *winning continuations of* w as the set $w^{-1}W = \{w' \in C^\omega \mid ww' \in W\}$ (this set is sometimes called a *left quotient* of W in the literature). Given an objective $W \subseteq C^\omega$, its *prefix preorder* $\preceq_W \subseteq C^* \times C^*$ is defined as $w_1 \preceq_W w_2$ if $w_1^{-1}W \subseteq w_2^{-1}W$. Its *prefix equivalence* $\sim_W \subseteq C^* \times C^*$ is defined as $w_1 \sim_W w_2$ if $w_1^{-1}W = w_2^{-1}W$. We denote $\prec_W = \preceq_W \setminus \sim_W$. We drop the subscript W when there is no ambiguity on the objective. The prefix preorder is a relation that is preserved by reading colors.

▶ **Lemma 1.** *Let $W \subseteq C^\omega$ be an objective. If $w_1 \preceq w_2$, then for all $w \in C^*$, $w_1 w \preceq w_2 w$.*

Starting from a general reachability or safety objective $W \subseteq C^\omega$ derived from a set $A \in C^*$, we can associate with W its minimal automaton $\mathcal{D}_W$ that "classifies" the equivalence classes of $\sim$. Formally, $\mathcal{D}_W = (Q, C, q_{\mathsf{init}}, \delta, F)$ where $Q = \{[w]_\sim \mid w \in C^*\}$ is the set of equivalence classes of $\sim$, $q_{\mathsf{init}} = [\varepsilon]_\sim$, $\delta([w]_\sim, c) = [wc]_\sim$, and $F = \{q_{\mathsf{fin}}\}$ where $q_{\mathsf{fin}} = [w]_\sim$ for some $w \in A$ (the choice of w does not matter). The transition function δ is well-defined: $w_1 \sim w_2$ implies $w_1 c \sim w_2 c$ for all $c \in C$. Notice that the final state of such an automaton is always absorbing, i.e., for all $c \in C$, $\delta(q_{\mathsf{fin}}, c) = q_{\mathsf{fin}}$. This matches the intuition that once a word of A is seen and the reachability (resp. safety) game is won (resp. lost), it stays that way for the rest of the game.

We have that a general reachability (resp. safety) objective W is equal to $\mathsf{Reach}(\mathcal{L}(\mathcal{D}_W))$ (resp. to $\mathsf{Safe}(\mathcal{L}(\mathcal{D}_W))$) – in examples, we will sometimes start from an automaton to generate an objective. Using the well-known Myhill-Nerode theorem [20], we obtain that a general reachability or safety objective W is regular if and only if $\sim$ has finitely many equivalence classes if and only if $\mathcal{D}_W$ is finite.

When considering a minimal automaton $\mathcal{D}_W = (Q, C, q_{\mathsf{init}}, \delta, F)$, for $q \in Q$, we abusively write $q^{-1}W$ for the set $w^{-1}W$, where w is any finite word such that $\delta^*(q_{\mathsf{init}}, w) = q$ (the choice of w does not matter). We extend $\preceq$ to automaton states ($q_1 \preceq q_2$ if $q_1^{-1}W \subseteq q_2^{-1}W$).

Preorders. Let $\preceq$ be a preorder on some set B. We say that two elements $b_1, b_2 \in B$ are *comparable for* $\preceq$ if $b_1 \preceq b_2$ or $b_2 \preceq b_1$. A set $\Gamma \subseteq B$ is a *chain for* $\preceq$ (resp. *antichain for* $\preceq$) if for all $b_1, b_2 \in \Gamma$, b_1 and b_2 are (resp. are not) comparable for $\preceq$. A preorder $\preceq$ is *well-founded* if every chain for $\preceq$ contains a minimal element for $\preceq$.

Memory structures. A *(chromatic) memory structure* is a tuple $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$ where M is a possibly infinite set of *states*, $m_{\mathsf{init}} \in M$ is an *initial state*, and $\alpha_{\mathsf{upd}} \colon M \times C \to M$ is a (deterministic, complete) *update function*. It is syntactically almost the same as a deterministic automaton, except that we do not specify final states. We recover notations α^*_{upd} and Π_{m_1,m_2} (for $m_1, m_2 \in M$) from automata. We let $\mathcal{M}_{\mathsf{triv}} = (\{m_{\mathsf{init}}\}, m_{\mathsf{init}}, (m_{\mathsf{init}}, c) \mapsto m_{\mathsf{init}})$ denote the only memory structure with a single state. The *size* of a memory structure is its number of states.

Strategies. Let $\mathcal{A} = (V, V_1, V_2, E)$ be an arena and $i \in \{1, 2\}$. A *strategy of $\mathcal{P}_i$ on $\mathcal{A}$* is a function $\sigma_i \colon \mathsf{Hists}_i(\mathcal{A}) \to E$ such that for all $\gamma \in \mathsf{Hists}_i(\mathcal{A})$, $\mathsf{out}(\gamma) = \mathsf{in}(\sigma_i(\gamma))$. Given a strategy σ_i of $\mathcal{P}_i$, we say that a play $\pi = e_1 e_2 \dots$ is *consistent with σ_i* if for all finite prefixes $\gamma = e_1 \dots e_j$ of π such that $\mathsf{out}(\gamma) \in V_i$, $\sigma_i(\gamma) = e_{j+1}$. For $v \in V$, we denote by $\mathsf{Plays}(\mathcal{A}, v, \sigma_i)$ the set of plays on $\mathcal{A}$ from v that are consistent with σ_i.

For $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$ a memory structure, a strategy σ_i of $\mathcal{P}_i$ on arena $\mathcal{A}$ is *based on (memory) $\mathcal{M}$* if there exists a function $\alpha_{\mathsf{nxt}} \colon V_i \times M \to E$ such that for all $v \in V_i$, $\sigma_i(\lambda_v) = \alpha_{\mathsf{nxt}}(v, m_{\mathsf{init}})$, and for all non-empty histories $\gamma \in \mathsf{Hists}_i(\mathcal{A})$, $\sigma_i(\gamma) = \alpha_{\mathsf{nxt}}(\mathsf{out}(\gamma), \alpha^*_{\mathsf{upd}}(m_{\mathsf{init}}, \mathsf{col}^*(\gamma)))$. A strategy is *memoryless* if it is based on $\mathcal{M}_{\mathsf{triv}}$. For conciseness, we sometimes abusively assume that a strategy of $\mathcal{P}_i$ based on $\mathcal{M}$ is a function $V_i \times M \to E$.

▶ Remark 2. This *chromatic* memory model only observes the sequence of *colors* seen, and not the precise edges that are taken during a play (i.e., the current memory state is determined by the word in C^* seen, not by the history in E^*). A memory structure observing the edges is sometimes called a *chaotic* memory [18] and, as was recently shown, may allow to play optimally with fewer memory states for some objectives [8]. However, this comes at the cost of needing to specialize the transition function of the memory structure for every arena – it does not provide an *arena-independent* memory structure [5]. The chaotic memory requirements of general safety objectives are characterized in [10] while, as far as we know, the chaotic memory requirements of general and regular reachability objectives are unknown. ⌟

Optimality. Let $\mathcal{G} = (\mathcal{A} = (V, V_1, V_2, E), W)$ be a game, and $v \in V$. We say that a strategy σ_1 of $\mathcal{P}_1$ on $\mathcal{A}$ is *winning from v for W* if for all $\pi \in \mathsf{Plays}(\mathcal{A}, v, \sigma_1)$, $\mathsf{col}^\omega(\pi) \in W$.

A strategy of $\mathcal{P}_1$ is *optimal for $\mathcal{P}_1$ in $(\mathcal{A}, W)$* if it is winning from all the vertices of $\mathcal{A}$ from which $\mathcal{P}_1$ has a winning strategy. We often write *optimal for $\mathcal{P}_1$ in $\mathcal{A}$* if the objective W is clear from the context.

▶ Remark 3. We stress that this notion of optimality requires a *single* strategy to be winning from *all* the winning vertices (a property sometimes called *uniformity*). Asking for uniformity may require strategies that are more complex to implement than just requiring winning strategies from individual vertices. Still, uniformity is a common requirement (see, e.g., [14, 21]) that comes at no extra cost in many well-studied situations [13, 12]. We discuss uniformity again in Remark 6.

Note also that there is no requirement on the behavior of an optimal strategy from vertices from which no strategy is winning, as we assume that the opponent plays rationally. In particular, even if winning becomes possible due to a mistake of the opponent after starting from a non-winning vertex, an optimal strategy needs not win. ⌟

Let $\mathcal{M}$ be a memory structure and $W \subseteq C^\omega$ be an objective. We say that $\mathcal{M}$ *suffices (to play optimally) for W (resp. in finite, finitely branching, one-player arenas)* if for all (resp. finite, finitely branching, one-player) arenas $\mathcal{A}$, $\mathcal{P}_1$ has an optimal strategy based on $\mathcal{M}$ in game $(\mathcal{A}, W)$.

3 Technical overview

In this section, we start with a more in-depth discussion of the related literature. We then present our main contributions (characterization of the memory requirements of safety objectives, of reachability objectives, and the computational complexity of the related decision problems) while describing and illustrating the main concepts used in our results. Complete proofs for the three kinds of contributions are available in the full version [4].

Related works. To classify the existing literature on memory for games, we identify two axes. The first is whether they concern chaotic memory or chromatic memory. The second is how the class of objectives is defined: either in automata-theoretic terms, typically as a subclass of ω-regular languages, or in topological terms, referring to the natural topology over the set of infinite words.

The result of Dziembowski, Jurdziński, and Walukiewicz [13] applies to the whole class of Muller objectives, which specify the set of colors which appears infinitely many times. It shows that Zielonka trees [23] can be used to compute chaotic memory requirements in polynomial time. Recently, Casares [8] has shown that this characterization does not extend to chromatic memory: deciding whether there is a memory structure of size k becomes NP-complete and equivalent to minimizing *transition-based Rabin automata*. In this direction, Casares, Colcombet and Lehtinen [9] showed that computing chaotic memory requirements for Muller objectives is equivalent to minimizing good-for-games automata. A result by Bouyer, Randour, and Vandenhove [7] provides a link between the chromatic memory requirements of all ω-regular objectives (not only Muller conditions) and their representation as *transition-based parity automata*, but with less tight bounds on the minimal memory structures.

Article [6] establishes the existence of finite-memory optimal strategies from topological properties of objectives. Although general reachability and safety objectives fit into their framework, there are major differences with our work: their framework is different (they study *concurrent* games that are not played *on graphs*), and their aim is to establish the existence of finite-memory optimal strategies for many objectives, but not to understand precisely the memory requirements of some class of objectives.

Regular objectives are also mentioned in [19], where the existence of finite-memory optimal strategies is shown for Boolean combinations of objectives involving regular objectives.

In another line of works, Gimbert and Zielonka [14] gave a characterization of all payoff functions (extending objectives to a quantitative setting) for which both players have memoryless optimal strategies, implying an important lifting result: the sufficiency of memoryless strategies in finite two-player arenas is implied by the existence of memoryless optimal strategies in both players' finite one-player arenas. Bouyer et al. [5] extended this to chromatic finite memory.

The work most related to the present paper is by Colcombet, Fijalkow, and Horn [10, 11], which gives a characterization of chaotic memory requirements for general safety objectives. Their constructions strongly rely on the model of chaotic memory; indeed, as a corollary of our results, we will see that already for regular safety objectives, chromatic and chaotic memory requirements do not coincide. Our first step is to obtain a characterization of chromatic memory requirements for (general and regular) safety objectives.

3.1 Monotony and safety objectives

Let us fix an objective $W \subseteq C^\omega$. In order to play optimally for W, a memory structure $\mathcal{M}$ needs to be able to distinguish between histories that are not comparable for $\preceq_W$: indeed, if two finite words $w_1, w_2 \in C^*$ are not comparable, we can construct an arena in which the opponent chooses between playing w_1 and playing w_2, and then the correct choice has to be made between a continuation only winning after w_1, and a continuation only winning after w_2. This motivates the following definition, which we call $\mathcal{M}$-*strong-monotony*.

▶ **Definition 4** ($\mathcal{M}$-strong-monotony). *Let $W \subseteq C^\omega$ be an objective and $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$ be a memory structure. We say that W is* $\mathcal{M}$-strongly-monotone *if for all $w_1, w_2 \in C^*$, $\alpha^*_{\mathsf{upd}}(m_{\mathsf{init}}, w_1) = \alpha^*_{\mathsf{upd}}(m_{\mathsf{init}}, w_2)$ implies that w_1 and w_2 are comparable for $\preceq_W$.*

Notice also that W is $\mathcal{M}$-strongly-monotone if and only if $\overline{W}$ is $\mathcal{M}$-strongly-monotone (as being comparable for $\preceq_W$ is equivalent to being comparable for $\preceq_{\overline{W}} = \succeq_W$). Although stated differently, a property called *strong monotony* was introduced in [1] and coincides with our definition of $\mathcal{M}_{\mathsf{triv}}$-strong-monotony. We can therefore see our definition as a reformulation and a generalization to handle arbitrary memory structures, rather than only the "memoryless memory structure" $\mathcal{M}_{\mathsf{triv}}$.

The discussion above implies that for a memory $\mathcal{M}$, $\mathcal{M}$-strong-monotony is necessary for $\mathcal{M}$ to be sufficient to play optimally. Depending on the type of objective (regular or general), we specify a class of arenas in which $\mathcal{M}$-strong-monotony can already be shown to be necessary. Intuitively, regularity allows to distinguish distinct objectives with ultimately periodic words, which can be encoded into a finite arena.

▶ **Lemma 5** (Necessity of $\mathcal{M}$-strong-monotony). *Let W be an objective and $\mathcal{M}$ a memory structure.*

1. *If W is regular and $\mathcal{M}$ suffices to play optimally for W in all finite one-player arenas, then W is $\mathcal{M}$-strongly-monotone.*
2. *In the general case, if $\mathcal{M}$ suffices to play optimally for W in all finitely branching one-player arenas, then W is $\mathcal{M}$-strongly-monotone.*

Complete proofs for this section can be found in [4, Section 4].

▶ Remark 6. This is the only result relying on the "uniformity" assumption (see Remark 3). This assumption is crucial to obtain this lemma with a hypothesis about *one-player* arenas. We provide additional details about the (small) cost of requiring uniformity w.r.t. memory requirements in two-player games in [4, Section 4]. ⌟

In the case of general reachability or safety objectives, it is useful to reformulate the notion of $\mathcal{M}$-strongly-monotone objectives using chains. Given a general reachability or safety objective W, its minimal automaton $\mathcal{D}_W = (Q, C, q_{\mathsf{init}}, \delta, F)$, and a memory structure $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$, we can associate with each state $m \in M$ the set $\Gamma^W_m \subseteq Q$ of states of $\mathcal{D}_W$ that can be reached "simultaneously". Formally, for $m \in M$,

$$\Gamma^W_m = \{\delta^*(q_{\mathsf{init}}, w) \in Q \mid w \in C^*, \alpha^*_{\mathsf{upd}}(m_{\mathsf{init}}, w) = m\}.$$

We drop the superscript W if there is no ambiguity. The following property follows from the definitions.

▶ **Lemma 7.** *Let W be a general reachability or safety objective and $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$ be a memory structure. Objective W is $\mathcal{M}$-strongly-monotone if and only if for all $m \in M$, the set Γ_m is a chain for $\preceq_W$.*

Our initial definition of $\mathcal{M}$-strong-monotony required that any two finite words reaching the same state of $\mathcal{M}$ must be comparable; in this reformulation, we focus instead on the minimal automaton of W and require that states of the automaton that can be reached along with the same state of $\mathcal{M}$ are comparable.

Our first characterization states that for general safety objectives, $\mathcal{M}$-strong-monotony also implies that $\mathcal{M}$ suffices to play optimally. We state two variants of the results: in the first one, we assume that the preorder $\preceq$ induced by the objective is well-founded (which includes the regular case), and the result holds for all arenas; in the second one, we make no such assumption, but the result holds only for finitely branching arenas. We will discuss why we do not have the result with none of these hypotheses in Remark 10.

▶ **Theorem 8** (Characterization for safety). *Let W be a general safety objective, and $\mathcal{M}$ be a memory structure.*

1. *If $\preceq_W$ is well-founded (in particular, if W is regular), then $\mathcal{M}$ suffices to play optimally for W if and only if W is $\mathcal{M}$-strongly-monotone.*
2. *In the general case, $\mathcal{M}$ suffices to play optimally for W in all finitely branching arenas if and only if W is $\mathcal{M}$-strongly-monotone.*

Proof sketch. We provide an overview of the proof of Theorem 8 (complete proof in [4, Section 4]). We discuss here the sufficiency of $\mathcal{M}$-strong-monotony (the necessity is easier and was stated in Lemma 5).

Let $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$ and let $\mathcal{D}_W = (Q, C, q_{\mathsf{init}}, \delta, F)$ be the minimal automaton of W. We assume that W is $\mathcal{M}$-strongly-monotone. Let $\mathcal{A} = (V, V_1, V_2, E)$ be an arena. As per the hypotheses, we require that $\preceq$ is well-founded or that $\mathcal{A}$ is finitely branching.

We want to build a strategy σ optimal for $\mathcal{P}_1$ in $\mathcal{A}$ such that $\sigma\colon V_1 \times M \to E$ is based on $\mathcal{M}$. The key to the proof is to understand the following sets of states of $\mathcal{D}_W$ in order to know what to play in each pair $(v, m) \in V_1 \times M$. For $v \in V$, $m \in M$, we define

$$Q_{v,m} = \{q \in \Gamma_m \mid \mathcal{P}_1 \text{ has a winning strategy for objective } q^{-1}W \text{ from } v\}.$$

States in $Q_{v,m}$ are states of $\mathcal{D}_W$ that could be reached while the memory state is m, by definition of Γ_m. Moreover, $\mathcal{M}$-strong-monotony tells us that each Γ_m is a chain for $\preceq$ (Lemma 7), so each $Q_{v,m}$ is too.

For given $v \in V_1$ and $m \in M$, we distinguish three possibilities.

- If $Q_{v,m}$ is empty, then this means that there is no state of Γ_m for which $\mathcal{P}_1$ can win from v (i.e., for all $q \in \Gamma_m$, $\mathcal{P}_1$ has no winning strategy for $q^{-1}W$ from v). In this case, we can define $\sigma(v, m)$ arbitrarily, as there is no hope to win.
- If $Q_{v,m}$ has a minimum $q_{v,m}$ for $\preceq$, then this minimum represents the worst (for $\preceq$) state of Γ_m for which $\mathcal{P}_1$ still has a winning strategy. We define $\sigma(v, m)$ as the edge played by such a winning strategy. Intuitively, this is the most robust way to play as it is a winning move for the worst possible state of Γ_m for which winning is possible. Notice that a non-empty $Q_{v,m}$ always has a minimum when $\preceq$ is well-founded.
- In case $Q_{v,m}$ is non-empty and has no minimum, then $\preceq$ is not well-founded, so we work under the hypothesis that $\mathcal{A}$ is finitely branching. We can then consider, for each $q \in Q_{v,m}$, the set of edges E_q that can be taken from v to win for $q^{-1}W$. Each E_q is finite and non-empty, and they are ordered for inclusion. We can then show that their intersection is non-empty, so there is an edge that can be taken in v to win for all $q \in Q_{v,m}$. We define $\sigma(v, m)$ as such an edge.

We have now defined a strategy σ based on $\mathcal{M}$ that makes local choices that are played by winning strategies for as many states of Γ_m (where m is the current memory state) as possible. Using the fact that W is a general safety condition, one can prove that this strategy is in fact optimal. ◀

A corollary of this characterization, by comparing to the characterization for chaotic memory in [10], is that chromatic and chaotic memory requirements differ already for regular safety objectives. We provide an instructive example below. Note that this provides a new simple kind of counterexample to Kopczyński's conjecture [18], which Casares [8] had already falsified with a Muller objective.

▶ **Example 9.** Let $C = \{a, b, c, d\}$. We consider the regular language recognized by the finite automaton $\mathcal{D}$ depicted in Figure 1 (left). It accepts the finite words that first see both a and b (in any order, possibly interspersed with c's and d's), and then see both c and d (in any order, possibly interspersed with a's and b's). This language can be described by the regular expression $C^*(aC^*b \mid bC^*a)C^*(cC^*d \mid dC^*c)C^*$. We write W for the induced regular safety objective: $W = \mathsf{Safe}(\mathcal{L}(\mathcal{D}))$.

The main claim is that the chaotic memory requirements for W are two states, which is easily obtained from the existing characterization [10] (this is the size of a maximal antichain for $\preceq$), while the chromatic requirements for W are three states. We depict a memory structure $\mathcal{M}$ with three states which makes W $\mathcal{M}$-strongly-monotone in Figure 1 (right). To check that W is indeed $\mathcal{M}$-strongly-monotone, we have to check that there is no pair of words $w_1, w_2 \in C^*$ such that w_1 and w_2 reach the same state of $\mathcal{M}$, but reach non-comparable states in $\mathcal{D}$. The only two pairs of non-comparable states in $\mathcal{D}$ are q_a and q_b, and q_c and q_d (besides these, states are ordered for $\preceq$ from right to left). We can check that for this choice of $\mathcal{M}$, $\Gamma_{m_1} = \{q_{\mathsf{init}}, q_a\}$, $\Gamma_{m_2} = \{q_b, q_{ab}, q_d, q_{cd}\}$, $\Gamma_{m_3} = \{q_b, q_{ab}, q_c, q_{cd}\}$. As these are all chains for $\preceq$, we have that W is $\mathcal{M}$-strongly-monotone.

It is not possible to find a chromatic memory structure $\mathcal{M}$ with *two* states which makes W $\mathcal{M}$-strongly-monotone (this can be checked by trying to assign transitions to two states while distinguishing non-comparable states, and observing that all cases fail). ⌟

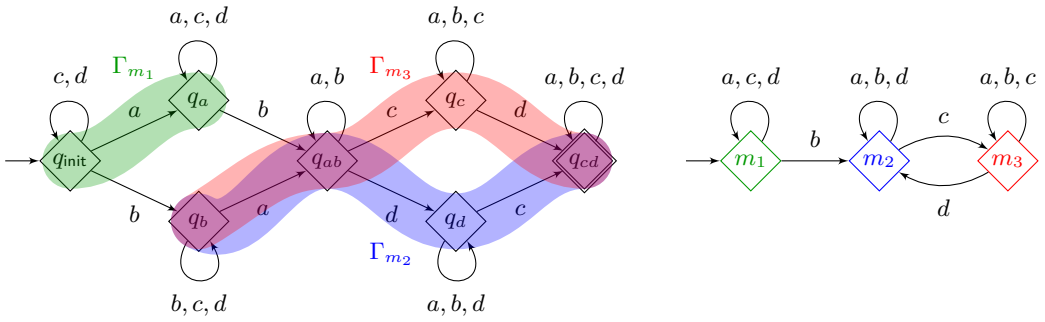


Figure 1 Example 9: automaton $\mathcal{D}$ (left) and a minimal memory structure $\mathcal{M}$ (right) such that $\mathsf{Reach}(\mathcal{L}(\mathcal{D}))$ and $\mathsf{Safe}(\mathcal{L}(\mathcal{D}))$ are $\mathcal{M}$-strongly-monotone. In figures, diamonds are used to depict automaton states and memory states, and accepting states are depicted with a double border.

To conclude this section, we discuss why, with neither the well-foundedness hypothesis nor the finitely branching hypothesis from Theorem 8, we cannot expect such a characterization.

▶ Remark 10. If the prefix preorder of an objective W is not well-founded, then there is an infinite decreasing sequence of finite words $w_1 \succ w_2 \succ \dots$ in C^*. This means that for all $i \geq 1$, there is $w'_i \in C^\omega$ such that $w_i w'_i \in W$, but for $j > i$, $w_j w'_i \notin W$. We can then build the infinitely branching arena depicted in Figure 2 in which $\mathcal{P}_2$ first chooses a word w_j, and $\mathcal{P}_1$ can win by playing a word w'_i with $i \geq j$. This requires infinite memory, even if W is $\mathcal{M}_{\mathsf{triv}}$-strongly-monotone. ⌟

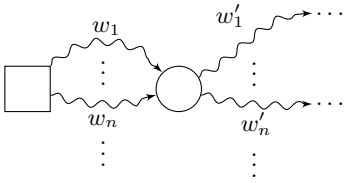


Figure 2 Infinite branching arena in which $\mathcal{P}_1$ needs memory beyond the $\mathcal{M}$-strong-monotony property in Remark 10. In figures, circles (resp. squares) represent arena vertices controlled by $\mathcal{P}_1$ (resp. $\mathcal{P}_2$), i.e., in V_1 (resp. V_2). Squiggly arrows indicate a sequence of edges.

3.2 Capturing progress and reachability objectives

To play optimally for general and regular reachability objectives with a memory $\mathcal{M}$, $\mathcal{M}$-strong-monotony is necessary (Lemma 5) but not enough: the following example shows that the memory structure must keep track of *progress*.

▶ **Example 11.** Let $C = \{a, b\}$. We consider the regular language $b^*a^+bC^*$ of words that have to see at least one a, followed by at least one b. This language is recognized by the finite automaton $\mathcal{D}$ in Figure 3 (left). We write W for the induced regular reachability objective: $W = \mathsf{Reach}(\mathcal{L}(\mathcal{D}))$.

In the arena in Figure 3 (center), $\mathcal{P}_1$ may win by starting a play with ab, but not without memory. The intuition is that playing a first makes some progress (it reaches an automaton state with more winning continuations), but is not sufficient to win, even if repeated. Therefore, in our memory structures, if a word makes some progress but without guaranteeing the win when repeated, we want the memory state to change upon reading that word. The memory structure in Figure 3 (right) is sufficient for W; in particular, seeing the first a, which makes progress from q_{init} to q_a, changes the memory state. ⌟

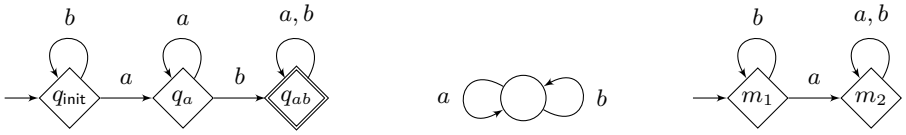


Figure 3 Example 11: automaton $\mathcal{D}$ (left), an arena requiring memory for $\mathsf{Reach}(\mathcal{L}(\mathcal{D}))$ (center), and a minimal sufficient memory structure (right).

We formalize this intuition in the following definition, which is a generalization of the *progress-consistency* property [3]. Notation Π_{m_1,m_2}, representing the finite words read from memory state m_1 to memory state m_2, was defined in Section 2.

▶ **Definition 12** ($\mathcal{M}$-progress-consistency)**.** *Let W be an objective and $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$ be a memory structure. We say that W is* $\mathcal{M}$-progress-consistent *if for all $m \in M$, for all $w_1 \in \Pi_{m_{\mathsf{init}},m}$, for all $w_2 \in \Pi_{m,m}$,*

$$w_1 \prec w_1w_2 \implies w_1(w_2)^\omega \in W.$$

Intuitively, this says that if it is possible to come back to the same memory state while reading a "word that makes progress" (i.e., that improves our situation by putting us in a position with more winning continuations), then repeating this word infinitely often from that point onward must be winning. The notion of $\mathcal{M}_{\mathsf{triv}}$-progress-consistency corresponds to the previous definition of *progress-consistency* [3].

The discussion above shows that $\mathcal{M}$-progress-consistency is necessary for a memory structure $\mathcal{M}$ to be sufficient to play optimally. As for $\mathcal{M}$-strong-monotony, we distinguish the regular case from the general case.

▶ **Lemma 13** (Necessity of $\mathcal{M}$-progress-consistency). *Let W be an objective and $\mathcal{M}$ a memory structure.*

1. *If W is regular and $\mathcal{M}$ suffices to play optimally for W in all finite one-player arenas, then W is $\mathcal{M}$-progress-consistent.*
2. *In the general case, if $\mathcal{M}$ suffices to play optimally for W in all finitely branching one-player arenas, then W is $\mathcal{M}$-progress-consistent.*

Complete proofs for this section can be found in [4, Section 5]. The following example should help the reader form the right intuition about $\mathcal{M}$-progress-consistency.

▶ **Example 14.** Let $C = \{a, b\}$. We consider the regular language of words containing *ababa* as a (non-necessarily contiguous) subword, recognized by the finite automaton $\mathcal{D}$ in Figure 4 (left). We consider the memory structure $\mathcal{M}$ remembering whether a or b was last seen, depicted in Figure 4 (right). The regular reachability objective $W = \mathsf{Reach}(\mathcal{L}(\mathcal{D}))$ is $\mathcal{M}$-progress-consistent. Indeed, let us first consider $m = m_b$ in the definition of $\mathcal{M}$-progress-consistency. A finite word w_1 reaching m_b in $\mathcal{M}$ necessarily reaches q_{init}, q_{ab}, or q_{abab} in Q (excluding the final state from the reasoning, as no progress is possible from it). After w_1, words w_2 that both (i) make progress ($w_1 \prec w_1w_2$) and (ii) are a cycle on m_b necessarily see both a and b. Therefore, $w_1(w_2)^\omega$ is always a winning word. The same reasoning holds for $m = m_a$. Notice that the memory states from the memory structure do not carry enough information to ascertain when a word of the language has been seen (i.e., when the game is won).

The upcoming Theorem 16 implies that $\mathcal{M}$ suffices to play optimally for $\mathcal{P}_1$. ⌟

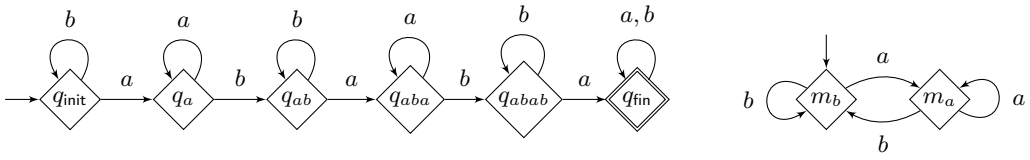


Figure 4 Example 14: automaton $\mathcal{D}$ (left) and memory structure $\mathcal{M}$ (right).

This need to capture *progress* was not necessary to understand the memory requirements of safety objectives, which may be explained by the following reasoning.

▶ Remark 15. Unlike general reachability objectives, all general safety objectives are $\mathcal{M}_{\mathsf{triv}}$-progress-consistent. Here is a proof of this statement. Let $W \subseteq C^\omega$ be a general safety objective. Let $w_1, w_2 \in \Pi^{\mathcal{M}_{\mathsf{triv}}}_{m_{\mathsf{init}}, m_{\mathsf{init}}} = C^*$ be such that $w_1 \prec w_1w_2$. This implies that w_1w_2, and therefore w_1, have a non-empty set of winning continuations. Assume by contradiction that $w_1(w_2)^\omega \notin W$. As W is a general safety objective, there is a smallest $n \geq 1$ such that $w_1(w_2)^n$ has no winning continuation. Hence, $w_1(w_2)^{n-1}$ still has some winning continuations, so $w_1(w_2)^n \prec w_1(w_2)^{n-1}$. This is a contradiction, as $w_1 \prec w_1w_2$ implies that $w_1(w_2)^{n-1} \preceq w_1w_2(w_2)^{n-1} = w_1(w_2)^n$ by Lemma 1. This property is, at least intuitively, a reason hinting that the memory requirements of safety objectives are lower and easier to understand than those for their complement reachability objective. ⌟

We have now discussed two necessary properties for a memory $\mathcal{M}$ to be sufficient to play optimally for an objective. For regular reachability objectives, it appears that the conjunction of these two properties is also sufficient.

▶ **Theorem 16** (Characterization for reachability). *Let W be a regular reachability objective and $\mathcal{M}$ be a finite memory structure. Memory $\mathcal{M}$ suffices to play optimally for W if and only if W is $\mathcal{M}$-strongly-monotone and $\mathcal{M}$-progress-consistent.*

Proof sketch. We provide an overview of the proof of Theorem 16 (complete proof in [4, Section 5]). We discuss here the sufficiency of $\mathcal{M}$-strong-monotony and $\mathcal{M}$-progress-consistency (their necessity is easier and was stated in Lemmas 5 and 13).

Let $\mathcal{D}_W = (Q, C, q_{\mathsf{init}}, \delta, F)$ be the minimal automaton of W (which is finite as W is regular), and $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$. We assume that W is $\mathcal{M}$-strongly-monotone and $\mathcal{M}$-progress-consistent. Let $\mathcal{A} = (V, V_1, V_2, E)$ be a (possibly infinite) arena. We construct an optimal strategy based on memory $\mathcal{M}$, using the same idea as in the proof for safety objectives (Theorem 8): we once again consider a strategy based on $\mathcal{M}$ making choices that are "locally optimal". We then show, thanks to our hypotheses ($\mathcal{M}$-strong-monotony and $\mathcal{M}$-progress-consistency), that this strategy must be optimal.

For $v \in V_1$, $m \in M$, we define

$$q_{v,m} = \min_{\preceq}\{q \in \Gamma_m \mid \mathcal{P}_1 \text{ has a winning strategy for objective } q^{-1}W \text{ from } v\}.$$

Every set Γ_m is a chain using $\mathcal{M}$-strong-monotony (Lemma 7) and is finite since $\mathcal{D}_W$ is finite. Hence, the minimum $q_{v,m}$ exists (except when the set is empty, but that means that the game cannot be won anymore – we ignore this case). Let $\sigma_{v,m}$ be a strategy winning for $q_{v,m}^{-1}W$ from v.

Now, just like for the proof for safety, we want to define $\sigma(v, m)$ as the first edge taken by $q_{v,m}$ from v – we play locally reasonable edges played by good strategies and hope that this creates a "globally" optimal strategy. However, this does not work in general, as any choice for the strategies $\sigma_{v,m}$ may not be good: indeed, such strategies may be winning, but may make unnecessary moves delaying the achievement of the objective. For instance, in the arena of Figure 3, a strategy playing bab^ω is winning for $q_{\mathsf{init}}^{-1}W$, but not as fast as possible (it takes three moves to create a word in $\mathcal{L}(\mathcal{D})$, while it is possible to do it in two moves). If, by imitating the first move of this strategy, we define $\sigma(v, m_1) = (v, b, v)$, we then get stuck and σ plays the losing word b^ω.

A way to remedy this is by formally defining the "time" taken by a strategy to guarantee a win, and choosing strategies $\sigma_{v,m}$ that win in the least time. When considering infinite two-player arenas, this time has to be defined using ordinals. If we do this, it is possible (though still quite involved) to show that σ defined as above is indeed optimal, thanks to $\mathcal{M}$-progress-consistency. ◀

▶ Remark 17. Unlike safety objectives, our characterization is only shown to hold for *regular* reachability objectives. We discuss in [4, Section 5] why our proof technique does not apply to general reachability objectives (even with $\preceq$ well-founded and finite branching of the arenas). ⌟

For objectives beyond reachability and safety, $\mathcal{M}$-strong-monotony and $\mathcal{M}$-progress-consistency may not imply the sufficiency of $\mathcal{M}$ to play optimally. For instance, with $C = \{a, b\}$, let us consider the objective

$$W = \{w \in C^\omega \mid a \text{ and } b \text{ are both seen infinitely often}\},$$

which is ω-regular (it can be recognized by a *deterministic Büchi automaton* with two states), but is not a general reachability nor safety objective. Objective W is $\mathcal{M}_{\mathsf{triv}}$-strongly-monotone and $\mathcal{M}_{\mathsf{triv}}$-progress-consistent, but $\mathcal{M}_{\mathsf{triv}}$ does not suffice to play optimally.

Lift for regular objectives. As a by-product of our results, we observe that for regular objectives, our characterizations deal with arbitrary arenas of any cardinality, but the properties used in the characterizations are already necessary in *finite one-player* arenas. This means that strategy-wise, to accomplish a regular objective, all the complexity already appears in finite graphs with no opponent. For the specific class of regular objectives that we study, this strengthens so-called *one-to-two-player lifts* from the literature [14, 5].

▶ **Theorem 18** (Finite-to-infinite, one-to-two-player lift). *Let W be a regular (reachability or safety) objective and $\mathcal{M}$ be a finite memory structure. Memory $\mathcal{M}$ suffices to play optimally for W (in all arenas) if and only if $\mathcal{M}$ suffices to play optimally for W in finite one-player arenas.*

Proof. The implication from left-to-right holds as this is the same property quantified over fewer arenas. We argue the other implication for each case.

For regular safety objectives W, we showed that if $\mathcal{M}$ suffices in finite one-player arenas, then W is $\mathcal{M}$-strongly-monotone (by Lemma 5 as W is regular), which implies that $\mathcal{M}$ suffices in all arenas (by Theorem 8 as W is a safety condition with a well-founded preorder).

For regular reachability objectives W, we showed that if $\mathcal{M}$ suffices in finite one-player arenas, then W is $\mathcal{M}$-strongly-monotone and $\mathcal{M}$-progress-consistent (by Lemmas 5 and 13 as W is regular), which implies that $\mathcal{M}$ suffices in all arenas (by Theorem 16 as W is a regular reachability objective). ◀

3.3 The complexity of finding small memory structures

We finally discuss the computational complexity of finding small memory structures for regular objectives. We formalize the question as two decision problems: given a regular reachability or safety objective, how much memory is required to play optimally for this objective?

Memory-Safe

Input: A finite automaton $\mathcal{D}$ inducing the regular safety objective $W = \mathsf{Safe}(\mathcal{L}(\mathcal{D}))$ and an integer $k \in \mathbb{N}$.

Question: Does there exist a memory structure $\mathcal{M}$ of size at most k which suffices to play optimally for W?

Memory-Reach

Input: A finite automaton $\mathcal{D}$ inducing the regular reachability objective $W = \mathsf{Reach}(\mathcal{L}(\mathcal{D}))$ and an integer $k \in \mathbb{N}$.

Question: Does there exist a memory structure $\mathcal{M}$ of size at most k which suffices to play optimally for W?

It follows from our characterizations (Theorems 8 and 16) that Memory-Safe is equivalent to asking whether there is a memory structure $\mathcal{M}$ of size at most k such that $\mathsf{Safe}(\mathcal{L}(\mathcal{D}))$ is $\mathcal{M}$-strongly-monotone, and Memory-Reach whether there is a memory structure $\mathcal{M}$ of size at most k such that $\mathsf{Reach}(\mathcal{L}(\mathcal{D}))$ is $\mathcal{M}$-strongly-monotone and $\mathcal{M}$-progress-consistent.

▶ Remark 19. The way k is encoded (in binary or in unary) has no impact on the complexity. Indeed, the input consists of the number k together with a (deterministic) automaton describing the objective. Since the automaton is an upper bound on the memory requirements (for both Memory-Safe and Memory-Reach), the problem is non-trivial only when k is smaller than the size of the automaton. Therefore, the size of the input is dominated by the size of the automaton in the non-trivial cases. ⌟

▶ **Theorem 20** (Complexity of Memory-Safe and Memory-Reach). *Both Memory-Safe and Memory-Reach are NP-complete.*

For NP-hardness, we construct a reduction from the Hamiltonian cycle problem which works for both Memory-Safe and Memory-Reach. Complete proofs for this section can be found in [4, Section 6].

Our main insight is to reformulate the notion of $\mathcal{M}$-strong-monotony (NP-membership of Memory-Safe follows from this reformulation). Let $W = \mathsf{Safe}(\mathcal{L}(\mathcal{D}))$ be a regular objective and $\mathcal{M} = (M, m_{\mathsf{init}}, \alpha_{\mathsf{upd}})$ be a memory structure. In Example 9, we have seen how to go from a memory structure $\mathcal{M}$ such that W is $\mathcal{M}$-strongly-monotone to a covering of the states of $\mathcal{D}$ by chains of states. We formulate exactly the requirements for such coverings in order to have a point of view equivalent to $\mathcal{M}$-strong-monotony. For $\Gamma \subseteq Q$ a set of automaton states and $c \in C$ a color, we define $\delta(\Gamma, c) = \{\delta(q, c) \mid q \in \Gamma\}$.

▶ **Definition 21** (Monotone decomposition). *Let $\mathcal{D} = (Q, C, q_{\mathsf{init}}, \delta, F)$ be an automaton. We say that the sets $\Gamma_1, \dots, \Gamma_k \subseteq Q$ form a* monotone decomposition of $\mathcal{D}$ *if*
(a) $Q = \bigcup_{i=1}^{k} \Gamma_i$,
(b) *for all $c \in C$, for all $i \in \{1, \dots, k\}$, there is $j \in \{1, \dots, k\}$ such that $\delta(\Gamma_i, c) \subseteq \Gamma_j$, and*
(c) *for all $i \in \{1, \dots, k\}$, Γ_i is a chain for $\preceq$.*

Note that the sets Γ_i do not have to be disjoint (as was illustrated in Example 9). If we only consider requirements (a) and (b) of this definition, we recover the definition of an *admissible decomposition*, which can be used to quotient an automaton [15]. Here, we add the additional requirement (c) that each set of states is a chain for $\preceq$. Note that there always exists an admissible decomposition with just one set (by taking $\Gamma_1 = Q$), but finding a small *monotone* decomposition may not be so easy. This point of view in terms of monotone decompositions turns out to be equivalent to our initial point of view in terms of $\mathcal{M}$-strong-monotony in the following sense.

▶ **Lemma 22.** *Let $\mathcal{D}$ be an automaton and W be equal to $\mathsf{Safe}(\mathcal{L}(\mathcal{D}))$ or $\mathsf{Reach}(\mathcal{L}(\mathcal{D}))$. Automaton $\mathcal{D}$ admits a monotone decomposition with k sets if and only if W is $\mathcal{M}$-strongly-monotone for some memory structure $\mathcal{M}$ of size k.*

It is instructive to reformulate the characterization of *chaotic* memory requirements from [10]: the original phrasing was that the number of memory states necessary and sufficient to play optimally for the safety objective W is the size of the largest antichain of $\preceq_W$. Using our terminology and Dilworth's theorem, it is equivalent to the smallest number of chains required to cover all states; that is, decompositions satisfying (a) and (c) in Definition 21, but not necessarily (b). Hence, it is smaller in general.

We finish this section with an overview of the proof of Theorem 20 (complete proofs in [4, Section 6]).

Proof sketch of Theorem 20. We first discuss membership in NP, and then NP-hardness.

Membership in NP. Problem Memory-Safe with inputs $\mathcal{D}$ and k was shown in Theorem 8 to be equivalent to the existence of a memory structure $\mathcal{M}$ of size k such that W is $\mathcal{M}$-strongly-monotone. This second problem is itself equivalent by Lemma 22 to the existence of a monotone decomposition of $\mathcal{D}$ with k sets. A monotone decomposition is a polynomial-size witness, and checking whether k sets of states form a monotone decomposition is done in polynomial time by checking the three conditions in the definition. This shows NP-membership of Memory-Safe.

For Memory-Reach, we use memory structures as polynomial-size witnesses. Let $\mathcal{D}$ be a finite automaton and $k \in \mathbb{N}$. Given a memory structure $\mathcal{M}$ of size k, we want to decide in polynomial time whether objective $\mathsf{Reach}(\mathcal{L}(\mathcal{D}))$ is $\mathcal{M}$-strongly-monotone and $\mathcal{M}$-progress-consistent. We have discussed how to check $\mathcal{M}$-strong-monotony in polynomial time through monotone decompositions. Checking $\mathcal{M}$-progress-consistency in polynomial time is slightly more involved and is described in [4, Section 6]: we reduce $\mathcal{M}$-progress-consistency to checking a polynomial number of emptiness queries of intersections of regular languages recognized by deterministic finite automata.

NP-hardness. As mentioned above, we prove NP-hardness of Memory-Safe using a reduction from the Hamiltonian cycle problem. The proof also applies to Memory-Reach, but we move this discussion to [4, Section 6]. In the following, a *(directed) graph* is a tuple $G = (V, E)$ with $E \subseteq V \times V$. A *Hamiltonian cycle of* G is a sequence $(u_1, \dots, u_n)$ in which each state of V appears exactly once, $(u_i, u_{i+1}) \in E$ for all i, $1 \leq i < n$, and $(u_n, u_1) \in E$.

We start from a directed graph $G = (V, E)$ and we intend to build an automaton $\mathcal{D}_G$ such that G has a Hamiltonian cycle if and only if $\mathcal{D}_G$ has a monotone decomposition with k sets, for a well-chosen k. We write $|V| = n$ and $|E| = m$. We assume $m \geq n$, otherwise G cannot have a Hamiltonian cycle. We define $\mathsf{Automaton}(G)$ as the automaton $(Q, \Sigma, \delta, q_{\mathsf{init}}, F)$ with $Q = V \uplus E$, $\Sigma = \{\mathsf{in}, \mathsf{out}\}$, and transitions such that for $v \in V$, $\delta(v, \mathsf{in}) = \delta(v, \mathsf{out}) = v$, and for $e = (v_1, v_2) \in E$, $\delta(e, \mathsf{in}) = v_1$ and $\delta(e, \mathsf{out}) = v_2$. We ignore q_{init} and F at the moment. This definition is inspired from a reduction in [2] (although the rest of the proof is different).

We also consider the *cycle graph with* n *vertices* $C_n = (V_C, E_C)$, with $V_C = \{v_1^C, \dots, v_n^C\}$ and $E_C = \{e_1^C, \dots, e_n^C\}$ such that $e_i^C = (v_i^C, v_{i+1}^C)$ for $1 \leq i < n$ and $e_n = (v_n^C, v_1^C)$. We now consider an automaton $\mathcal{D}_G = (Q, \Sigma, \delta, q_{\mathsf{init}}, F)$ based on the disjoint union $\mathsf{Automaton}(C_n) \uplus \mathsf{Automaton}(G)$ along with three extra states q_{init}, $\bot$, and $\top$. We illustrate this part of the construction in Figure 5.

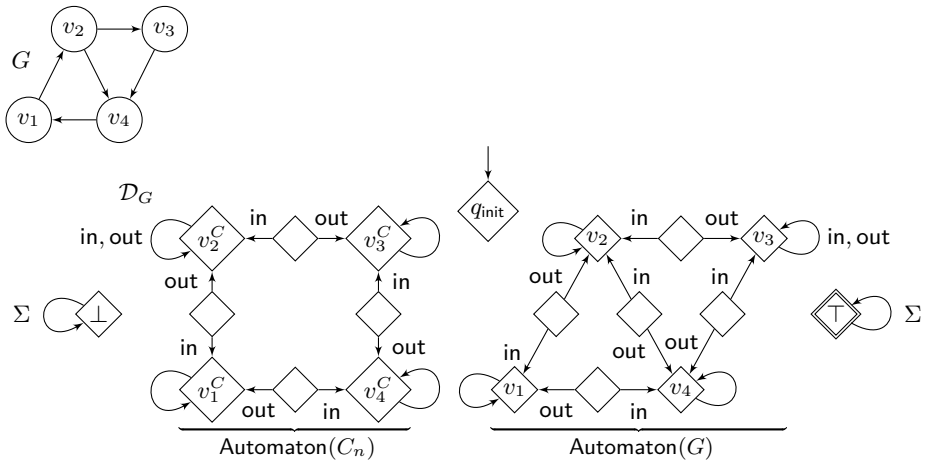


Figure 5 Illustration of automaton $\mathcal{D}_G$ starting from a graph G with four vertices. This is only a part of the full construction in [4, Section 6] to give an overview of the proof.

What is now missing is a way to induce an interesting ordering $\preceq$ – intuitively, we want $\bot$ to be the smallest state, $\top$ to be the largest, and all automaton states corresponding to vertices (resp. edges) of $\mathsf{Automaton}(C_n)$ to be smaller than all automaton states corresponding to vertices (resp. edges) of $\mathsf{Automaton}(G)$, while making all other pairs of states non-comparable. We can get this ordering by adding a letter to Σ for each state of $\mathcal{D}_G$ and defining the right transitions from q_{init} and to $\bot$ and $\top$.

In this way, we have that chains of $\mathcal{D}_G$ for $\preceq$ have at most 4 states, and chains with four states contain either $\bot$, $\top$, a vertex of C_n and a vertex of G, or $\bot$, $\top$, an edge of C_n and an edge of G. Moreover, the largest antichain of $\mathcal{D}_G$ for $\preceq$ has $n+m+1$ elements and is achieved by $V \cup E \cup \{q_{\mathsf{init}}\}$. By a counting argument, it is then possible to cover all states with $n+m+1$ chains if only if every vertex (resp. edge) of C_n is in a chain with one vertex (resp. edge) of G. A covering with $n+m+1$ chains therefore induces a bijection between V_C and V and an injection from E_C to E. To form a monotone decomposition, these chains still have to satisfy condition b from Definition 21. If it is possible to find $n+m+1$ such chains, we can show by reading in and out from chains containing edges that the cycle on C_n transfers to a Hamiltonian cycle on G. Reciprocally, if G has a Hamiltonian cycle, then we can find a natural correspondence between vertices (resp. edges) of C_n and vertices (resp. edges) of G that allows to define a monotone decomposition with $n+m+1$ sets. We have that G has a Hamiltonian cycle if and only if $\mathcal{D}_G$ has a monotone decomposition in $k = n+m+1$ sets. ◀

4 Conclusion

We have characterized the minimal memory structures sufficient to play optimally for regular reachability and safety objectives. In doing so, we were able to prove that related decision problems about regular objectives were NP-complete. Our characterizations were encoded into a SAT solver that automatically generates a minimal memory structure given a finite automaton as an input (link in Section 1).

This article can be seen as one step toward understanding more generally the (chromatic or chaotic) memory requirements of all ω-regular objectives, as well as synthesizing minimal memory structures for them. The chaotic memory requirements of regular reachability objectives are still unknown, as well as the chromatic memory requirements of larger classes of ω-regular objectives (such as, e.g., the objectives recognized by deterministic Büchi automata).

References

1 Alessandro Bianco, Marco Faella, Fabio Mogavero, and Aniello Murano. Exploring the boundary of half-positionality. *Annals of Mathematics and Artificial Intelligence*, 62(1-2):55–77, 2011. doi:10.1007/s10472-011-9250-1.

2 Kellogg S. Booth. Isomorphism testing for graphs, semigroups, and finite automata are polynomially equivalent problems. *SIAM Journal on Computing*, 7(3):273–279, 1978. doi:10.1137/0207023.

3 Patricia Bouyer, Antonio Casares, Mickael Randour, and Pierre Vandenhove. Half-positional objectives recognized by deterministic Büchi automata. In Bartek Klin, Sławomir Lasota, and Anca Muscholl, editors, *Proceedings of the 33rd International Conference on Concurrency Theory, CONCUR 2022, Warsaw, Poland, September 12–16, 2022*, volume 243 of *LIPIcs*, pages 20:1–20:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CONCUR.2022.20.

4 Patricia Bouyer, Nathanaël Fijalkow, Mickael Randour, and Pierre Vandenhove. How to play optimally for regular objectives? *CoRR*, abs/2210.09703, 2022. doi:10.48550/arXiv.2210.09703.

5 Patricia Bouyer, Stéphane Le Roux, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. Games where you can play optimally with arena-independent finite memory. *Logical Methods in Computer Science*, 18(1), 2022. doi:10.46298/lmcs-18(1:11)2022.

6 Patricia Bouyer, Stéphane Le Roux, and Nathan Thomasset. Finite-memory strategies in two-player infinite games. In Florin Manea and Alex Simpson, editors, *Proceedings of the 30th EACSL Annual Conference on Computer Science Logic, CSL 2022, Göttingen, Germany, February 14–19, 2022*, volume 216 of *LIPIcs*, pages 8:1–8:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CSL.2022.8.

7 Patricia Bouyer, Mickael Randour, and Pierre Vandenhove. Characterizing omega-regularity through finite-memory determinacy of games on infinite graphs. *TheoretiCS*, 2:1–48, 2023. doi:10.46298/theoretics.23.1.

8 Antonio Casares. On the minimisation of transition-based Rabin automata and the chromatic memory requirements of Muller conditions. In Florin Manea and Alex Simpson, editors, *Proceedings of the 30th EACSL Annual Conference on Computer Science Logic, CSL 2022, Göttingen, Germany, February 14–19, 2022*, volume 216 of *LIPIcs*, pages 12:1–12:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CSL.2022.12.

9 Antonio Casares, Thomas Colcombet, and Karoliina Lehtinen. On the size of good-for-games Rabin automata and its link with the memory in Muller games. In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, Paris, France, July 4–8, 2022*, volume 229 of *LIPIcs*, pages 117:1–117:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.117.

10 Thomas Colcombet, Nathanaël Fijalkow, and Florian Horn. Playing safe. In Venkatesh Raman and S. P. Suresh, editors, *Proceedings of the 34th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2014, New Delhi, India, December 15–17, 2014*, volume 29 of *LIPIcs*, pages 379–390. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014. doi:10.4230/LIPIcs.FSTTCS.2014.379.

11 Thomas Colcombet, Nathanaël Fijalkow, and Florian Horn. Playing safe, ten years later. *CoRR*, abs/2212.12024, 2022. doi:10.48550/arXiv.2212.12024.

12 Thomas Colcombet and Damian Niwiński. On the positional determinacy of edge-labeled games. *Theoretical Computer Science*, 352(1-3):190–196, 2006. doi:10.1016/j.tcs.2005.10.046.

13 Stefan Dziembowski, Marcin Jurdziński, and Igor Walukiewicz. How much memory is needed to win infinite games? In *Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science, LICS 1997, Warsaw, Poland, June 29 – July 2, 1997*, pages 99–110. IEEE Computer Society, 1997. doi:10.1109/LICS.1997.614939.

14 Hugo Gimbert and Wieslaw Zielonka. Games where you can play optimally without any memory. In Martín Abadi and Luca de Alfaro, editors, *Proceedings of the 16th International Conference on Concurrency Theory, CONCUR 2005, San Francisco, CA, USA, August 23–26, 2005*, volume 3653 of *Lecture Notes in Computer Science*, pages 428–442. Springer, 2005. doi:10.1007/11539452_33.

15 Abraham Ginzburg and Michael Yoeli. Products of automata and the problem of covering. *Transactions of the American Mathematical Society*, 116:253–266, 1965. URL: http://www.jstor.org/stable/1994117.

16 Yuri Gurevich and Leo Harrington. Trees, automata, and games. In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, STOC 1982, San Francisco, CA, USA, May 5–7, 1982*, pages 60–65. ACM, 1982. doi:10.1145/800070.802177.

17 Alexey Ignatiev, António Morgado, and João Marques-Silva. PySAT: A Python toolkit for prototyping with SAT oracles. In Olaf Beyersdorff and Christoph M. Wintersteiger, editors, *Proceedings of the 21st International Conference on the Theory and Applications of Satisfiability Testing, SAT 2018, Held as Part of FloC 2018, Oxford, UK, July 9–12, 2018*, volume 10929 of *Lecture Notes in Computer Science*, pages 428–437. Springer, 2018. doi:10.1007/978-3-319-94144-8_26.

18 Eryk Kopczyński. *Half-positional Determinacy of Infinite Games*. PhD thesis, Warsaw University, 2008.

19 Stéphane Le Roux, Arno Pauly, and Mickael Randour. Extending finite-memory determinacy by Boolean combination of winning conditions. In Sumit Ganguly and Paritosh K. Pandya, editors, *Proceedings of the 38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, Ahmedabad, India, December 11–13, 2018*, volume 122 of *LIPIcs*, pages 38:1–38:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.FSTTCS.2018.38.

20 Anil Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9(4):541–544, 1958. doi:10.2307/2033204.

21 Pierre Ohlmann. Characterizing positionality in games of infinite duration over infinite graphs. *TheoretiCS*, 2, 2023. doi:10.46298/theoretics.23.3.

22 Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969. doi:10.2307/1995086.

23 Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998. doi:10.1016/S0304-3975(98)00009-7.

Monadic NIP in Monotone Classes of Relational Structures

Samuel Braunfeld
Computer Science Institute of Charles University (IUUK), Prague, Czech Republic

Anuj Dawar
Department of Computer Science and Technology, University of Cambridge, UK

Ioannis Eleftheriadis
Department of Computer Science and Technology, University of Cambridge, UK

Aris Papadopoulos
School of Mathematics, Univesity of Leeds, UK

Abstract

We prove that for any monotone class of finite relational structures, the first-order theory of the class is NIP in the sense of stability theory if, and only if, the collection of Gaifman graphs of structures in this class is nowhere dense. This generalises results previously known for graphs to relational structures and answers an open question posed by Adler and Adler (2014). The result is established by the application of Ramsey-theoretic techniques and shows that the property of being NIP is highly robust for monotone classes. We also show that the model-checking problem for first-order logic is intractable on any monotone class of structures that is not (monadically) NIP. This is a contribution towards the conjecture that the hereditary classes of structures admitting fixed-parameter tractable model-checking are precisely those that are monadically NIP.

2012 ACM Subject Classification Theory of computation → Finite Model Theory; Mathematics of computing → Combinatorics

Keywords and phrases Model theory, finite model theory, structural graph theory, model-checking

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.119

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Funding *Samuel Braunfeld*: Project 21-10775S of the Czech Science Foundation (GAČR), European Union's Horizon 2020 research and innovation programme (grant agreement No 810115 – Dynasnet).
Anuj Dawar: EPSRC grant EP/T007257/1.
Ioannis Eleftheriadis: George and Marie Vergottis Scholarship awarded through Cambridge Trust, Onassis Foundation Scholarship, Robert Sansom Studentship.
Aris Papadopoulos: Leeds Doctoral Scholarship.

Acknowledgements We want to thank the referees for their numerous improvements to the text, and for suggesting the simplified argument in Section 6.

1 Introduction

The development of stability theory in classical model theory, originating with Shelah's classification programme fifty years ago [19, 2], has sought to distinguish *tame* first-order theories from *wild* ones. A key discovery is that combinatorial configurations serve as dividing lines in this classification.

Separately, in the development of finite model theory, there has been in interest in investigating *tame* classes of finite structures. Here tameness can refer to *algorithmic tameness*, meaning that algorithmic problems that are intractable in general may be tractable on a tame class; or it can refer to *model-theoretic tameness*, meaning that the class enjoys

some desirable model-theoretic properties that are absent in the class of all finite structures. See [6] for an exposition of these notions of tameness. The tame classes that arise in this context are often based on notions taken from the study of sparse graphs [15] and usually extended to classes of relational structures beyond graphs by applying them to the *Gaifman graphs* of such structures.

In the context of algorithmic tameness of sparse classes, this line of work culminated in the major result of Grohe et al. [10] showing that the problem of model checking first-order sentences is fixed-parameter tractable (FPT) on any class of graphs that is *nowhere dense*. This generalized a sequence of earlier results showing the tractability of the model checking problem on classes of graphs satisfying other notions of sparsity. Moreover, it is also known [13] that this is the limit of tractability for *monotone* classes of graphs. That is to say that (under reasonable assumptions) any monotone class of graphs in which first-order model checking is FPT is necessarily nowhere dense. These results underline the centrality of the notion of nowhere denseness in the study of sparse graph classes.

A significant line of recent research has sought to generalize the methods and results on tame sparse classes of graphs to more general classes that are not necessarily sparse. Interestingly, this has tied together notions of tameness arising in finite model theory and those in classical model theory. Notions arising from stability theory play an increasingly important role in these considerations (see [16, 8], for example). Central to this connection is the realisation that for well-studied notions of sparseness in graphs, the first-order theory of a sparse class $\mathcal{C}$ is stable. Thus, stability-theoretic notions of tameness, applied to the theory of a class of finite structures, generalize the notions of tameness emerging from the theory of sparsity.

A key result connecting the two directions is that a monotone class of finite graphs is stable if, and only if, it is nowhere dense. This connection between stability and combinatorial sparsity was established in the context of infinite graphs by Podewski and Ziegler [17] and extended to classes of finite graphs by Adler and Adler [1]. Indeed, for monotone classes of graphs, stability is a rather robust concept as the theory of such a class is stable if, and only if, it is NIP (that is, it does not have the independence property) and these conditions on monotone classes are in turn equivalent to it being monadically stable and monadically NIP (these notions are formally defined in Section 2 below).

A question posed by Adler and Adler is whether their result can be extended from graphs to structures in any finite relational language. We settle this question in the present paper by establishing Theorem 1 below. In the following $\mathsf{Gaif}(\mathcal{C})$ (respectively $\mathsf{Inc}(\mathcal{C})$) denotes the collection of Gaifman graphs (resp. incidence graphs) of structures in the class $\mathcal{C}$. Note that the extension from graphs to relational structures requires considerable combinatorial machinery in the form of Ramsey-theoretic results, which we detail in later sections. We also relate the characterization to the tractability of the classes. In summary, our key results are stated in the following theorem. See Section 2 for all the relevant definitions.

▶ **Theorem 1.** *Let $\mathcal{C}$ be a monotone class of finite structures in a finite relational language. Then, the following are equivalent:*

1. *$\mathcal{C}$ is NIP;*
2. *$\mathcal{C}$ is monadically NIP;*
3. *$\mathcal{C}$ is stable;*
4. *$\mathcal{C}$ is monadically stable;*
5. *$\mathsf{Gaif}(\mathcal{C})$ is nowhere dense;*
6. *$\mathsf{Inc}(\mathcal{C})$ is nowhere dense; and*
7. *(assuming $\mathsf{AW}[*] \neq \mathsf{FPT}$) $\mathcal{C}$ admits fixed-parameter tractable model checking.*

Moreover, the equivalence of the first six notions also holds for classes containing infinite structures.

Thus, for monotone classes of relational structures, the picture is clear. Beyond monotone classes, not every NIP class is stable or monadically NIP. However, it has been conjectured [22, 9] that for any hereditary class $\mathcal{C}$ of structures, the model checking problem on $\mathcal{C}$ is fixed-parameter tractable if, and only if, $\mathcal{C}$ is NIP. This has previously been established for monotone classes of graphs (by the results of Adler and Adler, combined with those of Grohe et al.) and for hereditary classes of ordered graphs by results of Bonnet et al.[3]. Our results also extend the classes for which this conjecture is verified to all monotone classes of relational structures.

We establish some necessary definitions and notation in Sections 2 and 3. The proof of Theorem 1 occupies the next two sections. The equivalence of the first four notions for any monotone class $\mathcal{C}$ is due to Braunfeld and Laskowski [4]. The equivalence of the fifth and sixth notions follows by results in sparsity theory (see [15]) which we recall in Section 6. We, therefore, establish the equivalence of the first with the fifth and the seventh. In Section 4 we show that if $\mathsf{Gaif}(\mathcal{C})$ is not nowhere dense, then $\mathcal{C}$ admits a formula with the independence property. That nowhere density of $\mathsf{Gaif}(\mathcal{C})$ implies tractability is implicit in [10]. We establish the converse of this statement in Section 5. Finally, we give an argument that $\mathsf{Gaif}(\mathcal{C})$ being nowhere dense implies monadic stability in Section 6.

2 Preliminaries

We assume familiarity with first-order logic and the basic concepts of model theory. We have tried to make this paper as self-contained as possible, but refer the reader to [11] for background and undefined notation. Throughout this paper, $\mathcal{L}$ denotes a finite, first-order, relational language. We write $\mathrm{ar}(R)$ for the arity of each relation symbol $R \in \mathcal{L}$. Tuples of elements or variables are denoted by overlined letters and given a tuple $\bar{a}$ and $k \leq |\bar{a}|$, we write $\bar{a}(k)$ to denote the k-th element of $\bar{a}$. Often we abuse notation and treat tuples as unordered sets; whether we refer to the ordered tuple or the unordered set should be clear from the context. For $n \in \mathbb{N}$, we write $[n]$ for the set $\{1, \dots, n\}$.

We adopt the convention of allowing finitely many constant symbols (i.e. parameters) in $\mathcal{L}$-formulas. Syntactically, these are to be understood as additional free variables, while semantically these have a fixed interpretation in every $\mathcal{L}$-structure. This is purely a notational convenience and has no effect on the applicability of our results. By a further abuse of notation, we do not distinguish between a parameter p and its interpretation p^M in an $\mathcal{L}$-structure, M.

2.1 Graphs and relational structures

An $\mathcal{L}$-structure is denoted by $(M, R^M)_{R \in \mathcal{L}}$, where M is its underlying set and $R^M \subseteq M^{\mathrm{ar}(R)}$ is the interpretation of the relation symbol $R \in \mathcal{L}$ in M. We write $\mathfrak{C}(\mathcal{L})$ for the class of all $\mathcal{L}$-structures. By abusing notation, often we do not distinguish between an $\mathcal{L}$-structure and its underlying set. For an $\mathcal{L}$-structure M and a subset $A \subseteq M$ we denote by $M[A]$ the substructure of M induced by A, i.e. the structure on domain A with $R^A = R^M \cap A$ for all $R \in \mathcal{L}$. A pointed $\mathcal{L}$-structure is a pair $(M, \bar{m})$ where $\bar{m}$ is a tuple of $|\bar{m}|$ labelled points of M. By the *equality type* of a tuple $\bar{m}$ from an $\mathcal{L}$-structure M, we mean the set $\Delta_=(\bar{m})$ of atomic formulas $\eta(\bar{x})$ using only the equality symbol such that $M \models \eta(\bar{m})$.

A *homomorphism* from an $\mathcal{L}$-structure M to an $\mathcal{L}$-structure N is a map $f : M \to N$ satisfying such that for all relation symbols $R \in \mathcal{L}$ and tuples $\bar{m} \in M^{\mathrm{ar}(R)}$, if $\bar{m} \in R^M$ then $f(\bar{m}) \in R^N$. A homomorphism of pointed structures $f : (M, \bar{m}) \to (N, \bar{n})$ is understood as a homomorphism $f : M \to N$ of the underlying $\mathcal{L}$-structures such that $f(\bar{m}) = \bar{n}$.

By a graph G we mean an $\{E\}$-structure such that $E^G \subseteq G^2$ is a symmetric, irreflexive binary relation. We write $E(G)$ rather than E^G for the edge set of a graph. Given a graph G and $r \in \mathbb{N}$, we write $G^{(r)}$ for the *r-subdivision of G*, i.e. the graph obtained by replacing every edge of G by a path of length $r+1$. We denote by K_n the complete graph on n vertices and by $K_{t,t}$ the complete bipartite graph with parts of size t. We write $G = (U, V; E)$ for a bipartite graph with parts U and V and edge set $E \subseteq U \times V$, and write $\mathfrak{B}$ for the class of all bipartite graphs.

We recall two ways of constructing a graph from a given relational structure M. First, the *Gaifman graph* of M which is the graph on vertex set M, whose edges are precisely the pairs (u, v) such that u and v appear together in a relation of M. Second, the *Incidence graph* of M which is the the bipartite graph with elements of M in one part, all tuples in all relations in the other part, and edges denoting membership of an element u in a tuple $\bar{v}$. More formally:

▶ **Definition 2** (Gaifman/Incidence graph). *Given an $\mathcal{L}$-structure $(M, R^M)_{M\in\mathcal{L}}$ we define the* Gaifman graph *of M, denoted $\mathsf{Gaif}(M)$, to be the graph on vertex set M with edges:*

$$E := \{(x, y) : \exists R \in \mathcal{L} \exists v_1, \ldots, v_{\mathrm{ar}(R)-2} \exists \sigma \in \mathcal{S}_{\mathrm{ar}(R)} (\sigma(x, y, v_1, \ldots, v_{\mathrm{ar}(i)-2}) \in R^M)\},$$

where $\mathcal{S}_n$ the symmetric group on n elements. Moreover, we define the Incidence graph *of M, denoted $\mathsf{Inc}(M)$, to be the bipartite graph $(M, \bigsqcup_{R\in\mathcal{L}} M^R, E')$, where*

$$E' := \{(x, \bar{z}) : x \in \bar{z}\}$$

For a class of relational structures $\mathcal{C}$, all in the same language, we define the Gaifman class *of $\mathcal{C}$ to be $\mathsf{Gaif}(\mathcal{C}) := \{\mathsf{Gaif}(M) : M \in \mathcal{C}\}$. Likewise, we define $\mathsf{Inc}(\mathcal{C}) := \{\mathsf{Inc}(M) : M \in \mathcal{C}\}$.*

2.2 Sparsity and stability

Throughout this paper, $\mathcal{C}$ refers to a class of $\mathcal{L}$-structures or graphs. We write $\mathrm{Th}(\mathcal{C})$ for the common theory of the class, i.e. the set of all first-order $\mathcal{L}$-sentences that hold in all structures in $\mathcal{C}$. We say that a class $\mathcal{C}$ is:

- *hereditary*, if $\mathcal{C}$ is closed under induced substructures, i.e. if $(M, R^M)_{R\in\mathcal{L}} \in \mathcal{C}$ then $(M', R^M \cap M')_{R\in\mathcal{L}} \in \mathcal{C}$ for any $M' \subseteq M$.
- *monotone*, if $\mathcal{C}$ is closed under weak substructures, i.e. if $(M, R^M)_{R\in\mathcal{L}} \in \mathcal{C}$ then $(M', R^{M'})_{R\in\mathcal{L}} \in \mathcal{C}$ for any $M' \subseteq M$ and $R^{M'} \subseteq R^M$.

▶ **Definition 3.** *Let $\mathcal{C}$ be a class of graphs. We say that $\mathcal{C}$ is* nowhere dense *if for every $r \in \mathbb{N}$ there is some $n \in \mathbb{N}$ such that for all $G \in \mathcal{C}$ we have that $K_n^{(r)}$ is not a subgraph of G.*

Nowhere density was introduced by Nešetřil and Ossona de Mendez [14], as a structural property of classes of finite graphs that generalises numerous well-behaved classes, including graphs of bounded degree, planar graphs, graphs excluding a fixed minor and graphs of bounded expansion. Nowhere dense classes play an important role in algorithmic graph theory, as several computationally hard problems become tractable when restricted to such classes.

Let us now recall some core notions of tameness from classification theory, adapted from the context of infinite structures to that of classes of (not necessarily infinite) structures.

▶ **Definition 4** (Order/Independence Property). *Let $\mathcal{C}$ be a class of $\mathcal{L}$-structures. We say that an $\mathcal{L}$-formula $\phi(\bar{x}, \bar{y})$ has:*

1. The Order Property in $\mathcal{C}$ *if for all $n \in \mathbb{N}$ there is some $M_n \in \mathcal{C}$ and sequences $(\bar{a}_i)_{i\in[n]}$ and $(\bar{b}_j)_{j\in[n]}$ of tuples from M_n such that:*

$$M_n \vDash \phi(\bar{a}_i, \bar{b}_j) \text{ if, and only if, } i < j.$$

2. The Independence Property in $\mathcal{C}$ *if for all bipartite graphs* $G = (U, V; E) \in \mathfrak{B}$ *there is some* $M_G \in \mathcal{C}$ *and sequences of tuples* $(\bar{a}_i)_{i \in U}$ *and* $(\bar{b}_j)_{j \in V}$ *such that:*

$$M_G \vDash \phi(\bar{a}_i, \bar{b}_j) \text{ if, and only if, } (i, j) \in E.$$

We say that $\mathcal{C}$ *is* stable *if no formula has the order property in* $\mathcal{C}$*. We say that* $\mathcal{C}$ *is* NIP *(No Independence Property) if no formula has the independence property in* $\mathcal{C}$*.*

An easy application of compactness reveals that a class $\mathcal{C}$ is stable (resp. NIP) if, and only if, all completions of $\mathrm{Th}(\mathcal{C})$ are stable (resp. NIP) in the standard model-theoretic sense (see for instance [20] for the standard model-theoretic definitions).

Given a class $\mathcal{C}$ of $\mathcal{L}$-structures and an expansion $\mathcal{L}' = \mathcal{L} \cup \{P_i : i \in I\}$ by unary predicates, we say that a class $\mathcal{C}'$ of $\mathcal{L}'$-structures is a *monadic expansion* of $\mathcal{C}$ if $\mathcal{C} = \{M' \restriction_{\mathcal{L}} : M' \in \mathcal{C}'\}$, where for an $\mathcal{L}'$-structure M' we write $M' \restriction_{\mathcal{L}}$ for the $\mathcal{L}$*-reduct* of M', i.e. the $\mathcal{L}$-structure obtained from M' by simply forgetting each relation symbol not in $\mathcal{L}$. In other words, $\mathcal{C}'$ is a monadic expansion of $\mathcal{C}$ if, for each structure $M \in \mathcal{C}$, $\mathcal{C}'$ contains at least one copy of M expanded with unary predicates which are interpreted freely, and no other structures.

▶ **Definition 5** (Monadic Stability/NIP)**.** *Let* $\mathcal{C}$ *be a class of* $\mathcal{L}$*-structures. We say that* $\mathcal{C}$ *is* monadically stable *(resp.* monadically NIP*) if all monadic expansions* $\mathcal{C}'$ *of* $\mathcal{C}$ *are stable (resp. NIP).*

The relationship between sparsity and stability is captured by the following theorem, which was established by Podewski and Ziegler [17], in the context of infinite graphs, and much later translated to the context of graph classes by Adler and Adler [1].

▶ **Theorem 6** (Adler, Adler [1]; Podewski, Ziegler [17])**.** *Let* $\mathcal{C}$ *be a nowhere dense class of graphs. Then* $\mathcal{C}$ *is monadically stable. Moreover, the following are equivalent when* $\mathcal{C}$ *is monotone:*

1. $\mathcal{C}$ *is NIP;*
2. $\mathcal{C}$ *is monadically NIP;*
3. $\mathcal{C}$ *is stable;*
4. $\mathcal{C}$ *is monadically stable;*
5. $\mathcal{C}$ *is nowhere dense.*

Furthermore, Adler and Adler asked if Theorem 6 can be generalised to arbitrary relational structures with finite signature. Recently, Braunfeld and Laskowski established a collapsing phenomeon akin to Theorem 6 for relational structures.

▶ **Theorem 7** (Braunfeld, Laskowski, [4])**.** *Let* $\mathcal{C}$ *be a hereditary class of structures. Then* $\mathcal{C}$ *is monadically NIP (resp. monadically stable) if, and only if,* $\mathcal{C}$ *is NIP (resp. stable). Moreover, if* $\mathcal{C}$ *is monotone then* $\mathcal{C}$ *is NIP if, and only if, it is stable.*

In light of the above, Theorem 1 answers the question of Adler and Adler affirmatively by connecting the picture arising in Theorem 7 with the sparsity-theoretic properties of the Gaifman class.

2.3 Model-checking

By model-checking on a class $\mathcal{C}$ we refer to the following parametrised decision problem:

Given: A FO-sentence ϕ and a structure $M \in \mathcal{C}$.

Parameter: $|\phi|$.

Decide: Whether or not M satisfies ϕ.

▶ **Definition 8.** *We say that $\mathcal{C}$ is* tractable*, or that the model-checking problem on a class $\mathcal{C}$ is fixed-parameter tractable, if there is an algorithm that decides on input (M, ϕ) whether $G \models \phi$, in time $f(|\phi|) \cdot |M|^{\mathcal{O}(1)}$ for some computable function f.*

Model-checking on the class of all graphs is complete with respect to the complexity class $\mathsf{AW}[*]$, which is conjectured to strictly contain the class FPT. We shall assume throughout that $\mathsf{AW}[*] \neq \mathsf{FPT}$.

All hereditary classes of graphs and relational structures that are known to admit tractable model-checking are NIP. Moreover, the robustness of NIP in hereditary classes hints at its potential necessity for tractability. This is the basis of the following conjecture:

▶ **Conjecture 9** ([22, 9, 3])**.** *Let $\mathcal{C}$ be a hereditary class of relational structures. Then $\mathcal{C}$ is tractable if, and only if, $\mathcal{C}$ is NIP.*

There is good evidence for a positive answer to this conjecture. Indeed, it is known to hold for:

- Monotone classes of graphs, where NIP coincides with nowhere density [10];
- Hereditary classes of ordered graphs, where NIP coincides with bounded twin-width [21, 3].

Although it is not explicitly stated in this form, a careful examination of the argument of [10] reveals that the following holds.

▶ **Theorem 10** (Grohe, Kreutzer, Siebertz, [10])**.** *Let $\mathcal{C}$ be a class of relational structures such that $\mathsf{Gaif}(\mathcal{C})$ is nowhere dense. Then $\mathcal{C}$ admits fixed-parameter tractable model-checking.*

2.4 Interpretations

Interpretations in classical model theory allow us to find structures in some language in a definable way inside a definable quotient of structures in some other language, mimicking, for instance, the way one can find the rational numbers inside the integers.

In our case, we focus on a restricted version of interpretations, which we call *simple interpretations* (possibly with parameters). Intuitively, a class of $\mathcal{L}'$-structures $\mathcal{D}$ can be interpreted in a class of $\mathcal{L}$-structures $\mathcal{C}$ if there is a uniform way of defining every structure in $\mathcal{D}$, in some (Cartesian power of some) structure from $\mathcal{C}$. More formally:

▶ **Definition 11** (Simple interpretation)**.** *Let $\mathcal{L}, \mathcal{L}'$ be two finite relational languages. A* simple interpretation with parameters $I : \mathfrak{C}(\mathcal{L}) \to \mathfrak{C}(\mathcal{L}')$ *consists of the following data:*

- *A* domain formula $\delta(\bar{x}, \bar{v}) \in \mathcal{L}$ *and, a function d which to each $M \in \mathfrak{C}(\mathcal{L})$ associates a tuple $\bar{d}(M)$ from $M^{|\bar{v}|}$.*
- *For each k-ary relation symbol $R(y_1, \ldots, y_k) \in \mathcal{L}'$ an* interpreting formula $\phi_R(x_1, \ldots, x_k, \bar{v}_R) \in \mathcal{L}$*, where $|\bar{x}_i| = |\bar{x}|$, for each $i \in [k]$, and a function c_R which to each $M \in \mathfrak{C}(\mathcal{L})$ associates a tuple $\bar{c}_R(M)$ from $M^{|\bar{v}_R|}$.*

In order to make our discussion of interpretations easier, we adopt the following notation. Given $M \in \mathfrak{C}(\mathcal{L})$ we write $I(M)$ for the $\mathcal{L}'$ structure on the set $\delta(M) := \{a \in M : M \vDash \delta(a, \bar{d}(M))\}$ with:

$$I(M) \vDash R(a_1, \ldots, a_k) \text{ if, and only if, } M \vDash \phi_R(a_1, \ldots, a_k, \bar{c}_R(M)),$$

for each k-ary relation symbol $R \in \mathcal{L}'$ and $a_1, \ldots, a_k \in \delta(M, \bar{d}(M))$. This dually gives a map $\widehat{I} : \mathcal{L}' \to \mathcal{L}$ mapping $\mathcal{L}'$-formulas to $\mathcal{L}$-formulas with parameters, such that for any $\mathcal{L}'$-sentence ϕ we have that:

$$M \models \widehat{I}(\phi) \text{ if, and only if, } I(M) \models \phi.$$

In order to be able to reduce the problem of FO model-checking from one class of structures to another, possibly in a different language, we are interested in interpretations that can be computed in polynomial time. More precisely we define the following notion:

▶ **Definition 12** (Polynomial interpretation). *Given classes of structures $\mathcal{C} \subseteq \mathfrak{C}(\mathcal{L})$ and $\mathcal{D} \subseteq \mathfrak{C}(\mathcal{L}')$ we say that $\mathcal{D}$ is* polynomially interpreted in $\mathcal{C}$, with parameters, *if there are:*

1. *A simple interpretation with parameters, $I : \mathfrak{C}(\mathcal{L}) \to \mathfrak{C}(\mathcal{L}')$, as in Definition 11, such that the functions d and $(c_R)_{R\in\mathcal{L}'}$ are computable in polynomial time; and*
2. *a polynomial-time computable map $f : \mathcal{D} \to \mathcal{C}$ such that for all $D \in \mathcal{D}$ we have that $D = I(f(D))$.*

In this case, we write $\mathcal{D} \leq_{\mathsf{P}} \mathcal{C}$.

The next lemma justifies why polynomial interpretations are particularly useful.

▶ **Lemma 13.** *The relations $\leq_{\mathsf{P}}$ is a quasi-order on the collection of classes of structures in finite relational languages. Moreover $\leq_{\mathsf{P}}$ preserves tractability, i.e. if $\mathcal{C}$ is tractable and $\mathcal{D} \preceq_{\mathsf{P}} \mathcal{C}$, then $\mathcal{D}$ is tractable.*

Proof. The first part of the lemma is immediate, so let us only discuss the second part. We reduce the problem of model checking in $\mathcal{D}$ to model checking in $\mathcal{C}$. Given an $\mathcal{L}'$-sentence ϕ and an $\mathcal{L}'$-structure $M \in \mathfrak{C}(\mathcal{L}')$, we can compute, by assumption, in polynomial time an $\mathcal{L}$-structure $f(D) \in \mathcal{C}$ such that $M = I(f(D))$. By assumption, we can also compute $I(f(D))$ in polynomial time, since the parameters in the domain and interpreting formulas are computable from M in polynomial time. Then, we have that:

$$f(D) \models \widehat{I}(\phi) \text{ if, and only if, } I(f(D)) = M \models \phi,$$

where $\widehat{I}(\phi)$ is obtained, essentially, as in the discussion after Definition 11, which can clearly be done in polynomial time, from ϕ. Since $\mathcal{C}$ is tractable, it follows that $\mathcal{D}$ is tractable. ◀

2.5 Ramsey Theory

A core technique that is used repeatedly in our arguments is that if a finite structure is large enough, then patterns in it are inevitable. This is the main idea of Ramsey theory, the relevant tools from which we recall here. The notation we use is standard, given a set S and $k \in \mathbb{N}$ we write $[S]^{(k)}$ for the collection of all k-element subsets of S.

▶ **Theorem 14** (Ramsey's Theorem, [18]). *There is a computable function $\mathcal{R} : \mathbb{N}^3 \to \mathbb{N}$ such that for all $m, k, r \in \mathbb{N}$ and for every colouring $\chi : [\mathcal{R}(m, k, r)]^{(k)} \to [r]$ there exists some $S \subseteq [\mathcal{R}(m, k, r)]$ of size m which is monochromatic.*

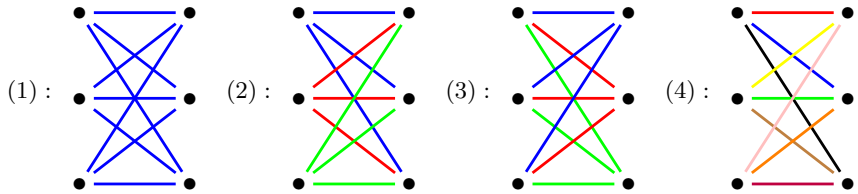
Another standard theorem from Ramsey theory that we make use of is the following well-known variant of Theorem 14:

▶ **Theorem 15** (Bipartite Ramsey Theorem). *There is a computable function $\mathcal{P} : \mathbb{N}^2 \to \mathbb{N}$ such that for all $m, r \in \mathbb{N}$ and all edge colourings of the complete bipartite graph $K_{\mathcal{P}(m,r),\mathcal{P}(m,r)}$ with r colours, there are subsets A, B of the two parts, both of size m, which induce a monochromatic copy of $K_{m,m}$.*

We also need to make use of the following Ramsey-theoretic result, where the number of colours is allowed to be possibly infinite. Of course, in this case, we cannot expect to find monochromatic subsets. Nonetheless, we can ensure that the behaviour of the colouring falls into one of few "canonical" cases on a large enough set. The original canonical Ramsey theorem is due to Erdős and Rado [7], but for the purposes of this paper, we are only interested in the bipartite version in its effective form.

▶ **Theorem 16** (Bipartite Canonical Ramsey Theorem, [12]). *There is a computable function $\mathcal{K} : \mathbb{N} \to \mathbb{N}$ such that for every $n \in \mathbb{N}$ and every edge-colouring of the complete bipartite graph $K_{\mathcal{K}(n),\mathcal{K}(n)}$ there exist subsets X,Y of the two parts, both of size n, such that one of the following occurs for all $x, x' \in X$ and $y, y' \in Y$:*

1. $\chi(x, y) = \chi(x', y')$;
2. $\chi(x, y) = \chi(x', y')$ *if, and only if,* $x = x'$;
3. $\chi(x, y) = \chi(x', y')$ *if, and only if,* $y = y'$;
4. $\chi(x, y) = \chi(x', y')$ *if, and only if,* $x = x'$ *and* $y = y'$.



Henceforth, we shall say that an edge colouring of a complete bipartite graph is *canonical of type* 1 (resp. $2, 3, 4$) if it satisfies condition 1 (resp. $2, 3, 4$) from Theorem 16 for all edges. More generally, we say that such a colouring is *canonical* whenever it is canonical of any type.

3 Path formulas

Recall that a formula $\phi(\bar{x})$ is called *primitive positive* if it has the form $\exists \bar{y} \psi(\bar{x}, \bar{y})$, where ψ is a conjunction of atomic formulas. Primitive positive formulas are also known as *conjunctive queries* in the database theory literature. The following association of a canonical structure with a primitive positive formula and conversely a canonical such formula with a finite structure goes back to Chandra and Merlin [5].

▶ **Definition 17** (Canonical structures). *Given a primitive positive formula $\phi(\bar{x}) = \exists \bar{y} \psi(\bar{x}, \bar{y})$ we define a pointed $\mathcal{L}$-structure $(\mathcal{M}_\phi, \bar{x})$ whose domain is the set $\{v_1, \ldots, v_r\}$ of variables of ϕ, and where each $R \in \mathcal{L}$ is interpreted as follows:*

$$\mathcal{M}_\phi \vDash R(v_1, \ldots, v_n) \text{ if, and only if, } R(v_1, \ldots, v_n) \text{ appears as a conjunct in } \psi(\bar{x}, \bar{y}).$$

The pointed elements $\bar{x}$ precisely correspond to the free variables of ϕ. This structure is unique, up to isomorphism, and we call it the canonical structure of ϕ.

Similarly, for every pointed $\mathcal{L}$-structure $(A, \bar{x})$ we may associate a primitive positive formula $\phi_A(\bar{x})$ so that $(\mathcal{M}_{\phi_A}, \bar{x}) = (A, \bar{x})$. We call this formula the *canonical formula* of $(A, \bar{x})$. Let $\phi(\bar{x})$ be a primitive positive formula and $(\mathcal{M}_\phi, \bar{x})$ its canonical structure. It is easy to see that for any $\mathcal{L}$-structure A and $\bar{a} \in A$ we have that $A \vDash \phi(\bar{a})$ if, and only if, there exists a homomorphism (of pointed structures) $h : (\mathcal{M}_\phi, \bar{x}) \to (A, \bar{a})$.

In our analysis, we argue that whenever a monotone class of relational structures has the independence property then this is witnessed by a certain kind of primitive positive formula. In the case of graphs, it is implicit in the work of Adler and Adler that the canonical structure of this primitive positive formula is a path in the standard graph-theoretic sense, i.e. a tuple $(x_1, \ldots, x_n)$ of pairwise distinct elements such that $E(x_i, x_{i+1})$ for all $i \in [n-1]$.

In this section, we introduce the analogue of (graph) paths that witnesses the independence property in general relational structures. We start with the following rather technical definition.

▶ **Definition 18** (Path). *By a* path of length n, *we mean an* $\mathcal{L}$*-structure* $\mathbf{P}$ *consisting of a sequence of pairwise disjoint tuples each consisting of pairwise different elements* $\bar{e}_1, \dots, \bar{e}_n$ *such that:*

- $\mathbf{P} = \bigcup_{i\in[n]} \bar{e}_i$*;*
- $|\bar{e}_i \cap \bar{e}_{i+1}| = 1$*, for all* $i < n$*;*
- $\bar{e}_i \not\subseteq \bar{e}_{i+1}$ *and* $\bar{e}_{i+1} \not\subseteq \bar{e}_i$*, for all* $i < n$*;*
- $\bar{e}_i \cap \bar{e}_j = \emptyset$*, for all* $j \in [n] \setminus \{i-1, i, i+1\}$*;*
- $R_i(\bar{e}_i)$*, for exactly one relation symbol* $R_i \in \mathcal{L}$*;*
- $R(\bar{a}) \implies \bar{a} = \bar{e}_i$ *for some* $i \in [n]$*, for all relation symbols* $R \in \mathcal{L}$ *and all tuples* $\bar{a} \in \mathbf{P}$*.*

We write $S(\boldsymbol{P}) = \bar{e}_1 \setminus \bar{e}_2$ *and call these the starting vertices, while we write* $F(\mathbf{P}) = \bar{e}_n \setminus \bar{e}_{n-1}$ *and call these the finishing vertices. We refer to the tuples* $\bar{e}_i$ *as the steps of the path, and to the singletons in* $\bar{e}_i \cap \bar{e}_{i+1}$ *as the joints of the path.*

Given a primitive positive formula $\phi(\bar{x}, \bar{y}, \bar{z})$ (where $\bar{z}$ is possibly empty), we say that ϕ is a *path formula* if there are $x_0 \in \bar{x}$ and $y_0 \in \bar{y}$ such that $\mathcal{M}_\phi$ is a path with $x_0 \in S(\mathcal{M}_\phi)$ and $y_0 \in F(\mathcal{M}_\phi)$. Similarly, we call ϕ a *simple path formula* if $\bar{x} \subseteq S(\mathcal{M}_\phi)$ and $\bar{y} \subseteq F(\mathcal{M}_\phi)$.

Note that technically, no graph G can be a path under the above definition. Indeed, the last condition ensures that $E(G)$ cannot be symmetric as no permutation of a tuple appearing in a relation R can appear in any other relation from $\mathcal{L}$. To avoid confusion, we always refer to paths in the standard graph-theoretic sense as *graph paths*.

Intuitively, a path formula $\phi(\bar{x}, \bar{y})$ plays the role of a higher arity graph path from $\bar{x}$ to $\bar{y}$. However, under enough symmetry, it is possible that we cannot definably tell the direction of ϕ, i.e. $\bar{x}$ and $\bar{y}$ look the same within ϕ. This is formalised in the following definition, and is important in the proof of Theorem 27.

▶ **Definition 19** (Symmetric path). *A* symmetric path *is a path* $\mathbf{P}$ *of length* n*, such that* $R_i = R_{n+1-i}$ *for all* $i \in [n]$*. A* symmetric path formula $\phi(\bar{x}, \bar{y}, \bar{z})$ *is a simple path formula with* $|\bar{x}| = |\bar{y}| = m$ *such that* $\mathcal{M}_\phi$ *is a symmetric path and there is an automorphism* f *of* $\mathcal{M}_\phi$ *which maps* $\bar{x} = (x_1, \dots, x_m) \mapsto (y_{\sigma(1)}, \dots, y_{\sigma(m)})$ *and* $\bar{y} = (y_1, \dots, y_m) \mapsto (x_{\sigma^{-1}(1)}, \dots, x_{\sigma^{-1}(m)})$*, for some* $\sigma \in \mathcal{S}_m$ *which is not the identity permutation. Moreover, if* ϕ *contains parameters then these must be fixed by* f*.*

Given an $\mathcal{L}$-structure and a graph path in $\mathsf{Gaif}(M)$, we may produce a path formula that describes a "type" for this path. This idea is captured by the following definition which is relevant for the proof of Lemma 21.

▶ **Definition 20** (Path type). *Let* M *be an* $\mathcal{L}$*-structure, and* $S = (u_1, \dots, u_n)$ *a graph path in* $\mathsf{Gaif}(M)$*. For every* $i \in [n-1]$ *we may associate a relation symbol* $R_i \in \mathcal{L}$*, elements* $v_{i,1}, \dots, v_{i,\mathrm{ar}(R_i)}$*, and a permutation* $\sigma_i \in \mathcal{S}_{\mathrm{ar}(R_i)}$ *such that* $\sigma_i(u_i, u_{i+1}, \bar{v}_i) \in R_i^M$*. Then we call the formula*

$$\phi(x, y, z_2, \dots, z_{n-1}) = \\ \exists \bar{v}_i \dots \bar{v}_{n-1}(R_1(\sigma_1(x, z_2, \bar{v}_1)) \wedge R_2(\sigma_2(z_2, z_3, \bar{v}_2)) \wedge \dots \wedge R_{n-1}(\sigma_{n-1}(z_{n-1}, y, \bar{v}_{n-1})))$$

a path type *for the graph path* $u_1, \dots, u_n$*.*

It is easy to see that whenever $S = (u_1, \dots, u_n)$ is a graph path in $\mathsf{Gaif}(M)$, then there is a path type ϕ for S such that $M \models \phi(u_1, u_n, u_2, \dots, u_{n_1})$. Clearly, this is not uniquely determined by S, as for the same graph path $u_1, \dots, u_n$ in $\mathsf{Gaif}(M)$ we can possibly obtain different sequences of relations $R_i, \dots R_{i-1}$ and permutations $\sigma_1, \dots, \sigma_{i-1}$ as in Definition 20.

4 From somewhere density to IP

The main result in this section is Theorem 23, where we prove that for any monotone class $\mathcal{C}$ of relational structures whose Gaifman class is somewhere dense, there is a path formula (in the sense of Definition 18) which codes the edge relation of all bipartite graphs uniformly over $\mathcal{C}$.

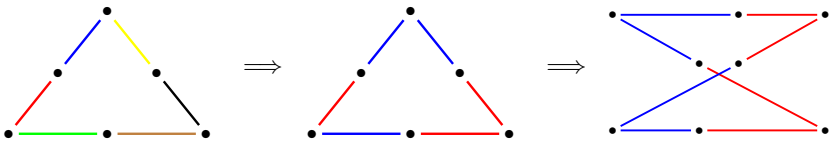
We work towards this theorem via two preparatory lemmas, which have the benefit of applying to classes that are not necessarily monotone. Intuitively, Lemma 21 tells us that if $\mathcal{C}$ is a monotone class of relational structure whose Gaifman class is somewhere dense, then we can find a path formula that codes the edge relation of all finite complete bipartite graphs in $\mathcal{C}$.

▶ **Lemma 21.** *Let $\mathcal{C}$ be a class of $\mathcal{L}$-structures such that $\mathsf{Gaif}(\mathcal{C})$ is somewhere dense. Then there is a path formula $\phi(x, y, \bar{z}) = \exists \bar{w} \psi(x, y, \bar{z}, \bar{w})$ of length ≥ 2 whose joints are precisely the variables in $\bar{z}$, and for each $n \in \mathbb{N}$ there is some $M_n \in \mathcal{C}$ and pairwise distinct elements $(a_i)_{i\in[n]}, (b_j)_{j\in[n]}, (\bar{c}_{i,j})_{(i,j)\in[n]^2}$ from M_n such that*

$$M_n \models \phi(a_i, b_j, \bar{c}_{i,j}), \text{ for all } i, j \in [n].$$

Proof. If $\mathsf{Gaif}(\mathcal{C})$ is somewhere dense, then there exists $r \in \mathbb{N}$ such that for all $n \in \mathbb{N}$ there is some $M_n \in \mathsf{Gaif}(\mathcal{C})$ with $K_n^{(r)} \leq \mathsf{Gaif}(M_n)$. Without loss of generality, we may assume that $r \geq 1$. Indeed, if $r = 0$ then $K_n^1 \leq K_{n^2} \leq \mathsf{Gaif}(M_{n^2})$ so we may pass to a subsequence of $(M_n)_{n\in\mathbb{N}}$ and relabel the indices appropriately.

For every $i < j$ from $[n]$ let $S^n_{i,j}$ be the graph path in $\mathsf{Gaif}(M_n)$ corresponding to the r-subdivision of the edge (i, j) from K_n, directed from i to j. Let $q \in \mathbb{N}$ be the maximum arity of a relation symbol $R \in \mathcal{L}$. Observe that there are at most $p = (|\mathcal{L}| \times q!)^{r+1}$ path types for each graph path $S^n_{i,j}$. By Ramsey's theorem we may find for each n some $\Sigma_n \subseteq [\mathcal{R}(n, 2, q)]$ of size n such that $S^{\mathcal{R}(n,2,q)}_{i,j}$ have the same path type for all $i < j$ from Σ_n. By passing to a subsequence of $(M_n)_{n\in\mathbb{N}}$ and relabelling indices, we may therefore assume that all the $S^n_{i,j}$ have the same path type. Let this be ϕ_n. Since there are only finitely many possible path types for every n, we may prune the sequence $(M_n)_{n\in\mathbb{N}}$ once again to ensure that the same path type $\phi(x, y, \bar{z})$ is obtained for all $n \in \mathbb{N}$. By definition, the joints of $\mathcal{M}_\phi$ are precisely the variables in $\bar{z}$, while $\mathcal{M}_\phi$ has length ≥ 2 since $r \geq 1$.



Work in M_{2n} and let $(a_i)_{i\in[n]}$ be the elements corresponding to $1, \ldots, n$ from $K^{(r)}_{2n}$, and $(b_j)_{j\in[n]}$ be those corresponding to $n + 1, \ldots, 2n$. Moreover, let $\bar{c}_{i,j}$ be the tuples obtained by removing a_i and b_j from the beginning and end respectively of the graph path $S^{2n}_{i,n+j}$. It is clear that the elements $(a_i)_{i\in[n]}, (b_j)_{j\in[n]}, (\bar{c}_{i,j})_{i,j\in[n]}$ are pairwise distinct. Since the path type of $S^{2n}_{i,n+j}$ is equal to ϕ for all $i, j \in [n]$, it follows that

$$M_{2n} \models \phi(a_i, b_j, \bar{c}_{i,j}), \text{ for all } i, j \in [n].$$

We finally pass to the subsequence $(M_{2n})_{n\in\mathbb{N}}$ and relabel. ◀

Having established that we may encode the edge relation of any complete bipartite graph, we want to use monotonicity in order to encode the edge relation of arbitrary bipartite graphs, and consequently, to witness the independence property. To achieve this, we must

ensure that the tuples used in the encoding are "sufficiently disjoint" so that the removal of the desired relations does in fact translate to the removal of an encoded edge. The following lemma is a step toward this.

▶ **Lemma 22.** *Let $\mathcal{C}$ be a class of $\mathcal{L}$-structures such that $\mathsf{Gaif}(\mathcal{C})$ is somewhere dense. Then there is a path formula $\phi(\bar{x},\bar{y},\bar{z}) = \exists\bar{w}\psi(\bar{x},\bar{y},\bar{z},\bar{w})$ of length ≥ 2 with parameters $\bar{p}$ whose joints are precisely the elements of $\bar{z}$, and for every $n \in \mathbb{N}$ there is some $M_n \in \mathcal{C}$ and tuples $(\bar{a}_i)_{i\in[n]}$, $(\bar{b}_j)_{j\in[n]}$, $(\bar{c}_{i,j})_{(i,j)\in[n]^2}$, $(\bar{d}_{i,j})_{i,j\in[n]^2}$ from M_n such that the following hold for all $i,i',j,j' \in [n]$:*

1. *$M_n \models \psi(\bar{a}_i,\bar{b}_j,\bar{c}_{i,j},\bar{d}_{i,j})$;*
2. *$\bar{a}_i(k) \neq \bar{a}_{i'}(k)$, for $i \neq i'$ and all $k \in [|\bar{x}|]$;*
3. *$\bar{b}_j(k) \neq \bar{b}_{j'}(k)$, for $j \neq j'$ and all $k \in [|\bar{y}|]$;*
4. *$\bar{c}_{i,j}(k) \neq \bar{c}_{i',j'}(k)$ and $\bar{c}_{i,j}(k) \neq \bar{c}_{i,j}(l)$, for $(i,j) \neq (i',j')$ and all $k \neq l$ from $[|\bar{z}|]$;*
5. *$\bar{d}_{i,j}(k) \neq \bar{d}_{i',j'}(k)$, for $(i,j) \neq (i',j')$ and all $k \in [|\bar{w}|]$.*

Proof. Let $\phi(x,y,\bar{z}) = \exists\bar{w}\psi(x,y,\bar{z},\bar{w})$ and $(M_n)_{n\in\mathbb{N}}$ be as in Lemma 21. For clarity, we write $(a^n_i)_{i\in[n]}, (b^n_j)_{j\in[n]}, (\bar{c}^n_{i,j})_{(i,j)\in[n]^2}$ to denote the elements of M_n from the same lemma. For each $n \in \mathbb{N}$, and for each pair $(i,j) \in [n]^2$, pick a tuple $\bar{d}^n_{i,j}$ of elements from M_n consisting of some arbitrarily fixed existential witnesses to $M_n \models \phi(a^n_i, b^n_j, \bar{c}^n_{i,j})$, i.e. $M_n \models \psi(a^n_i, b^n_j, \bar{c}^n_{i,j}, \bar{d}^n_{i,j})$ for all $i,j \in [n]$.

Let $m = |\bar{d}_{i,j}|$. By m applications of Theorem 16, we may assume that whether $\bar{d}^n_{i,j}(k) = \bar{d}^n_{i',j'}(k)$ depends on one of the four canonical cases from that theorem, and not on n. Indeed, for every $n \in \mathbb{N}$ and each $k \in [m]$, define colourings $\chi_{n,k}(i,j) = \bar{d}^n_{i,j}(k)$ of the edges of $K_{n,n}$. Let $\mathcal{K} : \mathbb{N} \to \mathbb{N}$ be the computable function guaranteed by Theorem 16 and write $\mathcal{K}^m$ for the composition of $\mathcal{K}$ with itself m times. It follows that the complete bipartite graph with parts of size $\mathcal{K}^m(n)$ contains subsets A_n, B_n of the two parts of size n, which induce a copy of $K_{n,n}$ on which $\chi_{\mathcal{K}^m(n),k}$ is canonical for all $k \in [m]$. We may thus restrict the argument on the subsequence $(M_{\mathcal{K}^m(n)})_{n\in\mathbb{N}}$ and the elements $a_i^{\mathcal{K}^m(n)}, b_j^{\mathcal{K}^m(n)}, \bar{c}_{i,j}^{\mathcal{K}^m(n)}, \bar{d}_{i,j}^{\mathcal{K}^m(n)}$ for $i \in A_n$ and $j \in B_n$ and relabel appropriately. For every $n \in \mathbb{N}$, after the relabelling, we have thus obtained a tuple $\bar{t}_n \in [4]^m$ such that $\chi_{n,k}$ is canonical of type $\bar{t}_n(k)$. Since there are only finitely many such $\bar{t}_n$, by the pigeonhole principle we may consider a subsequence of $(M_n)_{n\in\mathbb{N}}$ for which $\bar{t}_n$ is constant and equal to some $\bar{t} \in [4]^m$, and relabel once more.

We now proceed to sequentially remove elements from the tuples $\bar{d}^n_{i,j}$, and to either name them by a parameter, or to append them to one of a^n_i or b^n_j. Since $\bar{t}$ is constant for all n, exactly the same process is carried out to all tuples $\bar{d}_{i,j}$, and so we may concurrently move the corresponding variables from ϕ. So, if we fall into Case 1 for some k, i.e. if $\bar{t}(k) = 1$, then $\bar{d}_{i,j}(k)$ is the same for all i,j, and so we may name it by a parameter and remove it from every $\bar{h}_{i,j}$. If we fall into Case 2, then $\bar{d}_{i,j}(k) = \bar{d}_{i',j'}(k)$ if, and only if, $i = i'$. Then, for every $i \in [n]$ we may remove the common element $\bar{d}_{i,j}(k)$ from each $\bar{d}_{i,j}$ and append it to a_i, turning it into a tuple $\bar{a}_i$. We then adjust ϕ accordingly by shifting the corresponding variable v_k from $\bar{v}$ to x, which also becomes a tuple $\bar{x}$. Case 3 is symmetric to Case 2, only now we append $\bar{d}_{i,j}(k)$ to $\bar{b}_j$ and shift the variable v_k to $\bar{y}$. We may therefore assume that we fall into Case 4 for all the remaining $k \in [m]$.

We argue that the resulting formula and tuples satisfy the requirements of the lemma. Clearly, $M_n \models \phi(\bar{a}_i,\bar{b}_j,\bar{c}_{i,j},\bar{d}_{i,j})$ for all $n \in \mathbb{N}$ and $i,j \in [n]$. Condition 2 is also satisfied, since the original singletons $(a_i)_{i\in[n]}$ were pairwise disjoint, while for every $i \neq i'$ and $k \in [m]$ the elements $\bar{d}_{i,j}(k)$ and $\bar{d}_{i',j}(k)$, appended to a_i and $a_{i'}$ respectively, come from an instance of Case 2, and are therefore pairwise distinct. Likewise, condition 3 is satisfied. Since we have not interfered with the tuples $\bar{c}_{i,j}$ in the above process and these contain pairwise distinct elements by Lemma 21, Condition 4 is also satisfied. Finally, Condition 5 is trivially satisfied since the elements remaining in $\bar{d}_{i,j}$ fall into Case 4. ◀

▶ **Theorem 23.** *Let $\mathcal{C}$ be a monotone class of $\mathcal{L}$-structures such that $\mathsf{Gaif}(\mathcal{C})$ is somewhere dense. Then there is a path formula $\phi(\bar{x},\bar{y}) = \exists \bar{w}\psi(\bar{x},\bar{y},\bar{w})$ with parameters $\bar{p}$ and for each bipartite graph $G = (U,V;E) \in \mathfrak{B}$ there is some $M_G \in \mathcal{C}$ and sequences of tuples $(\bar{a}_u)_{u\in U}$ $(\bar{b}_v)_{v\in V}$, $(\bar{h}_{u,v})_{(u,v)\in E}$ from M_G such that:*

1. *$M_G \models \phi(\bar{a}_u,\bar{b}_v)$ if, and only if, $(u,v) \in E$ (so, in particular $\mathcal{C}$ is not NIP);*
2. *If $(u,v) \in E$ then $M_G \models \psi(\bar{a}_u,\bar{b}_v,\bar{h}_{u,v})$;*
3. *The equality type of $\bar{p}_{u,v} = \bar{a}_u^\frown \bar{b}_v^\frown \bar{h}_{u,v}$ is constant for all $(u,v) \in E(G)$;*
4. *Any two tuples in $\{\bar{a}_u,\bar{b}_v,\bar{h}_{u,v} : u \in U, v \in V\}$ are disjoint and do not intersect the parameters $\bar{p}$.*

Proof. Let $\phi(x,y,\bar{z}) = \exists \bar{w}\psi(x,y,\bar{z},\bar{w})$, with parameters $\bar{p}$, and $(M_n)_{n\in\mathbb{N}}$ be as in Lemma 22. For clarity, we again write $(a_i^n)_{i\in[n]}, (b_j^n)_{j\in[n]}, (\bar{c}_{i,j}^n)_{(i,j)\in[n]^2}$ to denote the elements from that lemma coming from M_n. Consider the tuples $\bar{p}_{i,j}^n = \bar{a}_i^n \frown \bar{b}_j^n \frown \bar{c}_{i,j}^n \frown \bar{d}_{i,j}^n$, and let $q = |\bar{p}_{i,j}^n|$. Observe that for every $n \in \mathbb{N}$, at most $q \cdot |\bar{p}|$ many tuples $\bar{p}_{i,j}^n$ intersect the parameters $\bar{p}$ because of the conditions in Lemma 22. By working with suitably large n and avoiding these tuples, we may relabel so that no $\bar{p}_{i,j}^n$ intersects $\bar{p}$.

For $i,j,k,l \in [n]$, we say that the tuples $\bar{p}_{i,j}^n$ and $\bar{p}_{k,l}^n$ *intersect trivially* whenever

$$\bar{p}_{i,j} \cap \bar{p}_{k,l} = \begin{cases} \bar{p}_{i,j}, & \text{if } i = k \wedge j = l \\ \bar{a}_i, & \text{if } i = k \wedge j \neq l \\ \bar{b}_j, & \text{if } i \neq k \wedge j = l \\ \emptyset, & \text{otherwise.} \end{cases}$$

Letting $f(n) = q \cdot (n-1)^2 + n$, we claim that for all $n \in \mathbb{N}$ and all $m \geq f(n)$ we may find a set $A_n \subseteq [f(n)]$ of size n so that $\bar{p}_{i,j}^m$ and $\bar{p}_{k,l}^m$ intersect trivially for all $i,j,k,l \in A_n$.

We show this by induction. Indeed, for $n = 1$ this is trivially true as $A_1 = [1]$ works for all $m \geq 1$. Suppose that the claim holds for $n-1$ and fix $m \geq f(n)$. Since $f(n) \geq f(n-1)$, by the induction hypothesis there is some $A_{n-1} \subseteq [f(n-1)] \subseteq [f(n)]$ of size $n-1$ so that $\bar{p}_{i,j}^m$ and $\bar{p}_{k,l}^m$ intersect trivially for all $i,j,k,l \in A_{n-1}$. Notice, that because of Lemma 22, for every fixed $\bar{p}_{i,j}$, there are at most q tuples $\bar{p}_{k,l}$ that do not intersect trivially with it. Hence, there are at most $q \cdot (n-1)^2$ elements $l \in [f(n)]$ such that $\bar{p}_{i,j}^m$ and $\bar{p}_{k,l}^m$ do not intersect trivially for all $i,j,k \in A_{n-1}$. Since $[f(n)]$ contains an additional n elements, we are guaranteed to find some $l \in [f(n)]$, which is not one of the $n-1$ elements of A_{n-1}, such that $\bar{p}_{i,j}^m$ and $\bar{p}_{k,l}^m$ intersect trivially for all $i,j,k \in A_{n-1}$. We may therefore let $A_n = A_{n-1} \cup \{l\}$.

Hence, we may consider the subsequence $(M_{f(n)})_{n\in\mathbb{N}}$ and relabel the tuples appropriately, so that all tuples $\bar{p}_{i,j}^n, \bar{p}_{k,l}^n$ intersect trivially for all $n \in \mathbb{N}$ and $i,j,k,l \in [n]$. Furthermore, by an application of Theorem 15, we may assume that the tuples $\bar{p}_{i,j}^n$ have the same equality type for all $i,j \in [n]$ and all $n \in \mathbb{N}$. More precisely, for every pair $(i,j) \in [n]^2$ let $\Delta_n(i,j) := \Delta_=(\bar{p}_{i,j}^n)$. Letting $q = |\bar{p}_{i,j}|$, it is easy to see that there are at most $p = 2^{q^2}$ sets $\Delta_n(i,j)$. It follows by Theorem 15, that there are subsets A, B of $[\mathcal{P}(n,2,p)]$ of size n such that $\Delta_{\mathcal{P}(n,2,p)}(i,j)$ is constant for all $i \in A$, $j \in B$. Hence, we may relabel appropriately so that $\Delta_n(i,j)$ is constant for all $i,j \in [n]$. Since there are only finitely many such sets, the pigeonhole principle implies that we may prune the sequence $(M_n)_{n\in\mathbb{N}}$ so that $\Delta_n(i,j)$ is uniformly constant for all $n \in \mathbb{N}$.

It follows that no tuple $\bar{a}_i^n$ can intersect a tuple $\bar{b}_j^n$. Indeed, since the equality types are constant, and in particular $\Delta_n(i,j) = \Delta_n(i,j')$, if $\bar{a}_i^n$ and $\bar{b}_j^n$ had an element in common then $\bar{b}_j^n(k) = \bar{b}_{j'}^n(k)$ for some k and all $j' \neq j$, contradicting the assumptions of Lemma 22. Likewise, no tuple $\bar{h}_{i,j}^n = \bar{c}_{i,j}^n \frown \bar{d}_{i,j}^n$ can intersect the tuples $\bar{a}_i$ or $\bar{b}_j$. Since the tuples $\bar{p}_{i,j}^n$ intersect trivially, this implies that any two tuples $\{\bar{a}_i^n, \bar{b}_j^n, \bar{h}_{i,j}^n : i,j \in A_n\}$ are pairwise disjoint, and furthermore do not intersect the parameters $\bar{p}$.

For every $n \in \mathbb{N}$, consider the weak substructure $M'_n \leq M_n$ consisting of the elements in $\bar{p}^n_{i,j}$ and the parameters $\bar{p}$, and containing solely the relations necessary to witness $M_n \models \psi(\bar{a}^n_i, \bar{b}^n_j, \bar{h}^n_{i,j})$. By monotonicity, $M'_n \in \mathcal{C}$. Notice that every tuple appearing in a relation of M'_n contains at least one element of $\bar{c}^n_{i,j}$ for some $i, j \in [n]$. Indeed, the elements of $\bar{c}_{i,j}$ correspond precisely to the joints of the paths $\phi(\bar{a}_i, \bar{b}_j)$, and since $\mathcal{M}_\phi$ has length ≥ 2 every path has at least one joint.

Finally, given $G = (U, V; E)$ with $U = V = [n]$, let $M_G \in \mathcal{C}$ be the induced substructure of M'_n obtained by removing $\bar{h}^n_{i,j}$ for all $(i, j) \notin E$. Since the tuples in $\{\bar{a}^n_i, \bar{b}^n_j, \bar{h}^n_{i,j} : i, j \in [n]\}$ are pairwise disjoint, it follows that $\bar{h}^n_{i,j} \in M_G$ for $(i, j) \in E(G)$. Hence, letting $\phi'(\bar{x}, \bar{y}) = \exists \bar{z} \phi(\bar{x}, \bar{y}, \bar{z})$, we see that $M_G \models \phi'(\bar{a}^n_i, \bar{b}^n_j)$ for all $(i, j) \in E(G)$. Moreover, $M_G \models \neg\phi(\bar{a}_i, \bar{b}_j)$ for $(i, j) \notin E(G)$. Indeed, since the elements of $\bar{c}^n_{i,j}$ are not in M_G for $(i, j) \notin E(G)$, the above observation implies that $M_G \models \neg\phi(\bar{a}_i, \bar{b}_j)$. ◀

Note that all of the above can be proved by working with an appropriate infinite model of $\mathrm{Th}(\mathcal{C})$ obtained by compactness, and applying the infinite versions of the different Ramsey theorems. We have chosen to give a finitistic proof, which is admittedly more involved, so that everything is carried out effectively. Therefore, if we assume that the *VC-dimension* of formulas in the class is computable, we may compute given r the maximum size of an r-subdivided clique occurring in the Gaifman graph of a structure in $\mathcal{C}$.

▶ **Definition 24.** *We say that a class $\mathcal{C}$ of structures is effectively NIP if there is a computable function $f : \mathbb{N} \to \mathbb{N}$ such that for all formulas $\phi(\bar{x}, \bar{y})$ and all structures $M \in \mathcal{C}$ there is no $n > f(|\phi|)$ and $(\bar{a}_i)_{i \in [n]}, (\bar{b}_J)_{J \subseteq [n]}$ with*

$$M \models \phi(\bar{a}_i, \bar{b}_J) \iff i \in J.$$

Recall that a class of graphs $\mathcal{C}$ is called *effectively nowhere dense* whenever there is a computable function $f : \mathbb{N} \to \mathbb{N}$ such that for all $r \in \mathbb{N}$ and for all $G \in \mathcal{C}$ we have that $K^{(r)}_{f(r)}$ is not a subgraph of G.

▶ **Corollary 25.** *Let $\mathcal{C}$ be a monotone and (monadically) NIP class of $\mathcal{L}$-structures in a finite relational language. Then $\mathsf{Gaif}(\mathcal{C})$ is nowhere dense. Moreover, if $\mathcal{C}$ is effectively NIP then $\mathsf{Gaif}(\mathcal{C})$ is effectively nowhere dense.*

5 Intractability

In this section, we prove that any monotone class of relational structures whose Gaifman class is somewhere dense polynomially interprets the class of all bipartite graphs, and is therefore intractable. Towards this, we first strengthen Theorem 23 to obtain a simple path formula ϕ as well as a computable function $\Phi : \mathfrak{B} \to \mathcal{C}$ such that ϕ codes the edge relation of G in $\Phi(G)$.

▶ **Lemma 26.** *Let $\mathcal{C}$ be a monotone class of $\mathcal{L}$-structures such that $\mathsf{Gaif}(\mathcal{C})$ is somewhere dense. Then there is a simple path formula $\phi(\bar{x}, \bar{y})$ with parameters $\bar{p}$ and a polynomial time computable function $\Phi : \mathfrak{B} \to \mathcal{C}$, such that for each bipartite graph $G = (U, V; E) \in \mathfrak{B}$ there are tuples $(\bar{a}_u)_{u \in U}$ $(\bar{b}_v)_{v \in V}$, $(\bar{h}_{u,v})_{(u,v) \in E}$ from $\Phi(G)$ satisfying:*

$$\Phi(G) \models \phi(\bar{a}_u, \bar{b}_v) \text{ if, and only if, } (u, v) \in E.$$

Given ϕ, the interpretation of the parameters $\bar{p}$ in $\Phi(G)$ can be computed in constant time from $G \in \mathfrak{B}$.

Proof. Let ϕ and $(M_G)_{G\in\mathfrak{B}}$ be as in Theorem 23. Consider the path $\mathcal{M}_\phi$. Observe that either there is a step $\bar{e}_i$ such that both $\bar{e}_i \cap \bar{x} = \bar{x}' \neq \emptyset$ and $\bar{e}_i \cap \bar{y} = \bar{y}' \neq \emptyset$, or there are $i < j$ and steps $\bar{e}_i$, $\bar{e}_j$ such that $\bar{e}_i \cap \bar{y} = \emptyset, \bar{e}_j \cap \bar{x} = \emptyset$ and $\bar{e}_i \cap \bar{x} = \bar{x}' \neq \emptyset, \bar{e}_j \cap \bar{y} = \bar{y}' \neq \emptyset$ and for all $k \in \{i+1, \dots, j-1\}$ we have that $\bar{e}_k \cap \bar{x} = \bar{e}_k \cap \bar{y} = \emptyset$. Consider the induced substructure $\mathcal{M}'$ of $\mathcal{M}_\phi$ consisting solely of the step $\bar{e}_i$ in the first case or the steps $\bar{e}_i, \dots, \bar{e}_j$ in the second, and let $\phi'(\bar{x}', \bar{y}') = \exists \bar{w}' \psi'(\bar{x}', \bar{y}', \bar{w}')$ be the canonical formula of $(\mathcal{M}', \bar{x}', \bar{y}')$. Clearly, ϕ' is a simple path formula, and it follows by construction that for each $G \in \mathfrak{B}$ we may pick minimal subtuples $\bar{a}'_u \subseteq \bar{a}_u, \bar{b}'_v \subseteq \bar{b}_v, \bar{c}'_{u,v} \subseteq \bar{c}_{u,v}\bar{h}'_{u,v} \subseteq \bar{h}_{u,v} \in M_G$ for all $u \in U, v \in V$ such that :

- $M_G \models \phi'(\bar{a}'_u, \bar{b}'_v)$ if, and only if $(u, v) \in E$, and
- $(u, v) \in E$ implies $M_G \models \phi'(\bar{a}'_u, \bar{b}'_v, \bar{c}'_{u,v}, \bar{h}'_{u,v})$.

Clearly, these new subtuples are mutually disjoint and do not intersect any of the parameters $\bar{p}' \subseteq \bar{p}$ that appear in ϕ'. We finally let M'_G be the induced substructure of M_G consisting solely of these subtuples. Since the equality type of all tuples $\bar{p}'_{u,v} = \bar{a}_u^\frown \bar{b}_v^\frown \bar{h}_{u,v}$ is uniformly constant by Theorem 23, it follows that M_G may be computed from $G = (U, V; E)$ by adding disjoint tuples $(\bar{a}_u^G)_{u\in U}, (\bar{b}_v^G)_{v\in V}, (\bar{h}_{u,v}^G)_{(u,v)\in E(G)}, \bar{p}^G$ of appropriate equality types to represent vertices and existential witnesses, and the relations specified by ϕ' to represent the edges. Clearly, the tuple $\bar{p}^G$ which interprets the parameters of ϕ' is obtained in constant time from G. ◀

With this, we proceed to show intractability for monotone classes with somewhere Gaifman class. Our proof is essentially based on the proof of [13, Theorem 6.1], which covers the case of graphs. There, monotonicity and somewhere density imply that for some $r \in \mathbb{N}$ we may find an r-subdivided copy of any finite graph G in our class. The aim is then to definably distinguish the native points of G from the subdivision points. Assuming this, G can be simply interpreted, defining an edge between two native points if there is a path of length r between them. The idea is to distinguish points by their degrees; however, while all subdivision points have degree two, other points in G may as well have degree two. To address this, we first pre-process G to obtain a graph G' by adding two pendant vertices to each non-isolated vertex. Then, G may be definably recovered from G', and moreover, given an r-subdivision of G', we can definably distinguish the subdivision points and the remaining points by their degrees. Our construction is essentially the same, although the degree of a subdivision point is bounded by the length of paths in the subdivision, rather than by two. Moreover, we ought to ensure that the formula coding paths is not symmetric, so as to avoid accidentally creating two disjoint copies of the graph we wish to interpret.

▶ **Theorem 27.** *Let $\mathcal{C}$ be a monotone class of $\mathcal{L}$-structures such that* $\mathsf{Gaif}(\mathcal{C})$ *is somewhere dense, and assume that* $\mathsf{AW}[*] \neq \mathsf{FPT}$. *Then* FO *model-checking on $\mathcal{C}$ is not fixed-parameter tractable.*

Proof. Let $\mathcal{C}$ satisfy the above, and assume that $\mathsf{AW}[*] \neq \mathsf{FPT}$. We argue that we may polynomially interpret the class of all bipartite graphs in $\mathcal{C}$.

Let $\phi(\bar{x}, \bar{y})$ be the simple path formula from Lemma 26. Without loss of generality, we may assume that ϕ is not symmetric (in the sense of Definition 19). Indeed, if ϕ is symmetric let $\sigma \in \mathcal{S}_n$ be the non-identity permutation from Definition 19, and consider the formula $\phi'(\bar{x}, \bar{y}) = \phi(\bar{x}, \sigma^{-1}(\bar{y}))$, where σ^{-1} is applied to the indices of $\bar{y}$. Clearly, ϕ' is no longer symmetric, while the tuples $(\bar{a}_u)_{u\in U}$ $(\sigma(\bar{b}_v))_{v\in V}$, $(\bar{h}_{u,v})_{(u,v)\in E}$ still satisfy the conditions in Lemma 26.

Now, let k the length of the path $\mathcal{M}_\phi$ and define the auxiliary map:

$$f : \mathfrak{B} \to \mathfrak{B}$$
$$G = (U, V; E) \mapsto (U', V'; E'),$$

where $U' := U \sqcup \{\dot{u}_{v,1}, \dots, \dot{u}_{v,k+1} : v \in V\}$, $V' := V \sqcup \{\dot{v}_{u,1}, \dots, \dot{v}_{u,k+1} : u \in U\}$, and $E' := E \sqcup \{(u, \dot{v}_{u,i}) : u \in U, i \in [k+1]\} \sqcup \{(v, \dot{u}_{v,i}) : v \in V, i \in [k+1]\}$.

This is clearly computable in polynomial time. Given $G = (U, V; E) \in \mathfrak{B}$, consider $\Phi \circ f(G) \in \mathcal{C}$ given from Theorem 23, and let:

$$\theta_U(\bar{x}) := \exists^{>k} \bar{y} \phi(\bar{x}, \bar{y}) \wedge \bar{x} \neq \bar{p} \text{ and } \theta_V(\bar{y}) := \exists^{>k} \bar{x} \phi(\bar{x}, \bar{y}) \wedge \bar{y} \neq \bar{p},$$

where $\bar{p}$ are the parameters of ϕ. Without loss of generality, we may assume that $|\bar{x}| = |\bar{y}|$, for if $m = |\bar{y}| < |\bar{x}| = n$, then we may take $\theta_V(\bar{y}, y_{m+1}, \dots, y_n)$ to be $\theta_V(\bar{y}) \wedge \bigwedge_{i=m}^{n-1}(y_i = y_{i+1})$, and similarly if $|\bar{x}| < |\bar{y}|$. So, let $\theta(\bar{x}) = \theta_V(\bar{x}) \vee \theta_U(\bar{x})$.

Observe that G is an induced subgraph of $f(G)$, so we may view $\Phi(G)$ as an induced substructure of $\Phi \circ f(G)$. Letting $\bar{p}_{u,v} = \bar{a}_u \frown \bar{b}_v \frown \bar{h}_{u,v}$, it holds that $\bar{p}_{u,v} \cap \bar{p} = \emptyset$ and

$$\bar{p}_{u,v} \cap \bar{p}_{u',v'} = \begin{cases} \bar{a}_u & \text{if } u = u'; \\ \bar{b}_v & \text{if } v = v'; \\ \emptyset & \text{otherwise.} \end{cases}$$

whenever $(u, v) \neq (u', v')$. Hence, the only non-parameter elements that appear more than k times within a path are those in the tuples $\bar{a}_u$ and $\bar{b}_v$ for $u \in U$ and $v \in V$, i.e. those tuples corresponding to the elements of G. Since ϕ is not symmetric, it follows that $\theta(\Phi \circ f(G)) = \{\bar{a}_u, \bar{b}_v : u \in U, v \in V\}$, and so the pair $I = (\theta(\bar{x}), \phi(\bar{x}, \bar{y}))$ is an interpretation with computable parameters such that $I(\Phi \circ f(G)) = G$ for all $G \in \mathfrak{B}$. It follows that $\mathfrak{B} \leq_P \mathcal{C}$, and therefore $\mathcal{C}$ is not tractable. ◀

6 From nowhere density to monadic stability

Here we establish that a class of structures with nowhere dense Gaifman graphs is monadically stable. This argument relies on the extension of Theorem 6 to coloured digraphs, and the equivalence of nowhere density for the classes of Gaifman graphs and incidence graphs.

▶ **Lemma 28** ([15, Proposition 5.7]). *Let $\mathcal{C}$ be a class of structures in a finite relational language. Then $\mathsf{Gaif}(\mathcal{C})$ is nowhere dense if, and only if, $\mathsf{Inc}(\mathcal{C})$ is nowhere dense.*

We enrich the definition of incidence graphs by colouring the edges to distinguish between the various relations in the original language, and to indicate that a point in the domain corresponds to the i^{th} point of an incident tuple. We also direct the edges from points in the original domain to incident tuples. This will allow us to easily recover the original structure via a simple interpretation.

▶ **Definition 29** (Coloured incidence graphs). *Let M be an $\mathcal{L}$-structure in a finite relational language. Write n for the maximum arity of a relation symbol in $\mathcal{L}$, and let $E_\mathcal{L}$ be the language containing binary relation symbols $\{R_i : i \in [n], R \in \mathcal{L}\}$. We define the* coloured incidence graph *of M to be the $E_\mathcal{L}$-structure $\mathsf{Inc}^{\mathsf{c}}(M)$ on domain $M \sqcup \bigsqcup_{R \in \mathcal{L}} R^M$ such that for all $R \in \mathcal{L}$ and $i \in [n]$*

$$(u, \bar{v}) \in R_i^{\mathsf{Inc}^{\mathsf{c}}(M)} \text{ if, and only if, } \bar{v} \in R^M \text{ and } \bar{v}(i) = u.$$

For a class $\mathcal{C}$ of $\mathcal{L}$-structures we write $\mathsf{Inc}(\mathcal{C})$ for the class $\{\mathsf{Inc}(M) : M \in \mathcal{C}\}$.

▶ **Theorem 30.** *Let $\mathcal{C}$ be a class of structures in a finite relational language. If $\mathsf{Gaif}(\mathcal{C})$ is nowhere dense, then $\mathcal{C}$ is monadically stable.*

Proof. By Lemma 28, $\mathsf{Gaif}(\mathcal{C})$ being nowhere dense implies that $\mathsf{Inc}(\mathcal{C})$ is nowhere dense. In turn, this implies that $\mathsf{Inc}^{\mathbf{c}}(\mathcal{C})$ is monadically stable by the generalisation of Theorem 6 to coloured directed graphs, mentioned in both [1] and [17]. It is easily observed that $\mathcal{C}$ is simply interpreted in $\mathsf{Inc}^{\mathbf{c}}(\mathcal{C})$ by the formulas $\delta(x) = \neg\exists y \bigvee_{R\in\mathcal{L}} \bigvee_{i\in[n]} R_i(y,x)$ and $\phi_R(x_1,\dots,x_{\mathrm{ar}(R)}) = \exists z \bigwedge_{i\in[\mathrm{ar}(R)]} R_i(x_i,z)$ for $R \in \mathcal{L}$. Since interpretations preserve monadic stability, $\mathcal{C}$ is monadically stable as well. ◀

An alternative proof of the theorem above is indicated in [1], which does not pass through incidence graphs but instead explicitly codes the relations into a graph via gadgets.

The hypothesis in the following corollary is weaker than demanding that $\mathsf{Gaif}(\mathcal{C})$ be nowhere dense, as witnessed by the class of finite cliques with countably many edge colours and no two edges receiving the same colour.

▶ **Corollary 31.** *Let $\mathcal{C}$ be a class of structures in an infinite relational language. If for every reduct to a finite language, $\mathsf{Gaif}(\mathcal{C}^-)$ is nowhere dense, then $\mathcal{C}$ is monadically stable.*

Proof. The failure of monadic stability is witnessed by a single formula in some unary expansion, which only uses finitely many relations. ◀

▶ **Corollary 32.** *Let M be a relational structure such that for every reduct M^- to a finite language, for every $r \in \mathbb{N}$ there is some $n \in \mathbb{N}$ with $K_n^{(r)} \not\leq \mathsf{Gaif}(M^-)$. Then M is monadically stable.*

Proof. Let $\mathcal{C}$ be the class of finite substructures of M. Given the assumption, the previous lemma implies $\mathcal{C}$ is monadically stable. By [4], this implies M is monadically stable. ◀

7 Conclusion

Our paper settles the question of Adler and Adler, showing that tameness for a monotone class of relational structures can be completely recovered from the structural sparsity of its Gaifman class. We believe that many results from the theory of sparse graphs will generalise to relational structures by working with the Gaifman class, and we plan to exhibit such generalisations in future work.

Although this has not been addressed thus far, monotonicity as defined for classes of relational structures does not fully correspond to monotonicity in the standard graph-theoretic sense. Indeed, in the graph-theoretic sense, a monotone class of graphs is one closed under removal of *undirected* edges, that is, simultaneous removal of pairs of relations $E(u,v), E(v,u)$. However, a monotone class of $\{E\}$-structures is one where we can remove any E relation (so possibly we can turn an undirected edge into a directed one). In future work, we aim to address this subtle difference by introducing *symmetrically monotone* classes, so that our results can extend to broader classes of relational structures, such as classes of undirected hypergraphs closed under removal of hyperedges.

Finally, our paper makes a significant contribution towards Conjecture 9, settling it for the case of monotone classes of structures. While the machinery used in this paper will certainly assist in tackling the full conjecture, we believe that new techniques are required for this task. Here, it is important to understand the role of linear orders in the collapse of monadic NIP and bounded twin-width for hereditary classes of ordered graphs, and to identify which model-theoretic conditions generalise this phenomenon to arbitrary hereditary graph classes.

References

1 Hans Adler and Isolde Adler. Interpreting nowhere dense graph classes as a classical notion of model theory. *European Journal of Combinatorics*, 36:322–330, 2014. doi:10.1016/j.ejc.2013.06.048.

2 John T. Baldwin. *Fundamentals of Stability Theory*. Perspectives in Logic. Cambridge University Press, 2017.

3 Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, Pierre Simon, Stéphan Thomassé, and Szymon Toruńczyk. Twin-width IV: Ordered graphs and matrices, 2021. doi:10.48550/arXiv.2102.03117.

4 Samuel Braunfeld and Michael C. Laskowski. Existential characterizations of monadic NIP, 2022. doi:10.48550/arXiv.2209.05120.

5 Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In John E. Hopcroft, Emily P. Friedman, and Michael A. Harrison, editors, *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*, pages 77–90. ACM, 1977. doi:10.1145/800105.803397.

6 Anuj Dawar. Finite model theory on tame classes of structures. In *MFCS*, volume 4708 of *Lecture Notes in Computer Science*, pages 2–12. Springer, 2007.

7 Paul Erdős and Richard Rado. A combinatorial theorem. *Journal of the London Mathematical Society*, 1(4):249–255, 1950.

8 Jakub Gajarský, Michal Pilipczuk, and Szymon Torunczyk. Stable graphs of bounded twin-width. In Christel Baier and Dana Fisman, editors, *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, pages 39:1–39:12. ACM, 2022. doi:10.1145/3531130.3533356.

9 Jakub Gajarský, Petr Hliněný, Daniel Lokshtanov, Jan Obdržálek, and M. S. Ramanujan. A new perspective on FO model checking of dense graph classes, 2018. arXiv:1805.01823.

10 Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding first-order properties of nowhere dense graphs. *J. ACM*, 64(3), June 2017. doi:10.1145/3051095.

11 Wilfrid Hodges. *Model Theory*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1993. doi:10.1017/CBO9780511551574.

12 Alexandr Kostochka, Dhruv Mubayi, and Jacques Verstraëte. Turán problems and shadows II: Trees. *Journal of Combinatorial Theory, Series B*, 122:457–478, 2017. doi:10.1016/j.jctb.2016.06.011.

13 Stephan Kreutzer and Anuj Dawar. Parameterized complexity of first-order logic. *Electron. Colloquium Comput. Complex.*, TR09-131, 2009. URL: https://eccc.weizmann.ac.il/report/2009/131.

14 Jaroslav Nešetřil and Patrice Ossona De Mendez. On nowhere dense graphs. *European Journal of Combinatorics*, 32(4):600–617, 2011.

15 Jaroslav Nešetřil and Patrice Ossona De Mendez. *Sparsity: graphs, structures, and algorithms*, volume 28. Springer Science & Business Media, 2012.

16 Michal Pilipczuk, Sebastian Siebertz, and Szymon Torunczyk. Parameterized circuit complexity of model-checking on sparse structures. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 789–798. ACM, 2018. doi:10.1145/3209108.3209136.

17 Klaus-Peter Podewski and Martin Ziegler. Stable graphs. *Fund. Math*, 100(2):101–107, 1978.

18 F. P. Ramsey. On a Problem of Formal Logic. *Proceedings of the London Mathematical Society*, s2-30(1):264–286, January 1930. doi:10.1112/plms/s2-30.1.264.

19 Saharon Shelah. *Classification theory and the number of nonisomorphic models*, volume 92 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam-New York, 1978.

20 Pierre Simon. *A Guide to NIP Theories*. Lecture Notes in Logic. Cambridge University Press, 2015. doi:10.1017/CBO9781107415133.

21 Pierre Simon and Szymon Toruńczyk. Ordered graphs of bounded twin-width, 2021. doi:10.48550/arXiv.2102.06881.

22 Algorithms, logic, and structure workshop in Warwick, open problem session. URL: https://warwick.ac.uk/fac/sci/maths/people/staff/daniel_kral/alglogstr/openproblems.pdf. Accessed: 2023-05-02.

Compositionality of Planar Perfect Matchings

A Universal and Complete Fragment of ZW-Calculus

Titouan Carette
Centre for Quantum Computer Science, Faculty of Computing, University of Latvia, Riga, Latvia

Etienne Moutot
CNRS, I2M, Aix-Marseille Université, Marseille, France

Thomas Perez
Université de Lyon, ENS de Lyon, France

Renaud Vilmart
Université Paris-Saclay, ENS Paris-Saclay, Inria, CNRS, LMF, 91190, Gif-sur-Yvette, France

Abstract

We exhibit a strong connection between the matchgate formalism introduced by Valiant and the ZW-calculus of Coecke and Kissinger. This connection provides a natural compositional framework for matchgate theory as well as a direct combinatorial interpretation of the diagrams of ZW-calculus through the perfect matchings of their underlying graphs.

We identify a precise fragment of ZW-calculus, the planar W-calculus, that we prove to be complete and universal for matchgates, that are linear maps satisfying the matchgate identities. Computing scalars of the planar W-calculus corresponds to counting perfect matchings of planar graphs, and so can be carried in polynomial time using the FKT algorithm, making the planar W-calculus an efficiently simulable fragment of the ZW-calculus, in a similar way that the Clifford fragment is for ZX-calculus. This work opens new directions for the investigation of the combinatorial properties of ZW-calculus as well as the study of perfect matching counting through compositional diagrammatical technics.

2012 ACM Subject Classification Theory of computation → Quantum computation theory; Theory of computation → Equational logic and rewriting; Mathematics of computing → Matchings and factors

Keywords and phrases Perfect Matchings Counting, Quantum Computing, Matchgates, ZW-Calculus, String Diagrams, Completeness

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.120

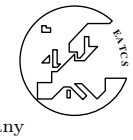
Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `https://arxiv.org/abs/2302.08767`

Funding *Titouan Carette*: acknowledges support from the ERDF project 1.1.1.5/18/A/020 "Quantum33 algorithms: from complexity theory to experiment".
Renaud Vilmart: acknowledges support from the PEPR integrated project EPiQ ANR-22-PETQ-0007 part of Plan France 2030, the ANR projects TaQC ANR-22-CE47-0012 and HQI ANR-22-PNCQ-0002, as well as the European project HPCQS.

1 Introduction

A quantum computation mapping n qubits to m qubits corresponds to an isometric linear map $\mathbb{C}^{2^n} \to \mathbb{C}^{2^m}$. Due to the exponential size of their matrix representation, those linear maps are traditionally depicted as quantum circuits, an assemblage of elementary quantum gates similar to the more common boolean circuits. Given a quantum circuit $n \to m$, evaluating a coefficient of the corresponding $2^m \times 2^n$ matrix (i.e. evaluating the circuit



with a given input) typically requires an exponential time. However, there are some specific classes of quantum circuits – or fragments –, that can be classically simulated in polynomial time. Examples are the Clifford fragment (as asserted by the Gottesman-Knill theorem) as well as the fragment that will particularly interest us in this paper, the nearest-neighbour matchgates [24]. Investigating those tractable fragments allows a better understanding of the computational advantage of quantum computing. The reference for all elementary results on quantum circuits is [19].

Taking the diagrammatical circuit representation seriously led to developing graphical languages for quantum computing [7]. Those languages are equational theories described by elementary gates and local identities between diagrams. Such languages come with an interpretation into linear maps. A language is said universal for a class of linear maps if any linear map in the class is the interpretation of a diagram in the language. A language is said complete if two diagrams with the same interpretation are equivalent up to the equational theory, which means that they can be rewritten from one to the other using the local rewriting rules of the equational theory. In general, completeness is the most challenging property to prove.

The first quantum graphical language to appear was the ZX-calculus in 2008 [7]. It was rapidly known to be universal for all linear maps. However, providing a complete set of rewriting rules took another ten years (see [26] for an history of completeness) and first required a translation through another language, the ZW-calculus [12, 13].

The ZW-calculus was introduced in [8] as a graphical representation of the two kinds of tripartite entanglement for qubits, namely the GHZ-states and W-states. It then appeared that this calculus had very nice algebraic properties allowing the internal encoding of arithmetical operations. Those properties allowed the ZW-calculus to be the first proven universal and complete language for linear maps [12]. Despite this historical importance, the ZW-calculus gathered less attention than other languages, seen as more connected to quantum computing. Still, we must mention interesting connections with fermionic quantum computing [11], and recent works importing some ZW-calculus primitives into ZX-calculus to exploit their algebraic properties [20, 28]. In this paper, we show that ZW-calculus has very strong connections with a specific family of quantum circuits: the matchgates.

Matchgates were introduced in 2002 by Valiant [24]. They are linear maps defined by counting the perfect matching of a graph from which we remove some vertices depending on the inputs. This underlying combinatorial structure allows to classically simulate the corresponding quantum circuits by using the Polynomial FKT algorithm for perfect matchings counting [22, 15]. The theory of matchgates was then developed further to the concept of holographic algorithms [25]. We can notice that if some connections between graphical languages and holographic algorithms have been investigated [2], we are not aware of any diagrammatical approach to the original concept of matchgate before the present work, except a mention in [11].

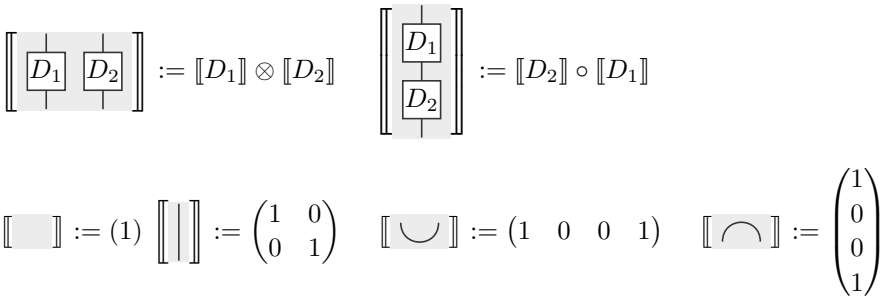
The main contribution of this paper is the introduction of a fragment of the ZW-calculus, that we call planar W-calculus. We show that this language is universal and complete for the planar matchgate fragment of quantum computation. The completeness proof relies on designing a normal form and a rewriting strategy to reach it. We also define a pro of matchgate computations by showing the compositionality of the matchgate identities introduced in [4]. The combinatorial characterisation of matchgate computations then directly follows from the correspondence with the graphical language. Hence one can see this paper as a reformulation of matchgate theory in a compositional framework.

The paper is structured as follows. Section 2 introduces our graphical primitives, their interpretation as linear maps and their combinatorial properties: the interpretation of a diagram can be deduced by counting the number of perfect matching of the underlying weighted graph. We present the generators and elementary rewrite rules of the language as

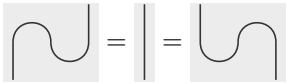
well as an essential syntactic sugar: the fermionic swap that emulates the swap gate, which is not part of our language. Section 3 introduces the normal form and proves the completeness of the language. In Section 4, we properly define a pro of matchgates characterised as the linear maps satisfying the matchgate identities. We show that our language is universal for matchgates, i.e., that the interpretation of a diagram is always a matchgate and that all matchgates correspond to a diagram. Finally, in Section 5, we sketch future directions of research suggested by the connection we identified between ZW-calculus and perfect matching counting.

2 Perfect Matchings and Planar W-Calculus

We define our fragment of the ZW-calculus, the *planar W-calculus*, denoted p**W**, by defining its diagrams. Any diagram with n inputs and m outputs $D : n \to m$ is interpreted as a linear map $[\![D]\!] : {}^{2^n} \to {}^{2^m}$ inductively as follows:



In particular, note that we do *not* use the usual swap diagram, hence the name *planar*. We do have, however, the so-called cup and cap satisfying the "snake equations":

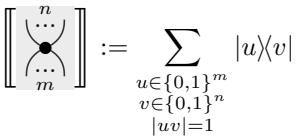


In the following, with $D : n \to n$, we may use the following notation: $D^{\otimes \vec{b}}$ when $\vec{b}$ is a bitstring, to represent $D^{b_1} \otimes ... \otimes D^{b_n}$ with $D^0 = id_n$ and $D^1 = D$. We call a diagram D a scalar if it has no input and no output, i.e. $D : 0 \to 0$. In the category-theoretic terminology, such a collection of diagrams defines a pro, a strict monoidal category whose monoid of objects is generated by a unique element, and not a prop, which requires the category to be symmetric, i.e. to have swap diagrams. Furthermore, the presence of the cups and caps make the category a compact-closed pro. We define **Qubit** to be the prop whose $n \to m$ morphisms are linear maps ${}^{2^n} \to {}^{2^m}$. Hence $[\![\cdot]\!] : \text{p}\mathbf{W} \to \mathbf{Qubit}$ is a pro morphism.

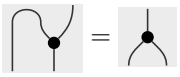
We add the two following generators: the black spider and the binary white spider, whose interpretations are detailed in the next sub-sections.

2.1 Black Spider

To manipulate binary words $\alpha \in \{0,1\}^n$ and $\beta \in \{0,1\}^m$, we will denote $\alpha \oplus \beta \in \{0,1\}^n$ the bitwise XOR (if $n = m$), $\alpha \cdot \beta \in \{0,1\}^{n+m}$ the concatenation, $|\alpha| \in \{0, ..., n\}$ the Hamming weight, i.e., the number of ones in the word α, and $|\alpha|_2 \in \{0,1\}$ the parity of this weight, 0 if even and 1 if odd. The *black spider* (or black node) is given by the following interpretation:

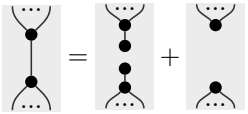


In other words, the black spiders gives an output 1 if and only if exactly one of its legs (either input or outputs) has value $|1\rangle$ and all the others $|0\rangle$. As inputs and outputs behave exactly the same, one can use cup and caps in order to transform inputs into outputs and vice-versa:



Moreover, as input order do not matter, one can bend the wires and move black spiders around, without altering the resulting linear map, we say that the black nodes are flexsymmetric [6]. Flexsymmetry of the black spider allows us to see diagrams as graphs with fixed inputs and outputs edges. Fixing the input and outputs edges, any graph isomorphism preserves the semantics.

With this graphical interpretation in mind, one can understand the interpretation of a scalar diagram, composed of only black spiders, as counting the number of perfect matchings in the underlying graph. To see this, one can use the interpretation of a single edge, which simply is the identity $|0\rangle\langle 0| + |1\rangle\langle 1|$. This interpretation gives a useful insight in the diagrams: given an edge, one can partition the set of perfect matchings between those that have this edge and those that don't:

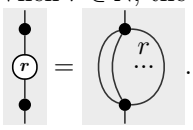


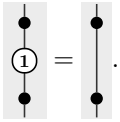
In the case where the graph is an actual graph, without half edges, the resulting map is a scalar (no input or outputs). One can show by induction that this scalar corresponds to the number of ways of choosing a set of edges such that each vertex is covered by exactly one edge. In other ways, *the number of perfect matchings* of the graph.

2.2 Binary White Spider

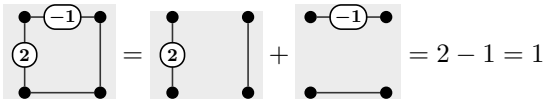
The last generator of the planar W-calculus is the *binary white spider*, given, for any $r \in$, by:


$:= \begin{pmatrix} 1 & 0 \\ 0 & r \end{pmatrix}$

which corresponds to the usual binary white spider with weight r of the ZW-calculus. This binary spider corresponds to having a weight r on an edge of the graph. When $r \in \mathbb{N}$, the interpretation is straightforward: the white spider can be replaced by r edges:  .

And in particular,  .

Let us interpret the white spiders as weights on the edges of a planar graph G with black spiders on their vertices. Consider one perfect matching of the same graph G' without weights and consider one perfect matching P of G'. If the edge e that belongs to P has a weight $r \in \mathbb{N}$, then it can be replaced by r edges. In other words, the single perfect matching P is replaced by r perfect matchings when e has weight r. By doing this for every edges, one can see that each perfect matching in G' corresponds to a perfect matching of G with a *weight* that is the product of all its edge weights, instead of weight 1 in G'. For $r \in$, one cannot replace a white spider by a given number of edges, but the interpretation is the same: the edge contribute to the perfect matchings that contain it with a *weight* r.

▶ **Example 1.** 

$= 2 - 1 = 1$

Diagrams generated by the black and binary white node, within the framework described at the beginning of the section, are called p**W**-diagrams.

2.3 The FKT Algorithm

In general, counting the number of perfect matchings in a graph is an #P-complete problem [23]. However, for planar graphs the same problem turns out to be surprisingly easy, as Fisher, Temperley and Kastelyn showed that it is in P [22, 14]. The main idea behind the algorithm is that for planar graphs, it is possible to find a good orientation of the edges (called a Pfaffian orientation) in polynomial time such that the number of perfect matchings is the Pfaffian of the adjacency matrix A (actually its skew-symmetric version, called Tutte matrix) of the oriented graph. A result due to Cayley then shows that the Pfaffian is the square root of the determinant of A.

Note that one can find such an orientation for any planar graph, even weighted with complex weights, and the equality $pf(A) = \sqrt{det(A)}$ still holds. Therefore, computing the total *weight* of perfect matchings in a complex-weighted graph is in P.

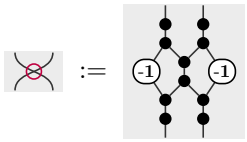
▶ **Proposition 2.** *Let D be a scalar* p**W***-diagram. Then $[\![D]\!]$ is computable in polynomial time in the number of black nodes.*

2.4 Fermionic Swap

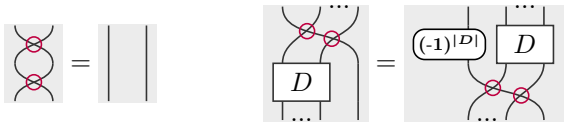
The usual ZW-calculus does have another generator that we did not explicitly include in our fragment, called the *fermionic swap*:


$:= \sum_{x,y\in\{0,1\}} (-1)^{xy} \, |xy\rangle\langle yx|$

However, it turns out that the fermionic swap is just syntactic sugar, and it is actually in our fragment:



Notice that the previous equation also appears in [5] to relate planar and non-planar matchgates. It is very useful to treat this piece of diagram as a generator of its own, especially as a particular kind of swap, which shares a lot of (but not all) properties of the symmetric braiding of props. In particular:



where $|D|$ is the number of black nodes in the diagram D.

3 Completeness

The planar W-calculus is introduced with an equational theory, given in Figure 1, relating together diagrams with the same semantics. This builds upon the more general equational theories for non-planar ZW-calculi, presented in [12, 13]. We write $\mathrm{pW} \vdash D_1 = D_2$ when one can turn diagram D_1 into diagram D_2 by applying the equations of Figure 1 locally.

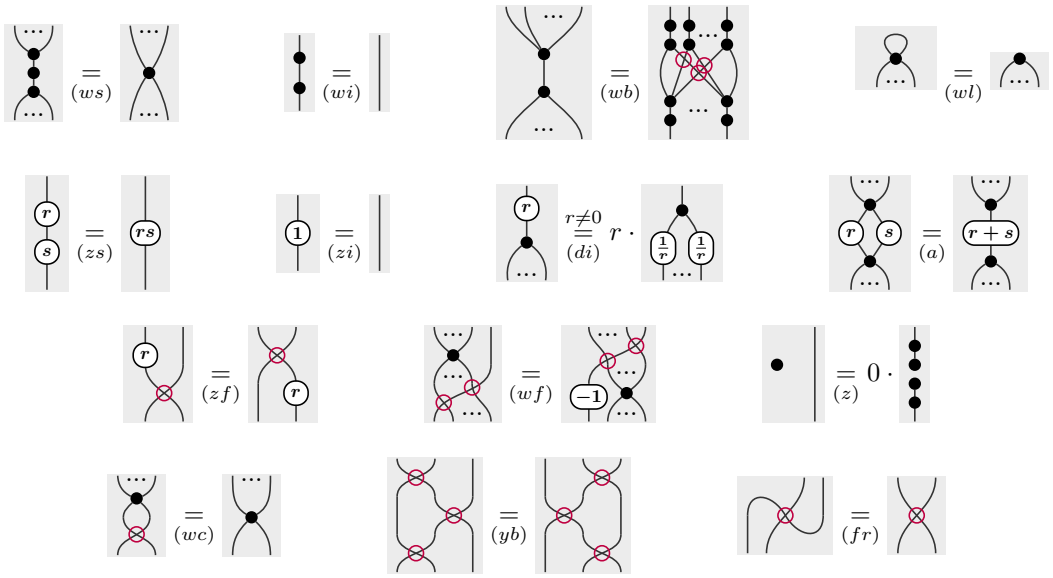


Figure 1 Axioms of the planar W-calculus.

▶ **Proposition 3.** *The equational theory of Figure 1 preserves the semantics:*

$$\mathrm{pW} \vdash D_1 = D_2 \quad \Longrightarrow \quad [\![D_1]\!] = [\![D_2]\!]$$

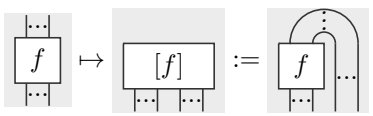
In the following, we will show that the converse also holds, that is, that whenever two diagrams have the same semantics, they can be turned into one another using the equational theory. Intuitively, this implies that the equational theory completely captures the interaction of generators with one another in the fragment.

To show this result, we give a notion of normal form, which we call W-graph-state with X-gates (WGS-X for short), then a refinement of that normal form (reduced WGS-X form) which can be shown to be unique, and we give a rewrite strategy (derivable from the equational theory) to turn any pW-diagram into this form.

3.1 Normal Form

The first step we take towards defining a normal form is a simplification, making use of the compact structure of the underlying pro, where we relate maps and states:

▶ **Proposition 4.** *There is an isomorphism between* $\mathrm{pW}(n, m)$ *and* $\mathrm{pW}(0, n + m)$ *defined as such:*



This isomorphism allows us only to consider states rather than maps in the following.

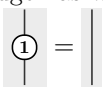
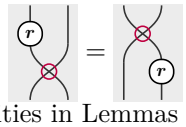
Then, we define W-graph-states, by first defining ordered weighted graphs:

▶ **Definition 5** (Ordered Weighted Graph). *$G = (V, E, w)$ is called an ordered weigthed graph if:*

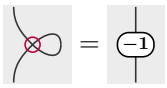
- *V is a set endowed with a total order $\prec$ (or equivalently a sequence)*
- *$E \subset V \times V$ is such that $(u, v) \in E \implies u \prec v$*
- *$w : E \to \mathbb{C} \setminus \{0\}$ maps each edge to its weight*

▶ **Definition 6** (W-Graph-State). *Let $G = (V, E, w)$ be an ordered weighted graph. Then,* $\mathrm{WGS}(G)$ *is defined as the* p**W***-diagram where:*

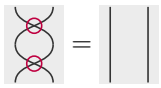
- *Each vertex in V gives a W-spider linked to an output through an additional*  *(the order on V gives the order of the outputs)*
- *Each (weighted) edge (u, v) gives a white dot with parameter $w((u, v))$ linked to the W-spiders obtained from u and v*
- *All wire crossings in* $\mathrm{WGS}(G)$ *are fermionic swaps* 
- *No output wire crosses another wire*
- *There are no self-intersecting wires*

When an edge has weight 1 we may ignore the white dot and represent the edge as a simple wire, since  . Notice that there are several ways to build $\mathrm{WGS}(G)$, but all of them are equivalent thanks to  and the axioms on the fermionic swap  , together with the provable identities in Lemmas 7 and 8:

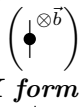
▶ **Lemma 7.**



▶ **Lemma 8.**

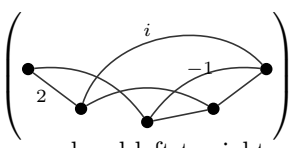
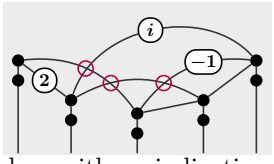


▶ **Definition 9** (WGS-X form). *We say that a* p**W***-state D on n qubits is in:*

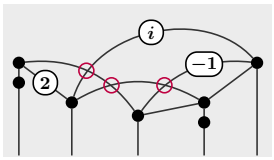
- ***WGS-X form*** *if there exist $s \in \mathbb{C}$, $G = ([1, n], E, w)$ an ordered graph, and $\vec{b} \in \{0, 1\}^n$ such that $D = s \cdot$*  $\circ\, \mathrm{WGS}(G)$.
- ***pseudo-WGS-X form*** *if it is in WGS-X form with potentially vertices linked to several outputs, additional*  *$(r \neq 0)$ on wires that do not correspond to edges in the graph, and potentially fermionic swaps*  *between outputs.*
- ***reduced WGS-X form*** *(rWGS-X) if it is in WGS-X form and:*

$$\forall i,\ (b_i = 0 \implies \nexists j,\ (i, j) \in E)$$

i.e. $b_i = 0$ is only possible if vertex i has no neighbour on its right.

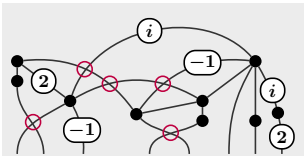
▶ **Example 10.** $\mathrm{WGS}\left($  $\right) =$  where in the starting graph, vertices are ordered left to right, and edges with no indication of weight have weight 1.

If $\vec{b} = (0, 1, 1, 0, 1)$, then the obtained WGS-X state is:

$s \cdot$ 

where we used the fact that  is an involution to simplify the diagram. The WGS-X state is however not reduced, as both the first and fourth qubits have additional  applied to them, but still have neighbours on their right.

Finally, the following diagram is an example of a pseudo-WGS-X state:



3.2 Rewrite Strategy

We define in this section a rewrite strategy, derived from the equational theory, that will terminate in a normal form (WGS-X). Doing this naively is made difficult by the potential presence of fermionic swaps  wherever we are looking for patterns to rewrite. Thankfully, the last 5 equations in Figure 1, together with the above Lemmas 7 and 8 essentially tell us that we can treat those as usual swaps with the only catch that removing self loops or moving wires past black nodes adds a -1 weight to the wires.

In the upcoming rewrite strategy, we will hence only specify the patterns without potential fermionic swaps inside. Should there be some present, it is understood that they will be moved out of the pattern, before the rewrite occurs. The rules necessary for the rewrite strategy are given in Figure 2.

▶ **Proposition 11.** *The rewrite rules of Figure 2 are derivable from the equational theory of Figure 1 and hence are sound.*

For the rewrite strategy to terminate, we need to distinguish between different types of nodes:

▶ **Definition 12** (Boundary Node / Internal Node). *A node is a* boundary node of type 1 *if it is linked directly to an output (potentially through a white node). A node is a* boundary node of type 0 *if it is connected to a binary boundary node of type 1.*
We say that a black node of D is internal *if it is not a boundary node.*

Notice that diagrams in WGS-X, rWGS-X, or pseudo-WGS-X form, do not have internal nodes. The crux of the upcoming rewrite strategy is precisely to remove internal nodes. The rewrite strategy is laid out as follows:

▶ **Definition 13** (Rewrite Strategy). *The rewrite strategy is defined in 3 steps:*

1. *Apply the rewrites of Figure 2 in any order but following constraints, until none apply anymore. The diagram ends up in pseudo-WGS-X form.*

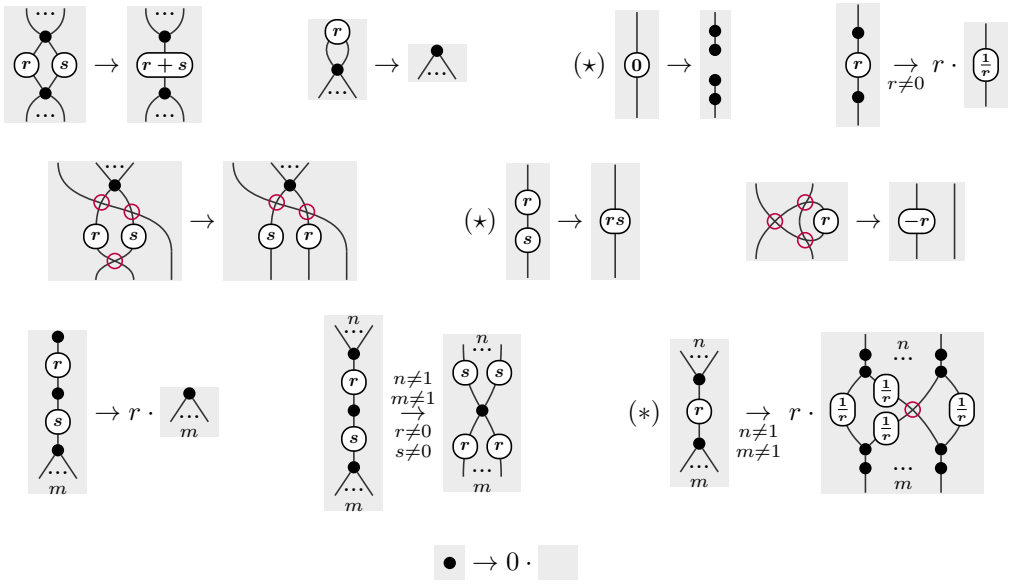


Figure 2 Rewrite rules. Rule $(\star)$ corresponds to the third axiom of Figure 1 with weights, i.e. the edges form a complete bipartite graph between the black nodes, with a fermionic swap at each crossing and weight $\frac{1}{r}$ on each edge. All the rules in this figure, except $(\star)$, are supposed to apply when any of the white nodes are replaced by identity (i.e. when their weight is 1). Rule $(*)$ can only be applied if at least one of the black nodes is internal, and if none of the other rules applies.

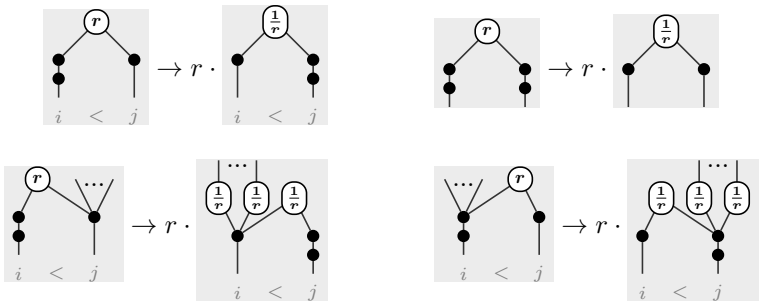
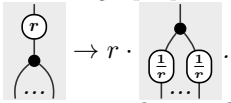


Figure 3 Rules for reduced WGS-X form, together with rule $(*)$ when the leftmost black node is a type-0 boundary node.

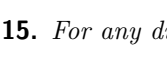
2. *First, whenever a type-1 boundary is linked to $n > 1$ outputs directly, apply*  *to the $n-1$ rightmost such outputs (the top black node then becomes a type-0 boundary node, the bottom one a type-1 boundary node). Then, push all potential fermionic swaps*  *between outputs inside the graph part. Finally, move boundary weights up into the edges of the WGS using*  *. The diagram ends up in WGS-X form.*
3. *Whenever a type-0 vertex in the graph has a right neighbour, depending on the arity of the nodes, apply rule $(*)$ or one of the rules of Figure 3 between the two nodes (and apply any other possible rule before going on).*

A claim was made in Definition 13 about the form of the diagram at the end of each step. Those claims are going to be proven in the following (Proposition 14). At the same time, we are going to show that the rewrite terminates.

▶ **Proposition 14** (Termination in rWGS-X form)**.** *The rewrite strategy terminates in a polynomial number of rewriting steps. Moreover, after Step 1 of the rewrite, the diagram is indeed in pseudo-WGS-X form, after Step 2, it is in WGS-X form, and after Step 3, it is in rWGS-X form.*

A key property of planar W calculus is that any multi-edge can be merged thanks to the first rule of Figure 2. In the case of the general ZW-calculus, the presence of both swaps and fermionic swaps do not allow us to merge all multi-edges. Indeed, different parallel edges, between the two same nodes, can make fermionic swaps with different other edges. The problem is that with the presence of multi-edges, rule $(*)$ does not necessarily lead to a reduction in the number of nodes and our rewriting strategy may not terminate.

An important operation on WGS-X states that has a simple graphical interpretation is the following:

▶ **Lemma 15.** *For any diagram D in WGS-X form $(s, G, \vec{b})$, applying*  *on the ith output can be turned into the WGS-X form $(s, G \setminus \{i\}, \vec{b} \setminus b_i)$, where $G \setminus \{i\}$ is defined as the graph G from which vertex i is removed (together with all edges linked to i and their weights), and similarly $\vec{b} \setminus b_i$ is defined as the sequence $\vec{b}$ from which ith element is removed.*

This allows us to prove the following:

▶ **Lemma 16.** *For any diagram D in WGS-X form $(s, G, \vec{b})$:*

$$[\![D]\!] = 0 \iff s = 0$$

We may then prove that 0-diagrams can be put in a very well-defined form:

▶ **Lemma 17.** *Let D be a WGS-X state such that $[\![D]\!] = 0$. Then D can be put in the WGS-X form $(0, G = ([1, n], \emptyset, \_), \vec{0})$, i.e.:*

$\mathrm{pW} \vdash D = 0 \cdot$ 

We are now able to prove the completeness of the equational theory.

▶ **Theorem 18.** *Let D_1 and D_2 be two* pW*-diagrams. Then:*

$$[\![D_1]\!] = [\![D_2]\!] \iff \mathrm{pW} \vdash D_1 = D_2$$

This last theorem, together with the fact that the rewriting in rWGS-X form is polynomial (Proposition 14) makes the problem of deciding whether two pW-diagrams are semantically equivalent a P problem.

4 Matchgates

This section aims at characterising exactly the linear maps that W-diagrams represent.

4.1 Matchgate Identities

Valiant first introduced matchgate identities to characterise $2 \to 2$ matchgates, a family of linear maps described in a combinatorial way [24]. In [4], the matchgate identities have been extended to characterise matchgates of any size. In the literature, there is a close link between

matchgate identities and the Grassman-Plucker identities applied to Pfaffians. It is not the case here, as the diagrammatic technics allow us to directly link matchgate identities to matchings without the intermediate of the Pfaffian. We can fully recover the connection with Pfaffians through the Fetcher-Kasteleyn-Temperley algorithm for counting perfect matchings [15, 22], more details on this are outlined in Section 5. Many of the proofs of this section are inspired by the very useful clarification of matchgate theory presented in [5]. Notice that contrary to the literature that differentiates between matchgrids, matchcircuits or matchnets, we will only use the term matchgate for any linear map satisfying the matchgate identities.

Recall that for binary words $\alpha \in \{0,1\}^n$ and $\beta \in \{0,1\}^m$, $\alpha \oplus \beta \in \{0,1\}^n$ is the bitwise XOR (if $n = m$), $\alpha \cdot \beta \in \{0,1\}^{n+m}$ the concatenation, $|\alpha| \in \{0, ..., n\}$ the Hamming weight, i.e., the number of ones in the word α, and $|\alpha|_2 \in \{0,1\}$ the parity of this weight, 0 if even and 1 if odd.

▶ **Definition 19** (Matchgate Identities). *A tensor $\Gamma \in \mathbb{C}^{2^n}$ satisfies the **matchgate identities** (MGIs) if for all $\alpha, \beta \in \{0,1\}^n$:*

$$\sum_{k=1}^{|\alpha \oplus \beta|} (-1)^k \Gamma_{\alpha \oplus e_{p_k}} \Gamma_{\beta \oplus e_{p_k}} = 0$$

Where $e_{p_k} \in \{0,1\}^n$ is the binary word which is zero everywhere except in position p_k, which is the kth position in the set $\{p_1, ..., p_{|\alpha \oplus \beta|}\} \subseteq \{1, ..., n\}$ of positions in which the words α and β differs.

The matchgate identities are not linear, so the set of matchgates is not a subspace of the vector space $\mathbb{C}^{2^n}$ but an algebraic variety [4]. In general, those identities are not algebraically independent, i.e. are not all strictly necessary to describe match-tensors.

Indeed, there are numerous symmetries in those identities. For example, the case $\alpha = \beta$ directly gives empty sums and exchanging α and β gives the same identity. Interestingly, one can replace half of the identities with a parity condition.

▶ **Proposition 20** (Parity condition [5]). *If Γ satisfies the matchgate identities then it satisfies the **parity condition**: for all $\alpha, \beta \in \{0,1\}^n$, $|\alpha|_2 \neq |\beta|_2 \Rightarrow \Gamma_\alpha \Gamma_\beta = 0$.*

The parity condition splits match-tensors into two groups, the one with odd parity, such that $|\alpha|$ even implies $\Gamma_\alpha = 0$, and the one of even parity, such that $|\alpha|$ odd implies $\Gamma_\alpha = 0$. In particular, the parity condition directly implies that all terms in identities with $|\alpha|_2 \neq |\beta|_2$ are zero. Notice that the parity condition is not sufficient. We still need matchgate identities in general.

However, the parity condition is sufficient for $n \leq 3$, but not anymore for $n = 4$, the original case considered by Valiant [24]. In particular, for $n = 0$, the matchgate identities are trivially true; hence they are satisfied by all scalars (processes $0 \to 0$).

4.2 The Pro of Matchgates

We will now use the matchgates to define a pro. So far, matchgate identities have been used to characterise vectors seen as tensors, without consideration of inputs and outputs. To apply them to linear maps $f : n \to m$, we will use the state form: $[f] : 0 \to n + m$ described in Proposition 4. It allows us to define matchgates.

▶ **Definition 21** (Matchgates). *A **matchgate** is a linear map $f : \mathbb{C}^{2^n} \to \mathbb{C}^{2^m}$ such that $[f]$ satisfies the matchgate identities.*

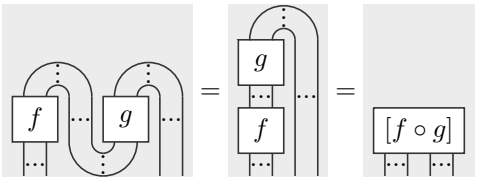
We would like to define a sub-pro of **Qubit** whose processes are matchgates, however, there are a few properties to check before that. We start by showing stability by the tensor product.

▶ **Lemma 22.** *Given two linear maps $f : a \to b$ and $g : c \to d$ whose state forms $[f] \in \mathbb{C}^{2^{a+b}}$ and $[g] \in \mathbb{C}^{2^{c+d}}$ satisfy the matchgate identities, then $[f \otimes g] \in \mathbb{C}^{2^{a+c+b+d}}$ satisfies the matchgate identities.*

The next thing to check is stability by composition; this follows from the following result:

▶ **Lemma 23.** *If $\Gamma \in \mathbb{C}^{2^{n+2}}$ satisfies the matchgate identities, then the tensor obtained by contracting two consecutive indices satisfies the matchgate identities.*

Notice that the consecutive indices assumption is essential here. Without it, we could easily construct the swap gate that does not satisfy the matchgate identities. To be able to contract consecutive indices is enough to show the stability by composition. The idea is to iterate contraction on consecutive indices until we obtain enough cups to use the snake equation, pictorially:

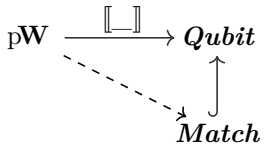


Now that we have stability by tensor and composition, it only remains to show the identities are matchgates. id_0 is a scalar, so directly a matchgate. The state-form of id_1 is the cap which is a matchgate as it satisfies the parity condition (sufficient for $n = 2$). The fact that all id_n are matchgates follows from stability by the tensor product. We can now state the main theorem of this subsection.

▶ **Theorem 24 (Match).** *The matchgates form a pro **Match**, which is a sub-pro of **Qubit**.*

Notice that **Match** is compact closed since the cup and the cap are both matchgates. Hence we can also use process/state duality in **Match** without any worry. As expected, all W-diagrams represent matchgates.

▶ **Lemma 25.** *The functor $[\![\_]\!] : \mathrm{pW} \to$ **Qubit** factorises through **Match**, i.e., the interpretations of diagrams in W are matchgates.*



Proof. We have to prove that the interpretation of any pW diagram is a matchgate. To do so, as matchgates are stable by composition and tensor product we only have to check that the interpretations of the generators are matchgates. The state forms of the generators have at most three outputs (n-ary spiders can be decomposed into binary and ternary spiders), so it is sufficient to check the parity condition, which is indeed satisfied by the interpretations of the generators. ◀

4.3 Universality

Now that we proved that all pW-diagrams represent matchgates, it remains to show that all matchgates can be represented by a pW diagram, in other words, that pW is universal for **Match**. This will require a few additional properties of matchgates, adapting some results of [5].

▶ **Lemma 26.** *If Γ satisfies the matchgate identities and $\Gamma_{\mathbf{0}} = 1$, where $\mathbf{0}$ is binary word full of 0, then it is uniquely determined by its coefficients Γ_α where $|\alpha| = 2$.*

Proof. If $|\alpha| = 0$ then we already know that $\Gamma_\alpha = 1$ and the parity condition implies that $\Gamma_\alpha = 0$ if $|\alpha| = 1$. We show that for all α with $3 \leq |\alpha|$, we can express Γ_α from coefficients Γ_β where all βs have strictly smaller Hamming weights. Let i be the first position where α and $\mathbf{0}$ differ, the matchgate identity corresponding to $\alpha \oplus e_i$ and $\mathbf{0} \oplus e_i$ is:

$$\sum_{k=1}^{|\alpha|} (-1)^k \Gamma_{\alpha \oplus e_i \oplus e_{p_k}} \Gamma_{e_i \oplus e_{p_k}} = 0$$

Here the p_k are exactly the position where α is 1, in particular $i = p_1$ so:

$$\Gamma_\alpha = \Gamma_\alpha \Gamma_{\mathbf{0}} = \sum_{k=2}^{|\alpha|} (-1)^k \Gamma_{\alpha \oplus e_i \oplus e_{p_k}} \Gamma_{e_i \oplus e_{p_k}}$$

For $k \geq 2$, We have $|e_i \oplus e_{p_k}| = 2$ and $|\alpha \oplus e_i \oplus e_{p_k}| = |\alpha| - 2$ so Γ_α is completely determined by coefficients corresponding to strictly smaller Hamming weight. It follows that all Γ_α can be expressed from the Γ_βs with $|\beta| = 2$. ◀

We will now be able to reuse the normal form from Section 3 to construct diagrams representing any matchgate.

▶ **Lemma 27** (Universality)**.** pW *is universal for* ***Match****.*

Proof. Relying on process/state duality, we only consider states $0 \to n$. Given Γ satisfying the matchgate identities, we will construct a W diagram D such that $[\![D]\!] = \Gamma$. We start by considering the case where $\Gamma_{\mathbf{0}} = 1$. Then we construct a weighted graph G on n vertices setting the weight of the edge (i, j) to $\Gamma_{e_i \oplus e_j}$. We then take D to be the diagram in graph form corresponding to G. By construction we then have $[\![D]\!]_{\mathbf{0}} = 1$ and $[\![D]\!]_{e_i \oplus e_j} = \Gamma_{e_i \oplus e_j}$ for all $i \neq j$. Furthermore, by Lemma 25, $[\![D]\!]$ is a matchgate so by Lemma 26, $[\![D]\!] = \Gamma$.

Now if $\Gamma_{\mathbf{0}} \neq 1$: First if $\Gamma_{\mathbf{0}} \neq 0$ then $\Gamma' = \frac{1}{\Gamma_{\mathbf{0}}}\Gamma$ is of the right form so we can obtain D by adding a floating edge of weight $\Gamma_{\mathbf{0}}$ to the diagram D' representing Γ'. The last case is $\Gamma_{\mathbf{0}} = 0$, then if $\Gamma = 0$ we can represent Γ by any diagram and a floating black node, else let β be such that $\Gamma_\beta \neq 0$, then Γ' defined as $\Gamma'_\alpha = \Gamma_{\alpha \oplus \beta}$ satisfies $\Gamma'_{\mathbf{0}} \neq 0$ and there is a diagram D' representing Γ'. A diagram D representing Γ is then obtained by plugging binary black nodes to the outputs of D' corresponding to the positions where β is 1. ◀

Notice that since **Match** is a sub-pro of **Qubit**, the completeness proof of Section 3 still holds in **Match**. It provides us with a universal and complete graphical language for matchgates.

▶ **Theorem 28.** pW *is universal and complete for* ***Match****.*

5 Further Work

The proper definition and axiomatisation of the p**W**-calculus pave the way to diverse investigations of the connection between combinatorics and quantum computing. We briefly outline in this last section some very promising directions that are the subjects of ongoing research.

5.1 New Simulation Techniques for Quantum Circuits

The identification of a fragment of the ZX-calculus exactly corresponding to the efficiently simulable Clifford gate [3] allows to design new rewrite-based simulation technics for quantum circuits introduced in [16]. Those algorithms have a parametrised complexity which is polynomial in the number of Clifford gated but exponential in the number of T-gates (a gate outside of the Clifford fragment sufficient to reach approximate universality).

Similarly, we have identified an efficiently simulable fragment of ZW-calculus: the p**W**-calculus exactly corresponding to matchgates. Adding the swap gate to p**W** we obtain another fragment of ZW which is exactly the fermionic ZW-calculus introduced in [11]. This calculus is universal for **Qubit** modulo an encoding trick: the dual-rail encoding. Equivalently, LFM is ZW where white nodes are contrived to have even arities, so adding arity one white nodes (corresponding to preparing $|+\rangle$ states) is enough to recover the full ZW-calculus, which is universal for **Qubits**. This situation suggests the possibility of designing rewrite-based simulation algorithms with complexities parametrised by the number of swap gates and/or $|+\rangle$ preparation. It would lead to a brand new kind of quantum simulation technics exploiting the combinatorial structure of matchgate and directly connected to classical perfect matching counting algorithms.

5.2 Combinatorial Interpretation of Full ZW-Calculus

In Section 2, we provided a combinatorial interpretation of p**W**-diagrams *via* perfect matchings in planar graphs. This combinatorial approach directly extends to LFM-calculus *via* perfect matchings in arbitrary graphs (which is #P-complete). Furthermore, we can also extend the interpretation to the full ZW-calculus, where white nodes can have arbitrary arities. To do so, we must consider hypergraph matchings, i.e., subsets of hyperedges covering each vertex exactly once. The arbitrary arity white nodes here play the role of hyperedges, and the black nodes, the role of vertices. Thus, the interpretation of ZW-scalars is the number of hypergraph matchings of the hypergraph underlying the diagram. Notice that hypergraph matching is also presented as the set cover problem in the literature. The full ZW-calculus could offer new perspectives on set cover in the same way that p**W** does for perfect matchings. In particular, some reduction results appear to have very clear diagrammatical proofs.

Aside from perfect matchings, it seems that graphical languages can encode other counting problems on graphs or hypergraphs. Designing such languages could shed a new tensorial/diagrammatical light on the corresponding combinatorial problems. Those approaches are reminiscent of the recent ZH-based algorithm for #Sat, introduced in [17] and related works linking graphical languages and counting complexity [9, 10]. Conversely, the question of applying similar combinatorial interpretations to other graphical languages as ZX-calculus [7], or ZH-calculus [1] is also worth being investigated.

5.3 Towards a Diagrammatic Approach of Perfect Matching Counting

In Section 2, we used the Fletcher-Kasteleyn-Temperley algorithm as a black box to compute the interpretation of p**W**-scalars in polynomial time. However, it seems possible to achieve the same result with purely diagrammatical technics. In fact, applying the rewriting strategy described in Section 3 to a scalar reduces it to a normal form from which we can directly read the interpretation. It seems very probable that this requires only a polynomial number of rewrites.

This provides a way to count perfect matchings without referring to Pfaffian computation, and conversely, it gives a new algorithm to compute Pfaffians based on rewriting.

The FKT algorithm only applies to a specific class of graphs, called Pfaffian graphs, i.e., the graphs admitting a Pfaffian orientation. In particular, all planar graphs are Pafaffian [14]. It seems that Pfaffian orientiation are directly connected to the behavior of fermionic swap and their lack of naturality which introduces -1 weights in the edges. More generally, all graphs not containing $K_{3,3}$ are Pfaffian [27, 18] (we recall that planar graphs are precisely the graphs not containing neither $K_{3,3}$ nor K_5 as minors). Moreover, there also exists a polynomial time algorithm for K_5-minor-free graphs [21] based on graph decomposition. There is a large amount of work in perspective, re-expressing in diagrammatic terms those different variations and understanding adequately how our rewriting rules could encode the minor constraints.

Formalising and implementing those different algorithms is the object of ongoing work. The main difficulty is to identify the suitable data structures to manipulate the topological data of a given diagram, equivalently, the specific planar embedding of the corresponding graph.

References

1 M Backens and A Kissinger. Zh: A complete graphical calculus for quantum computations involving classical non-linearity. *Electronic Proceedings in Theoretical Computer Science*, 287:23–42, 2019.

2 Miriam Backens. A new holant dichotomy inspired by quantum computation. *arXiv preprint*, 2017. arXiv:1702.00767.

3 Miriam Backens, Simon Perdrix, and Quanlong Wang. Towards a minimal stabilizer zx-calculus. *Log. Methods Comput. Sci.*, 16(4), 2020. doi:10.23638/LMCS-16(4:19)2020.

4 Jin-yi Cai, Vinay Choudhary, and Pinyan Lu. On the theory of matchgate computations. *Theory Comput. Syst.*, 45(1):108–132, 2009. doi:10.1007/s00224-007-9092-8.

5 Jin-Yi Cai and Aaron Gorenstein. Matchgates revisited. *Theory Comput.*, 10:167–197, 2014. doi:10.4086/toc.2014.v010a007.

6 Titouan Carette. *Wielding the ZX-calculus, Flexsymmetry, Mixed States, and Scalable Notations. (Manier le ZX-calcul, flexsymétrie, systèmes ouverts et limandes)*. PhD thesis, University of Lorraine, Nancy, France, 2021. URL: https://tel.archives-ouvertes.fr/tel-03468027.

7 Bob Coecke and Ross Duncan. Interacting quantum observables. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 298–310. Springer, 2008. doi:10.1007/978-3-540-70583-3_25.

8 Bob Coecke and Aleks Kissinger. The compositional structure of multipartite quantum entanglement. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part II*, volume 6199 of *Lecture Notes in Computer Science*, pages 297–308. Springer, 2010. doi:10.1007/978-3-642-14162-1_25.

9 Niel de Beaudrap, Aleks Kissinger, and Konstantinos Meichanetzidis. Tensor network rewriting strategies for satisfiability and counting. In Benoît Valiron, Shane Mansfield, Pablo Arrighi, and Prakash Panangaden, editors, *Proceedings 17th International Conference on Quantum Physics and Logic, QPL 2020, Paris, France, June 2 - 6, 2020*, volume 340 of *EPTCS*, pages 46–59, 2020. doi:10.4204/EPTCS.340.3.

10 Niel de Beaudrap, Aleks Kissinger, and John van de Wetering. Circuit extraction for zx-diagrams can be #p-hard. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 119:1–119:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.119.

11 Giovanni de Felice, Amar Hadzihasanovic, and Kang Feng Ng. A diagrammatic calculus of fermionic quantum circuits. *Log. Methods Comput. Sci.*, 15(3), 2019. doi:10.23638/LMCS-15(3:26)2019.

12 Amar Hadzihasanovic. A diagrammatic axiomatisation for qubit entanglement. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 573–584. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.59.

13 Amar Hadzihasanovic, Kang Feng Ng, and Quanlong Wang. Two complete axiomatisations of pure-state qubit quantum computing. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 502–511. ACM, 2018. doi:10.1145/3209108.3209128.

14 P. W. Kasteleyn. Graph theory and crystal physics. *Graph Theory and Theoretical Physics*, 1967.

15 P.W. Kasteleyn. The statistics of dimers on a lattice: I. the number of dimer arrangements on a quadratic lattice. *Physica*, 27(12):1209–1225, 1961. doi:10.1016/0031-8914(61)90063-5.

16 Aleks Kissinger, John van de Wetering, and Renaud Vilmart. Classical simulation of quantum circuits with partial and graphical stabiliser decompositions. In François Le Gall and Tomoyuki Morimae, editors, *17th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2022, July 11-15, 2022, Urbana Champaign, Illinois, USA*, volume 232 of *LIPIcs*, pages 5:1–5:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.TQC.2022.5.

17 Tuomas Laakkonen, Konstantinos Meichanetzidis, and John van de Wetering. A graphical #sat algorithm for formulae with small clause density. *CoRR*, abs/2212.08048, 2022. doi:10.48550/arXiv.2212.08048.

18 Charles HC Little. An extension of kasteleyn's method of enumerating the 1-factors of planar graphs. In *Combinatorial Mathematics: Proceedings of the Second Australian Conference*, pages 63–72. Springer, 1974.

19 Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

20 Razin A Shaikh, Quanlong Wang, and Richie Yeung. How to sum and exponentiate hamiltonians in zxw calculus. *arXiv preprint*, 2022. arXiv:2212.04462.

21 Simon Straub, Thomas Thierauf, and Fabian Wagner. Counting the number of perfect matchings in k5-free graphs. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*, pages 66–77. IEEE Computer Society, 2014. doi:10.1109/CCC.2014.15.

22 H. N. V. Temperley and Michael E. Fisher. Dimer problem in statistical mechanics-an exact result. *The Philosophical Magazine: A Journal of Theoretical Experimental and Applied Physics*, 6(68):1061–1063, 1961. doi:10.1080/14786436108243366.

23 Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979. doi:10.1016/0304-3975(79)90044-6.

24 Leslie G. Valiant. Quantum circuits that can be simulated classically in polynomial time. *SIAM J. Comput.*, 31(4):1229–1254, 2002. doi:10.1137/S0097539700377025.

25 Leslie G Valiant. Holographic algorithms. *SIAM Journal on Computing*, 37(5):1565–1594, 2008.

26 John van de Wetering. Zx-calculus for the working quantum computer scientist. *arXiv preprint*, 2020. arXiv:2012.13966.

27 Vijay V. Vazirani. NC algorithms for computing the number of perfect matchings in k3, 3-free graphs and related problems. In Rolf G. Karlsson and Andrzej Lingas, editors, *SWAT 88, 1st Scandinavian Workshop on Algorithm Theory, Halmstad, Sweden, July 5-8, 1988, Proceedings*, volume 318 of *Lecture Notes in Computer Science*, pages 233–242. Springer, 1988. doi:10.1007/3-540-19487-8_27.

28 Quanlong Wang and Richie Yeung. Differentiating and integrating zx diagrams. *arXiv preprint*, 2022. arXiv:2201.13250.

Deterministic Regular Functions of Infinite Words

Olivier Carton
Université Paris Cité, CNRS, IRIF, F-75013, France
Institut Universitaire de France, Paris, France

Gaëtan Douéneau-Tabot
Université Paris Cité, CNRS, IRIF, F-75013, France
Direction générale de l'armement – Ingénierie des projets, Paris, France

Emmanuel Filiot
Université libre de Bruxelles & F.R.S.-FNRS, Brussels, Belgium

Sarah Winter
Université libre de Bruxelles & F.R.S.-FNRS, Brussels, Belgium

Abstract

Regular functions of infinite words are (partial) functions realized by deterministic two-way transducers with *infinite* look-ahead. Equivalently, Alur et. al. have shown that they correspond to functions realized by deterministic Muller streaming string transducers, and to functions defined by MSO-transductions. Regular functions are however not computable in general (for a classical extension of Turing computability to infinite inputs), and we consider in this paper the class of *deterministic regular functions* of infinite words, realized by deterministic two-way transducers *without* look-ahead. We prove that it is a well-behaved class of functions: they are computable, closed under composition, characterized by the *guarded* fragment of MSO-transductions, by deterministic Büchi streaming string transducers, by deterministic two-way transducers with *finite* look-ahead, and by finite compositions of sequential functions and one fixed basic function called map-copy-reverse.

2012 ACM Subject Classification Theory of computation → Transducers

Keywords and phrases infinite words, streaming string transducers, two-way transducers, monadic second-order logic, look-aheads, factorization forests

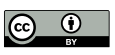
Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.121

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Funding This work was partially supported by the Fonds National de la Recherche Scientifique – F.R.S.-FNRS – under the MIS project F451019F.

1 Introduction

Transducers extend automata with output mechanisms, turning finite state machines from language acceptors to computational models for functions. Inspired by a seminal work by Engelfriet and Hoogeboom [22], the last decade has seen an increasing interest in characterizing the class of functions defined by deterministic two-way transducers over finite words (2-dT), now called the class of *regular functions of finite words*. This class admits several (effective) characterizations: it corresponds to the functions definable by MSO-transductions [22], by an MSO-based logic on origin graphs [15], by an extension of regular expressions called combinator expressions [4, 5, 20], and computed by *copyless streaming string transducers* (SST) (a deterministic one-way model which uses registers to store and update partial output words [2]). Moreover, the class of regular functions over finite words is closed under composition [11], and it has decidable equivalence problem [25].



▶ **Example 1.1.** Let Σ be an alphabet, the function map-copy-reverse : $(\Sigma \uplus \{|\})^* \to (\Sigma \uplus \{|\})^*$ takes any word of the form $u_1|\dots|u_n$ where each u_i is $|$-free, and outputs $u_1|\widetilde{u_1}|\dots|u_n|\widetilde{u_n}$, where $\widetilde{u_i}$ is the mirror image of u_i. The function map-copy-reverse is regular.

Regular functions can also be characterized as the compositions of sequential functions (functions computed by deterministic *one-way* finite transducers) and map-copy-reverse [6].

Regular functions of infinite words. The class of regular functions has been extended to infinite words in [3], and defined as the class of functions definable by MSO-transductions over infinite words. Equivalently, they have been shown to be the functions realized by deterministic two-way transducers with regular look-ahead, and by streaming string transducers with a Muller selection condition (the register holding the final output word is selected depending on the set of states seen infinitely often). As for finite words, regular functions of infinite words are closed under composition, and have decidable equivalence problem [3].

▶ **Example 1.2.** Let $\Sigma = \{a, b, c\}$ be an alphabet, and consider the function double : $\Sigma^\omega \to \Sigma^\omega$ which behaves like the identity function except that any occurrence of a is replaced by aa if there exists a b in the future of that occurence. For example, $(ab)^\omega$ is mapped to $(aab)^\omega$ and $acaab(ac)^\omega$ is mapped to $aacaaaab(ac)^\omega$. The function double is regular, as it can be realized by a one-way transducer which, when reading an a, uses regular look-aheads to determine whether there exists a b or not in the future, and produces either a or aa accordingly.

▶ **Example 1.3.** Let $\Sigma' = \{a, b, 1, 2\}$ and consider the function copy which maps:

- $u_1\sigma_1 u_2 \dots \sigma_n u \mapsto u_1^{\sigma_1}\sigma_1 \dots u_n^{\sigma_n}\sigma_n u$ where $u_1u_2\dots u_n u \in \{a,b\}^\omega$ and $\sigma_1, \dots, \sigma_n \in \{1,2\}$;
- $u_1\sigma_1 \dots u_i\sigma_i \cdots \mapsto u_1^{\sigma_1}\sigma_1 \dots u_i^{\sigma_i}\sigma_i \dots$ (if there are infinitely many $\sigma_i \in \{1,2\}$).

For example, $\mathsf{copy}(ab2a1b^\omega) = abab2a1b^\omega$ and $\mathsf{copy}((a2)^\omega) = (aa2)^\omega$. The function copy is regular, for instance realized by a deterministic two-way transducer which, using two-wayness, makes one or two passes on the blocks u_i, depending on whether they are followed by $\sigma_i = 2$. On the first pass, it always outputs what it reads, so that if no separator in $\{1,2\}$ is ever read again (which means it is reading the infinite suffix u), then it outputs u.

Despite the robustness of the class of regular functions of infinite words, witnessed by its various characterizations and algorithmic properties, they suffer from a severe downside when it comes to computability. Indeed, there are regular functions of infinite words which are *not* computable. At this point, we make clear what is meant by computability, since the input is infinite. We refer the reader to [18, 19] (and the references therein) for a formal definition of computability, and rather give intuitions here. A function f of infinite words is computable if there is a Turing machine with an infinite read-only tape which contains some infinite input word u in the domain of the function, a bidirectional working tape, and a write-only left-to-right output tape, such that by reading longer and longer input prefixes, the machine writes longer and longer prefixes of $f(u)$ on the output tape. Informally, it is an algorithm which takes the input as a stream and is able to produce the output as a stream, so that infinitely often, at least one output symbol is produced. For instance, the function double above is not computable. On reading prefixes of the form ac^n for increasing values of n, it can safely output one a symbol, but not more. Indeed, if it outputs one more a, then it is a wrong output for continuation c^ω, and if it outputs a c, then it is a wrong output for continuation b^ω, as $\mathsf{double}(ac^nc^\omega) = ac^\omega$ and $\mathsf{double}(ac^nb^\omega) = aac^nb^\omega$. Its implementation by a two-way transducer indeed requires an infinite look-ahead to check the absence of a b in the future. On the other hand, copy is realized by a deterministic two-way transducer with *no* look-ahead, so it is computable. So, deterministic two-way transducers

with (infinite) look-ahead, and their equivalent model Muller streaming string transducers, *cannot* be considered as models of computation for infinite word functions. This was observed in [19], where it is shown that the problem of deciding whether a given regular function of infinite words is computable is PSPACE-C. On the other hand, deterministic two-way transducers *without look-ahead* are a proper model of computation for functions of infinite words.

Deterministic regular functions of infinite words. Motivated by the latter observation, the class of functions computed by deterministic two-way transducers *without* look-ahead, coined the class of *deterministic regular functions*, was introduced in [9], where it is shown that they are also equivalently computed by Büchi SST (BSST). In BSST, there is one special designated register $\mathfrak{out}$ in which to write the output word, which is required to be updated with at least one new symbol infinitely often. For example, copy can be implemented by a single-state BSST with two registers $\mathfrak{out}$ and $\mathfrak{r}$, updated as follows. On reading $\sigma \in \{a, b\}$, it performs the updates $\mathfrak{out} \mapsto \mathfrak{out}.\sigma$ and $\mathfrak{r} \mapsto \mathfrak{r}.\sigma$, on reading 1, it does $\mathfrak{out} \mapsto \mathfrak{out}.1$ and $\mathfrak{r} \mapsto \varepsilon$, and on reading 2, it does $\mathfrak{out} \mapsto \mathfrak{out}.\mathfrak{r}.2$ and $\mathfrak{r} \mapsto \varepsilon$.

Several important questions remain on the class of deterministic regular functions, such as whether it is closed under composition, whether it can be logically characterized by a natural fragment of MSO-transductions, and whether they can be obtained as finite compositions of "simple" functions. In this paper, we provide positive answers to these questions.

Contributions. Concerning the class of deterministic regular functions, our main results are:

- its effective closure under composition;
- its characterization by means of finite compositions of sequential functions and an extension of map-copy-reverse to infinite words;
- a logical characterization by a natural syntactic fragment of MSO-transductions, the guarded fragment, called MSOT_g.

An MSO-transduction is defined as an MSO-interpretation, where the predicates of the output word structure, namely the successor and label relations, are defined by MSO formulas with two and one free first-order variables respectively, interpreted over a fixed number of copies of the input. The guarded fragment is defined by a classical restriction (see e.g. [24] and references therein) on the MSO formulas composing the MSO-transduction. They have to be prefixed by an existential quantifier $\exists \mathsf{g}$, where g is a word position, and all quantifiers of the formula are guarded by the guard $x \leq \mathsf{g}$ (and $\forall x \in X, x \leq \mathsf{g}$ for any set variable X). So, guarded MSO formulas on infinite words, only speak about finite prefixes. Consider again the function copy. Two copies of the input are needed to account for potential duplication of the blocks, but the presence or not of a successor edge between two nodes of the output word structure, only depends on local properties, which are definable by guarded MSO formulas. E.g., such a property may be "if position $x+1$ is labeled 2, then there is a successor between the 1st copy of x and the 2nd copy of first position of the block to which x belongs".

In general, guarded MSO formulas can test non-local properties, which is the main source of technical difficulties in the paper. It is illustrated by the next example.

▶ **Example 1.4.** The function replace : $\{0, a, b\}^\omega \multimap \{a, b\}^\omega$ of domain $\mathsf{Dom}(\mathsf{replace}) = \{u \in \{0, a, b\}^\omega : |u|_a{=}\infty \text{ or } |u|_b{=}\infty\}$ and mapping $0^{n_1}\sigma_1 0^{n_2}\sigma_2 \cdots \mapsto \sigma_1{}^{n_1+1}\sigma_2{}^{n_2+1}\cdots$ if $\sigma_i \in \{a, b\}$ and $n_i \in \mathbb{N}$, is deterministic regular. Replacing a zero at position x by a or b depends on the next non-zero symbol in the future of x, which can be arbitrarily faraway, but occurs in a finite prefix if $u \in \mathsf{Dom}(\mathsf{replace})$. This property is expressible with a guarded MSO formula, which defines the position holding this non-zero symbol as a guard.

Proof techniques and additional results. We now give an overview of the proof techniques used to show the logical characterization, along with some other interesting and useful results. We prove that deterministic two-way transducers (2-dT) are expressively equivalent to MSOT_g. The conversion of 2-dT into MSOT_g is standard and follows the same line as [22]. The converse is more involved and requires new techniques. First, we convert MSOT_g-transductions into deterministic two-way transducers with *finite look-ahead* (2-$\mathsf{dT}^{\mathsf{FLA}}$), which account for non-local, but finite, properties, as illustrated before. 2-$\mathsf{dT}^{\mathsf{FLA}}$ are equipped with regular languages of finite words on their transitions, which act as finite look-aheads in the following sense: when the reading head is at some position i of an infinite word u, in some state q, a transition from q with look-ahead L is enabled if there exists a position $j \geq i$, called witness, such that the infix $u[i{:}j]$ starting at position i and ending at position j, belongs to L. If no transition is enabled at state q, the computation fails. To ensure determinism, if several transitions are enabled, only the transition with minimal (i.e. smallest) witness j is triggered, and a disjointness requirement on the look-aheads make sure that this j is unique. The condition to consider only the transition with minimal witness j is crucial to ensure that 2-$\mathsf{dT}^{\mathsf{FLA}}$ define only computable functions. Indeed, a 2-$\mathsf{dT}^{\mathsf{FLA}}$ can be executed as follows: all finite look-aheads, supposed for instance to be finitely represented by DFA, are executed in parallel. By the minimality requirement for j and the disjointness of look-aheads, *as soon as* a prefix is accepted by one look-ahead DFA, the corresponding transition is triggered.

Adding look-aheads to two-way transducers in order to capture MSO-transductions is standard on finite words [22, 14], for example because the "moves" of the MSO-transduction depends on non-local properties. Look-aheads are then directly removed by using the closure under composition of deterministic two-way transducers [11]. Closure under composition of deterministic two-way transducers on infinite words is, to the best of our knowledge, unknown, and instead we give a direct proof of finite look-ahead removal. It is our main technical result: any 2-$\mathsf{dT}^{\mathsf{FLA}}$ is effectively equivalent to some 2-dT. To prove this result, classical techniques, such as Hopcroft-Ullman construction [1] or the tree outline construction [16] do not apply, as they heavily rely on the fact that words are finite. In our setting, we instead use a new technique, based on summarizing the computations of the look-aheads into trees which we prove to be bounded. As a side result of finite look-ahead removal, we prove that 2-dT (and so deterministic regular functions) are closed under composition. Classically, closure under composition of MSO-transductions is direct, by formula substitutions [14]. This technique however does not apply here, as the guarded MSO formulas are not syntactically closed under formula substitution, making the correspondence between MSOT_g and 2-dT crucial to obtain closure under composition of MSO_g-transductions.

Structure of the paper. In Section 2, we introduce the class of deterministic regular functions. In Section 3, we prove its closure under composition and the decomposition result. In Section 4, we introduce guarded MSO-transductions and state the logical characterization. Since its proof is based on a compilation into deterministic two-way transducers with finite look-ahead, we prove in Section 5 how to remove those look-aheads. Finally, we prove the logical characterization in Section 6. All transformations are effective in the paper. Some proofs are only sketched or simply omitted, but the proof details can be found in [10].

2 Deterministic regular functions

In this section, we introduce the class of *deterministic regular functions of infinite words* and recall that it can be described by two computation models: *deterministic two-way transducers* and *deterministic Büchi streaming string transducers*.

Notations. Letters Σ, Γ denote alphabets, i.e. finite sets of letters. The set Σ^* (resp. Σ^+, Σ^ω) denotes the set of finite words (resp. non-empty finite words, infinite words) over the alphabet Σ. Let $\Sigma^\infty := \Sigma^* \cup \Sigma^\omega$. If $u \in \Sigma^\infty$, we let $|u| \in \mathbb{N} \cup \{\infty\}$ be its length, $|u|_\sigma \in \mathbb{N} \cup \{\infty\}$ be the number of occurrences of $\sigma \in \Sigma$ and $u[i] \in \Sigma$ be the i-th letter of u for $1 \leq i \leq |u|$. If $1 \leq i \leq j \leq |u|$, $u[i{:}j]$ stands for $u[i] \cdots u[j]$. We write $u[i{:}]$ for $u[i{:}|u|]$. If $j > |u|$ we let $u[i{:}j] := u[i{:}|u|]$. If $j < i$ we let $u[i{:}j] := \varepsilon$. In this paper, functions are *by default* partial (i.e. possibly with non-total domain). A (partial) function f from S to T is denoted $f : S \rightharpoonup T$, and its domain is denoted $\mathsf{Dom}(f) \subseteq S$. A total function from S to T is denoted $f : S \to T$.

Two-way transducers. Let us recall the syntax of two-way transducers. We consider here that the machines work on infinite words, and have a Büchi acceptance condition.

▶ **Definition 2.1** (Two-way transducer). *A* deterministic two-way transducer (2-dT) *denoted* $\mathcal{T} = (\Sigma, \Gamma, Q, q_0, F, \delta, \lambda)$ *consists of:*

- *an input alphabet* Σ *and an output alphabet* Γ*;*
- *a finite set of states* Q *with an initial state* $q_0 \in Q$ *and a set of final states* $F \subseteq Q$*;*
- *a transition function* $\delta : Q \times (\Sigma \uplus \{\vdash\}) \rightharpoonup Q \times \{\triangleleft, \triangleright\}$*;*
- *an output function* $\lambda : Q \times (\Sigma \uplus \{\vdash\}) \rightharpoonup \Gamma^*$ *with same domain as* δ*.*

A *configuration* of $\mathcal{T}$ over $u \in (\Sigma \cup \{\vdash\})^\infty$ is a tuple (q, i) where $q \in Q$ is the current state and $1 \leq i \leq |u|$ is the current position of the reading head. The *transition relation* $\to$ is defined as follows. Given a configuration (q, i), let $(q', \star) := \delta(q, u[i])$. Then $(q, i) \to (q', i')$ whenever either $\star = \triangleleft$ and $i' = i - 1$ (move left), or $\star = \triangleright$ and $i' = i + 1$ (move right). A *run* over u is a (finite or infinite) sequence of consecutive configurations $(q_1, i_1) \to (q_2, i_2) \to \cdots$.

Now, we define the infinite output produced by $\mathcal{T}$ when given the infinite word $u \in \Sigma^\omega$ as input. First, we let $u[0] := \vdash$, i.e. we force the symbol $\vdash$ to be used to mark the beginning of the input. An *accepting* run is an infinite run that starts in $(q_0, 0)$, visits infinitely often configurations of the form (q, i) with $q \in F$ and such that $i_n \to \infty$ when $n \to \infty$ (without this last condition, the transducer may enter an infinite loop without reading its whole input). The partial function $f : \Sigma^\omega \rightharpoonup \Gamma^\omega$ computed by $\mathcal{T}$ is defined as follows. Let $u \in \Sigma^\omega$ be such that there exists a (unique) accepting run $(q_0^u, i_0^u) \to (q_1^u, i_1^u) \to \cdots$ labelled by $\vdash u$. Let $v := \prod_{j=1}^{\infty} \lambda(q_j^u, (\vdash u)[i_j^u]) \in \Gamma^* \cup \Gamma^\omega$ be the concatenation of the outputs produced along this run. If $v \in \Gamma^\omega$, we define $f(u) := v$. Otherwise $f(u)$ is undefined.

▶ **Definition 2.2.** *The class of* deterministic regular functions of infinite words *is the class of (partial) functions computed by deterministic two-way transducers.*

We have explained in Example 1.3 how to compute the function copy using a 2-dT (without look-aheads). Observe that the function replace from Example 1.4 can be computed in a similar fashion. Hence both functions are deterministic regular.

▶ **Example 2.3.** Let us extend the function map-copy-reverse of Example 1.1 to infinite words. Let Σ be an alphabet, we define map-copy-reverse : $(\Sigma \uplus \{|\})^\omega \to (\Sigma \uplus \{|\})^\omega$ as follows:

- map-copy-reverse$(u_1|u_2|\cdots) := u_1|\widetilde{u_1}|u_2|\widetilde{u_2}|\cdots$ with $u_i \in \Sigma^*$ for all $i \geq 0$;
- map-copy-reverse$(u_1|\cdots|u_n|u) := u_1|\widetilde{u_1}|\cdots|u_n|\widetilde{u_n}|u$ for $u_i \in \Sigma^*$ and $u \in \Sigma^\omega$.

This function is deterministic regular since we can build a 2-dT that processes twice each $|$-free factor (or only once for the last infinite one if it exists).

Büchi Streaming String Transducers. Now, we describe a model of a one-way machine with registers which captures deterministic regular functions of infinite words. Over finite words, it is well-known that deterministic two-way transducers are equivalent to *copyless streaming string transducers* [2]. A similar equivalence holds for the class of *regular functions of infinite words*, which can equivalently be described by *deterministic two-way transducers with regular look-aheads* or *copyless streaming string transducers with Muller conditions* [3]. However, Muller conditions enable to check regular properties of the infinite input, and thus describe functions which are not (Turing) computable [3]. Now, let us recall the model of *Büchi deterministic streaming string transducer* (BSST), introduced by Carton and Douéneau-Tabot in [9], that captures exactly the class of deterministic regular functions.

Formally, a Büchi deterministic streaming string transducer consists of a one-way deterministic automaton with a finite set $\mathfrak{R}$ of registers that store words from Γ^*. We use a distinguished register $\mathfrak{out}$ to store the output produced when reading an infinite word. The registers are modified when reading the input using *substitutions*, i.e. mappings $\mathfrak{R} \to (\Gamma \uplus \mathfrak{R})^*$. We denote by $\mathcal{S}_{\mathfrak{R}}^{\Gamma}$ the set of these substitutions. They can be extended morphically from $(\Gamma \uplus \mathfrak{R})^*$ to $(\Gamma \uplus \mathfrak{R})^*$ by preserving the elements of Γ.

▶ **Example 2.4** (Substitutions). Let $\mathfrak{R} = \{\mathfrak{r}, \mathfrak{s}\}$ and $\Gamma = \{b\}$. Consider $\tau_1 := \mathfrak{r} \mapsto b, \mathfrak{s} \mapsto b\mathfrak{r}\mathfrak{s}b$ and $\tau_2 := \mathfrak{r} \mapsto \mathfrak{r}b, \mathfrak{s} \mapsto \mathfrak{r}\mathfrak{s}$, then $\tau_1 \circ \tau_2(\mathfrak{r}) = \tau_1(\mathfrak{r}b) = bb$ and $\tau_1 \circ \tau_2(\mathfrak{s}) = \tau_1(\mathfrak{r}\mathfrak{s}) = bb\mathfrak{r}\mathfrak{s}b$.

▶ **Definition 2.5.** *A* Büchi deterministic streaming string transducer (BSST) *denoted by* $\mathcal{T} = (\Sigma, \Gamma, Q, F, q_0, \delta, \mathfrak{R}, \mathfrak{out}, \lambda)$ *consists of:*

- *a finite input (resp. output) alphabet* Σ *(resp.* Γ*);*
- *a finite set of states* Q *with* $q_0 \in Q$ *initial and* $F \subseteq Q$ *final;*
- *a transition function* $\delta : Q \times \Sigma \rightharpoonup Q$*;*
- *a finite set of registers* $\mathfrak{R}$ *with a distinguished output register* $\mathfrak{out} \in \mathfrak{R}$*;*
- *an update function* $\lambda : Q \times \Sigma \rightharpoonup \mathcal{S}_{\mathfrak{R}}^{\Gamma}$ *such that for all* $(q, \sigma) \in \mathsf{Dom}(\lambda) = \mathsf{Dom}(\delta)$*:*
  - $\lambda(q, \sigma)(\mathfrak{out}) = \mathfrak{out} \cdots$*;*
  - *there is no other occurrence of* $\mathfrak{out}$ *among the* $\lambda(q, \sigma)(\mathfrak{r})$ *for* $\mathfrak{r} \in \mathfrak{R}$*.*

This machine defines a partial function $f : \Sigma^\omega \rightharpoonup \Gamma^\omega$ as follows. For $i \geq 0$ let $q_i^u := \delta(q_0, u[1{:}i])$ (when defined). For $i \geq 1$, we let $\lambda_i^u := \lambda(q_{i-1}^u, u[i])$ (when defined) and $\lambda_0^u(\mathfrak{r}) = \varepsilon$ for all $\mathfrak{r} \in \mathfrak{R}$. For $i \geq 0$, let $[\![\cdot]\!]_i^u := \lambda_0^u \circ \cdots \circ \lambda_i^u$. By construction $[\![\mathfrak{out}]\!]_i^u$ is a prefix of $[\![\mathfrak{out}]\!]_{i+1}^u$ (when defined). If $[\![\mathfrak{out}]\!]_i^u$ is defined for all $i \geq 0$, q_i^u is a state of F infinitely often, and $|[\![\mathfrak{out}]\!]_i^u| \to +\infty$, then we let $f(u) := \bigvee_i [\![\mathfrak{out}]\!]_i^u$ (the symbol $\vee$ is used to denote the unique $v \in \Gamma^\omega$ such that $[\![\mathfrak{out}]\!]_i^u$ is a prefix of v for all $i \geq 0$). Otherwise $f(u)$ is undefined.

▶ **Example 2.6.** The function replace from Example 1.4 can be computed by a BSST. For all $i \geq 1$, it crosses the block 0^{n_i} and computes 1^{n_i} and 2^{n_i} in two registers. Once it sees σ_i it adds in $\mathfrak{out}$ the register storing ${\sigma_i}^{n_i}$.

▶ **Definition 2.7** (Copyless, bounded copy). *We say that a substitution* $\tau \in \mathcal{S}_{\mathfrak{R}}^{B}$ *is* copyless *(resp.* K-bounded*) if for all* $\mathfrak{r} \in \mathfrak{R}$*,* $\mathfrak{r}$ *occurs at most once in* $\{\tau(\mathfrak{s}) : \mathfrak{s} \in \mathfrak{R}\}$ *(resp. for all* $\mathfrak{r}, \mathfrak{s} \in \mathfrak{R}$*,* $\mathfrak{r}$ *occurs at most* K *times in* $\tau(\mathfrak{s})$*). We say that a* BSST $\mathcal{T} = (\Sigma, \Gamma, Q, q_0, \delta, \mathfrak{R}, \mathfrak{out}, \lambda)$ *is* copyless *(resp.* K-bounded*) if for all* $u \in \Sigma^\omega$ *and* $i \leq j$ *such that* $\lambda_i^u \circ \cdots \circ \lambda_j^u$ *is defined, this substitution is copyless (resp.* K*-bounded).*

▶ Remark 2.8. The composition of two copyless substitutions is copyless, hence a BSST is copyless as soon as $\lambda(q, \sigma)$ is copyless for all $q \in Q$ and $\sigma \in \Sigma$. However, K-boundedness is not necessarily preserved under composition.

Observe that the BSST described in Example 2.6 is copyless. Now, we recall the result of Carton and Douéneau-Tabot that proves equivalence between two-way transducers, copyless, and bounded copy Büchi deterministic streaming string transducers.

▶ **Theorem 2.9** ([9, Theorem 3.7]). *The following machines compute the same class of partial functions over infinite words:*

1. *deterministic two-way transducers* (2-dT)*;*
2. *K-bounded deterministic Büchi streaming string transducers* (K*-bounded* BSST)*;*
3. *copyless deterministic Büchi streaming string transducers* (*copyless* BSST)*.*

Furthermore, all the conversions are effective.

▶ Remark 2.10. The original proof of [9] which transforms a 2-dT into a BSST only considers machines where all states are final. Nevertheless, the proof can easily be adapted to transducers with non-final states. Furthermore, given a BSST (possibly with non-final states) one can build an equivalent BSST where all states are final by [9, Lemma D.1] (the Büchi conditions are hidden in the fact that the output must be infinite). All in all, all the models (with all states final or not) exactly capture the class of deterministic regular functions.

Finally, we recall the domains of deterministic regular functions. We say that a language is *Büchi deterministic* if it is accepted by a deterministic Büchi automaton (see e.g. [26]).

▶ **Proposition 2.11** ([9]). *If f is deterministic regular, then* Dom(f) *is Büchi deterministic.*

3 Composition and decomposition theorems

In this section, we show that deterministic regular functions are closed under composition, and that conversely they can be written as the composition of some "basic" functions.

It is known since [11] (resp. [3]) that the class of regular functions of finite (resp. infinite) words is closed under composition. We transport this result to deterministic regular functions of infinite words in Theorem 3.1. However, its proof is not an immediate extension of the regular case, and it illustrates the main difficulty of this paper: since look-aheads are not allowed, it is complex for a 2-dT to check if some property happens *after* its current position.

▶ **Theorem 3.1.** *Deterministic regular functions are (effectively) closed under composition.*

Proof idea. The approach is to compose the two transducers directly (using a product construction); the difficulty in the composition of two computations arises when one transducer is moving forward and the other backward. In that case, we need to rewind the computation of the transducer that moves backward by one computation step.

To recover the previous configuration look-ahead comes in handy. As mentioned above, (infinite) look-aheads are not permitted, but we use a weaker form of *finite* look-aheads (to be introduced in Section 5) which does not increase the expressiveness of deterministic two-way transducers over infinite words (and can be effectively removed), see Theorem 5.2. Finite look-aheads account for non-local but finite properties. The look-ahead we define basically re-traces the computation that the two-way transducer has taken so far. Note that this is indeed a finite property as only a prefix of the input has been visited by the computation of the two-way transducer. ◀

As an easy consequence of Theorem 3.1, let us observe that deterministic regular functions (effectively) preserve Büchi deterministic languages by inverse image. Analogue results hold for regular functions of finite (resp. infinite) words with regular languages.

▶ **Proposition 3.2.** *If $f : \Sigma^\omega \rightharpoonup \Gamma^\omega$ is deterministic regular and $L \subseteq \Gamma^\omega$ is Büchi deterministic, then $f^{-1}(L) \subseteq \Sigma^\omega$ is (effectively) Büchi deterministic.*

Proof. The function $f \circ \mathsf{id}_L$ (where $\mathsf{id}_L : \Gamma^\omega \rightharpoonup \Gamma^\omega$ is the identity function restricted to L) is deterministic regular. Its domain $f^{-1}(L)$ is Büchi deterministic by Proposition 2.11. ◀

Let us now focus on the converse of Theorem 3.1, i.e. showing that any deterministic regular function can be written as a composition of "basic" functions. As mentioned in introduction, regular functions of finite words can be written as compositions of map-copy-reverse (see Example 1.1) and sequential functions (computed by one-way transducers).

▶ **Theorem 3.3** ([6, Theorem 13])**.** *Over finite words, a function is regular if and only if it can (effectively) be written as a composition of* map-copy-reverse *and sequential functions.*

To state our similar result for deterministic regular functions of infinite words, we first recall formally the definition of sequential functions of infinite words.

▶ **Definition 3.4** (Sequential functions)**.** *A* deterministic one-way transducer *is a* 2-dT $(\Sigma, \Gamma, Q, q_0, F, \delta, \lambda)$ *such that for all $q \in Q$ and $\sigma \in (\Sigma \uplus \{\vdash\})$, $\delta(q, \sigma)$ has shape $(\_, \triangleright)$ (when defined). The class of (partial) functions over infinite words computed by one-way deterministic transducers is called* sequential functions of infinite words.

▶ **Example 3.5.** Any function that replaces some letter of its input by another letter is sequential. The functions replace and map-copy-reverse of Examples 1.4 and 2.3 are *not* sequential (this can be shown using a pumping argument). Observe that replace can be written as the composition of: a sequential function that replaces each $\sigma_i \in \{1, 2\}$ by $\sigma_i|$, the function map-copy-reverse, and finally a sequential function that uses the first copy of each block to determine the value of σ_i, and transforms the (mirror) second copy accordingly.

Now, we state the decomposition result, that also uses map-copy-reverse from Example 2.3. Its proof is somehow technical and it illustrates once more the main difficulty of this paper: deterministic regular functions are not able to check many properties about the "future".

▶ **Theorem 3.6.** *A function is deterministic regular if and only if it can (effectively) be written as a composition of* map-copy-reverse *and sequential functions of infinite words.*

Proof idea. In the case of finite words, the proofs of [7, 6] rely on Simon's factorization forests theorem [27]. They first build a factorization forest, and then use its structure to simulate the runs of a transducer. Furthermore, over finite words, such forests can be computed by a *rational function*, which is a composition of sequential functions and map-copy-reverse. We follow a similar proof sketch for infinite words, but the main issue is that factorization forests can no longer be computed by a composition of sequential functions and map-copy-reverse (their structure may depend on regular properties of the input). Thus we use instead a weakened version of forests, introduced by Colcombet under the name of *forward Ramseyan splits* [12]. Such splits can be computed with a sequential function. Our new techniques show how to simulate the runs of a transducer by using a forward Ramseyan split. ◀

4 Guarded MSO-transductions

In this section, we define the logic MSO over finite and infinite words, as well as MSO-transductions, and its guarded fragment. We also state the logical characterization of deterministic regular functions (Theorem 4.8).

MSO on infinite words. Infinite words over Σ are seen as structures of domain $\mathbb{N}$, over the signature $\mathcal{W}_\Sigma = \{S(x,y), (\sigma(x))_{\sigma\in\Sigma}\}$ which consists of the successor predicate $S(x,y)$, naturally interpreted as the successor over $\mathbb{N}$, and unary predicates $\sigma(x)$ for all $\sigma \in \Sigma$, interpreted as the set of positions labelled σ. Given an infinite word $u \in \Sigma^\omega$, we denote by G_u the structure it induces, and just u when it is clear that u denotes the structure G_u.

Monadic second-order formulas are defined as first-order logic formulas, which can additionally use quantifiers $\exists X, \forall X$ over sets of positions, and membership atomic formulas of the form $x \in X$, where x is a first-order variable while X is a set variable. We denote by $\mathsf{MSO}[\Sigma, S, \leq]$ (or just MSO when the predicates are clear from the context), the set of monadic second-order formulas over the word signature $\mathcal{W}_\Sigma$ extended with the order predicate $\leq$ (interpreted by the natural order on $\mathbb{N}$). It is well-known that the predicate $\leq$ is syntactic sugar. The semantics is defined as expected (details can be found in [28, 14] for instance). For a formula ϕ with sets of free first-order and set variables $\overline{x}, \overline{X}$ (we use the tuple notation which implicitly assumes an order between variables), we may write it $\phi(\overline{x}, \overline{X})$ to explicit the free variables of ϕ. We also denote by $Free(\phi)$ the free (first-order and set) variables of ϕ.Given a word w, an n-tuple of positions $\overline{p}$ of w and an m-tuple $\overline{P}$ of sets of positions of w, we write $w \models \phi(\overline{p}, \overline{P})$ to mean that the structure induced by w is a model of ϕ under assignments $\overline{p}$ and $\overline{P}$.

▶ **Example 4.1.** The formula $\mathsf{first}(x) = \forall y \cdot \neg S(y,x)$ is satisfied by any word and position x such that x is the first position to the left.

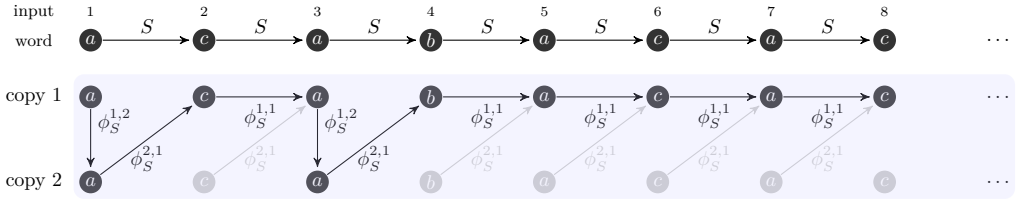
Over an alphabet Σ, any *closed* formula $\phi \in \mathsf{MSO}$ *defines* a regular language $L_\phi = \{u \in \Sigma^\omega \mid u \models \phi\}$. By Büchi-Elgot-Trakhtenbrot's theorem [29, 8, 21], it is known MSO defines precisely the class of regular languages over alphabet Σ: for any language L over Σ, L is regular if and only if $L = L_\phi$ for some $\phi \in \mathsf{MSO}$. MSO formulas can also be interpreted over finite word structures, whose domains are the (finite) set of word positions. It is also well-known that a language of finite words is regular iff it is MSO-definable.

MSO-transductions of infinite words. MSO-transductions define transformations of graph structures, and have been studied in the context of finite words by Engelfriet and Hoogeboom in [22] (see also [14] for a more recent introduction to MSO-transductions). The main result of [22] is a Büchi-like theorem: a function of finite words is MSO-definable if and only if it is regular (i.e. recognizable by a deterministic two-way transducer). This result was then lifted to functions of infinite words in [3], but deterministic two-way transducers may need infinite look-aheads to capture the full expressive power of MSO-transductions.

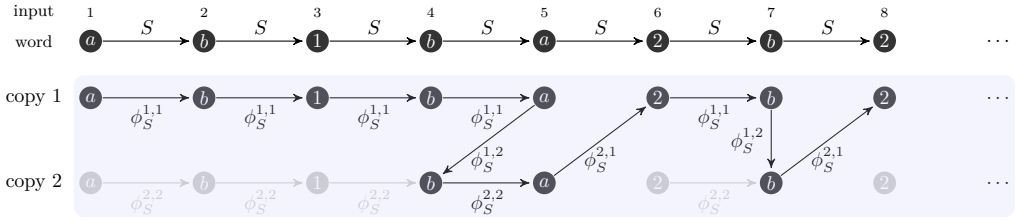
In an MSO-transduction, the output word structure is defined via an MSO interpretation over a fixed number k of copies of the input word (seen as a structure). Therefore, the nodes of the output word are copies 1 to k of the nodes of the input word. Output nodes are pairs (i,c) (often denoted i^c), for every copy c and input node i.

The output label and successor predicates are defined by MSO formulas with one and two free first-order variables respectively, interpreted over the input structure. For instance, over the output alphabet $\Gamma = \{a,b\}$, to set all the output labels to a, one just specifies the formulas $\phi_a^c(x) = \top$ and $\phi_b^c(x) = \bot$ for all copies c. The output successor predicate relates input nodes of possibly different copies, and is therefore defined by formulas of the form $\phi_S^{c,d}(x,y)$, indexed by copies $c, d \in \{1, \ldots, k\}$.

Finally, there is one distinguished copy c_0 together with a formula $\phi_{\text{fst}}^{c_0}(x)$, which must be satisfied by at most one node x. Intuitively, if the output structure is a word, this formula defines the first node of the output word. The domain of the output structure is composed of



(a) Input and output structures of the MSO-transduction for the function double of Example 1.2 on input word $acab(ac)^\omega$.



(b) Input and output structures of the MSO-transduction for the function copy of Example 1.3 on input word $ab1ba2b2\dots$.

Figure 1

all nodes that can be reached from the initial node x^{c_0} by following multiple successor edges. In general, the output structure of an input word u by an MSO-transduction $\mathcal{T}$ might not be an infinite word structure, in which case u is not in the domain of the function defined by $\mathcal{T}$.

Formally, an MSO-*transduction* over an input alphabet Σ and output alphabet Γ is a tuple $\mathcal{T} = (k, (\phi^c_\gamma(x))_{1\le c\le k, \gamma\in\Gamma}, (\phi^{c,d}_S(x,y))_{1\le c,d\le k}, c_0, \phi^{c_0}_{\text{fst}}(x))$ where $k \in \mathbb{N}\setminus\{0\}$, $1 \le c_0 \le k$ and for all input $u \in \Sigma^\omega$, there is at most one position i such that $u \models \phi^{c_0}_{\text{fst}}(i)$. We may omit c_0 in the tuple above.

We now formally define the semantics of MSO-transductions. Let $u \in \Sigma^\omega$ and $N \subseteq \mathbb{N}\times\{1,\dots,k\}$. We first define the set of output nodes that can be reached from N in zero or more steps. We let $\mathsf{Post}^0_u(N) = N$ and for all $\ell > 0$,

$$\mathsf{Post}^\ell_u(N) = \{j^d \mid \exists i^c \in \mathsf{Post}^{\ell-1}_u(N) \cdot u \models \phi^{c,d}_S(i,j)\} \text{ and } \mathsf{Post}^*_u(N) = \bigcup_{\ell\ge 0} \mathsf{Post}^\ell_u(N)$$

Given an MSO-transduction $\mathcal{T}$ as above, and input word $u \in \Sigma^\omega$, the *output structure*, denoted $\mathcal{T}(u)$, is the structure over signature $\mathcal{W}_\Gamma$ defined by the following interpretation:

- the domain is $D = \mathsf{Post}^*_u(\{i^{c_0} \mid u \models \phi^{c_0}_{\text{fst}}(i)\})$ (note that the argument of Post^*_u is either empty or a singleton)
- a node $i^c \in D$ is labelled $\gamma \in \Gamma$ if $u \models \phi^c_\gamma(i)$
- a node j^d is a successor of a node i^c if $u \models \phi^{c,d}_S(i,j)$.

The output structure $\mathcal{T}(u)$ may not be a word structure. For instance, a node might have multiple labels, $\mathcal{T}(u)$ may contain cycles, or branching. So we restrict semantically the function defined by $\mathcal{T}$ to word structures. Formally, the *function defined by* $\mathcal{T}$ is the function $[\![\mathcal{T}]\!] : \Sigma^\omega \rightharpoonup \Gamma^\omega$ whose graph is:

$$\{(u,v) \in \Sigma^\omega \times \Gamma^\omega \mid G_v \text{ (the structure associated with } v) \text{ is isomorphic to } \mathcal{T}(u)\}$$

We denote by MSOT the set of MSO-transductions and say that a function $f : \Sigma^\omega \rightharpoonup \Gamma^\omega$ is MSOT-definable if $f = [\![\mathcal{T}]\!]$ for some $\mathcal{T} \in \mathsf{MSOT}$.

▶ **Example 4.2.** We consider again the function double of Example 1.2, illustrated on Figure 1a and show how to define it with an MSO-transduction. Since some a must be duplicated, two copies are needed, so $k = 2$. Labels are preserved: $\phi^c_\sigma(x) = \sigma(x)$ for all $c \in \{1, 2\}$ and $\sigma \in \Sigma$. The first copy c_0 is 1, and $\phi^{c_0}_{\text{fst}}(x) = \mathsf{first}(x)$. The successor formulas distinguish if there is a b in the future or not. First, from the 2nd to the 1st copy, there is always a successor relation from a node to its successor in copy 1: $\phi^{2,1}_S(x, y) = S(x, y)$. There is a successor from x^1 to y^2 if $x = y$, x is labelled a and there is a b in the remaining infinite suffix starting at x: $\phi^{1,2}_S(x, y) = a(x) \wedge (x = y) \wedge \exists z \cdot x \leq z \wedge b(z)$. On the first copy, it depends on the label of the input: $\phi^{1,1}_S(x, y) = S(x, y) \wedge (a(x) \to (\forall z \geq x \cdot \neg b(z))$. On the second copy, there is never a predicate edge: $\phi^{2,2}_S = \bot$. On Figure 1a, the interpretation of those formulas is depicted, in bold if they are part of the output word, in light grey otherwise. One can see that the output structure induced by all the descendants of the first node (by the transitive closure of the successor relation) is isomorphic to the structure $G_{aacaab(ac)^\omega}$.

The function copy of Example 1.3, illustrated in Figure 1b, is definable by an MSOT with two copies ($k = 2$). Formulas $\phi^{c_0}_{\text{fst}}$ and ϕ^c_σ are the same as for double. Then:

$$\phi^{1,1}_S(x, y) = \phi^{2,2}_S(x, y) = S(x, y) \wedge \neg 2(y) \qquad \phi^{2,1}_S(x, y) = S(x, y) \wedge 2(y)$$
$$\phi^{1,2}_S(x, y) = \exists \mathsf{g} \cdot y < x \leq \mathsf{g} \wedge 2(\mathsf{g}) \wedge \forall z \leq y \cdot (S(z, y) \to (1(z) \vee 2(z))) \wedge$$
$$\forall t \cdot (y \leq t \leq x) \to (a(t) \vee b(t))$$

The class of regular functions of infinite words has been defined in [3] as the class of functions recognizable by deterministic two-way transducers extended with regular (infinite) look-ahead: to take a transition, such a transducer can query a regular oracle on the infinite current suffix (given as a deterministic parity automaton for example). Equivalently, this class corresponds to functions recognizable by (deterministic) SST: they work as BSST but are not forced to output the content of a special register infinitely often. Instead, the output of a run depends on the set of states that are seen infinitely often along that run, and can be "computed" only once the infinite input has been processed (see [3]) for more details. The following provides a logical characterization of the class of regular functions:

▶ **Theorem 4.3** ([3]). *A function $f : \Sigma^\omega \rightharpoonup \Gamma^\omega$ is regular if and only if it is* MSOT*-definable.*

The definition of MSOT in [3] is slightly different, but equivalent, to the definition we take in this paper.

Guarded MSO-transductions of infinite words. Guarded MSO formulas are a syntactical restriction of MSO formulas. This restriction requires all the free variables and quantifiers to be guarded by a first-order variable g, in the sense that quantifiers should only talk about positions which are *before* g (i.e. smaller than g). Intuitively, the satisfiability of a guarded formula on an infinite word only depends on the finite prefix up to position g. Formally, given two first-order variables x and g, we let $\mathsf{G}(x, \mathsf{g})$ be the formula $x \leq \mathsf{g}$ (x is guarded by g), and for a set variable X, we let $\mathsf{G}(X, \mathsf{g})$ be the formula $\forall x \in X, \mathsf{G}(x, \mathsf{g})$. Then, an MSO formula φ is *guarded by some variable g* if it is equal to $\psi(\mathsf{g}) \wedge \bigwedge_{\alpha \in Free(\psi)} \mathsf{G}(\alpha, \mathsf{g})$ for some $\psi(\mathsf{g})$ such that all its quantified subformulas, i.e. subformulas of the form $QX \cdot \psi'$ or $Qx \cdot \psi'$ for some $Q \in \{\exists, \forall\}$, are in one of the following forms:

$$(1)\ \forall x \cdot \mathsf{G}(x, \mathsf{g}) \to \zeta \quad (2)\ \exists x \cdot \mathsf{G}(x, \mathsf{g}) \wedge \zeta \quad (3)\ \forall X \cdot \mathsf{G}(X, \mathsf{g}) \to \zeta \quad (4)\ \exists X \cdot \mathsf{G}(X, \mathsf{g}) \wedge \zeta$$

An MSO formula is *guarded* if it is of the form $\exists \mathsf{g} \cdot \varphi$ where φ is guarded by g. We denote by MSO_g the set of guarded MSO-formulas. For conciseness, we may write $\forall x : \mathsf{g} \cdot \zeta$ instead of $\forall x \cdot \mathsf{G}(x, \mathsf{g}) \to \zeta$, and $\exists x : \mathsf{g} \cdot \zeta$ instead of $\exists x \cdot \mathsf{G}(x, \mathsf{g}) \wedge \zeta$ (and similarly for set variables).

▶ **Example 4.4.** All the formulas of the MSO-transduction of Example 4.2 defining the function double are guarded, or trivially equivalent to a guarded formula. For example, the formula $\mathsf{first}(x)$ is equivalent to the guarded formula $\exists \mathsf{g} \cdot x \leq \mathsf{g} \wedge \forall y \leq \mathsf{g} \cdot \neg S(y, x)$.

The order predicate $x \leq y$ is definable by the guarded formula $\exists \mathsf{g} \cdot x \leq \mathsf{g} \wedge y \leq \mathsf{g} \wedge y = \mathsf{g}$. Since $\neg(x \leq y)$ is equivalent to $y \leq x \wedge y \neq x$, we easily get that any MSO_g-formula ϕ is equivalent to an MSO_g-formula ψ in which the order predicate is only used to guard quantifiers, by existentially quantifying a global guard, guarding all the local guards used to define the atomic formulas of the form $z \leq t$ occurring in ϕ (assumed to occur positively).

▶ Remark 4.5. MSO_g formulas only talk about prefixes, in the following sense: If $\varphi = \exists \mathsf{g} \cdot \psi(\mathsf{g})$ is a closed guarded formula and $w \in \Sigma^\omega$, then $w \models \varphi$ if and only if there exists a finite prefix u of w such that $u \models \psi(\ell)$, where ℓ is the last position of u. This allows us to get the following immediate characterization: A language $L \subseteq \Sigma^\omega$ is MSO_g-definable if and only if there exists a regular language $F \subseteq \Sigma^*$ such that $L = F\Sigma^\omega$.

▶ **Definition 4.6** (Guarded MSO-transductions)**.** *A* guarded MSO-transduction *(*MSOT_g*) is an* MSO*-transduction all formulas of which are guarded.*

▶ **Example 4.7.** As explained in Example 4.4, all formulas of the MSO-transduction of Example 4.2 defining double are guarded, or trivially equivalent to a guarded formula.

We can now state the logical characterization of deterministic regular functions:

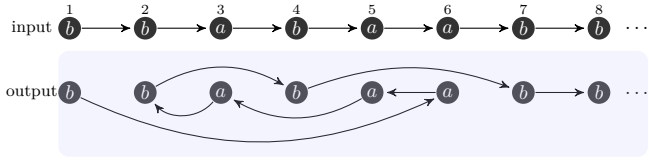
▶ **Theorem 4.8** (Logical characterization)**.** *A function $f : \Sigma^\omega \rightharpoonup \Gamma^\omega$ is deterministic regular if and only if it is* MSOT_g*-definable.*

The proof is given in Section 6. As an application of this result, since deterministic regular functions are (effectively) closed under composition by Theorem 3.1, we obtain that MSOT_g are (effectively) closed under composition as well. This is a well-known result for MSOT over finite strings [22], infinite strings [3] and more generally any structure [13], yet with purely logic-based and direct proofs, while we use here involved automata-based arguments (look-ahead removal). Indeed, composition closure of MSOT is obtained by formula substitutions. To compose two MSOT $\mathcal{T}_2 \circ \mathcal{T}_1$, the predicates occurring in $\mathcal{T}_2$ are substituted by their definition in $\mathcal{T}_1$. Such a direct proof idea does not work in the guarded fragment MSOT_g, as guarded formulas are not closed under negation.

Guarded MSO-transductions with order. We conclude this section by discussing an alternative definition of MSO_g-transductions, denoted $\mathsf{MSOT}_g[\leq]$, where instead of defining the output successor relation, it requires to define the total order $\leq$ of the output structure, with MSO_g formulas. This however allows to define uncomputable functions (in the sense of [18], see also Section 1), as stated by the following proposition:

▶ **Proposition 4.9.** *There exists an* $\mathsf{MSOT}_g[\leq]$ *which defines an uncomputable function.*

To prove this proposition, we show that the following uncomputable function h is definable with $\mathsf{MSOT}_g[\leq]$. Let $\Sigma = \Gamma = \{a, b\}$ and $\mathsf{er}_b : \Sigma^* \to \Sigma^*$ the (erasing) morphism defined by $\mathsf{er}_b(a) = a$ and $\mathsf{er}_b(b) = \varepsilon$. The function $h : \Sigma^\omega \rightharpoonup \Gamma^\omega$ is defined on inputs of the form bub^ω, for $u \in \{a, b\}^*$, by $h(bub^\omega) = b\mathsf{er}_b(u)b^\omega$. It can be shown that h is definable by a 1-copy $\mathsf{MSOT}_g[\leq]$. An example of output structure on input $bbabaab^\omega$ is given below (we depict only the successor predicate and not the order):



The output order formula for instance states that the b occurrences are ordered according to their input order, while the a occurrences are ordered in reverse. Moreover, it states that the first b occurrence is smaller than any a occurence, and that any a occurrence is smaller that any b occurrence but the first one.

Without the guarded restriction, it is known that two definitions of MSOT, with successor or with order, both define the class of regular functions of infinite words.

5 Two-way transducers with finite look-ahead

We extend deterministic two-way transducers with finite look-ahead. Transitions are additionally labelled by a regular language of *finite words*, called (finite) look-ahead. A transition with look-ahead L can only be taken if the remainder of the input sequence has a prefix that belongs to L. Such a finite prefix is called a *look-ahead witness* for L. To ensure determinism, if several look-aheads succeed, it is required that there is a unique shortest look-ahead witness. The transducer follows the transition which minimizes the length of the witness. If no look-aheads succeed the computation fails.

▶ **Definition 5.1** (Finite look-ahead). *A* deterministic two-way transducer with finite look-ahead (2-dT$^{\mathsf{FLA}}$) *is a tuple* $\mathcal{T} = (\Sigma, \Gamma, Q, q_0, F, \delta, \lambda)$ *where* $\Sigma, \Gamma, Q, q_0, F, \lambda$ *are defined as for deterministic two-way transducers w/o look-ahead,* δ *is a transition function* $Q \times (\Sigma \uplus \{\vdash\}) \times \mathcal{R}^*(\Sigma)) \rightharpoonup Q \times \{\triangleright, \triangleleft\}$ *where* $\mathcal{R}^*(\Sigma)$ *is the set of all regular languages of* finite *words over* Σ*. The function* δ *is required to have finite domain. The* look-ahead *for a transition* $(q, \sigma, L) \mapsto (q, d)$ *is* L*. Furthermore, we require that if* $\delta(q, \sigma, L)$ *and* $\delta(q, \sigma, L')$ *are defined, then* $L \cap L' = \emptyset$ *for all* $L, L' \in \mathcal{R}^*(\Sigma)$*,* $q \in Q$ *and* $\sigma \in \Sigma$*. Finally, it is assumed that the look-ahead languages are represented by deterministic finite automata.*

The semantics of a deterministic two-way transducer with finite look-ahead remains unchanged compared to the model without look-ahead. The only difference in the presence of look-ahead is when a transition is enabled: A transition with look-ahead L can only be taken if the remainder of the input sequence has a prefix that belongs to L. Formally, in a configuration (q, i) over input u, a transition of the form $\delta(q, \sigma, L)$ where $L \subseteq \Sigma^*$ is enabled if $u[i] = \sigma$ and there exists some $i < j$ such that $u[i{+}1{:}j] \in L$. The word $u[i{+}1{:}j]$ is called a *witness* for L. To ensure determinism, whenever the transducer is in a configuration (q, i), if several look-aheads $L_1, \ldots, L_k$ are enabled, the triggered transition is the unique (ensured by the disjointness requirement) transition with shortest witness.

Removing finite look-ahead. We know that infinite look-ahead is strictly more expressive than finite look-ahead. The natural question is how much expressiveness is gained by adding finite look-ahead to deterministic two-way transducers w/o look-ahead. As already explained in the introduction, any function defined by such a transducer is (Turing machine) computable: A Turing machine can memorize where it is in the input, verify which look-ahead succeeds, and continue the computation from the memorized position. A two-way transducer does not have the ability to memorize a position arbitrarily far away in the input. Hence, verifying (in the absence of some look-ahead "oracle") that some finite prefix of the remainder of

the input is a witness for some look-ahead and returning to a specific position becomes a problem to be solved. This problem is not unique to two-way transducers over infinite words, it also appears when some regular property of the remainder of a finite input word must be checked and subsequently the two-way transducer must return to the position it has been in before checking the property. On finite words, this task can be handled using the Hopcroft-Ullman [1] or the improved tree-outline construction [16]. However, these constructions rely on the fact that the input word is finite. We prove that this task can be also accomplished for infinite words using different techniques.

In the following, we show that no expressiveness is gained by allowing finite look-ahead.

▶ **Theorem 5.2** (Finite look-ahead removal). *Given a* 2-$\mathsf{dT}^{\mathsf{FLA}}$*, one can effectively construct an equivalent* 2-dT*.*

Proof sketch. The proof is divided into two parts. The main part is to translate a given 2-$\mathsf{dT}^{\mathsf{FLA}}$ into an equivalent BSST with bounded copy. We then use Theorem 2.9 to obtain an equivalent 2-dT. Given a deterministic two-way transducer *without* look-ahead, the standard approach to obtain an equivalent SST is to simulate the right-to-right runs of the deterministic two-way transducer on the so-far read prefix of the infinite input, store their outputs in registers and compose these registers in the right way (with the output of the "main" left-to-right run) to re-create the output of the two-way transducer. Since the two-way transducer is deterministic there is a global bound on the number of different right-to-right runs on any prefix of the input. The constructions presented in [3, 17, 9] are all built on this idea. In [2], equivalence between SST and two-way transducers on finite words is shown but the work exhibits no direct translation.

Our goal is to design a similar construction for deterministic two-way transducers *with* finite look-ahead. The main difficulty is that there is no global bound on the number of different runs that can occur on a prefix, if one takes additionally into account all the runs of the look-ahead automata that have been triggered so far. Alternatively, such a transducer can be seen as a non-deterministic transducer, which guesses which finite look-ahead will succeed and verifies it a posteriori, but there can be many look-ahead automata running in parallel.

Hence, we extend the standard construction to go from a deterministic two-way transducer to an SST by additionally taking all the possible look-ahead choices into account. This approach results in a tree structure representation of the possible runs (similar to a standard run-tree of a non-deterministic automaton, here the non-determinism is the look-ahead choice). A branch in such a tree corresponds to a possible run and the nodes additionally contain information to detect when look-ahead choices succeed or are doomed to fail. The size of the tree representations is kept bounded by sharing information and a relevant pruning strategy. The strategy takes care of removing branches whose look-ahead choices *cannot succeed* and (prefixes of) branches where the look-ahead choices *already have succeeded*. Applying this construction to a deterministic two-way transducer without look-ahead yields the standard translation construction. ◀

6 Logic-transducer correspondence: proof of Theorem 4.8

In this section, we give an overview of the proof of the logical characterization of Theorem 4.8. We first prove that any deterministic regular function is MSOT_g-definable. The proof is standard and uses same ideas as for regular functions of finite words [22] and infinite words [3].

▶ **Lemma 6.1.** *If a function* $f : \Sigma^\omega \multimap \Gamma^\omega$ *is deterministic regular, then it is* MSOT_g*-definable.*

Proof. The main idea is to define in MSOT_g the runs of a 2-$\mathtt{dT}$. Each copy of the MSOT_g represents a state of the 2-$\mathtt{dT}$, and there is a successor edges between node x^p to node y^q, where x, y are input positions and p, q are states, if and only if there exists a *finite* run from configuration (p, x) to configuration (q, y) which produces output symbols only in configuration (p, x) and (q, y). This property can be expressed by an MSO_g formula. ◀

Proving the converse of Lemma 6.1 is more involved. We first go to an intermediate model with MSO instructions, in the spirit of [22], called *jumping* MSO_g-transducers, proved to be equivalent to 2-$\mathtt{dT}$. It is a finite-state model which can (i) test MSO_g properties of the current position (called look-around), (ii) test safety constraints defined by MSO formulas, and (iii) jump from one position to another one with binary MSO_g formulas. Formally, it has a finite set of states (all final), and transitions are of the form $p \xrightarrow{\phi_{\mathrm{la}}(x)|w,\phi_{\mathrm{mv}}(x,y),\phi_{\mathrm{sf}}(x)} q$ where p, q are states, $\phi_{\mathrm{la}}, \phi_{\mathrm{mv}}$ are MSO_g formulas, ϕ_{sf} is an MSO formula, and w is a finite word. Look-around occurring on transitions with same source state are assumed to be pairwise disjoint (their conjunction is not satisfiable). The initial configuration is $(q_0, 0)$ where q_0 is the initial state. Whenever it is in a configuration (q, i), over an infinite word $u \in \Sigma^\omega$, it enables the transitions whose look-around $\phi_{\mathrm{la}}(i)$ holds on u, and select the transition with shortest witness. Call t this transition. It triggers t only if there exists j such that $\phi_{\mathrm{mv}}(i, j)$ holds and for all $k \geq i$, $u[{:}k] \models \phi_{\mathrm{sf}}(i)$ (otherwise the computation fails). It then outputs γ and moves to some position j such that $\phi_{\mathrm{mv}}(i, j)$ holds. Note that there could be several j, and therefore several runs on the same input in general. We thus make the following *assumption*, which can be described informally as follows: for any reachable configuration of the transducer from the initial configuration, there is always a unique j. Formally, for all infinite sequence of configurations $(q_0, i_0 = 0)(q_1, i_1)(q_2, i_2)\ldots$, for all $k \geq 0$, for any transition t triggered from configuration (q_k, i_k) to (q_{k+1}, i_{k+1}), if $\phi_{\mathrm{mv}}(x, y)$ is the jumping formula of t, then i_{k+1} is the unique position such that $\phi_{\mathrm{mv}}(i_k, i_{k+1})$ holds. As for two-way transducers, a sequence of configurations $(q_0, i_0 = 0)(q_1, i_1)\ldots$ is *accepting* if $\lim_{k\to\infty} i_k = \infty$ and it produces an infinite word.

We show that this model defines deterministic regular functions:

▶ **Lemma 6.2.** *Any jumping* MSO_g*-transducer defines a deterministic regular function.*

Sketch of proof. The proof goes in two steps. First, it is shown that jumping MSO_g-transducers are equivalent to *walking* MSO_g-transducers, i.e. MSO_g-transducers which moves (backward or forward) between successive positions. This step is standard (it appears e.g. in [22] in the non-guarded setting). Then, walking MSO_g-transducers are shown to be equivalent to an extension of 2-$\mathtt{dT}$ with finite look-around and safety constraints, then proved to be equivalent to 2-$\mathtt{dT}$ by transforming look-arounds into look-aheads, and then removing look-aheads (based on the techniques of Section 5) and safety constraints. ◀

▶ **Lemma 6.3.** *Any* MSO_g*-transduction is equivalent to a jumping* MSO_g*-transducer.*

Proof. Let $\mathcal{T} = (k, (\phi^c_\gamma)_{c\in[k],\gamma\in\Gamma}, (\phi^{c,d}_S)_{c,d\in[k]}, \phi^{c_0}_{\mathrm{fst}}(x))$ be an MSOT_g defining f. We construct a jumping MSO_g-transducer $\mathcal{T}'$ equivalent to $\mathcal{T}$. The set of states of $\mathcal{T}'$ is $\{0, 1\ldots, k\}$. In state 0, $\mathcal{T}'$ first jumps to the initial position, i.e. the position y which satisfies $\phi^{c_0}_{\mathrm{fst}}(y)$ and moves to state c_0. This is done by a transition going from state 0 to state c_0, with the trivial look-around and safety constraint $\top$, and the move $\phi_{\mathrm{mv}}(x, y) := \mathsf{first}(x) \wedge \phi^{c_0}_{\mathrm{fst}}(y)$. Then, it follows the successor relation of $\mathcal{T}$, and uses the label formulas to determine which label to output. Using safety constraints, $\mathcal{T}'$ also makes sure that the output graph structure is a word structure. In particular, they express that for any reachable node, there is exactly one label and at most one successor. There is no need to check that there is *at least* one

successor, because if there is none, then the run of $\mathcal{T}'$ stops and the input is not accepted, which is consistent with the semantics of $\mathcal{T}$ (the input is also rejected by $\mathcal{T}$ in that case). There is also no need to check that there is no cycle, because if there is some, then $\mathcal{T}'$ will never visit all input positions, and hence the input will be rejected, which is again consistent with the semantics of $\mathcal{T}$. Formally, for all copies $c, d \in \{1, \dots, k\}$ and output label γ, since $\phi_S^{c,d}(x,y)$ and $\phi_\gamma(x)$ are guarded, there are of the form $\phi_S^{c,d}(x,y) = \exists \mathsf{g} \cdot \psi_S(x,y,\mathsf{g})$ and $\phi_\gamma(x) = \exists \mathsf{g} \cdot \psi_\gamma(x,\mathsf{g})$. Then we add the following transition to $\mathcal{T}'$, from c to d:

$$c \xrightarrow{\phi_{\mathrm{la}}(x) := \exists \mathsf{g} \exists z \leq \mathsf{g} \cdot \psi_S^{c,d}(x,z,\mathsf{g}) \wedge \psi_\gamma^c(x,\mathsf{g}) \wedge \mathsf{disj}_{c,d,\gamma}(x,\mathsf{g}) \mid \gamma, \phi_{\mathrm{mv}}(x,y) := \phi_S^{c,d}(x,y), \phi_{\mathrm{sf}}(x)} d$$

in which $\mathsf{disj}_{c,d,\gamma}(x,\mathsf{g}) = \forall \mathsf{g}' \leq \mathsf{g} \cdot \bigwedge_{\gamma' \neq \gamma} \neg \psi_{\gamma'}^c(x,\mathsf{g}') \wedge \bigwedge_{d' \neq d} \forall z' \leq \mathsf{g}' \cdot \neg \psi_S^{c,d'}(x,z',\mathsf{g}')$ ensures disjointness of the look-around, and $\phi_{\mathrm{sf}}(x)$ equals

$$\begin{array}{ll} (\bigwedge_{d' \neq d} \forall y \cdot \neg \phi_S^{c,d'}(x,y)) \wedge & \text{no successor of } x \text{ in any copy } d' \neq d \\ (\forall y \forall y' \cdot (\phi_S^{c,c}(x,y) \wedge \phi_S^{c,c}(x,y')) \rightarrow y = y') \wedge & \text{at most one successor of } x \text{ in copy } c \\ (\bigwedge_{\gamma' \neq \gamma} \neg \phi_{\gamma'}^c(x) & \text{no other label for } x \end{array}$$

At this point, we remind the reader that safety constraints are not required to be defined by guarded formulas, as they are regular properties of finite words. However, the look-around and jumping formulas must be guarded, and it is indeed the case in the transition above.

Finally, note that $\mathcal{T}'$ satisfies the requirement that on infinite sequences of configurations $(q_0, i_0) \dots$, for all $k \geq 0$, i_{k+1} is the unique successor of i_k by the jumping formula. Indeed, if a sequence of configurations of $\mathcal{T}'$ is infinite, it implies that all safety constraints are satisfied, and they precisely make sure that there is no branching. ◀

As a corollary of Lemmas 6.3 and 6.2, we obtain the converse direction of Theorem 4.8:

▶ **Corollary 6.4.** *Any* MSOT_g*-definable function* f *is deterministic regular.*

7 Conclusion

In this paper, we have shown that the class of deterministic regular functions is characterized by computational models such as deterministic two-way transducers, deterministic two-way transducers with *finite* (regular) look-aheads, Büchi SST, by the logical formalism of *guarded* MSO-transductions, and by finite compositions of sequential functions and map-copy-reverse. The transformations between those models are effective. We have also shown that it is closed under composition, by extending to infinite words the known composition closure of deterministic two-way transducers, yet with new proof techniques. It is also conjectured that the class of deterministic regular functions is equal to the class of *continuous* regular functions (for the Cantor topology). It is already known that it includes the continuous letter-to-letter rational functions [23] and the strictly larger class of continuous rational functions [9]. All this, together with the fact that deterministic regular functions are computable, unlike regular functions, shows the robustness of this class.

References

1 Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. A general theory of translation. *Mathematical Systems Theory*, 3(3):193–221, 1969.

2 Rajeev Alur and Pavol Cerný. Expressiveness of streaming string transducers. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010*, volume 8 of *LIPIcs*, pages 1–12. Schloss Dagstuhl, 2010.

3 Rajeev Alur, Emmanuel Filiot, and Ashutosh Trivedi. Regular transformations of infinite strings. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012*, pages 65–74. IEEE Computer Society, 2012.

4 Rajeev Alur, Adam Freilich, and Mukund Raghothaman. Regular combinators for string transformations. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, page 9. ACM, 2014.

5 Nicolas Baudru and Pierre-Alain Reynier. From two-way transducers to regular function expressions. In *International Conference on Developments in Language Theory*, pages 96–108. Springer, 2018.

6 Mikołaj Bojańczyk and Rafał Stefański. Single-use automata and transducers for infinite alphabets. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

7 Mikołaj Bojańczyk. Polyregular Functions, 2018. doi:10.48550/arXiv.1810.08760.

8 J. R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6(1–6):66–92, 1960.

9 Olivier Carton and Gaëtan Douéneau-Tabot. Continuous rational functions are deterministic regular. In *47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022*, 2022.

10 Olivier Carton, Gaëtan Douéneau-Tabot, Emmanuel Filiot, and Sarah Winter. Deterministic regular functions of infinite words. *CoRR*, abs/2302.06672, 2023. doi:10.48550/arXiv.2302.06672.

11 Michal P. Chytil and Vojtěch Jákl. Serial composition of 2-way finite-state transducers and simple programs on strings. In *4th International Colloquium on Automata, Languages, and Programming, ICALP 1977*, pages 135–147. Springer, 1977.

12 Thomas Colcombet. A combinatorial theorem for trees. In *34th International Colloquium on Automata, Languages, and Programming, ICALP 2007*, 2007.

13 Bruno Courcelle. Monadic second-order definable graph transductions: A survey. *Theor. Comput. Sci.*, 126:53–75, 1994.

14 Bruno Courcelle and Joost Engelfriet. *Graph structure and monadic second-order logic: a language-theoretic approach*, volume 138. Cambridge University Press, 2012.

15 Luc Dartois, Emmanuel Filiot, and Nathan Lhote. Logics for word transductions with synthesis. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 295–304. ACM, 2018.

16 Luc Dartois, Paulin Fournier, Ismaël Jecker, and Nathan Lhote. On reversible transducers. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017*, volume 80 of *LIPIcs*, pages 113:1–113:12. Schloss Dagstuhl, 2017.

17 Luc Dartois, Ismaël Jecker, and Pierre-Alain Reynier. Aperiodic string transducers. *Int. J. Found. Comput. Sci.*, 29(5):801–824, 2018.

18 Vrunda Dave, Emmanuel Filiot, Shankara Narayanan Krishna, and Nathan Lhote. Synthesis of computable regular functions of infinite words. In *31st International Conference on Concurrency Theory (CONCUR 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

19 Vrunda Dave, Emmanuel Filiot, Shankara Narayanan Krishna, and Nathan Lhote. Synthesis of computable regular functions of infinite words. *Log. Methods Comput. Sci.*, 18(2), 2022.

20 Vrunda Dave, Paul Gastin, and Shankara Narayanan Krishna. Regular transducer expressions for regular transformations. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 315–324. ACM, 2018.

21 C. C. Elgot. Decision problems of finite automata design and related arithmetics. *In Transactions of the American Mathematical Society*, 98(1):21–51, 1961.

22 Joost Engelfriet and Hendrik Jan Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Transactions on Computational Logic (TOCL)*, 2(2):216–254, 2001.

23 Emmanuel Filiot and Sarah Winter. Synthesizing computable functions from rational specifications over infinite words. In *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual Conference*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

24 Erich Grädel. On the restraining power of guards. *J. Symb. Log.*, 64(4):1719–1742, 1999.

25 Eitan M Gurari. The equivalence problem for deterministic two-way sequential transducers is decidable. *SIAM Journal on Computing*, 11(3):448–452, 1982.

26 Dominique Perrin and Jean-Éric Pin. *Infinite words: automata, semigroups, logic and games*. Academic Press, 2004.

27 Imre Simon. Factorization forests of finite height. *Theor. Comput. Sci.*, 72(1):65–94, 1990. doi:10.1016/0304-3975(90)90047-L.

28 Wolfgang Thomas. Languages, automata, and logic. In *Handbook of formal languages*, pages 389–455. Springer, 1997.

29 Boris Avraamovich Trakhtenbrot. Finite automata and logic of monadic predicates (in Russian). *Dokl. Akad. Nauk SSSR*, 140:326–329, 1961.

Characterising Memory in Infinite Games

Antonio Casares
LaBRI, Université de Bordeaux, France

Pierre Ohlmann
University of Warsaw, Poland

Abstract

This paper is concerned with games of infinite duration played over potentially infinite graphs. Recently, Ohlmann (TheoretiCS 2023) presented a characterisation of objectives admitting optimal positional strategies, by means of universal graphs: an objective is positional if and only if it admits well-ordered monotone universal graphs. We extend Ohlmann's characterisation to encompass (finite or infinite) memory upper bounds.

We prove that objectives admitting optimal strategies with ε-memory less than m (a memory that cannot be updated when reading an ε-edge) are exactly those which admit well-founded monotone universal graphs whose antichains have size bounded by m. We also give a characterisation of chromatic memory by means of appropriate universal structures. Our results apply to finite as well as infinite memory bounds (for instance, to objectives with finite but unbounded memory, or with countable memory strategies).

We illustrate the applicability of our framework by carrying out a few case studies, we provide examples witnessing limitations of our approach, and we discuss general closure properties which follow from our results.

2012 ACM Subject Classification Theory of computation → Verification by model checking

Keywords and phrases Infinite duration games, Memory, Universal graphs

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.122

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: https://arxiv.org/abs/2209.12044

Funding *Pierre Ohlmann*: European Research Council (ERC), grant agreement No 948057 – BOBR.

Acknowledgements We thank Nathanaël Fijalkow, Rémi Morvan and Pierre Vandenhove for stimulating discussions around the topic.

This document contains hyperlinks. Each occurrence of a notion is linked to its definition. On an electronic device, the reader can click on words or symbols (or just hover over them on some PDF readers) to see their definition.

1 Introduction

1.1 Context

Games and strategy complexity. We study zero-sum turn-based games on graphs, in which two players, that we call Eve and Adam, take turns in moving a token along the edges of a given (potentially infinite) edge-coloured directed graph. Vertices of the graph are partitioned into those belonging to Eve and those belonging to Adam. When the token lands in a vertex owned by player X, it is this player who chooses where to move next. This interaction, which is sometimes called a play, goes on in a non-terminating mode, producing an infinite sequence

of colours. We fix in advance an objective W, which is a language of infinite sequences of colours; plays producing a sequence of colours in W are considered to be winning for Eve, and plays that do not satisfy the objective W are winning for the opponent Adam.

In order to achieve their goal, players use strategies, which are representations of the course of all possible plays together with instructions on how to act in each scenario. In this work, we are interested in optimal strategies for Eve, that is, strategies that guarantee a victory whenever this is possible. More precisely, we are interested in the complexity of such strategies, or in other words, in the succinctness of the representation of the space of plays. The simplest strategies are those that assign in advance an outgoing edge to each vertex owned by Eve, and always play along this edge, disregarding all the other features of the play. All the information required to implement such a strategy appears in the game graph itself. These strategies are called positional (or memoryless). However, in some scenarios, playing optimally requires distinguishing different plays that end in the same vertex; one should remember other features of plays. An example of such a game is given in Figure 1.

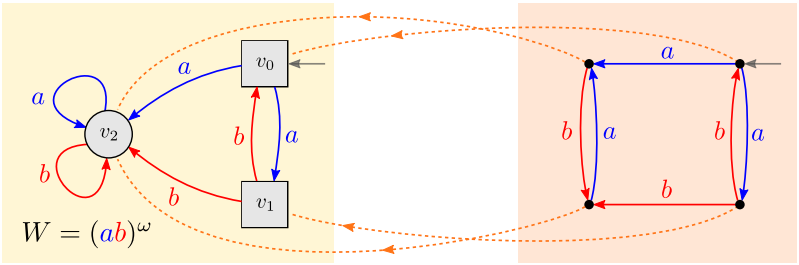


Figure 1 On the left, a game with objective $W = (ab)^\omega$; in words, Eve should ensure that the play alternates between a-edges and b-edges. We represent Eve's vertices as circles and Adam's as squares. On the right, a winning strategy for Eve which uses one state of memory for v_0, one state of memory for v_1, and two states of memory for v_2. Note that two states of memory for v_2 are required here: a positional strategy would always follow the same self-loop and therefore cannot win. One can prove that any game with objective W which is won by Eve can be won even when restricting to strategies with two states of memory, that is, the memory requirements for W is exactly two.

Given an objective W, the question we are interested in is:

"What is the minimal strategy complexity required for Eve to play optimally in all games with objective W?"

Positional objectives and universal graphs. As mentioned above, an important special case is that of positional objectives, those for which Eve does not require any memory to play optimally. A considerable body of research, with both theoretical and practical reach, has been devoted to the study of positionality. By now it is quite well-understood which objectives are positional for both players (bi-positional), thanks to the works of Gimbert and Zielonka [13] for finite game graphs, and of Colcombet and Niwiński [9] for arbitrary game graphs. However, a precise understanding of which objectives are positional for Eve – regardless of the opponent – remains somewhat elusive, even though this is a more relevant question in most application scenarios.

A recent progress in this direction was achieved by Ohlmann [19, 20], using totally ordered monotone universal graphs. Informally, an edge-coloured graph is universal with respect to a given objective W if it satisfies W (all paths satisfy W), and homomorphically embeds all graphs satisfying W. An ordered graph is monotone if its edge relations are monotone:

$$v \geq u \xrightarrow{c} u' \geq v' \implies v \xrightarrow{c} v', \text{ for every colour } c.$$

Ohlmann's main result is a characterisation of positionality (assuming existence of a neutral letter): an objective is positional if and only if it admits well-ordered monotone universal graphs.

From positionality to finite memory. Positional objectives have good theoretical properties and do often arise in applications (in particular, parity, Rabin or energy objectives). It is also true, however, that this class lacks in expressivity and robustness: only a handful of objectives are positional, and very few closure properties are known to hold for positional objectives.

In contrast, objectives admitting optimal finite memory strategies are much more general; for instance they encompass all ω-regular objectives [14] (in fact, it was recently established [3] that optimal finite chromatic memory for both players characterises ω-regularity). Moreover, in practice, finite memory strategies can be implemented by means of a program, and memory bounds for Eve directly translates in space and time required to implement controllers, which gives additional motivation for their systematic study.

Formally, when moving from positionality to finite memory, a few modelling difficulties arise, giving rise to a few different notions. Most prominently, one may or may not include uncoloured edges (ε-edges) in the game, over which the memory state cannot be updated; additionally one may or may not restrict to chromatic memories, meaning those that record only the colours that have appeared so far. We now discuss some implications of these two choices.

It is known that allowing ε-edges impacts the difficulty of the games, in the sense that it may increase the memory required for winning strategies [5, 15, 23], thus leading to two different notions of memory (that we call ε-memory and ε-free memory). It is natural to wonder whether one of the two notions should be preferred over the other. We argue that allowing ε-edges turns out to be more natural in many applications. First, we notice that currently existing characterisations of the memory (for Muller objectives [12] and for topologically closed objectives [8]) do only apply to the case of ε-memory. More importantly, games induced by logical formulas in which players are interpreted as the existential player (controlling existential quantifiers and disjunctions) and the universal player (controlling universal quantifiers and conjunctions) naturally contain ε-edges (along which the memory indeed should not be allowed to be updated).

It was originally conjectured by Kopczyński [15] that chromatic strategies have the same power than non-chromatic ones. It was not until recently that this conjecture was refuted [5], and since then several works have provided new examples separating both notions [6, 17, 18]. It now appears from recent dedicated works [2, 3, 4, 5] that chromatic memory is an interesting notion in itself.

The main challenge in the study of strategy complexity is to prove upper bounds on memory requirements of a given objective. A great feature of Ohlmann's result [20] is that it turns a question about games to a question about graphs, which are easier to handle. Despite its recent introduction, Ohlmann's framework has already proved instrumental for deriving general positionality results in the context of objectives recognised by finite Büchi automata [1].

1.2 Contribution

The present paper builds on the aforementioned work of Ohlmann by extending it to encompass the more general setting of finite (or infinite) memory bounds. This yields the first known characterisation results for objectives with given memory bounds, and provides a (provably) general tool for establishing memory upper bounds.

Doing so requires relaxing from totally to partially ordered graphs, while keeping the same monotonicity requirement, along with some necessary technical adjustments. We essentially prove that the memory of an objective corresponds to the size of antichains in its well-founded monotone universal graph; however it turns out that the precise situation is more intricate. It is summed up in Figure 2 and explained in more details below.

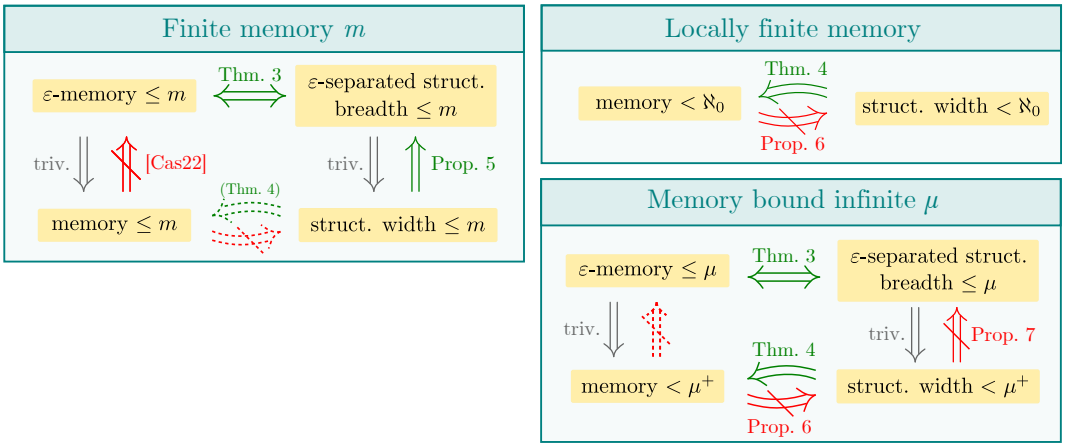


Figure 2 A summary of our main contributions. The three larger boxes correspond to the three regimes encompassed by our results: finite memory, locally finite memory and larger cardinal bounds. Each of the smaller boxes correspond to classes of objectives, where "struct." stands for "existence of well-founded monotone universal graphs"; for example, the box labelled "ε-separated struct. breadth $\leq m$" stands for "existence of ε-separated well-founded monotone universal graphs of breadth $\leq m$". The dotted implications follow from combining other implications in the figure. For $m = 1$, all notions collapse to a single equivalence, which corresponds to Ohlmann's characterisation.

It is convenient for us to define strategies directly as graphs (see Figure 1 for an example, and Section 2 for formal details), which allows us in particular to introduce new classes of objectives such as those admitting locally finite memory, discussed in more details below. For the well-studied case of finite memory bounds, our definition of memory coincides with the usual one.

Universal structures for memory. Our main contribution lies in introducing generalisations of Ohlmann's structures, and proving general connections between existence of such universal structures for a given objective W, and memory bounds for W (Section 3).

The first variant we propose is obtained by relaxing the monotonicity requirement to partially ordered graphs; Theorem 4 states that (potentially infinite) bounds on antichains of a well-founded monotone universal graph translate to memory bounds.

The second variant we propose, called ε-separated structures, is tailored to capture ε-memory. These are monotone graphs where the partial order coincides with $\xrightarrow{\varepsilon}$ and is constrained to be a disjoint union of well-orders; the breadth of such a graph refers to the number of such well-orders. Theorem 3 states that the existence of such universal structures of breadth μ actually characterises having ε-memory $\leq \mu$. Additionally, we define chromatic ε-separated structures (over which each colour acts uniformly), and establish that they capture ε-chromatic memory.

Applying (infinite) Dilworth's theorem we obtain that for finite m, one may turn any monotone graph of width m to an ε-separated one with breadth m (Proposition 5), and therefore in the setting of finite memory, the two notions collapse.

We are able to establish most (but not all) of our results in the more general framework of quantitative valuations; similarly as Ohlmann [20], we show how the notions instantiate in the qualitative case, and how they can be simplified assuming prefix-invariance properties.

Counterexamples for a complete picture. We provide additional negative results which set the limits of our approach, completing the picture in Figure 2. Namely, we build two families of counterexamples that are robust to larger cardinals; these give general separations of ε-free memory and ε-memory (Proposition 7), and negate the possibility of a converse for Theorem 4 (Proposition 6). This supports our informal claim that ε-memory is better behaved than ε-free memory.

Closure properties. Finally, we discuss how our characterisations can be exploited for deriving closure properties on some classes of objectives (Section 4). Apart from Ohlmann's result on lexicographic products of prefix-independent positional objectives [20], no such closure properties are known. Extending Ohlmann's proof to our framework, we prove that if W_1 and W_2 are prefix-independent objectives with ε-memory m_1 and m_2, then their lexicographical product $W_1 \ltimes W_2$ has ε-memory $\leq m_1 m_2$.

We then propose a new class of objectives with good properties, namely, objectives with locally finite memory: for each game, there exists a strategy which uses a finite (though possibly unbounded, even when the game is fixed) amount of memory states for each vertex. These objectives are connected with the theory of well-quasi orders (wqo), since they correspond to monotone universal graphs which are well-founded and have finite antichains. We obtain from the fact that wqo's are closed under intersections, that intersections of objectives with finite ε-memory have locally finite memory; an example is given by conjunctions of energy objectives which have unbounded finite memory even though energy objectives are positional. This hints at a general result, which is not implied by our characterisations but we conjecture to be true, that objectives with finite (possibly unbounded) memory are closed under intersection.

We end our paper by providing yet another application of our characterisation, establishing that prefix-independent Σ_2^0 objectives with finite memory are closed under countable unions. As of today, this is the only known (non-obvious) closure property pertaining to objectives with finite memory.

2 Preliminaries

For a finite or infinite word $w \in C^* \cup C^\omega$ we denote by w_i the letter at position i and by $|w|$ its length.

2.1 Graphs and morphisms

Graphs, paths and trees. A *C-pregraph* G, where C is a (potentially infinite) set of colours, is given by a set of vertices $V(G)$, and a set of coloured directed edges $E(G) \subseteq V(G) \times C \times V(G)$. We write $v \xrightarrow{c} v'$ for an edge (v, c, v'), say that it is outgoing from v, incoming in v' and has colour c. A *C-graph* G is a C-pregraph without sinks: from all $v \in V(G)$ there exists an outgoing edge $v \xrightarrow{c} v' \in E(G)$. We often say c-edges to refer to edges with colour c, and sometimes C'-edges for $C' \subseteq C$ for edges with colour in C'.

A *path* in a pregraph G is a finite or infinite sequence of edges of the form $\pi = (v_0 \xrightarrow{c_0} v_1)(v_1 \xrightarrow{c_1} v_2)\dots$, which for convenience we denote by $\pi = v_0 \xrightarrow{c_0} v_1 \xrightarrow{c_1} \dots$. We say that π is a path from v_0 in G. By convention, the empty path is a path from v_0, for any $v_0 \in V(G)$. If π is a finite path, it is of the form $v_0 \xrightarrow{c_0} v_1 \xrightarrow{c_1} \dots \xrightarrow{c_{n-1}} v_n$, and in this case we say that it is a path from v_0 to v_n in G.

Given a subset $X \subseteq V(G)$ of vertices of a pregraph G, we let $G|_X$ denote the *restriction* of G to X, which is the graph given by $V(G|_X) = X$ and $E(G|_X) = E(G) \cap (X \times C \times X)$. Given a vertex $v \in V(G)$, we let $G[v]$ denote the restriction of G to vertices reachable from v.

A *C-tree* (resp. *C-pretree*) T is a C-graph (resp. C-pregraph) with an identified vertex $t_0 \in V(T)$ called its *root*, with the property that for each $t \in V(T)$, there is a unique path from t_0 to t. Note that since graphs have no sinks, trees are necessarily infinite. We remark that $T[t]$ represents the *subtree rooted at t* (if T is a tree, $T[t]$ is also a tree with root t).

When it is clear from context, we omit C and simply say "a graph" or "a tree".

The *size* of a graph G (and by extension, of a tree) is the cardinality of $V(G)$.

Morphisms. A *morphism* ϕ between two graphs G and H is a map $\phi\colon V(G) \to V(H)$ such that for each edge $v \xrightarrow{c} v' \in E(G)$ it holds that $\phi(v) \xrightarrow{c} \phi(v') \in E(H)$. We write $\phi : G \to H$ in this case, and sometimes say that H embeds G. Note that morphisms preserve paths: if $v_0 \xrightarrow{c_0} v_1 \xrightarrow{c_1} \dots$ is a path in G, then $\phi(v_0) \xrightarrow{c_0} \phi(v_1) \xrightarrow{c_1} \dots$ is a path in H. An *isomorphism* is a bijective morphism whose inverse is a morphism; two graphs are isomorphic if they are connected by an isomorphism (stated differently, they are the same up to renaming the vertices). The composition of two morphisms is a morphism.

2.2 Valuations, games, strategies and memory

Valuations and objectives. A *C-valuation* is a map $\mathrm{val} : C^\omega \to X$, where X is a complete linear order (that is, a total order in which all subsets have both a supremum and an infimum). The *value* $\mathrm{val}_G(v_0)$ of a vertex $v_0 \in V(G)$ in a graph G is the supremum value of infinite paths from v, where the value of an infinite path $\pi = v_0 \xrightarrow{c_0} v_1 \xrightarrow{c_1} \dots$ is defined to be $\mathrm{val}(\pi) = \mathrm{val}(c_0 c_1 \dots)$.

In the important special case where $X = \{\bot, \top\}$, $\bot < \top$, we identify val with $W = \mathrm{val}^{-1}(\bot) \subseteq C^\omega$, and say that val (or W) is an *objective*. In a graph G, a path with value $\bot$ (equivalently, whose sequence of colours belongs to W) is said to *satisfy* W, and a vertex v_0 with value $\bot$ (equivalently, all paths from v_0 satisfy W) is also said to satisfy W. A graph is said to satisfy W if all its vertices satisfy it.

Games. A *C-game* is a tuple $\mathcal{G} = (G, V_{\mathrm{Eve}}, v_0, \mathrm{val})$, where G is a C-graph, V_{Eve} is a subset of $V(G)$, $v_0 \in V(G)$ is an identified initial vertex, and $\mathrm{val} : C^\omega \to X$ is a C-valuation. We interpret V_{Eve} to be the set of vertices controlled by the first player, *Eve*, and we will write $V_{\mathrm{Adam}} = V(G) \setminus V_{\mathrm{Eve}}$ for the vertices controlled by her opponent, *Adam*. A game is played as follows: starting from v_0, successive moves are played where the player controlling the current vertex v chooses an outgoing edge $v \xrightarrow{c} v'$ and proceed to v'. This interaction goes on forever, producing and infinite path π from v_0. Eve's goal is to minimise the value of the produced path π, whereas Adam aims to maximise it.

In this paper, we are interested in questions of strategy complexity for Eve: if she wins, how much memory is required/sufficient? Formally, these are independent of questions of determinacy (is there a winner?). As a result, we will only ever consider strategies for Eve.

Strategies. A *strategy* in the game $\mathcal{G}$ is a tuple $\mathcal{S} = (S, \pi_{\mathcal{S}}, s_0)$ where S is a graph, $\pi_{\mathcal{S}}$ is a morphism $\pi_S \colon S \to G$ called the *$\mathcal{S}$-projection* and $s_0 \in V(S)$ satisfying:

- $\pi_{\mathcal{S}}(s_0) = v_0$,
- for all $v \in V_{\text{Adam}}$, all outgoing edges $v \xrightarrow{c} v' \in E(G)$ and all $s \in \pi_{\mathcal{S}}^{-1}(v)$, there is $s' \in \pi^{-1}(v')$ such that $s \xrightarrow{c} s' \in E(S)$.

Note that the requirements that S is a graph and $\pi_{\mathcal{S}}$ a morphism impose that for all $v \in V_{\text{Eve}}$ and $s \in \pi_{\mathcal{S}}^{-1}(v)$, s has an outgoing edge $s \xrightarrow{c} s' \in E(S)$ satisfying $\pi_{\mathcal{S}}(s) = v \xrightarrow{c} \pi_{\mathcal{S}}(s') \in E(G)$.

We remark that we do not impose that for each $v \in V_{\text{Eve}}$ and $s \in \pi_{\mathcal{S}}^{-1}(v)$, s has *exactly one* outgoing edge. Stated differently, non-determinism is allowed in this definition of strategy. As the upcoming definition of value of a strategy will clarify, we can interpret that Adam decides how to resolve this non-determinism.

On an informal level, a strategy $\mathcal{S} = (S, \pi_{\mathcal{S}}, s_0)$ from $v_0 \in G$ is used by Eve to play in the game $\mathcal{G}$ as follows:

- whenever the game is in a position $v \in V(G)$, the strategy is in a position $s \in \pi_{\mathcal{S}}^{-1}(v)$;
- initially, the position in the game is v_0, and the position in the strategy is $s_0 \in \pi_{\mathcal{S}}^{-1}(v_0)$;
- if the position v in the game belongs to V_{Adam}, and Adam chooses the edge $v \xrightarrow{c} v'$ in G, then the strategy state is updated following an edge $s \xrightarrow{c} s'$ in S with $\pi_{\mathcal{S}}(s') = v'$, which exists by definition of $\mathcal{S}$ (if multiple options exist, Adam chooses one);
- if the position v in the game belong to V_{Eve}, then the strategy specifies at least one successor $s \xrightarrow{c} s'$ from the current $s \in \pi^{-1}(v)$, and the game proceeds along the edge $v \xrightarrow{c} \pi(s')$ (if multiple options exist in the strategy, which corresponds to the non-determinism mentioned above, then Adam chooses one).

Note that infinite sequences of colours produced when playing as above are exactly labels of infinite paths from s_0 in S.

The *value* $\text{val}(\mathcal{S})$ of a strategy $\mathcal{S}$ is $\text{val}_S(s_0)$. The *value* $\text{val}(\mathcal{G})$ of a game is the infimum value among its strategies. If val is an objective, we say that $\mathcal{S}$ is *winning* if $\text{val}_S(s_0) = \bot$, and we say that Eve *wins* a game $\mathcal{G}$ if $\text{val}(\mathcal{G}) = \bot$.

The following observation is standard (it is usually taken as the definition of a strategy).

▶ **Lemma 1.** *The value of a game is reached with strategies that are trees.*

Memory. For a strategy $\mathcal{S} = (S, \pi_{\mathcal{S}}, s_0)$, we interpret the fibres $\pi_{\mathcal{S}}^{-1}(v)$ as memory spaces. Given a cardinal μ, we say that $\mathcal{S}$ has *memory* strictly less than μ, (resp. less than μ) if for all $v \in V(G)$, $|\pi_{\mathcal{S}}^{-1}(v)| < \mu$ (resp. $|\pi_{\mathcal{S}}^{-1}(v)| \leq \mu$). As it will appear later on, it is convenient for us to be able to use both strict and non-strict inequalities. By means of clarity and conciseness, we usually simply write "$\mathcal{S}$ has memory $< \mu$" (resp. $\leq \mu$) instead of "$\mathcal{S}$ has memory stricly less than μ (resp. less than μ)".

We say that a valuation val has *memory strictly less than* μ, or $< \mu$, (resp. less than μ, or $\leq \mu$) if in all games with valuation val, the value is reached with strategies with memory $< \mu$.

Conversely, we say that val has *memory at least* μ, or $\geq \mu$, if it does not have memory $< \mu$: there exists a game with valuation val in which Eve cannot reach the value with strategies with memory $< \mu$.

We say that val is *positional* if it has memory ≤ 1.

Product strategies, chromatic strategies. A strategy $\mathcal{S} = (S, \pi_{\mathcal{S}}, s_0)$ in the game $\mathcal{G}$ is a *product strategy* over a set M if $V(S) \subseteq V(G) \times M$, with $\pi_{\mathcal{S}}(v, m) = v$. We call the elements of M *memory states*. Note that the memory in a product strategy over M is $\leq |M|$, since

fibers are included in M. A product strategy is *chromatic* if there is a map $\delta : M \times C \to M$ such that for all $(v, m) \xrightarrow{c} (v', m') \in E(S)$ we have $m' = \delta(m, c)$. We say in this case that δ is the *update function* of $\mathcal{S}$. In words, the update of the memory state in a chromatic strategy depends only on the current memory state and the colour that is read. A valuation val has *chromatic memory* $< \mu$ (resp. $\leq \mu$) if in all games with valuation val, the value is reached with chromatic strategies with memory $< \mu$ (resp. $\leq \mu$).

ε-games and ε-strategies. Fix a set of colours C, a fresh colour $\varepsilon \notin C$, and let $C^\varepsilon = C \sqcup \{\varepsilon\}$. The *$C$-projection* of an infinite sequence $w \in (C^\varepsilon)^\omega$ is the (finite or infinite) sequence $w_C \in C^* \cup C^\omega$ obtained by removing all ε's in w. Given a C-valuation $\mathrm{val} : C^\omega \to X$, define its *$\varepsilon$-extension* val^ε to be given by

$$\mathrm{val}^\varepsilon(w) = \begin{cases} \mathrm{val}(w_C), & \text{if } |w_C| = \infty, \\ \inf\limits_{w' \in C^\omega} \mathrm{val}(w_C w'), & \text{otherwise.} \end{cases}$$

It is the unique extension of val with ε as a strongly neutral colour, in the sense of Ohlmann [20]. In particular, if W is an objective and $w \in C^*$, $w\varepsilon^\omega \in W^\varepsilon$ unless w has no winning continuation in W.

An *ε-game* $\mathcal{G}$ is a C^ε-game with valuation val^ε. An *ε-strategy* over such a game is a product strategy $\mathcal{S} = (S, \pi_\mathcal{S}, s_0)$ over some set M such that $(v, m) \xrightarrow{\varepsilon} (v', m') \in E(S)$ implies $m = m'$. Intuitively, Eve is not allowed to update the state of the memory when an ε-edge is traversed. The *memory of an ε-strategy* is defined to be $|M|$. A valuation val has *ε-memory* $< \mu$ (resp. $\leq \mu$) if in all ε-games with valuation val^ε, the value is attained by ε-strategies with memory $< \mu$ (resp. $\leq \mu$).

Note that by definition, a chromatic strategy over M with update function δ is an ε-strategy if and only if for all $m \in M$ it holds that $\delta(m, \varepsilon) = m$. We call such a strategy an *ε-chromatic strategy*. A valuation val has *ε-chromatic memory* $< \mu$ (resp. $\leq \mu$) if in all ε-games with valuation val^ε, the value is attained by ε-chromatic strategies with memory $< \mu$ (resp. $\leq \mu$).

Whenever we want to emphasise that we consider games (resp. strategies, memory) without ε, we might add the adjective *ε-free*.

2.3 Monotonicity and universality

Monotonicity. A partially ordered graph $(G, \leq)$ is *monotone* if

$u \geq v \xrightarrow{c} v' \geq u'$ implies $u \xrightarrow{c} u'$ in G.

A partially ordered graph $(G, \leq)$ is called *well-monotone* if it is monotone and it is well-founded as a partial order. We say that the *width* of a partially ordered graph is $< \mu$ (resp. $\leq \mu$) if it does not contain antichains of size μ (resp. of size strictly greater than μ).

ε-separation. An *ε-separated monotone graph over a set M* is a C^ε-graph G such that $\xrightarrow{\varepsilon}$ defines a partial order making G monotone ($v \leq v' \iff v' \xrightarrow{\varepsilon} v \in E(G)$), and moreover $V(G)$ is partitioned into $(V_m)_{m \in M}$ such that for all $m \in M$, $\xrightarrow{\varepsilon}$ induces a total order over V_m, and there are no ε-edges between different parts: $v \xrightarrow{\varepsilon} v' \in E(G)$ implies that $v, v' \in V_m$ for some $m \in M$. See Figure 3. We define the *breadth* of such a graph as $|M|$.

An ε-separated monotone graph G over M is *chromatic* if there is a map $\delta : M \times C \to M$ such that for all $v \xrightarrow{c} v' \in E(G)$ with $v \in V_m$ and $v' \in V_{m'}$ we have $m' = \delta(v, m)$. We also say in this case that δ is the *update function* of G.

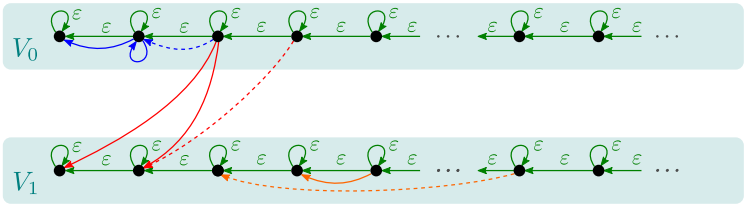


Figure 3 An ε-separated chromatic monotone graph of breadth 2. Note that $\xrightarrow{\varepsilon}$ defines a total order on each V_i (edges following from transitivity are not represented). Many edges which follow from monotonicity are not depicted, the dotted edges give a few examples.

Universality. Given a C-valuation val, a C-graph G and a cardinal κ, we say that G is (κ, val)-*universal* if for all C-trees T of cardinality $< \kappa$, there exists a morphism $\phi : T \to G$ such that $\mathrm{val}_G(\phi(t_0)) \leq \mathrm{val}_T(t_0)$, where t_0 is the root of T. We say that ϕ *preserves the value* at the root to refer to this property (we remark that, in that case, $\mathrm{val}_G(\phi(t_0)) = \mathrm{val}_T(t_0)$, since the other inequality always holds).

▶ Remark 2. An example where the definition with graphs is too constrained to capture memory is given in Proposition 22 from the full version [7].

3 Universal structures characterise memory

Statement of the main results. We start with our characterisations of ε-memory and ε-chromatic memory via (chromatic) ε-separated universal graphs.

▶ **Theorem 3.** *Let* val *be a valuation. If for all cardinals κ there exists an ε-separated (chromatic) and well-monotone $(\kappa, \mathrm{val}^\varepsilon)$-universal graph of breadth $\leq \mu$, then* val *has ε(-chromatic)-memory $\leq \mu$. The converse holds if* val *is an objective (in both the chromatic and non-chromatic cases).*

Our second result concerns ε-free memory. It is stated with strict inequalities, which are relevant in this case and allow for more precision. However, we do not have a converse statement; in fact, the converse cannot hold (see also Figure 2 and Proposition 7).

▶ **Theorem 4.** *Let* val *be a valuation. If for all cardinals κ there exists a well-monotone (κ, val)-universal graph of width $< \mu$, then* val *has ε-free memory $< \mu$.*

As we will see in Proposition 5, the two results above collapse for finite cardinals.

We give the main ideas of the proofs of these two theorems, in both cases we extend the proofs from Ohlmann [20]. The full proofs can be found in Sections 3.2 and 3.3 in the full version [7].

Proof sketch of Theorem 4 and of $\Longrightarrow$ in Theorem 3. We discuss the proof of Theorem 4 (the proof of the first implication in Theorem 3 follows the same structure). In this case, assuming existence of a universal structure, we prove upper bounds in the memory of a valuation. This is done using a strategy-folding procedure that is guided by the morphism towards the universal structure. Let $(U, \leq)$ be a well-monotone (κ, val)-universal graph of width $< \mu$. Suppose that $\mathcal{G}$ is a game of cardinality $\leq \kappa$ with valuation val, and let $\mathcal{T} = (T, \pi_{\mathcal{T}}, t_0)$ be a strategy for Eve given by a tree. By universality of U, there is a morphism $\phi\colon T \to U$ preserving the value at the root of T.

For each vertex v of the game we consider the set $\phi(\pi_{\mathcal{T}}^{-1}(v))$ in U. Since U is well-founded and of width $< \mu$, the set M_v of minimal elements of $\phi(\pi_{\mathcal{T}}^{-1}(v))$ has size strictly less than μ. This allows us to define a strategy over $\bigcup_v \{v\} \times M_v$ which simulates the strategy $\mathcal{T}$ as follows: we take a representative $t_{(v,m)} \in \pi_{\mathcal{T}}^{-1}(v)$, and for each $m \in M_v$, we follow the decisions made at $t_{(v,m)}$ when we are in (v, m).

To define the update of the memory, for each move $v \xrightarrow{c} v' \in E(G)$ and edge $t_{(v,m)} \xrightarrow{c} t' \in E(T)$, we consider the image $\phi(t') \in U$. By definition, there is an element m' in $M_{v'}$ smaller that $\phi(t')$, so we let $(v, m) \xrightarrow{c} (v', m')$. By monotonicity it follows that this strategy has the same value than $\mathcal{T}$. If U is assumed to be (chromatic) ε-separated, it follows directly that the obtained strategy is a (chromatic) ε-strategy. ◀

Proof sketch of $\Longleftarrow$ in Theorem 3. We prove the following result: given a C^ε-tree T satisfying an objective W, there exists an ε-separated well-monotone graph U of breadth $\leq \mu$ and a morphism $T \to U$ preserving the value at the root. Once this is proved, applying it to the tree T_{Univ} consisting of a root connected by an ε-edge to every C^ε-tree $< \kappa$ satisfying W yields a (κ, W^ε)-universal graph.

In order to prove this result, we consider the following game: Adam controls the vertices from T, and for each non-empty set $A \subseteq V(T)$, we add a vertex v_A controlled by Eve with ε-edges back and forth from any vertex in A. This game is won by Eve: whenever Adam chooses an edge $t \xrightarrow{\varepsilon} v_A$ she just need to respond $v_A \xrightarrow{\varepsilon} t$. Since W has ε-memory $\leq \mu$, Eve has a winning ε-strategy $\mathcal{S}$ over $V(S) = V(G) \times M$ with $|M| \leq \mu$.

We define the wanted morphism $\phi\colon T \to S$ in a top-down fashion using the properties of a strategy: $\phi(t_0) = s_0$ and if $\phi(t) = (t, m)$ and $t \xrightarrow{c} t' \in E(T)$, we set $\phi(t') = (t', m')$ where m' is such that $(t, m) \xrightarrow{c} (t', m') \in E(S)$. This morphism preserves the value of t_0, because $\mathcal{S}$ is a winning strategy. With some addition technical tweaks we transform S into an ε-separated graph U of breadth $\leq \mu$ while maintaining a value-preserving morphism $\phi\colon T \to U$. ◀

Applying Dilworth's Theorem [11], we prove that the two notions of graphs collapse (both characterise ε-memory) when dealing with objectives and finite memory bounds.

▶ **Proposition 5.** *Let W be an objective and $m \in \mathbb{N}$. If for all cardinals κ there exists a well-monotone graph which is (κ, W)-universal and has width $\leq m$, then for all cardinals κ there is also an ε-separated well-monotone (κ, W^ε)-universal graph of breadth $\leq m$, and therefore W has ε-memory $\leq m$.*

An objective $W \subseteq C^\omega$ is said to be *prefix-independent* if for all colours $c \in C$ it holds that $cW = W$. It is not difficult to prove (this was already done by Ohlmann [20]) that when considering prefix-independent objectives, one can use a simpler definition of universality, namely, a graph U is (κ, W)-*universal (for prefix-independent objectives)* if U satisfies W and embeds any tree of cardinality $< \kappa$ which satisfies W.

Some concrete examples. We start by illustrating the notions presented until now and some methods to derive universality proofs with a few simple concrete examples of objectives.

For many more examples, as well as the missing proofs of this paragraph, we refer to the Section 4 of the full version [7]. There, we also re-obtain in our framework the general characterisations of [8] for topologically closed objectives, and of [12] for Muller objectives.

Objective $W_1 = \{w \in \{a, b\}^\omega \mid a$ and b occur infinitely often in $w\}$. We show, for each cardinal κ, an ε-separated chromatic and well-monotone $(\kappa, W_1^\varepsilon)$-universal graph of breadth 2. This implies that the ε-chromatic memory of W_1 is at most 2.

Fix a cardinal number κ and consider the graph U from Figure 4. It is easy to check that U is an ε-separated monotone graph over the set $M = \{a, b\}$ and that it is indeed chromatic and satisfies W. We sketch a universality proof. Since W_1 is prefix-independent, we show that U embeds any tree of cardinality $< \kappa$ which satisfies W.

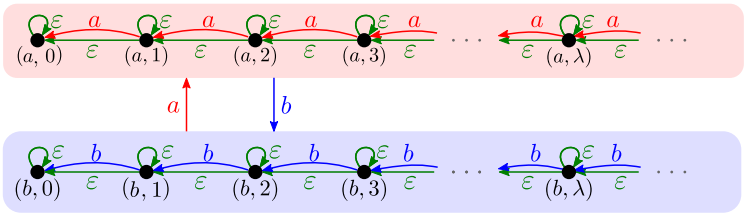


Figure 4 Universal graph for W_1. The order coincides with $\xrightarrow{\varepsilon}$ (as required by the definition of ε-separated graphs). Edges following from monotonicity are not represented. An edge between boxes indicates that all edges are put between vertices in the respective boxes.

Let T be a C-tree of size $< \kappa$ which satisfies W, and let t_0 be its root. Note that all paths from t_0 eventually visit a b-edge; there is in fact an ordinal $\lambda_0 < \kappa$ (defined by induction) which counts the maximal amount of a-edges seen from t_0 before a b-edge is seen; we set $\phi(t_0)$ to be (a, λ_0).

Then for each edge $t_0 \xrightarrow{c} t \in E(T)$ we proceed as follows.

- If $c \in \{a, \varepsilon\}$, we iterate exactly the same process on t, but the ordinal count will on the number of a's will have decreased (or even strictly decreased if $c = a$) from t_0 to t, which guarantees that $\phi(t_0) \xrightarrow{a} \phi(t)$ is indeed an edge in U.
- If $c = b$, then we iterate the same process of t but inverting the roles of a and b; thus $\phi(t)$ is of the form (b, λ_b) for some $\lambda_b < \kappa$, and the edge $\phi(t_0) \xrightarrow{b} \phi(t)$ belongs to U, as required.

This concludes the top-down construction of ϕ and the universality proof.

It is not difficult to find lower bounds to see that the ε-free memory of W_1 is ≥ 2. For example, a game with just one vertex controlled by Eve where she can choose to produce a or b provides this lower bound. Therefore, the exact memory of W_1 is 2, for all the different notions of memory.

Objective $W_2 = (C^*a)^m C^{\geq n} a C^\omega$ with $C = \{a, b\}$ and $m, n \geq 1$. We provide a universal graph of width $n+1$ which proves that the ε-memory is $\leq n+1$. A matching lower bound on the ε-free memory follows from the game depicted on Figure 5. We remark that from the minimal automaton for the regular language $L = (C^*a)^m C^{\geq n} a$ we only obtain an upper bound of $n + m + 1$ on the memory.

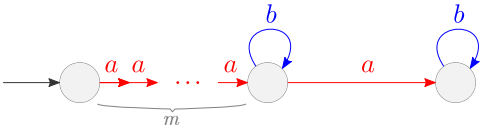


Figure 5 A game where Eve requires memory $n+1$ to ensure objective W_2.

The well-monotone graph U depicted in Figure 6 proves the $n+1$ upper bound on the ε-memory. Actually, it turns out that even the ε-chromatic memory of W_2 is $n+1$, which requires a more subtle construction presented in the full version.

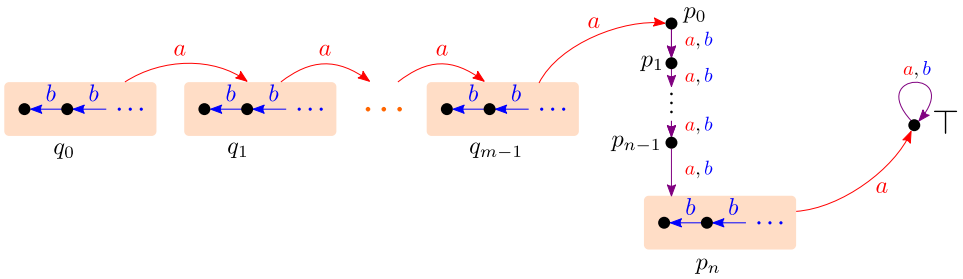


Figure 6 A well-monotone graph U which has width $n+1$ and is universal for W_2.

Objective $W_3 = \{w \in C^\omega \mid w$ contains infinitely often bb or (finitely often b and $aa)\}$ over $C = \{a, b, c\}$. Figure 7 depicts a deterministic parity automaton $\mathcal{A}$ of size 3 recognising W_3; this gives an upper bound of 3 on the memory of W_3. The game depicted on the right of Figure 7 witnesses that Eve requires ε-free memory ≥ 2: positional strategies are losing, but she wins by answering b to b and a to c.

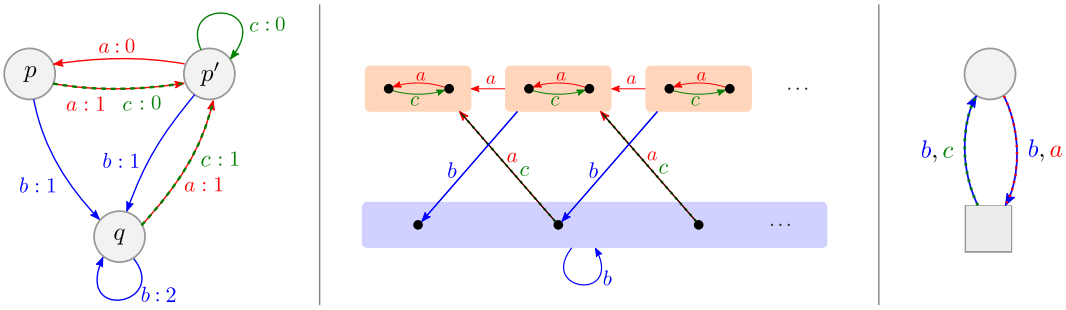


Figure 7 On the left, a deterministic parity automaton $\mathcal{A}$ with three states recognising W_3 (we use max-parity semantics). In the middle, an ε-separated chromatic universal graph U of breadth 2 for W_3; as always, edges following from monotonicity are omitted. On the right, a game witnessing that Eve requires ε-free memory ≥ 2.

The graph U depicted in the middle of Figure 7 is an ε-separated chromatic well-monotone universal graph for W_3 of breadth 2, providing the upper bound of 2 on all the types of memory for W_3.

Counterexamples. We now provide two negative results. First, we show that the converse of Theorem 4 does not hold, even in the case of objectives.

▶ **Proposition 6.** *For each cardinal μ, the objective $W_\mu = \{w_0 w_1 \cdots \in \mu^\omega \mid \forall i, w_i \neq w_{i+1}\}$ satisfies that*

1. *the ε-free memory of W_μ is ≤ 2;*
2. *the ε-free memory of W_μ^ε is $\geq \mu$; and therefore the ε-memory of W_μ is $\geq \mu$; and*
3. *there is κ such that any monotone (κ, W_μ)-universal graph has width $\geq \mu$.*

Second, we prove that Proposition 5 cannot hold if the bound on the size of the antichains of the graph is not finite.

▶ **Proposition 7.** *For any infinite cardinal μ, the objective $W_\mu = \{(w, w') \in (\mu \times \mu)^\omega \mid \nexists i$ such that $w_i < w_{i+1}$ and $w'_i < w'_{i+1}\}$ is such that*

- *for all cardinals κ there exists a well-monotone (κ, W_μ)-universal graph whose antichains have cardinality $< \aleph_0$; and*
- *there is an ε-game with objective W_μ^ε requiring ε-memory $\geq \mu$.*

4 Closure properties

Lexicographical products. We provide a study of lexicographical products, as introduced by Ohlmann [20], whose result we generalize to finite memory bounds.

Given two prefix-independent objectives W_1 and W_2 over disjoint sets of colours C_1 and C_2, we define their *lexicographical product* $W_1 \ltimes W_2$ over $C = C_1 \sqcup C_2$ by

$$W_1 \ltimes W_2 = \{w \in C^\omega \mid [w^2 \text{ is infinite and in } W_2] \text{ or } [w^2 \text{ is finite and } w^1 \in W_1]\},$$

where w^1 (resp. w^2) is the (finite or infinite) word obtained by restricting w to occurrences of letters from C_1 (resp. C_2) in the same order. Note that if w^2 is finite then w^1 is infinite, which is why the product is well defined.

We now define the *lexicographical product* $(U, \leq)$ of two ordered graphs $(U_1, \leq_1)$ and $(U_2, \leq_2)$. Intuitively, each vertex in U_2 is replaced by a copy of U_1. (see also Figure 8).

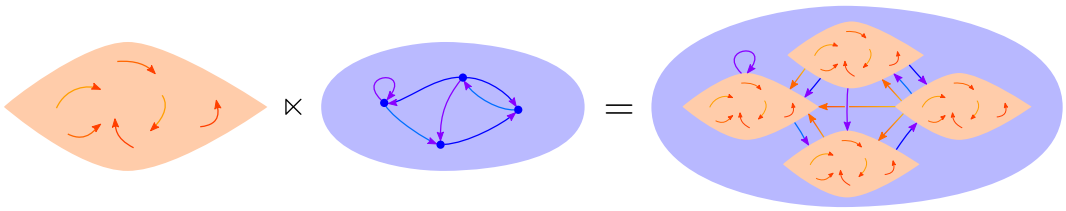


Figure 8 Illustration of the lexicographical product of two ordered graphs.

Formally $U_1 \ltimes U_2 = U$ is defined over the lexicographical product of $(V(U_1), \leq_1)$ and $(V(U_2), \leq_2)$, that is $V(U) = V(U_1) \times V(U_2)$ and $\leq$ is the lexicographical product of $\leq_1$ and $\leq_2$. Its edges are:

$$\begin{aligned} E(U) \quad = \quad & \{(u_1, u_2) \xrightarrow{c_1} (u_1', u_2') \mid c_1 \in C_1 \text{ and } (u_2 >_2 u_2' \text{ or } [u_2 = u_2' \text{ and } u_1 \xrightarrow{c_1} u_1'])\} \\ \cup \quad & \{(u_1, u_2) \xrightarrow{c_2} (u_1', u_2') \mid c_2 \in C_2 \text{ and } u_2 \xrightarrow{c_2} u_2'\}. \end{aligned}$$

We now state our main result in this section, a direct extension of [20, Theorem 18].

▶ **Theorem 8.** *Let W_1 and W_2 be two prefix-independent objectives over disjoint sets of colours C_1 and C_2. Let κ be a cardinal and let $(U_1, \leq)$ and $(U_2, \leq)$ be monotone graphs which are respectively (κ, W_1) and (κ, W_2)-universal. Then $U_1 \ltimes U_2$ is monotone and $(\kappa, W_1 \ltimes W_2)$-universal.*

Using Theorems 3 and 4 together with Proposition 5, we deduce the following result.

▶ **Corollary 9.** *Let W_1 and W_2 be two prefix-independent objectives over disjoint sets of colours C_1 and C_2, and assume that W_1 (resp. W_2) has ε-memory $\leq n_1 \in \mathbb{N}$ (resp. $\leq n_2$). Then, their lexicographical product $W_1 \ltimes W_2$ has ε-memory $\leq n_1 n_2$.*

Combining objectives with locally finite memory. When applied to $\mu = \aleph_0$, since well-founded orders with bounded antichains correspond to well-quasi-orders (wqo's), Theorem 4 states that the existence of universal monotone graphs which are wqo's for a given objective (or even, a valuation) entails *locally finite memory*, meaning that for any ε-free game there is an optimal strategy $\mathcal{S}$ such that for all vertices v, the amount of memory used at v (that is, the cardinality of $\pi_{\mathcal{S}}^{-1}(v)$) is finite. Unfortunately this is not a characterisation: Proposition 6 applied to $\mu = \aleph_0$ gives an objective with ε-free memory 2 but which does not admit such universal structures. Still, by combining our knowledge so far with a few additional insights

stated below, we may derive some strong closure properties pertaining to objectives with locally finite memory. In the sequel, we will simply say *monotone wqo* for a well-monotone graph whose antichains are finite.

Given two partially ordered sets $(U_1, \leq_1)$ and $(U_2, \leq_2)$, we define their *(direct) product* to be the partially ordered set $(U_1 \times U_2, \leq)$, where

$$(u_1, u_2) \leq (u_1', u_2') \iff [u_1 \leq u_1'] \text{ and } [u_2 \leq u_2'].$$

Note that if $\leq_1$ and $\leq_2$ are well-founded, then so is $\leq$. However, there may be considerable blowup on the size of antichains, for instance, $\omega \times \omega$ has arbitrarily large (finite) antichains whereas ω is a total order. However, it is a well-known fact that the product of two wqo's is also a wqo (see for instance [10]), that is, one may not go from finite to infinite antichains.

Given two partially ordered C-graphs $(G_1, \leq_1)$ and $(G_2, \leq_2)$, we define their *(direct) product* to be the partially ordered C-graph G defined over the product of $(V(G_1), \leq_1)$ and $(V(G_2), \leq_2)$ by

$$E(G) = \{(v_1, v_2) \xrightarrow{c} (v_1', v_2') \mid v_1 \xrightarrow{c} v_1' \in E(G_1) \text{ and } v_2 \xrightarrow{c} v_2' \in E(G_2)\}.$$

Note that if $(G_1, \leq_1)$ and $(G_2, \leq_2)$ are monotone, then so is their product. Therefore, if $(G_1, \leq_1)$ and $(G_2, \leq_2)$ are monotone wqo's, then so is their product. Our discussion hinges on the following result.

▶ **Lemma 10.** *Let κ be a cardinal, and $W_1, W_2 \subseteq C^\omega$ be two objectives. Let $(U_1, \leq_1)$ and $(U_2, \leq_2)$ be two C-graphs which are (κ, W_1) and (κ, W_2)-universal, respectively. Then their product U is $(\kappa, W_1 \cap W_2)$-universal.*

Therefore, by combining this lemma with the fact that wqo's are closed under product, we obtain that if two objectives W_1 and W_2 have monotone wqo's as universal graphs, then so does their intersection, hence, from Theorem 4, $W_1 \cap W_2$ has locally finite memory. In particular, thanks to Theorem 3, we get the following weak closure property.

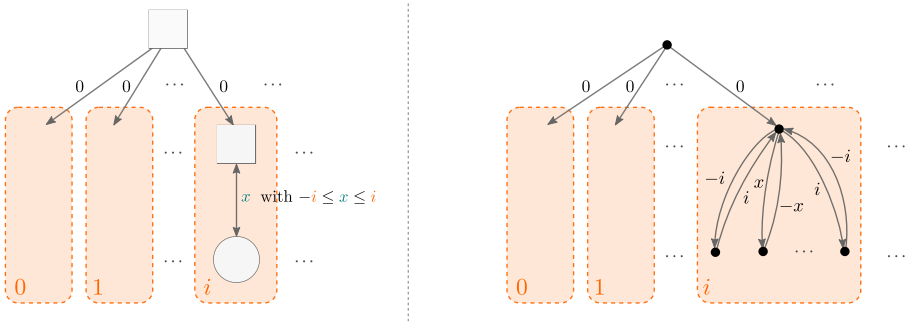


Figure 9 A game where initially, Adam chooses an upper bound i, then the players alternate in choosing integers in $[-i, i]$. Eve wins if the partial sums of the weights remain bounded both from above and below (bi-boundedness objective). She can ensure a win by simply playing the opposite of Adam in each round (this strategy is represented on the right-hand side), which requires unbounded but locally finite memory. Since bi-boundedness objectives are intersections of two positional objectives (being bounded from above and from below), our results in this section ensure that any game with a bi-boundedness objective has optimal locally finite memory strategies.

▶ **Corollary 11.** *Let W_1 and W_2 be two objectives which have monotone wqo's as universal graphs. Then so does $W_1 \cap W_2$. In particular the intersection of two objectives with finite ε-memory has locally finite memory.*

The upper bound stated in Corollary 11 is met: Figure 9 gives an example where W_1 and W_2 are positional but $W_1 \cap W_2$ has ε-free memory $> n$ for all $n \in \mathbb{N}$.

Although our results fall short of implying such a strong closure property, we may still state the following conjecture:

▶ **Conjecture 12.** *Objectives with ε-free memory $< \aleph_0$ are closed under intersection.*

Unions of prefix-independent $\mathbf{\Sigma_2^0}$ objectives. The Cantor topology on C^ω naturally provides a way to define general families of objectives that have been well-studied in the literature of formal languages (we refer to [21] for a general overview). In particular, some of these classes of objectives are given by the different levels of the Borel hierarchy; the lowest levels are Σ_1^0, consisting on the open subsets, and Π_1^0, consisting on the closed subsets. The level Σ_{n+1}^0 (resp. Π_{n+1}^0) contains the countable unions (resp. countable intersections) of subsets in Π_n^0 (resp. Σ_n^0).

We now prove that prefix-independent objectives in Σ_2^0 with ε-memory $\leq m \in \mathbb{N}$ are closed under countable unions. We recall that Σ_2^0 objectives are those of the form $W_{\mathcal{L}} = \{w \in C^\omega \mid w \text{ has finitely many prefixes in } \mathcal{L}\}$, where $\mathcal{L} \subseteq C^*$ is an arbitrary language of finite words [22].

▶ **Theorem 13.** *Prefix-independent Σ_2^0 objectives with ε-memory $\leq m \in \mathbb{N}$ are closed under countable unions.*

Our proof relies on the definition of the *direct sum* of a family of universal graphs (obtained by concatenating them) and the following lemma.

▶ **Lemma 14.** *Let $W_0, W_1, \dots \subseteq C^\omega$ be prefix-independent Σ_2^0 objectives, κ be a cardinal, and $U_0, U_1, \dots$ be C-graphs such that for each i, U_i is (W_i, κ)-universal. Let $W = \bigcup_i W_i$. Then the graph $U \ltimes \kappa$, where U is the direct sum of the U_i's, and κ is the edgeless graph with κ vertices, is (κ, W)-universal.*

Proof sketch. Let T be a tree of cardinality $< \kappa$ satisfying W. Since W is prefix-independent, proving that there is $t \in V(T)$ inducing a subtree $T[t]$ such that $T[t] \to U$ is enough to derive universality of $U \ltimes \kappa$ (in the full version [7], this useful fact is stated as Lemma 10). Since U is the direct sum of the U_i's and since each U_i is κ-universal for W_i, this amounts to showing that there is $i \in \mathbb{N}$ and $t \in T$ such that $T[t]$ satisfies W_i. Assume otherwise. Take $e = e_0 e_1 \cdots \in \mathbb{N}^\omega$ to be a word over the naturals with infinitely many occurrences of each natural, for instance $e = 010120123\dots$. For each $i \in \mathbb{N}$, let $\mathcal{L}_i \subseteq C^*$ be such that $W_i = \{w \in C^\omega \mid w \text{ has finitely many prefixes in } \mathcal{L}_i\}$.

We now construct an infinite path $\pi = \pi_0 \pi_1 \dots$ starting from the root t_0 in T such that for each i, the coloration $w_0 \dots w_i$ of $\pi_0 \dots \pi_i$ belongs to $\mathcal{L}_{e_i}$. This implies that the coloration w of π has infinitely many prefixes in each of the $\mathcal{L}_i$'s, therefore it does not belong to W, a contradiction. Assume $\pi = \pi_0 \dots \pi_{i-1} : t_0 \overset{w_0 \dots w_{i-1}}{\rightsquigarrow} t$ constructed up to π_{i-1}. Since by assumption, $T[t]$ does not satisfy W_{e_i}, there is a path $\pi' : t \overset{w}{\rightsquigarrow}$ such that $w \notin W_{e_i}$. By prefix-independence of W_{e_i}, we get $w_0 \dots w_{i-1} w \notin W_{e_i}$, thus w has a prefix w_i such that $w_0 \dots w_{i-1} w_i \in \mathcal{L}_{e_i}$; this allows us to augment π as required and conclude our proof. ◀

The theorem follows from Lemma 14, Theorem 3 and Proposition 5 and the fact that antichains in the well-founded graph $U \times \kappa$ are no larger than those in U.

5 Conclusion

In this paper, we have extended Ohlmann's work [20] to the study of the memory of objectives. We have introduced different variants of well-monotone universal graphs adequate to the various models of memory appearing in the literature, and we have characterised the memory of objectives through the existence of such universal graphs (Theorems 3 and 4).

Possible applications. We expect these results to have two types of applications. The first one is helping to find tight bounds for the memory of different families of objectives. We have illustrated this use of universal graphs by providing non-trivial tight bounds on the memory of some concrete examples. In the full version [7, Section 4], we further recover known results about the memory of topologically closed objectives [8] and Muller objectives [12]. While finding universal graphs and proving their correctness might be difficult, we believe that they are a useful support to guide our intuition, and they provide a standardised method to formalise proofs of upper bounds on memory requirements.

The second kind of application discussed in the paper is the study of combinations of objectives. We have used our characterizations to bound the memory requirements of finite lexicographical product of objectives (Section 4). We have also established that intersections of objectives with finite ε-memory always have locally finite ε-free memory. Finally, we have proved that prefix-independent Σ^0_2 objectives with finite ε-memory are closed under countable unions. We believe that the new angle offered by universal graphs will help to better understand general closure properties of memory.

Open questions. Many questions remain open. First of all, as discussed in Section 4, we have proved that objectives admitting universal monotone wqo's are closed by intersection. However, we do not know whether the larger class of objectives with unbounded finite ε-free memory is closed under intersection (Conjecture 12). A related question is therefore understanding what are exactly the objectives admitting universal monotone wqo's.

In the realm of positional objectives, a long-lasting open question is Kopczyński's conjecture [15]: are unions of prefix-independent positional objectives positional? This conjecture has recently been disproved for finite game graphs by Kozachinskiy [16], but it remains open for arbitrary game graphs. We propose a generalisation of Kopczyński's conjecture in the case of ε-memory.

▶ **Conjecture 15.** *Let $W_1 \subseteq C^\omega$ and $W_2 \subseteq C^\omega$ be two prefix-independent objectives with ε-memory $\leq n_1, n_2$, respectively. Then $W_1 \cup W_2$ has ε-memory $\leq n_1 n_2$.*

Objectives that are ω-regular (those recognised by a deterministic parity automaton, or, equivalently, by a non-deterministic Büchi automaton) have received a great deal of attention over the years. However, very little is known about their memory requirements, and even about their positionality. By now, thanks to a recent work of Bouyer, Casares, Randour and Vandenhove [1], which relies on Ohlmann's characterisation, positionality is understood for objectives recognised by deterministic Büchi automata.

Characterising positionality or memory requirements for other general classes of ω-regular objectives, such as those recognised by deterministic co-Büchi automata or by deterministic automata of higher parity index remains an open and challenging endeavour. Similarly, one may turn to (non-necessarily ω-regular) objectives with topological properties, for instance, it is not known by now which topologically open objectives (or, recognised by infinite deterministic reachability automata) are positional, or finite memory. We hope that the newly available tools presented in this paper will help progress in this direction.

References

1 Patricia Bouyer, Antonio Casares, Mickael Randour, and Pierre Vandenhove. Half-positional objectives recognized by deterministic Büchi automata. In *CONCUR*, volume 243, pages 20:1–20:18, 2022. doi:10.4230/LIPIcs.CONCUR.2022.20.

2 Patricia Bouyer, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. Arena-independent finite-memory determinacy in stochastic games. In *CONCUR*, volume 203, pages 26:1–26:18, 2021. doi:10.4230/LIPIcs.CONCUR.2021.26.

3 Patricia Bouyer, Mickael Randour, and Pierre Vandenhove. Characterizing omega-regularity through finite-memory determinacy of games on infinite graphs. In *STACS*, volume 219, pages 16:1–16:16, 2022. doi:10.4230/LIPIcs.STACS.2022.16.

4 Patricia Bouyer, Stéphane Le Roux, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. Games where you can play optimally with arena-independent finite memory. *Log. Methods Comput. Sci.*, 18(1), 2022. doi:10.46298/lmcs-18(1:11)2022.

5 Antonio Casares. On the minimisation of transition-based Rabin automata and the chromatic memory requirements of Muller conditions. In *CSL*, volume 216, pages 12:1–12:17, 2022. doi:10.4230/LIPIcs.CSL.2022.12.

6 Antonio Casares, Thomas Colcombet, and Karoliina Lehtinen. On the size of good-for-games Rabin automata and its link with the memory in Muller games. In *ICALP*, volume 229, pages 117:1–117:20, 2022. doi:10.4230/LIPIcs.ICALP.2022.117.

7 Antonio Casares and Pierre Ohlmann. Characterising memory in infinite games. *CoRR*, abs/2209.12044, 2022. doi:10.48550/arXiv.2209.12044.

8 Thomas Colcombet, Nathanaël Fijalkow, and Florian Horn. Playing safe. In *FSTTCS*, volume 29, pages 379–390, 2014. doi:10.4230/LIPIcs.FSTTCS.2014.379.

9 Thomas Colcombet and Damian Niwiński. On the positional determinacy of edge-labeled games. *Theor. Comput. Sci.*, 352(1-3):190–196, 2006. doi:10.1016/j.tcs.2005.10.046.

10 Stéphane Demri, Alain Finkel, Jean Goubault-Larrecq, Sylvain Schmitz, and Philippe Schnoebelen. Well-quasi-orders for algorithms. Lecture notes, Master MPRI, 2017. URL: https://wikimpri.dptinfo.ens-cachan.fr/lib/exe/fetch.php?media=cours:upload:poly-2-9-1v02oct2017.pdf.

11 Robert P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51(1):161–166, 1950. doi:10.2307/1969503.

12 Stefan Dziembowski, Marcin Jurdzinski, and Igor Walukiewicz. How much memory is needed to win infinite games? In *LICS*, pages 99–110. IEEE Computer Society, 1997. doi:10.1109/LICS.1997.614939.

13 Hugo Gimbert and Wieslaw Zielonka. Games where you can play optimally without any memory. In *CONCUR*, volume 3653 of *Lecture Notes in Computer Science*, pages 428–442. Springer, 2005. doi:10.1007/11539452_33.

14 Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *STOC*, pages 60–65, 1982. doi:10.1145/800070.802177.

15 Eryk Kopczyński. *Half-positional Determinacy of Infinite Games*. PhD thesis, Warsaw University, 2008.

16 Alexander Kozachinskiy. Energy games over totally ordered groups. *CoRR*, abs/2205.04508, 2022. doi:10.48550/arXiv.2205.04508.

17 Alexander Kozachinskiy. Infinite separation between general and chromatic memory. *CoRR*, abs/2208.02691, 2022. doi:10.48550/arXiv.2208.02691.

18 Alexander Kozachinskiy. State complexity of chromatic memory in infinite-duration games. *CoRR*, abs/2201.09297, 2022. arXiv:2201.09297.

19 Pierre Ohlmann. Characterizing positionality in games of infinite duration over infinite graphs. In *LICS*, pages 22:1–22:12, 2022. doi:10.1145/3531130.3532418.

20 Pierre Ohlmann. Characterizing Positionality in Games of Infinite Duration over Infinite Graphs. *TheoretiCS*, Volume 2, January 2023. doi:10.46298/theoretics.23.3.

21 Dominique Perrin and Jean-Éric Pin. *Infinite words - automata, semigroups, logic and games*, volume 141 of *Pure and applied mathematics series*. Elsevier Morgan Kaufmann, 2004.

22 Michał Skrzypczak. Topological extension of parity automata. *Information and Computation*, 228-229:16–27, 2013. doi:10.1016/j.ic.2013.06.004.

23 Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998.

Approximate Model Counting: Is SAT Oracle More Powerful Than NP Oracle?

Diptarka Chakraborty
National University of Singapore, Singapore

Sourav Chakraborty
Indian Statistical Institute, Kolkata, India

Gunjan Kumar
National University of Singapore, Singapore

Kuldeep S. Meel
National University of Singapore, Singapore

Abstract

Given a Boolean formula ϕ over n variables, the problem of model counting is to compute the number of solutions of ϕ. Model counting is a fundamental problem in computer science with wide-ranging applications in domains such as quantified information leakage, probabilistic reasoning, network reliability, neural network verification, and more. Owing to the #P-hardness of the problems, Stockmeyer initiated the study of the complexity of approximate counting. Stockmeyer showed that $\log n$ calls to an NP oracle are necessary and sufficient to achieve (ε, δ) guarantees. The hashing-based framework proposed by Stockmeyer has been very influential in designing practical counters over the past decade, wherein the SAT solver substitutes the NP oracle calls in practice. It is well known that an NP oracle does not fully capture the behavior of SAT solvers, as SAT solvers are also designed to provide satisfying assignments when a formula is satisfiable, without additional overhead. Accordingly, the notion of SAT oracle has been proposed to capture the behavior of SAT solver wherein given a Boolean formula, an SAT oracle returns a satisfying assignment if the formula is satisfiable or returns unsatisfiable otherwise. Since the practical state-of-the-art approximate counting techniques use SAT solvers, a natural question is whether an SAT oracle is more powerful than an NP oracle in the context of approximate model counting.

The primary contribution of this work is to study the relative power of the NP oracle and SAT oracle in the context of approximate model counting. The previous techniques proposed in the context of an NP oracle are weak to provide strong bounds in the context of SAT oracle since, in contrast to an NP oracle that provides only one bit of information, a SAT oracle can provide n bits of information. We therefore develop a new methodology to achieve the main result: a SAT oracle is no more powerful than an NP oracle in the context of approximate model counting.

2012 ACM Subject Classification Theory of computation → Oracles and decision trees

Keywords and phrases Model counting, Approximation, Satisfiability, NP oracle, SAT oracle

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.123

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Funding *Diptarka Chakraborty*: Supported in part by an MoE AcRF Tier 2 grant (MOE-T2EP20221-0009) and Google South & South-East Asia Research Award.
Gunjan Kumar: Supported in part by National Research Foundation Singapore under its NRF Fellowship Programme[NRF-NRFFAI1-2019-0004].
Kuldeep S. Meel: Supported in part by National Research Foundation Singapore under its NRF Fellowship Programme[NRF-NRFFAI1-2019-0004] and Campus for Research Excellence and Technological Enterprise (CREATE) programme, Ministry of Education Singapore Tier 2 grant MOE-T2EP20121-0011, and Ministry of Education Singapore Tier 1 Grant [R-252-000-B59-114].

1 Introduction

Let ϕ be a Boolean formula over n propositional variables. An assignment $s \in \{T, F\}^n$ is called a *satisfying assignment* if it makes ϕ evaluate to true. Let $\mathsf{sol}(\phi)$ denote the set of all satisfying assignments. The model counting problem is to compute $|\mathsf{sol}(\phi)|$ for a given ϕ. It is a fundamental problem in computer science and has numerous applications across different fields such as quantified information leakage, probabilistic reasoning, network reliability, neural network verification, and the like [12, 13, 17, 9, 8, 1]. The seminal work of Valiant [17] showed that the problem of model counting is #P-complete, and consequently, one is often interested in approximate variants of the problem. In this paper, we consider the following problem:

Approximate Model Counting

Input A formula ϕ, tolerance parameter $\varepsilon > 0$, and confidence parameter $\delta \in (0, 1)$.
Output Compute an estimate Est such that

$$\Pr\left[\frac{|\mathsf{sol}(\phi)|}{1+\epsilon} \leq \mathsf{Est} \leq (1+\epsilon)|\mathsf{sol}(\phi)|\right] \geq 1-\delta.$$

Stockmeyer [16] initiated the study of the complexity of approximate model counting. Stockmeyer's seminal paper made two foundational contributions: the first contribution was to define the query model that could capture possible natural algorithms yet amenable enough to theoretical tools to allow non-trivial insight. To this end, Stockmeyer proposed the query model wherein one can construct an arbitrary set S and query an NP oracle to determine if $|\mathsf{sol}(\phi) \cap S| \geq 1$. Stockmeyer showed that under the above-mentioned query model, $\log n$ calls to an NP oracle are necessary and sufficient (for a fixed ε and δ). Furthermore, Stockmeyer introduced a hashing-based algorithmic procedure to achieve the desired upper bound that makes $O(\log n)$ calls to NP-oracle. The lack of availability of powerful reasoning systems for problems in NP dissuaded the development of algorithmic frameworks based on Stockmeyer's hashing-based framework until the early 2000s [10].

Motivated by the availability of powerful SAT solvers, there has been a renaissance in the development of hashing-based algorithmic frameworks for model counting, wherein a call to an NP oracle is handled by an SAT solver in practice. The current state-of-the-art approximate model counter, ApproxMC [4], relies on the hashing-based framework and is able to routinely handle problems involving hundreds of thousands of variables. The past decade has witnessed a sustained interest in further enhancing the scalability of these approximate model counters. It is perhaps worth highlighting that Stockmeyer's query model captures queries by ApproxMC.

While the current state-of-the-art approximate model counters rely on the hashing-based framework, they differ significantly from Stockmeyer's algorithm for approximate model counting. The departures from Stockmeyer's algorithm have been deliberate and have often been crucial to attaining scalability. In particular, ApproxMC crucially exploits the underlying SAT solver's ability to return a satisfying assignment to attain scalability. In this context, it is worth highlighting that, unlike an NP oracle that only returns the answer Yes or No for a given Boolean formula, all the known SAT solvers are capable of returning a satisfying assignment if the formula is satisfiable without incurring any additional overhead. Observe that one would need n calls to an NP oracle to determine a satisfying assignment. From this viewpoint, an NP oracle does not fully capture the behavior of an SAT solver, and one needs a different notion to model the behavior of SAT solver.

Delannoy and Meel [7] sought to bridge the gap between theory and practice by proposing the notion of a SAT oracle. Formally, a SAT oracle takes in a Boolean formula ϕ as input and returns a satisfying assignment $s \in \mathsf{sol}(\phi)$ if ϕ is satisfiable and $\perp$, otherwise. It is worth highlighting that we may need n calls to an NP oracle to simulate a query to a SAT oracle, and therefore, it is conceivable for an algorithm to make $O(\log n)$ calls to a SAT oracle but $O(n \log n)$ calls to an NP oracle. Delannoy and Meel showcased precisely such behavior in the context of *almost-uniform generation*. Their proposed algorithm, UniSamp makes $O(\log n)$ calls to a SAT oracle and would require $O(n \log n)$ calls to an NP oracle if one were to replace a SAT oracle with an NP oracle. At the same time, it is not necessary that there would be a gap of n calls for every algorithm: simply consider the problem of determining whether a formula is satisfiable or not. Only one call to an NP oracle (and similarly to a SAT oracle) suffices.

Furthermore, the notion of the SAT oracle has the potential to be a powerful tool to explain the behavior of algorithms, as highlighted by Delannoy and Meel. Given access to an NP oracle, the sampling algorithm due to Jerrum, Valiant, and Vazirani [11] (referred to as JVV algorithm) makes $O(n^2 \log n)$ calls to an NP oracle as well as a SAT oracle, i.e., there are no savings from the availability of a SAT oracle. On the other hand, the algorithm, UniSamp makes $O(\log n)$ and $O(n \log n)$ calls to SAT and an NP oracle respectively. Therefore, the NP oracle model would indicate that one should expect the performance gap between JVV and UniSamp to be linear, while the SAT oracle model indicates an exponential gap. The practical implementations of JVV and UniSamp indeed indicate the performance gap between them to be exponential rather than linear. Therefore, analyzing problems under the SAT oracle model has the promise to have wide-ranging consequences.

In this paper, we analyze the complexity of the problem of approximate model counting given access to a SAT oracle. Our study is motivated by two observations:

O1 The modern state-of-the-art hashing-based techniques differ significantly from Stockmeyer's algorithmic procedure and, in particular, exploit the availability of SAT solvers. Yet, they make $O(\log n)$ calls to a SAT oracle, which coincides with the number of NP oracle calls in Stockmeyer's algorithmic procedure.

O2 Stockmeyer provided a matching lower bound of $\Omega(\log n)$ on the number of NP calls, which follows from the simple observation that for a fixed ε, there are $\Theta(n)$ possible outputs that an algorithm can return. Since every NP call returns an answer, Yes or No, the trace of an algorithm can be viewed as a binary tree such that every leaf represents a possible output value. Therefore, the height of the tree (i.e., the number of NP calls) must be $\Omega(\log n)$. Since a SAT oracle returns a satisfying assignment (i.e., provides n bits of information), the trace of the algorithm is no longer a binary tree, and therefore, Stockmeyer's analysis does not extend to the case of SAT oracles for approximate model counting.

To summarize, the best-known upper bound for SAT oracle calls for approximate model counting is $O(\log n)$, which matches the upper bound for NP oracle calls. However, the technique developed in the context of achieving a lower bound for NP oracle calls does not apply to the case of SAT oracle. Therefore, one wonders whether there exist algorithms with a lower number of SAT oracle calls. In other words, are SAT oracles more powerful than NP oracles for the problem of approximate model counting?

The primary contribution of this work is to resolve the above challenge. In contrast to the problem of uniform sampling, we reach a starkly different conclusion: SAT oracles are no more powerful than NP oracles in the context of approximate model counting. Formally, we prove the following theorem:

▶ **Theorem 1.1.** *For any $\epsilon, \delta \in (0,1)$, given a formula ϕ, computation of (ε, δ)-approximation of $|\mathsf{sol}(\phi)|$ requires $\tilde{\Omega}(\log n)$[1] queries to a SAT oracle.*

The establishment of the above theorem turned out to be highly challenging as the existing approaches in the context of NP oracles are not applicable to the SAT oracles. We provide an overview of our approach below.

1.1 Technical Overview

In order to provide the lower bound on the number of queries required by the SAT oracle, we work with a stronger SAT oracle model. In particular, an answer from a (standard) SAT oracle does not provide any extra guarantee/information other than that the returned assignment is a satisfying assignment of the queried formula. Our lower bound works even if we consider that the returned satisfying assignment is chosen randomly from the set of satisfying assignments. More specifically, we consider a stronger model, namely SAT-Sample *oracle*, which returns a uniformly chosen solution of a queried formula ϕ whenever the formula is satisfiable. It is worth remarking that while a SAT oracle can be simulated by only n queries to an NP oracle, the best-known technique to simulate SAT-Sample makes $O(n^2 \log n)$ queries to an NP oracle [2, 7]. We prove the following theorem which implies Theorem 1.1.

▶ **Theorem 1.2.** *For any $\epsilon < 1/2$ and any $\delta \leq 1/6$, given a formula ϕ, computation of (ε, δ)-approximation of $|\mathsf{sol}(\phi)|$ requires $\tilde{\Omega}(\log n)$ queries to a* SAT-Sample *oracle.*

Although we consider $\epsilon < 1/2$ and $\delta \leq 1/6$ in the above theorem and provide the proof accordingly, our proof works even for any constant $\epsilon, \delta \in (0,1)$. Another thing to remark is that in our proof, we allow even exponential (in the size of the original formula) size formula to be queried in the SAT-Sample oracle, making our result stronger than what is claimed in the above theorem.

Let us assume that Alg is an algorithm that (ϵ, δ)-approximates $|\mathsf{sol}(\phi)|$ for any given input ϕ (on n variables) by making q queries to a SAT-Sample oracle. We will refer to such an algorithm as a SAT-Sample counter. We would like to prove a lower bound on q.

The main technical difficulty in proving our lower bound results comes from the enormous power of a SAT-Sample oracle compared to an NP oracle. An NP oracle can only provide a YES or NO answer, restricting the number of possible answers (from the NP oracle) to 2^q for a q-query counter with an NP oracle. On the other hand, since a SAT-Sample oracle returns a (random) satisfying assignment (if a satisfying assignment exists), the number of possible answers can be 2^{nq}. Further, any counter can be adaptive – it can choose the next query adaptively based on the previous queries made and their corresponding answers. In general, proving a non-trivial (tight) lower bound for any adaptive algorithm turns out to be one of the notorious challenges, and the difficulty in proving such a lower bound arises in other domains like data structure lower bound, property testing, etc. One of the natural ways to prove any lower bound is to use the information-theoretic technique. However, one of the main challenges in applying such techniques in the adaptive setting is that conditional mutual information terms often involve complicated conditional distributions that are difficult to analyze.

To start with, we argue that we can assume that the SAT-Sample counter is "semi-oblivious" in nature. The number of satisfying assignments of a formula does not change by any permutation of the elements in $\{T, F\}^n$, and the SAT-Sample counter can only get

[1] The tilde hides a factor of $\log \log n$.

elements of $\mathsf{sol}(\phi)$ by querying the SAT-Sample oracle. So we argue that the only useful information of the i^{th} query set (that is, the set of satisfying assignments of the formula that is given to the SAT-Sample oracle) is the size of its intersection with the previous $(i-1)$ query sets and their corresponding answers. We formalize it in Section 3.1.

We next use Yao's minimax principle to prove a lower bound on the number of queries to a SAT-Sample oracle made by a deterministic "Semi-oblivious counter" when the input formula ϕ is drawn from a "hard" distribution.

For the hard distribution, we construct $O(n^{3/4})$ formulas ϕ_ℓ for each value of ℓ in the set $\{\lfloor n^{1/4} \rfloor, \lfloor n^{1/4} \rfloor + 1, \ldots, \lceil n^{3/4} \rceil\}$. The formulas ϕ_ℓ are chosen in such a way that $|\mathsf{sol}(\phi_\ell)| \approx 2|\mathsf{sol}(\phi_{\ell+1})|$ thereby approximately counting the number of satisfying assignments (upto a multiplicative $(1+\epsilon)$-factor for small constant ϵ) reduces to the problem of determining the value of ℓ. The hard distribution is obtained by picking an ℓ uniformly at random from the set $\{\lfloor n^{1/4} \rfloor, \lfloor n^{1/4} \rfloor + 1, \ldots, \lceil n^{3/4} \rceil\}$ and using the corresponding formula ϕ_ℓ.

Finally, we show the lower bound using information theory. At a high level, we show that the information gained about ℓ by the knowledge of obtained samples is small unless we make $\tilde{\Omega}(\log n)$ oracle calls (Lemma 9). Then we turn to Fano's Inequality (Theorem 3) which links the error probability of a counter to the total information gain. Showing that the information gained by samples is small boils down to showing that the KL-divergence of the conditional distribution over the samples is small for all formulas ϕ_ℓ (shown in the proof of the third part of Lemma 9). The difficulty in showing the above bound comes from the fact that the samples are adaptive and may not always be concentrated around the expectation. To overcome the above challenge, we first define an indicator random variable Y_i to denote whether, at the i^{th} query, the concentration holds (see the definition in Equation 10). Then we split it into cases: In the first case, we argue for the situation when concentration may not hold at some step of the algorithm (if $Y_i = 1$ for some $i \in [q]$). The second case is when concentration holds (if $Y_i = 0$ for all $i \in [q]$). We believe that the technique developed in this paper can be a general tool to show sampling lower bounds in a number of other settings.

2 Notations and Preliminaries

For any integer m, let $[m]$ denote the set of integers $\{1, 2, \ldots, m\}$. For a formula ϕ over variable set $\mathsf{vars}(\phi) = \{v_1, \ldots, v_n\}$, we denote by $\mathsf{sol}(\phi)$ the set of satisfying assignments of ϕ. If ϕ is not satisfiable then $\mathsf{sol}(\phi) = \emptyset$. We can interpret $\mathsf{sol}(\phi)$ as a subset of $\{T, F\}^n$. On the other hand, for any subset $A \subset \{T, F\}^n$ we denote by ψ_A the formula whose set of satisfying assignments is exactly A; that is, $\mathsf{sol}(\psi_A) = A$.

Oracles and query model

In our context of Boolean formulas, an *NP oracle* takes in a Boolean formula ϕ as input and returns Yes if ϕ is satisfiable (i.e., $\mathsf{sol}(\phi) \neq \emptyset$), and No, otherwise. Modern SAT solvers, besides determining whether a given formula is satisfiable or not, also return a satisfying assignment (arbitrarily) if the formula is satisfiable. This naturally motivates us to consider an oracle, namely SAT-Sample *oracle*, that takes in a Boolean formula ϕ as input and, if ϕ is satisfiable, returns a satisfying assignment uniformly at random from the set $\mathsf{sol}(\phi)$, and $\perp$, otherwise.

We rely on the query model introduced by Stockmeyer [16]: For a given ϕ whose model count we are interested in estimating, one can query the corresponding (NP/SAT) oracle with formulas of the form $\phi \wedge \psi_A$, where, as stated earlier, ψ_A is an (arbitrary) formula whose set of solutions is A. We will use ϕ_A as a shorthand to represent $\phi \wedge \psi_A$. Throughout

this paper, we consider the above query model with query access to the SAT-Sample oracle. One call to the SAT-Sample oracle will be called a SAT-Sample query. By abuse of notation, we sometimes say "A is queried" to refer to the formula ϕ_A.

k-wise independent hash functions

Let n, m, k be positive integers and let $H(n, m, k)$ denote the family of *k-wise independent hash functions* from $\{T, F\}^n$ to $\{T, F\}^m$. For any $\alpha \in \{T, F\}^m$, and $h \in H(n, m, k)$, let $h^{-1}(\alpha)$ denote the set $\{s \in \{T, F\}^n \mid h(s) = \alpha\}$.

It is well-known (e.g., see [5]) that for any integer n, m, k, one can generate an explicit family of k-wise independent hash functions in time and space $poly(n, m, k)$. Moreover, for any $\alpha \in \{T, F\}^m$, $h^{-1}(\alpha)$ (where $h \in H(n, m, k)$) can be specified by a Boolean formula of size $poly(n, m, k)$.

Concentration inequalities for limited independence

▶ **Lemma 1** ([15]). *If X is a sum of k-wise independent random variables, each of which is confined to $[0, 1]$ with $\mu = \mathbb{E}[X]$ then*

1. *For any $\gamma \leq 1$ and $k \geq \gamma^2\mu$, $\Pr[|X - \mu| \geq \gamma\mu] \leq exp(-\gamma^2\mu/3)$.*
2. *For any $\gamma \geq 1$ and $k \geq \gamma\mu$, $\Pr[|X - \mu| \geq \gamma\mu] \leq exp(-\gamma\mu/3)$.*

Basics of information theory

Let X and Y be two random variables over the space $\mathcal{X} \times \mathcal{Y}$. The mutual information $I(X;Y)$ between random variables X and Y is the reduction in the entropy of X given Y and hence

$$I(X;Y) = H(X) - H(X|Y) \leq H(X) \tag{1}$$

where $H(X) = -\sum_{x \in \mathcal{X}} \Pr[X = x] \log \Pr[X = x]$ is the Shannon entropy of X and $H(X|Y)$ is the conditional entropy of X given Y.

The *Kullback–Leibler divergence* or simply *KL divergence* (also called relative entropy) between two discrete probability distributions P and Q defined on same probability space $\mathcal{X}$ is given by :

$$KL(P||Q) := \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}$$

where p and q are probability mass functions of P and Q respectively.

If the joint distribution of X and Y is $Q_{X,Y}$ and marginal distributions Q_X and Q_Y respectively, then the mutual information $I(X;Y)$ can also be equivalently defined as:

$$I(X;Y) := KL(Q_{X,Y}||Q_X \times Q_Y).$$

For three random variables X, Y, Z, the *conditional mutual information* $I(X;Y|Z)$ is defined as

$$I(X;Y|Z) := \mathbb{E}_Z[KL(Q_{(X,Y)|Z}||Q_{X|Z} \times Q_{Y|Z})].$$

For any three random variables X, Y, Z, the *chain rule* for mutual information says that

$$I(X;(Y,Z)) = I(X;Y) + I(X;Z|Y).$$

If Z is a discrete random variable taking values in $\mathcal{Z}$ then we have

$$\mathbb{E}_Z[KL(Q_{(X,Y)|Z}||Q_{X|Z} \times Q_{Y|Z})] = \sum_{z\in\mathcal{Z}} Q_Z(z) \cdot KL(Q_{(X,Y)|Z=z}||Q_{X|Z=z} \times Q_{Y|Z=z})$$

$$= \sum_{z\in\mathcal{Z}} Q_Z(z) \cdot I(X;Y|Z=z).$$

▶ **Lemma 2** ([14]). *Let $P_X, P_Z, P_{Z|X}$ be the marginal distributions corresponding to a pair (X,Z), where X is discrete. For any auxiliary distribution Q_Z, we have*

$$I(X,Z) = \sum_x P_X(x) KL(P_{Z|X}(\cdot|x)||P_Z) \leq \max_x KL(P_{Z|X}(\cdot|x)||Q_Z).$$

▶ **Theorem 3** (Fano's inequality). *Consider discrete random variables X and $\hat{X}$ both taking values in $\mathcal{V}$. Then*

$$\Pr[\hat{X} \neq X] \geq 1 - \frac{I(X;\hat{X}) + \log 2}{\log|\mathcal{V}|}.$$

Consider the random variables $X, Z, \hat{X}$. If the random variable $\hat{X}$ depends only on Z and is conditionally independent on X, then we have

$$I(X;\hat{X}) \leq I(X;Z). \tag{2}$$

This inequality is known as the *data processing inequality*. For the further exposition, readers may refer to any standard textbook on information theory (e.g., [6]).

MiniMax theorem

Yao's minimax principle [18] is a standard tool to show lower bounds on the worst-case performance of randomized algorithms. Roughly speaking, it says that to show a lower bound on the performance of a randomized algorithm R, it is sufficient to show a lower bound on any deterministic algorithm when the instance is randomly drawn from some distribution.

Consider a problem over a set of inputs $\mathcal{X}$. Let Γ be some probability distribution over $\mathcal{X}$ and let $X \in \mathcal{X}$ be an input chosen as per Γ. Any randomized algorithm R is essentially a probability distribution over the set of deterministic algorithms, say $\mathcal{T}$. By Yao's minimax principle,

$$\max_{X\in\mathcal{X}} \Pr[R \text{ gives wrong answer on } X] \geq \min_{T\in\mathcal{T}} \Pr_{X\sim\Gamma}[T \text{ gives wrong answer on } X].$$

3 Lower Bound on the number of queries to SAT-Sample oracle

In this section, we will prove Theorem 1.2, which implies Theorem 1.1. Let Alg be an adaptive randomized algorithm that given as input ϕ over n variables $\mathsf{vars} = \{v_1, \ldots, v_n\}$ and output Est that is an (ϵ,δ)-approximation of $\mathsf{sol}(\phi)$. The only way Alg accesses the input ϕ is by making queries to the SAT-Sample oracle, that is, obtaining random satisfying assignments from $\mathsf{sol}(\phi_A)$, where $\phi_A = \phi \wedge \psi_A$. We will prove that Alg has to make at least $\tilde{\Omega}(\log n)$ such queries to the SAT-Sample oracle.

We will start by arguing that we can assume that the adaptive algorithm Alg has some more structure. In particular, in Section 3.1 we will argue (in the same lines as in [3]) that we can assume Alg is a *semi-oblivious counter* (Definition 4).

We use Yao's Min-max technique to argue that obtaining a lower bound on a (randomized) semi-oblivious counter is the same as obtaining a lower bound on a (deterministic) semi-oblivious counter when the input is drawn from the worst possible distribution over the set of formulas on n variables. In Section 3.2 we present the "hard" distribution that would help us prove the lower bound against any deterministic semi-oblivious counter. In Section 3.2.1 we present some properties of the hard instance that would be used for the final lower bound proof.

Finally in Section 3.3 we will use an information-theoretic argument to give a lower bound on the query complexity of any deterministic semi-oblivious counter and hence prove Theorem 1.2.

A note on the use of auxiliary variables in the queries to the SAT-Sample oracle

One thing we observe is that our lower bound proof does not assume that in the input formula ϕ all the variables are influential. In other words, we can assume that ϕ is on n variables, the actual number of variables in ϕ may be significantly less. All we need for our lower bound proofs to go through is that the queries to the SAT-Sample oracle made by the algorithm are to $\phi \wedge \psi_A$ where the ψ is a formula over n variables. And the lower bound on the query complexity that we prove (Theorem 1.2) is $\tilde{O}(\log n)$. Hence, as long as the number of variables used in the queries to the SAT-Sample oracle is at most polynomial in the actual number of variables in the input formula ϕ, our lower bound holds.

3.1 Semi-oblivious counter

Suppose given a formula ϕ over n variables, a counter Alg makes q calls to the SAT-Sample oracle with queried formulas $\phi_{A_1}, \cdots, \phi_{A_q}$ respectively, where each $A_i \subseteq \{T,F\}^n$. (Recall, $\phi_{A_i} = \phi \wedge \psi_{A_i}$, where ψ_{A_i} denote the formula having $\mathsf{sol}(\psi_{A_i}) = A_i$.) Note, the i-th SAT-Sample oracle call by the counter Alg is specified by the set A_i. During the i-th call (for $1 \leq i \leq q$), suppose the counter Alg receives a sample $s_i \in A_i \cup \{\perp\}$. Note that the oracle calls made by Alg can be *adaptive*, i.e., the sets $A_1, \cdots, A_q$ are not fixed in advance – the counter Alg fixes A_i only after seeing the samples $s_1, \cdots, s_{i-1}$ (outcomes of all the previous oracle calls).

We now define a special type of randomized SAT-Sample counter, referred to as *semi-oblivious counter*, which at any point of time queries the SAT-Sample oracle only by looking into the configuration of the previous step. We will later argue that to prove a query lower bound for general SAT-Sample counters, it suffices to consider semi-oblivious counters. In other words, semi-oblivious counters are as "powerful" as general SAT-Sample counters.

We first provide intuition for *semi-oblivious counter*. Note that permuting the variables of any formula ϕ permutes the set of satisfying assignments $\mathsf{sol}(\phi)$ but $|\mathsf{sol}(\phi)|$ is unchanged. Since a SAT-Sample counter needs to determine $|\mathsf{sol}(\phi)|$ only (not $\mathsf{sol}(\phi)$), the final output by the SAT-Sample counter, in some sense, should be based only on the relations between the samples and the query sets (not on their actual values). Before providing a formal definition, let us first introduce some terminology.

Given a family of sets $\mathcal{A} = \{A_1, \cdots, A_i\}$, (where $A_i \subseteq \{T,F\}^n$), the *atoms* generated by $\mathcal{A}$, denoted by $\mathsf{At}(\mathcal{A})$, are (at most) 2^i distinct sets of the form $\cap_{j=1}^{i} C_j$ where $C_j \in \{A_j, \{T,F\}^n \setminus A_j\}$. For example, if $i = 2$, then $\mathsf{At}(A_1, A_2) = \{A_1 \cap A_2, A_1 \setminus A_2, A_2 \setminus A_1, (A_1 \cup A_2)^c\}$.

▶ **Definition 4** (Semi-oblivious counter). *A semi-oblivious counter is a randomized algorithm T that, given any formula ϕ, at any step i, works in the following three phases:*

- ***Semi-oblivious choice:*** *Let $\mathcal{A}_{i-1} = \{A_1, \cdots, A_{i-1}\}$, $S_{i-1} = \{s_1, \cdots, s_{i-1}\}$, $C_{i-1} = \{c_1, \cdots, c_{i-1}\}$ be the set of first $i-1$ query sets, the set of first $i-1$ samples obtained, the set of first $i-1$ configurations, respectively. Only based on C_{i-1} (without knowing the set S_{i-1}), T does the following:*
  - *For each $A \in \mathsf{At}(\mathcal{A}_{i-1})$, it generates an integer k_i^A between 0 and $|A \setminus S_{i-1}|$. (k_i^A indicates how many unseen elements from the atom A of the previous query sets are to be included in the next query set.)*
  - *It chooses a set of indices $K_i \subseteq \{1, \cdots, i-1\}$. ($K_i$ specifies the index set of previous samples that are to be included in the next query set.)*
- ***Query set generation:*** *In this phase, it decides the query set A_i as follows:*
  - *Let us define the* candidate unseen set family *as*

$$\mathcal{U}_i := \{U \subseteq \{T,F\}^n \setminus S_{i-1} \mid \forall A \in \mathsf{At}(\mathcal{A}_{i-1}), |U_i \cap A| = k_i^A\}.$$

    The algorithm T chooses a set U_i uniformly at random from the candidate unseen set family $\mathcal{U}_{i-1}$.
  - *Let us denote $O_i := \{s_j \mid j \in K_i\}$. The algorithm T decides the query set to be $A_i = U_i \cup O_i$.*
- ***Oracle call:*** *It places a query to the* SAT-Sample *oracle with the formula ϕ_{A_i}. Let the i-th* configuration *c_i specify whether $s_i = \perp$, or for which $j \in K_i$, $s_i = s_j$, or for which $A \in \mathsf{At}(\mathcal{A}_{i-1})$, $s_i \in A \cap U_i$.*

In the end (after placing $q = q(n)$ SAT-Sample *oracle calls), depending on the set of all the configurations C_q, T outputs an estimate on the $|\mathsf{sol}(\phi)|$.*

From now on, for brevity, we use $\mathsf{At}(U_i)$ to denote the set $\{U_i \cap A \mid A \in \mathsf{At}(\mathcal{A}_{i-1})\}$. Next, we show that if there exists a general SAT-Sample counter, then there also exists a semi-oblivious counter. The proof is inspired by the argument used in [3] and is given in Appendix A.

▶ **Lemma 5.** *If there is an algorithm that, given any input ϕ on n variables, outputs an (ϵ, δ)-approximation of $|\mathsf{sol}(\phi)|$ while placing at most $q = q(n)$* SAT-Sample *oracle calls, then there also exists a (randomized) semi-oblivious counter that, given input ϕ, outputs an (ϵ, δ)-approximation of $|\mathsf{sol}(\phi)|$ while also placing at most q* SAT-Sample *oracle calls.*

Suppose all the internal randomness of a semi-oblivious counter is fixed. (Since in the proof of Theorem 1.1, we will first apply Yao's minimax principle, it suffices to only consider deterministic decision trees.) Then, a semi-oblivious counter T can be fully described by a decision tree R where the path from the root to any node v at depth i (more precisely, the edges of this path) corresponds to the configuration of the first $i-1$ samples. Note that fixing the configurations of the samples till $i-1$ queries (and the internal randomness) fixes the size of an atom $A \in At(A_1, \cdots, A_i)$ (and hence of each A_j for $j \leq i$). Formally,

(i) A path (from root) to any node v at depth i is associated with a sequence of query sets $\mathcal{A}_{i-1} = (A_1, \cdots, A_{i-1})$ such that the sizes of all atoms $A \in \mathsf{At}(\mathcal{A}_{i-1})$ are fixed.

(ii) The node v is labeled by a vector $\boldsymbol{k}_v = (k_i^A)_{A \in \mathsf{At}(\mathcal{A}_{i-1})}$ and a set $K_v \subseteq [i-1]$ which are used to determine the next query set $A_i = O_i \cup U_i$. (Again, $|U_i| = \sum_{A \in \mathsf{At}(\mathcal{A}_{i-1})} k_i^A$ and the set U_i is fixed.) A_i is used to place the next SAT-Sample oracle call.

(iii) For every possible value of the configuration at step i, there is a corresponding child of the node v, with the corresponding edge labeled by the value of the configuration.

For any node v, we use $A_v = O_v \cup U_v$ to denote the (random) query set (corresponding to the node v) determined by the $\boldsymbol{k}_v$ and K_v. Note that $|U_v| = \sum_{A \in \mathsf{At}(\mathcal{A}_{i-1})} k_i^A$. Further, we use $\mathcal{A}_v := (A_1, \cdots, A_v)$ for the sequence of query sets corresponding to a path to v and node v. Observe the number of possible outcomes of the counter T at any step i is at most $i + 2^i + 1 \leq 2^{q+1}$ (since $i \leq q$). So the total number of nodes in the decision tree corresponding to the semi-oblivious counter T is at most $2^{O(q^2)}$.

3.2 Hard instance

We will provide a set of inputs $\mathcal{X}$ (which, in our case, will be a set of formulas) and a distribution Γ over $\mathcal{X}$. Then we will show that any deterministic semi-oblivious counter D (note that D knows $\mathcal{X}$ and Γ) which receives as input a formula $\phi \in \mathcal{X}$ randomly drawn as per distribution Γ and returns an (ϵ, δ)-approximation of $\mathsf{sol}(\phi)$, must make $\tilde{\Omega}(\log n)$ queries to the SAT-oracle.

Let $k = (\log n)^9$. Let $\mathcal{X}$ be the set of all formulas (with n variables). We now define the hard distribution Γ over $\mathcal{X}$ as follows by describing the procedure of picking a formula in $\mathcal{X}$ according to Γ.

1. Pick $\ell \in \{\lfloor n^{1/4} \rfloor, \lfloor n^{1/4} \rfloor + 1, \ldots, \lceil n^{3/4} \rceil\}$ uniformly at random.
2. Draw a hash function $h_\ell \leftarrow H(n, \ell, k)$ uniformly at random.
3. Let ϕ_ℓ be the formula whose set of satisfying assignments is $h_\ell^{-1}(F^\ell)$. (Recall, $h_\ell : \{T, F\}^n \to \{T, F\}^\ell$.)
4. The formula ϕ_ℓ is the picked formula.

3.2.1 Properties of the hard instance

Let $f_\ell := \mathbb{E}[|\mathsf{sol}(\phi_\ell)|] = \mathbb{E}[|h_\ell^{-1}(F^\ell)|]$ for $\ell \in \{\lfloor n^{1/4} \rfloor, \lfloor n^{1/4} \rfloor + 1, \ldots, \lceil n^{3/4} \rceil\}$. Observe, it follows from the construction of ϕ_ℓ and the properties of hash functions that $f_\ell = \frac{2^n}{2^\ell}$.

▶ **Lemma 6.** *With probability at least $1 - n2^{-n/20}$, we have*

$$\text{for all } \ell,\ ||\mathsf{sol}(\phi_\ell])| - f_\ell| \leq 2^{-n/10} f_\ell. \tag{3}$$

Proof. It is straightforward to see that the variance of $|\mathsf{sol}(\phi_\ell)|$ is $Var[|\mathsf{sol}(\phi_\ell)|] \leq f_\ell$. So by Chebyshev's inequality,

$$\Pr\left[||\mathsf{sol}(\phi_\ell)| - f_\ell| \geq 2^{-n/5} f_\ell\right] \leq \frac{2^{n/5}}{f_\ell} \leq \frac{2^{n/5} \cdot 2^\ell}{2^n} \leq 2^{-n/20}.$$

The lemma now follows from a union bound over all ℓ. ◀

▶ **Definition 7.** *Once $\ell \in \{\lfloor n^{1/4} \rfloor, \lfloor n^{1/4} \rfloor + 1, \ldots, \lceil n^{3/4} \rceil\}$ has been picked in Step 1 of the construction of the hard instance (Section 3.2), let for any $S \subseteq \{T, F\}^n$*

$$\mathsf{N}_\ell(S) = \mathbb{E}\left[|\mathsf{sol}(\phi_\ell) \cap S|\right],$$

where the expectation is over the choice of the hash function is Step 2 of the construction of the hard instance.

Note that for any $S \subseteq \{T, F\}^n$ the value of $\mathsf{N}_\ell(S)$ is $|S|/2^\ell$.

▶ **Lemma 8.** *With probability at least $1 - \frac{2^{O(q^2)}}{n^{(\log n)^4}}$, the following holds:*
For any node v in the decision tree R and any atom $A \in \mathsf{At}(U_v)$,

1. *If $\mathsf{N}_\ell(U_v) < \frac{1}{n^{(\log n)^4}}$ then $|U_v \cap \mathsf{sol}(\phi_\ell)| = 0$. Similarly, if $\mathsf{N}_\ell(A) < \frac{1}{n^{(\log n)^4}}$ for any atom $A \in \mathsf{At}(U_v)$ then $|A \cap \mathsf{sol}(\phi_\ell)| = 0$*

2. *If* $\mathsf{N}_\ell(U_v) \geq (\log n)^5$ *then* $\frac{1}{2}\mathsf{N}_\ell(U_v) \leq |U_v \cap \mathsf{sol}(\phi_\ell)| \leq \frac{3}{2}\mathsf{N}_\ell(U_v)$. *Similarly, if* $\mathsf{N}_\ell(A) \geq (\log n)^5$ *then* $\frac{1}{2}\mathsf{N}_\ell(A) \leq |A \cap \mathsf{sol}(\phi_\ell)| \leq \frac{3}{2}\mathsf{N}_\ell(A)$
3. *If* $\mathsf{N}_\ell(U_v) \leq (\log n)^5$ *then* $|U_v \cap \mathsf{sol}(\phi_\ell)| \leq 2(\log n)^5$. *Similarly, if* $\mathsf{N}_\ell(A) \leq (\log n)^5$ *then* $|A \cap \mathsf{sol}(\phi_\ell)| \leq 2(\log n)^5$.

Proof. From Markov's inequality, we have

$$\Pr[|U_v \cap \mathsf{sol}(\phi_\ell)| \geq 1] \leq \Pr\left[|U_v \cap \mathsf{sol}(\phi_\ell)| \geq \left(\frac{1}{\mathsf{N}_\ell(U_v)} - 1\right)\mathsf{N}_\ell(U_v)\right] \leq 2\mathsf{N}_\ell(U_v)$$

Taking a union bound over all nodes v with $\mathsf{N}_\ell(U_v) < \frac{1}{n^{(\log n)^4}}$ and all possible values of ℓ (which can take $O(n^{3/4})$ values), we get the first part.

From the first part of the Lemma 1, by setting $\gamma = 1/2$, we have

$$\Pr[|U_v \cap \mathsf{sol}(\phi_\ell)| \geq \mathsf{N}_\ell(U_v)] \leq \exp\left(-\frac{\mathsf{N}_\ell(U_v)}{12}\right)$$

for all nodes v in R such that $\mathsf{N}_\ell(U_v) \geq (\log n)^5$ (note that we have $k = (\log n)^9 > \gamma^2\mathsf{N}_\ell(U_v)$). Taking a union bound over all such nodes v and all possible values of ℓ, we get the second bound.

Let $\gamma_v = \frac{(\log n)^5}{\mathsf{N}_\ell(U_v)}$. Since $k = (\log n)^9 > \gamma_v\mathsf{N}_\ell(U_v)$, from the second part of Lemma 1, we have

$$\Pr[|U_v \cap \mathsf{sol}(\phi_\ell)| \geq \gamma_v\mathsf{N}_\ell(U_v)] \leq \exp\left(-\gamma_v\frac{\mathsf{N}_\ell(U_v)}{3}\right) \leq O\left(\frac{1}{n^{(\log n)^4}}\right).$$

for all nodes v such that $\mathsf{N}_\ell(U_v) \leq (\log n)^5$. Taking a union bound over all such nodes v and all possible values of ℓ, we get the third part. ◀

3.3 Proof of Theorem 1.2

Proof of Theorem 1.2. By Lemma 5 and Yao's minmax theorem we can assume that our SAT-Sample counter Alg is a (deterministic) semi-oblivious counter whose input is a randomly chosen formula $\phi \in \phi_n$, as per distribution Γ and Alg returns Est which is an $(\epsilon, 2/3)$-approximation of $|\mathsf{sol}(\phi)|$. We will prove that Alg must make $q = \tilde{\Omega}(\log n)$ many SAT-oracle calls.

Recall the distribution Γ (Section 3.2) over the set of all formulas. We can assume that the input to Alg is ϕ_ℓ, where ℓ is uniformly drawn from the set $\{\lfloor n^{1/4}\rfloor, \lfloor n^{1/4}\rfloor + 1, \ldots, \lceil n^{3/4}\rceil\}$.

Consider the path taken by the semi-oblivious counter Alg in the decision tree. Let the ith query made by Alg (that is at vertex v_i) be $A_i = U_i \cup O_i$ (as in Definition 4). Let Z_i be the configuration (denoted as c_i in Definition 4) of the sample from A_i. Note that the domain of Z_i is $\Omega_i := O_i \cup \mathsf{At}(U_i) \cup \perp$.

Let Good be the event that the condition in Equation 3 (in Lemma 6) and the condition in Lemma 8 holds. Note that by Lemma 6 and Lemma 8 if $q \leq \log n$ then

$$\Pr[\mathsf{Good}] = 1 - o(1). \tag{4}$$

Let X be the random variable that takes values in $\{\lfloor n^{1/4}\rfloor, \lfloor n^{1/4}\rfloor + 1, \ldots, \lceil n^{3/4}\rceil\}$ uniformly at random (in Step 1 of the construction of hard instance). Note that by the triangle inequality

$$|\mathsf{Est} - |\mathsf{sol}(\phi_\ell)|| \geq \left|\mathsf{Est} - \frac{2^n}{2^\ell}\right| - \left|\frac{2^n}{2^\ell} - |\mathsf{sol}(\phi_\ell)|\right|. \tag{5}$$

By Lemma 6 we know that with probability at least $(1 - 1/6)$, we have $|\frac{2^n}{2^\ell} - |\mathsf{sol}(\phi_\ell)|| \leq \frac{1}{2^{n/10}} \cdot \frac{2^n}{2^\ell}$. On the other hand, since Alg outputs an (ϵ, δ)-approximation of $|\mathsf{sol}(\phi)|$ (with $\epsilon < 1/2$ and $\delta < 1/6$), Equation 5 implies that with probability at least $(1 - \frac{1}{6} - \delta) \geq \frac{2}{3}$ we have

$$\left|\mathsf{Est} - \frac{2^n}{2^\ell}\right| \leq \left(\epsilon + \frac{1}{2^{n/10}}\right)\frac{2^n}{2^\ell} \leq \frac{1}{2} \cdot \frac{2^n}{2^\ell}, \tag{6}$$

where the last inequality follows from the fact that $\epsilon \leq 1/3$. Since $|\frac{2^n}{2^\ell} - \frac{2^n}{2^{\ell'}}| > \frac{1}{2} \cdot \frac{2^n}{2^\ell}$ for any integer $\ell' \neq \ell$, so Equation 6 is satisfied only when $\hat{X}$ is same as the picked ℓ (that is $\hat{X} = X$) where,

$$\hat{X} = \underset{\ell \in \{\lfloor n^{1/4} \rfloor, \lfloor n^{1/4} \rfloor + 1, \ldots, \lceil n^{3/4} \rceil\}}{\arg\min} \left|\frac{2^n}{2^\ell} - \mathsf{Est}\right|.$$

Hence, assuming Good

$$\frac{1}{3} \geq \Pr[\hat{X} \neq X]. \tag{7}$$

By Fano's Inequality (Theorem 3)

$$\Pr[\hat{X} \neq X] \geq 1 - \frac{I(X; \hat{X})}{O(\log n)} \tag{8}$$

Since the final outcome of the algorithm is determined by the outcome at each step, i.e., $\boldsymbol{Z} = (Z_1, \ldots, Z_q)$, so by the data processing inequality (Equation 2), we have

$$I(X; \hat{X}) \leq I(X; Z_1, \ldots, z_q). \tag{9}$$

Let Y_i be the random variable that defined as

$$Y_i = \begin{cases} 1 & \text{if } \frac{1}{n^{(\log n)^4}} \leq \mathsf{N}_\ell(U_i) \leq n^{(\log n)^4} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Again by the data-processing inequality (Equation 2), we have

$$I(X; Z_1, \ldots, Z_q) \leq I(X; Y_1, Z_1, \ldots, Y_q, Z_q). \tag{11}$$

By the chain rule of mutual information, we have

$$I(X; Y_1, Z_1, \ldots, Y_q, Z_q) = \sum_{i \in [q]} I(X; Y_i, Z_i | Y_1, Z_1, \ldots, Y_{i-1}, Z_{i-1}) \tag{12}$$

Finally, we will show, in the following lemma, that conditioned on the fact Good happens we can upper bound $I(X; Y_1, Z_1, \ldots, Y_q, Z_q)$ by $O(\log \log n)$.

▶ **Lemma 9.** $I(X; (Y_1, Z_1, \ldots, Y_q, Z_q)) \leq q(O(\log \log n) + O(\log q) + \frac{2^{2q} poly(\log n)}{n^{(\log n)^3}})$.

We defer the proof of Lemma 9 and complete the proof of Theorem 1.2 assuming Lemma 9.

From the Equations 7, 8, 9, 11 and Lemma 9, we have that assuming Good happens

$$\begin{aligned}
\frac{1}{3} \geq & \Pr[\hat{X} \neq X] && \text{[From Equation 7]}\\
\geq & 1 - \frac{I(X;\hat{X})}{O(\log n)} && \text{[From Equation 8]}\\
\geq & 1 - \frac{I(X;Z_1,\ldots,z_q)}{O(\log n)} && \text{[From Equation 9]}\\
\geq & 1 - \frac{I(X;Y_1,Z_1,\ldots,Y_q,Z_q)}{O(\log n)} && \text{[From Equation 11]}\\
\geq & 1 - \frac{I(X;Y_1,Z_1,\ldots,Y_q,Z_q)}{O(\log n)} && \text{[From Equation 12]}\\
\geq & 1 - \frac{q \log\log n}{\log n} && \text{[From Lemma 9]}
\end{aligned}$$

Thus, from Equation 4, if $q \leq \log n$

$$1 - \frac{q \log\log n}{\log n} \leq \frac{1}{3} + \Pr[\mathsf{Good}] \leq \frac{1}{3} + O(1)$$

which implies

$$q = \Omega\left(\frac{\log n}{\log\log n}\right).$$ ◀

3.3.1 Proof of Lemma 9

▶ **Lemma 10.** *The following holds:*

1. *Conditioned on event that $Y_j = 1$ for some $j \leq i$,*
$$I(X; Z_i | Y_1, Z_1, \ldots, Y_{i-1}, Z_{i-1}, Y_i) \leq O(\log\log n),$$
2. $I(X, Y_i | Y_1, Z_1, \ldots, Y_{i-1}, Z_{i-1}) \leq 1$,
3. *Conditioned on the event that $Y_1 = 0, \ldots, Y_{i-1} = 0$,*
$$I(X, Z_i | Y_1, Z_1, \ldots, Y_{i-1}, Z_{i-1}, Y_i) \leq O(\log q) + \frac{2^{2q} poly(\log n)}{n^{(\log n)^3}}.$$

Proof. We will prove Part 1, 2, and 3 one by one.

Proof of Part 1. We will prove that conditioned on event that $Y_j = 1$ for some $j \leq i$,

$$I(X; Z_i | Y_1, Z_1, \ldots, Y_{i-1}, Z_{i-1}, Y_i) \leq O(\log\log n).$$

From (1), we have

$$I(X, Z_i | Y_1, Z_1, \ldots, Y_{i-1}, Z_{i-1}, Y_i) \leq H(X | Y_1, Z_1, \ldots, Y_{i-1}, Z_{i-1}, Y_i).$$

Note that if $Y_j = 1$ then by definition of Y_j we have $\frac{1}{n^{(\log n)^4}} \leq \frac{|U_j|}{2^\ell} \leq n^{(\log n)^4}$, that is,

$$\frac{|U_j|}{n^{(\log n)^4}} \leq 2^\ell \leq |U_j| n^{(\log n)^4}.$$

Note that by definition of the semi-oblivious counter the sets $|U_1|, \ldots, |U_i|$ are deterministically determined by $Z_1, \ldots, Z_i$. Thus, there are $O(\log(n^{(\log n)^4})) = O((\log n)^5)$ possible values of ℓ and hence

$$H(X | Y_1, Z_1, \ldots, Y_{i-1}, Z_{i-1}) \leq O(\log\log n).$$

This proves the first part.

Proof of Part 2. Since Y_i can take only binary values, we have

$$I(X, Y_i | Y_1, Z_1, \ldots, Y_{i-1}, Z_{i-1}) \leq 1.$$

This proves Part 2.

Proof of Part 3. We will now prove the upper bound on $I(X; (Y_i, Z_i) | Y_1, Z_1, \ldots, Y_{i-1}, Z_{i-1})$ for each $i \in [q]$, conditioned on $Y_j = 0$ for all $j \in [i]$.

Note that $Z_1, \ldots, Z_{i-1}$ fixes the size of O_i and each atoms in $\mathsf{At}(U_i)$. Note that the domain of Z_i, i.e., Ω_i is $\perp \cup O_i \cup \mathsf{At}(U_i)$. Let $r = |O_i| + 2 \leq q + 2$.

We define an auxiliary distribution $Q_{(Y_i,Z_i)}$ as follows:

$$Q_{(Y_i,Z_i)}(y_i, z_i) := Q_{Y_i}(y_i) Q_{Z_i|Y_i}(z_i | y_i)$$

where, $Q_{Y_i}(0) = Q_{Y_i}(1) = 1/2$ and

$$Q_{Z_i|Y_i}(z_i|y_i) = \begin{cases} \frac{1}{r}, & z_i \in O_i \cup \perp \\ \frac{1}{r} \cdot \frac{|z_i|}{|U_i|}, & z_i \in \mathsf{At}(U_i) \end{cases}$$

Let $P_X, P_Z, P_{Z|X}$ be the marginal distributions corresponding to a pair (X, Z). Conditioned on $Y_j = 0$ for all $j \in [i]$ and $Z_j = z_j$ for all $j \in [i-1]$ for any $(z_1, \ldots, z_{i-1}) \in \Omega_1 \times \cdots \times \Omega_{i-1}$, we have for any $\ell \in \mathcal{X}$ (note that, for brevity, we have ignored the conditioning on $Y_1, Z_1, \ldots, Y_{i-1}, Z_{i-1}$, in the expression below)

$$KL(P_{Z_i|X}(\cdot|X = \ell) || Q_{Z_i}) = \sum_{z_i \in \Omega_i} P_{Z_i|X}(z_i|X = \ell) \log \frac{P_{Z|X}(z_i|X = \ell)}{Q_{Z_i}(z_i)} \tag{13}$$

Note that if $z_i \in \perp \cup O_i$ then $Q_{Z_i}(z_i) = \frac{1}{r} \geq \frac{1}{q+2}$. Hence,

$$\frac{P_{Z_i|X}(z_i|X = \ell)}{Q_{Z_i}(z_i)} \leq q + 2 \leq 2q.$$

Now we consider the case when $z_i \in \mathsf{At}(U_i)$.

If $\mathsf{N}_\ell(z_i) \geq (\log n)^5$ then from Lemma 8 we have

$$P_{Z_i|X}(z_i|X = \ell) = \frac{|z_i \cap \mathsf{sol}(\phi_\ell)|}{|U_i \cap \mathsf{sol}(\phi_\ell)|} \leq 3\mathsf{N}_\ell(z_i)/\mathsf{N}_\ell(U_i).$$

Note that

$$Q_{Z_i}(z_i) = \frac{1}{r} \cdot \frac{|z_i|}{|U_i|} \geq 2q\mathsf{N}_\ell(z_i)/\mathsf{N}_\ell(U_i).$$

Therefore, we have

$$\frac{P_{Z_i|X}(z_i|X = \ell)}{Q_{Z_i}(z_i)} \leq O(q).$$

For the case when $\mathsf{N}_\ell(z_i) < \frac{1}{n^{(\log n)^4}}$, we have $|z_i \cap \mathsf{sol}(\phi_\ell)| = 0$. Hence the sum

$$\sum_{z_i} P_{Z_i|X}(z_i|X = \ell) \log \frac{P_{Z|X}(z_i|X = \ell)}{Q_{Z_i}(z_i)}$$

when, $z_i \in \perp \cup O_i$ or $z_i \in \mathsf{At}(U_i)$ such that $\mathsf{N}_\ell(z_i) \geq (\log n)^5$ or $\mathsf{N}_\ell(z_i) < \frac{1}{n^{(\log n)^4}}$, is at most $O(\log q)$.

Now we bound the sum

$$\sum_{z_i} P_{Z_i|X}(z_i|X=\ell)\log\frac{P_{Z|X}(z_i|X=\ell)}{Q_{Z_i}(z_i)}$$

when $z_i \in \mathsf{At}(U_i)$ such that

$$\frac{1}{n^{(\log n)^4}} < \mathsf{N}_\ell(z_i) < (\log n)^5.$$

If $\mathsf{N}_\ell(z_i) \le (\log n)^5$ then we have

$$|z_i \cap \mathsf{sol}(\phi_\ell)| \le 2(\log n)^5$$

and thus

$$P_{Z_i|X}(z_i|X=\ell) \le \frac{4(\log n)^5}{\mathsf{N}_\ell(U_i)}.$$

Note that

$$Q_{Z_i}(z_i) = \frac{1}{r}\cdot\frac{|z_i|}{|U_i|} \ge \frac{1}{2q}\mathsf{N}_\ell(z_i)/\mathsf{N}_\ell(U_i).$$

Hence,

$$\frac{P_{Z|X}(z_i|X=\ell)}{Q_{Z_i}(z_i)} \le O(q(\log n)^5/\mathsf{N}_\ell(z_i)).$$

Therefore, we have

$$\begin{aligned}
&\sum_{z_i:\frac{1}{n^{(\log n)^4}}<\mathsf{N}_\ell(z_i)\le(\log n)^5} P_{Z_i|X}(z_i|X=\ell)\log\frac{P_{Z|X}(z_i|X=\ell)}{Q_{Z_i}(z_i)}\\
&< \sum_{z_i:\frac{1}{n^{(\log n)^4}}<\mathsf{N}_\ell(z_i)\le(\log n)^5} \frac{4(\log n)^5}{\mathsf{N}_\ell(U_i)}\log(2q(\log n)^5/\mathsf{N}_\ell(z_i))\\
&\le 2^q\frac{8(\log n)^5}{n^{(\log n)^4}}\log(2q(\log n)^5 n^{(\log n)^4})\\
&\le \frac{2^{2q}poly(\log n)}{n^{(\log n)^4}}.
\end{aligned}$$

The second last inequality follows because there are at most 2^q possible values of such z_i, $\mathsf{N}_\ell(U_i) \ge n^{(\log n)^3}/2$ and $\mathsf{N}_\ell(z_i) \ge \frac{1}{n^{(\log n)^3}}$.

Now by Lemma 2 conditioned on the event that $Y_j = 0$ for all $j \le i$ we have

$$\begin{aligned}
I(X;Z_i|Y_1,Z_1,\ldots,Y_{i-1},Z_{i-1},Y_i) &\le KL(P_{Z_i|X}(\cdot|X=\ell)||Q_{Z_i})\\
&\le O(\log q) + \frac{2^{2q}poly(\log n)}{n^{(\log n)^3}}.
\end{aligned}$$

◀

Proof of Lemma 9. We will first prove that for any i

$$I(X;(Y_i,Z_i)|Y_1,Z_1,\ldots,Y_{i-1},Z_{i-1}) \le O(\log\log n) + O(\log q) + \frac{2^{2q}poly(\log n)}{n^{(\log n)^3}}.$$

By the chain rule of mutual information,

$$\begin{aligned}
&I(X;(Y_i,Z_i)|Y_1,Z_1,\ldots,Y_{i-1},Z_{i-1})\\
=&I(X;Y_i|Y_1,Z_1,\ldots,Y_{i-1},Z_{i-1})+I(X;Z_i|Y_1,Z_1,\ldots,Y_{i-1},Z_{i-1},Y_i)\\
\leq&O(\log\log n)+O(\log q)+\frac{2^{2q}poly(\log n)}{n^{(\log n)^3}},
\end{aligned}$$

where the last inequality follows from Lemma 10.

Again by the chain rule of mutual information, we have

$$\begin{aligned}
&I(X;(Y_1,Z_1,\ldots,Y_q,Z_q))\\
=&\sum_{i=1}^{q} I(X;(Y_i,Z_i)|Y_1,Z_1,\ldots,Y_{i-1},Z_{i-1})\\
\leq& q(O(\log\log n)+O(\log q)+\frac{2^{2q}poly(\log n)}{n^{(\log n)^3}}).
\end{aligned}$$

◀

4 Conclusion

In this paper, we study the power of SAT oracles in the context of approximate model counting and show a lower bound of $\tilde{\Omega}(\log n)$ on the number of oracle calls. This is in contrast to other settings where a SAT oracle is provably more powerful than an NP oracle. In fact, we prove that even with a much more powerful oracle (namely SAT-Sample oracle), the number of queries needed to approximately count the number of satisfying assignments of a Boolean formula is $\tilde{\Omega}(\log n)$.

References

1. Teodora Baluta, Shiqi Shen, Shweta Shinde, Kuldeep S Meel, and Prateek Saxena. Quantitative verification of neural networks and its security applications. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1249–1264, 2019.
2. Mihir Bellare, Oded Goldreich, and Erez Petrank. Uniform generation of NP-witnesses using an NP-oracle. *Information and Computation*, 163(2):510–526, 2000.
3. Sourav Chakraborty, Eldar Fischer, Yonatan Goldhirsh, and Arie Matsliah. On the power of conditional samples in distribution testing. *SIAM J. Comput.*, 45(4):1261–1296, 2016. doi:10.1137/140964199.
4. Supratik Chakraborty, Kuldeep S Meel, and Moshe Y Vardi. Algorithmic improvements in approximate counting for probabilistic inference: From linear to logarithmic sat calls. Technical report, Rice University, 2016.
5. Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company, 1989.
6. Thomas M. Cover and Joy A. Thomas. *Elements of information theory (2. ed.)*. Wiley, 2006.
7. Remi Delannoy and Kuldeep S Meel. On almost-uniform generation of SAT solutions: The power of 3-wise independent hashing. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 1–10, 2022.
8. Leonardo Duenas-Osorio, Kuldeep Meel, Roger Paredes, and Moshe Vardi. Counting-based reliability estimation for power-transmission grids. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31(1), 2017.
9. Matthew Fredrikson and Somesh Jha. Satisfiability modulo counting: A new approach for analyzing privacy properties. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–10, 2014.

10 Carla P Gomes, Ashish Sabharwal, and Bart Selman. Model counting: A new strategy for obtaining good bounds. In *AAAI*, volume 10, pages 1597538–1597548, 2006.

11 Mark R Jerrum, Leslie G Valiant, and Vijay V Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical computer science*, 43:169–188, 1986.

12 Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2):273–302, 1996.

13 Tian Sang, Paul Beame, and Henry A Kautz. Performing bayesian inference by weighted model counting. In *AAAI*, volume 5, pages 475–481, 2005.

14 Jonathan Scarlett and Volkan Cevher. An introductory guide to Fano's inequality with applications in statistical estimation. *arXiv preprint*, 2019. `arXiv:1901.00555`.

15 Jeanette P Schmidt, Alan Siegel, and Aravind Srinivasan. Chernoff–Hoeffding bounds for applications with limited independence. *SIAM Journal on Discrete Mathematics*, 8(2):223–250, 1995.

16 Larry Stockmeyer. The complexity of approximate counting. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 118–126, 1983.

17 Leslie G Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

18 Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (SFCS 1977)*, pages 222–227. IEEE Computer Society, 1977.

A Proof of Lemma 5

Consider any general SAT-Sample counter, T. We will show that there exists a semi-oblivious counter that performs similarly. Given a sequence of query-sample pairs $\{(A_1, s_1), \ldots, (A_{i-1}, s_{i-1})\}$, we say the query A_i is a good strategy by T (given $\{(A_1, s_1), \ldots, (A_{i-1}, s_{i-1})\}$) if the counter T can return the correct output by fixing the next query to A_i. It suffices to show that, given a sequence of query-sample pairs $\{(A_1, s_1), \ldots, (A_{i-1}, s_{i-1})\}$, if A_i is a good strategy then any A'_i is also a good strategy if $A'_i \cap \{s_1, \ldots, s_{i-1}\} = A_i \cap \{s_1, \ldots, s_{i-1}\}$ and $|A'_i \cap A| = |A_i \cap A|$ for atoms $A \in At(A_1, \ldots, A_{i-1})$. This means that to fix the next query, all it requires to fix the intersection size with each atom $A \in At(A_1, \ldots, A_i)$ and a subset of $\{s_1, \ldots, s_{i-1}\}$ (to be included in next query). We prove it in the following claim.

▷ Claim 11. Suppose A_i is a good strategy for $\{(A_1, s_1), \ldots, (A_{i-1}, s_{i-1})\}$. Consider A'_i such that $A'_i \cap \{s_1, \ldots, s_{i-1}\} = A_i \cap \{s_1, \ldots, s_{i-1}\}$ and $|A'_i \cap A| = |A_i \cap A|$ for atoms $A \in At(A_1, \ldots, A_{i-1})$. Then A'_i is also a good strategy for $\{(A_1, s_1), \ldots, (A_{i-1}, s_{i-1})\}$.

Proof. We denote by $\mathcal{S}_N$ the symmetric group acting on a set of size N. Any $\sigma \in \mathcal{S}_N$ can be thought of acting on any set of size N (by thinking the elements of the set as numbered $1, \ldots, N$ and σ acting on the set $[N]$). For any element x in the set, we will denote by $\sigma(x)$ the element after the action of σ. For any $\sigma \in \mathcal{S}_N$ and set A (with $|A| = N$) we denote by $\sigma(A)$ the following set $\sigma(A) := \{\sigma(x) \mid x \in A\}$.

Let $\sigma \in \mathcal{S}_{2^n}$ be a permutation acting on the set $\{T, F\}^n$. For any ϕ observe that $|\mathsf{sol}(\phi)| = |\sigma(\mathsf{sol}(\phi))|$. Since any counter estimates $|\mathsf{sol}(\phi)|$ only, we observe that if A_i is a good strategy for $\{(A_1, s_1), \ldots, (A_{i-1}, s_{i-1})\}$ then $\sigma(A_i)$ is also a good strategy for $\{(\sigma(A_1), \sigma(s_1)), \ldots, (\sigma(A_{i-1}), \sigma(s_{i-1})\}$ for any $\sigma : \{T, F\}^n \to \{T, F\}^n$ that preserves the atoms $\mathsf{At}(A_1, \ldots, A_{i-1})$ and the elements $\{s_1, \ldots, s_{i-1}\}$.

Since $|A'_i \cap A| = |A_i \cap A|$ for atoms $A \in At(A_1, \ldots, A_{i-1})$ and $A'_i \cap \{s_1, \ldots, s_{i-1}\} = A_i \cap \{s_1, \ldots, s_{i-1}\}$, there exists a σ such that $\sigma(A_j) = A_j$, $\sigma(s_j) = s_j$ for all $j \leq i-1$ and also $\sigma(A_i) = A'_i$. By our earlier observation, A'_i is also a good strategy for $\{(A_1, s_1), \ldots, (A_{i-1}, s_{i-1})\}$. ◁

The Identity Problem in $\mathbb{Z} \wr \mathbb{Z}$ Is Decidable

Ruiwen Dong ✉
Department of Computer Science, University of Oxford, UK

Abstract

We consider semigroup algorithmic problems in the wreath product $\mathbb{Z} \wr \mathbb{Z}$. Our paper focuses on two decision problems introduced by Choffrut and Karhumäki (2005): the *Identity Problem* (does a semigroup contain the neutral element?) and the *Group Problem* (is a semigroup a group?) for finitely generated sub-semigroups of $\mathbb{Z} \wr \mathbb{Z}$. We show that both problems are decidable. Our result complements the undecidability of the *Semigroup Membership Problem* (does a semigroup contain a given element?) in $\mathbb{Z} \wr \mathbb{Z}$ shown by Lohrey, Steinberg and Zetzsche (ICALP 2013), and contributes an important step towards solving semigroup algorithmic problems in general metabelian groups.

2012 ACM Subject Classification Computing methodologies → Symbolic and algebraic manipulation

Keywords and phrases wreath product, algorithmic group theory, identity problem, polynomial semiring, positive coefficients

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.124

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `https://arxiv.org/abs/2302.05939`

Funding The author acknowledges support from UKRI Frontier Research Grant EP/X033813/1.

Acknowledgements The author would like to thank Markus Schweighofer and David Sawall for useful discussions.

1 Introduction

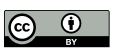
The computational theory of groups and semigroups is one of the oldest and most well-developed parts of computational algebra. Dating back to the work of Markov [24] in the 1940s, the area plays an essential role in analysing system dynamics, with notable applications in automata theory and program analysis [5, 9, 11, 16]. See [19] for an all-encompassing survey on this topic. Among the most prominent problems in this area are *Semigroup Membership* and *Group Membership*, proposed respectively by Markov and Mikhailova in the 1940s and 1960s. For these decision problems, we work in a fixed group G. The input is a finite set of elements $\mathcal{G} \subseteq G$, plus a distinguished element $g \in G$. Denote by $\langle \mathcal{G} \rangle$ the semigroup generated by $\mathcal{G}$, and by $\langle \mathcal{G} \rangle_{grp}$ the group generated by $\mathcal{G}$.

(i) *(Semigroup Membership)* decide whether $\langle \mathcal{G} \rangle$ contains g.
(ii) *(Group Membership)* decide whether $\langle \mathcal{G} \rangle_{grp}$ contains g.

In this paper, we consider two problems closely related to (i) and (ii): the *Identity Problem* and the *Group Problem*, both introduced by Choffrut and Karhumäki [9] in 2005.

(iii) *(Identity Problem)* decide whether $\langle \mathcal{G} \rangle$ contains the neutral element I of G.
(iv) *(Group Problem)* decide whether $\langle \mathcal{G} \rangle$ is a group, in other words, whether $\langle \mathcal{G} \rangle = \langle \mathcal{G} \rangle_{grp}$.

In general matrix groups, Semigroup Membership is undecidable by a classical result of Markov [24]. All four problems remain undecidable even for integer matrix groups of dimension four [4, 25]. Notably, using an embedding of the *Identity Correspondence Problem*, Bell and Potapov [4] showed undecidability of the Identity Problem and the Group Problem in the group $\mathsf{SL}(4, \mathbb{Z})$ of 4×4 integer matrices of determinant one. On the other hand, in the



group $\mathsf{SL}(2,\mathbb{Z})$, all four problems are decidable with various degrees of complexity [3, 9, 21]. In particular, the Identity Problem and the Group Problem in $\mathsf{SL}(2,\mathbb{Z})$ are both **NP**-complete by a result of Bell, Hirvensalo, and Potapov [3].

In this paper we focus on these decision problems in the *wreath product* $\mathbb{Z} \wr \mathbb{Z}$. The wreath product is a fundamental construction in group and semigroup theory. A great number of important groups can be constructed using the wreath product, notably *metabelian groups*. Metabelian groups are groups whose commutator is abelian: these are the simplest generalization of abelian groups. Algorithmic problems in metabelian groups have been the focus of active research since the 1970s [2, 8, 26], with a classic result of Romanovskii [27] showing decidability of Group Membership in all finitely presented metabelian groups. A key part of Romanovskii's proof is to embed metabelian groups into quotients of wreath products. In fact, the Magnus embedding theorem [23] states that every finitely generated free metabelian group can be embedded in a wreath product $\mathbb{Z}^m \wr \mathbb{Z}^n$. Therefore, understanding the wreath product $\mathbb{Z} \wr \mathbb{Z}$ is the most crucial step towards studying general metabelian groups. Apart from its interest within group theory, the wreath product also plays an important role in the algebraic theory of automata. The Krohn–Rhodes theorem [20] states that every finite semigroup (and correspondingly, every finite automaton) can be decomposed into elementary components using wreath products.

One easy way to understand the wreath product $\mathbb{Z} \wr \mathbb{Z}$ is through its isomorphism to a matrix group [23] over the Laurent polynomial ring $\mathbb{Z}[X^{\pm}]$:

$$\mathbb{Z} \wr \mathbb{Z} \cong \left\{ \begin{pmatrix} X^b & y \\ 0 & 1 \end{pmatrix} \;\middle|\; y \in \mathbb{Z}[X^{\pm}], b \in \mathbb{Z} \right\}. \tag{1}$$

Consider the four aforementioned decision problems in $\mathbb{Z} \wr \mathbb{Z}$. Since $\mathbb{Z} \wr \mathbb{Z}$ is metabelian [17], the classic result of Romanovskii [27] shows decidability of Group Membership in $\mathbb{Z} \wr \mathbb{Z}$. If we retrace the proof of Romanovskii, one can reduce Group Membership in $\mathbb{Z} \wr \mathbb{Z}$ to solving systems of linear equations over the ring $\mathbb{Z}[X^{\pm}]$, which can then be decided using Gröbner bases. For Semigroup Membership in $\mathbb{Z} \wr \mathbb{Z}$, Lohrey, Steinberg and Zetzsche showed its undecidability using an encoding of 2-counter machines [22]. Decidability of the Identity Problem and the Group Problem in $\mathbb{Z} \wr \mathbb{Z}$ remained an intricate open problem. A recent paper by Dong [12] gave a partial decidability result when the generators all satisfy $b = \pm 1$. Dong's idea was to represent a product of elements in $\mathbb{Z} \wr \mathbb{Z}$ as a walk on $\mathbb{Z}$. When the generators all satisfy $b = \pm 1$, this walk can be decomposed into simple cycles, and the Group Problem reduces to solving a *single* homogeneous linear equation over the semiring $\mathbb{N}[X]$. Extending Dong's result to arbitrary generators is highly challenging: the structure of the walk becomes much more complex when we allow steps of arbitrary length. In this paper, we combine a series of new ideas from graph theory and algebraic geometry to show full decidability of the Identity Problem and the Group Problem in $\mathbb{Z} \wr \mathbb{Z}$.

The first main idea of this paper is to reduce both problems to solving a *system* of homogeneous linear equations over the semiring $\mathbb{N}[X^{\pm}]$, in addition to two *degree constraints*. We use a highly non-trivial graph theoretic construction to establish this reduction. The second main idea of this paper is to generalize a local-global principle by Einsiedler, Mouat and Tuncel [13] to solve these linear equations with degree constraints. In particular, the original local-global principle by Einsiedler et al. is not compatible with the degree constraints which are essential in our reduction. We introduce new ideas to prove a generalized local-global principle that incorporates these additional degree constraints.

We now mention some other known decidability results in wreath products. In [14], Ganardi, König, Lohrey and Zetzsche showed that for every non-trivial finitely generated abelian group G, the *knapsack problem* in $G \wr \mathbb{Z}$ is **NP**-complete. Notably, this result applies to $\mathbb{Z} \wr \mathbb{Z}$. In [22], Lohrey, Steinberg and Zetzsche showed decidability of the *Rational Subset*

Membership Problem (which subsumes all four decision problems mentioned in the beginning) in the wreath product $H \wr V$, where H is a finite and V is virtually free. In [7], Cadilhac, Chistikov and Zetzsche proved decidability of Rational Subset Membership in the *Baumslag-Solitar groups* $\mathsf{BS}(1,p)$. This group can be considered as an analogue of $(\mathbb{Z}/p\mathbb{Z}) \wr \mathbb{Z}$ "with carrying". In [17], Kharlampovich, López, and Myasnikov showed decidability of solving Diophantine equations in certain metabelian groups, including $\mathbb{Z} \wr \mathbb{Z}$ and $\mathsf{BS}(1,p)$. Many of these results are closely related to automata theory, which we draw inspiration from.

A natural follow-up to our work would be trying to solve the Identity Problem and the Group Problem in *all* finitely presented metabelian groups. This boils down to deciding both problems in quotients of $\mathbb{Z}^m \wr \mathbb{Z}^n$. One encounters some difficulties when trying to generalize our approach to $\mathbb{Z}^m \wr \mathbb{Z}^n$. Notably, decomposition of walks in $\mathbb{Z}^n$ are much more complex, and we can no longer reduce these problems to solving a finite system of equations. We also point out that one cannot go much further beyond metabelian groups (which are 2-step solvable groups), since there exist 3-step solvable groups with undecidable word problem [18].

2 Preliminaries

Words, semigroups and graphs

Let G be an arbitrary group. Let $\mathcal{G} = \{g_1, \ldots, g_a\}$ be a finite set of elements in G. Considering $\mathcal{G}$ as an alphabet, denote by $\mathcal{G}^*$ the set of words over $\mathcal{G}$. For an arbitrary word $w = g_{i_1} g_{i_2} \cdots g_{i_m} \in \mathcal{G}^*$, by multiplying consecutively the elements appearing in w, we can evaluate w as an element $\pi(w)$ in G. We say that the word w *represents* the element $\pi(w)$. The semigroup $\langle \mathcal{G} \rangle$ generated by $\mathcal{G}$ is hence the set of elements in G that are represented by *non-empty* words in $\mathcal{G}^*$.

A word w over the alphabet $\mathcal{G}$ is called *full-image* if every letter in $\mathcal{G}$ has at least one occurrence in w. The following observation shows that deciding the Group Problem amounts to finding a full-image word representing the neutral element.

▶ **Lemma 2.1.** *Let $\mathcal{G} = \{g_1, \ldots, g_a\}$ be a set of elements in a group G. The semigroup $\langle \mathcal{G} \rangle$ is a group if and only if the neutral element I of G is represented by a full-image word over $\mathcal{G}$.*

The following lemma shows that decidability of the Group Problem implies decidability of the Identity Problem.

▶ **Lemma 2.2** ([4])**.** *Given a finite subset $\mathcal{G}$ of a group G, the semigroup $\langle \mathcal{G} \rangle$ contains the neutral element I if and only if there exists a non-empty subset $\mathcal{H} \subseteq \mathcal{G}$ such that $\langle \mathcal{H} \rangle$ is a group. In particular, if the Group Problem is decidable in the group G, then the Identity Problem is also decidable.*

For detailed definition of graph theory terms, see [6]. All graphs considered in this paper will be directed multigraphs. For a graph G, we denote by $V(G)$ its set of vertices and by $E(G)$ its set of edges. For a (directed) edge e, we denote by $s(e)$ the starting vertex of e.

A *loop* is an edge that starts and ends at the same vertex. A *circuit* is a path that starts and ends at the same vertex. An *Euler path* of a graph G is a path that uses each edge exactly once. An *Euler circuit* is an Euler path that starts and ends at the same vertex. We call a graph *Eulerian* if it contains an Euler circuit. It is easy to see that attaching a circuit to an Eulerian graph still results in an Eulerian graph.

Laurent polynomials and the wreath product $\mathbb{Z} \wr \mathbb{Z}$

A (univariate) *Laurent polynomial* with coefficients over $\mathbb{R}$ is an expression of the form

$$f = \sum_{i=p}^{q} a_i X^i, \quad \text{where } p, q \in \mathbb{Z} \text{ and } a_i \in \mathbb{R}, i = p, p+1, \ldots, q.$$

If $p > q$, then f is understood to be zero. Otherwise, $p \leq q$, and we suppose $a_p \neq 0$, $a_q \neq 0$. In this case, we call p the *negative degree* of f, denoted by $\deg_-(f)$, and call q the *positive degree* of f, denoted by $\deg_+(f)$. We call a_p the *negative leading coefficient* of f, denoted by $\mathrm{lc}_-(f)$, and call a_q the *positive leading coefficient* of f, denoted by $\mathrm{lc}_+(f)$. Define additionally $\deg_-(0) = +\infty$, $\deg_+(0) = -\infty$ and $\mathrm{lc}_-(0) = \mathrm{lc}_+(0) = 0$. In this paper, all polynomials will be univariate Laurent polynomials. The set of all polynomials with coefficients over $\mathbb{R}$ forms a ring and is denoted by $\mathbb{R}[X^{\pm}]$. One can define $\mathbb{Q}[X^{\pm}]$ and $\mathbb{Z}[X^{\pm}]$ similarly by restricting the coefficients a_i to $\mathbb{Q}$ and $\mathbb{Z}$.

Given a tuple of polynomials $\boldsymbol{f} = (f_1, \ldots, f_n) \in (\mathbb{R}[X^{\pm}])^n$ and $r \in \mathbb{R}$, one naturally defines the evaluation $\boldsymbol{f}(r) \coloneqq (f_1(r), \ldots, f_n(r)) \in \mathbb{R}^n$. The definition of leading coefficients also extends to tuples of polynomials by $\mathrm{lc}_*(\boldsymbol{f}) \coloneqq (\mathrm{lc}_*(f_1), \ldots, \mathrm{lc}_*(f_n)) \in \mathbb{R}^n$, where $* \in \{+, -\}$.

Consider the semiring $\mathbb{R}_{\geq 0}[X^{\pm}]$ of polynomials with positive coefficients: these are expressions of the form $f = \sum_{i=p}^{q} a_i X^i$ where $p, q \in \mathbb{Z}$ and $a_i \in \mathbb{R}_{\geq 0}, i = p, p+1, \ldots, q$. Define further $\mathbb{R}_{\geq 0}[X^{\pm}]^* \coloneqq \mathbb{R}_{\geq 0}[X^{\pm}] \setminus \{0\}$. One can define $\mathbb{N}[X^{\pm}]$ and $\mathbb{N}[X^{\pm}]^*$ similarly by restricting the coefficients a_i to $\mathbb{N}$.

An element $f = \sum_{i=p}^{q} a_i X^i \in \mathbb{R}_{\geq 0}[X^{\pm}]^*$ is called *gap-free* if $a_i \neq 0$ for all $i = p, p+1, \ldots, q$. Is it easy to see that, given arbitrary $M, N \in \mathbb{Z}_{>0}$ and $f \in \mathbb{R}_{\geq 0}[X^{\pm}]^*$, the polynomial $(X^{-M} + X^{-M+1} + \cdots + X^N)^n \cdot f$ is gap-free for all large enough n.

A *monomial* is a polynomial $f = a_i X^i$ with only one term (including zero). Let $d \geq 1$ be a positive integer. One can define the semirings

$$\mathbb{N}[X^{\pm d}] \coloneqq \left\{ \sum_{i=p}^{q} a_{di} X^{di} \in \mathbb{N}[X^{\pm}] \right\}, \quad \mathbb{N}[X^{\pm d}]^* \coloneqq \mathbb{N}[X^{\pm d}] \setminus \{0\}.$$

These are polynomials whose monomials have degrees divisible by d. Similarly, if $a_{di} \neq 0$ for all $i = p, \ldots, q$, we will call $\sum_{i=p}^{q} a_{di} X^{di}$ gap-free. Note that whether a polynomial is gap-free depends on the polynomial ring we consider it in.

Similarly one can define the rings $\mathbb{Z}[X^{\pm d}]$, $\mathbb{Q}[X^{\pm d}]$ and $\mathbb{R}[X^{\pm d}]$. Furthermore, we define the field of rational functions $\mathbb{Q}(X)$ to be the set of expressions of the form $\frac{f}{g}$, where $f, g \in \mathbb{Q}[X^{\pm}]$. Similarly, $\mathbb{Q}(X^d)$ is defined as the set of expressions $\frac{f}{g}$, where $f, g \in \mathbb{Q}[X^{\pm d}]$.

The wreath product $\mathbb{Z} \wr \mathbb{Z}$ has several equivalent definitions. Here, we introduce the one most convenient to our purpose.

▶ **Definition 2.3.** The wreath product $\mathbb{Z} \wr \mathbb{Z}$ is a group whose elements are pairs of the form (y, b), where $y \in \mathbb{Z}[X^{\pm}]$ and $b \in \mathbb{Z}$. The neutral element in $\mathbb{Z} \wr \mathbb{Z}$ is given by $(0, 0)$. Multiplication is defined by $(y, b) \cdot (y', b') = (y + X^b \cdot y', b + b')$, and inversion is defined by $(y, b)^{-1} = (X^{-b} \cdot y, -b)$. Note that the element (y, b) corresponds to the matrix $\begin{pmatrix} X^b & y \\ 0 & 1 \end{pmatrix}$ under the isomorphism (1) in the introduction.

The wreath product $\mathbb{Z} \wr \mathbb{Z}$ can be embedded into the larger group $\mathbb{Q}(X) \rtimes \mathbb{Z}$, whose elements are pairs of the form (y, b) with $y \in \mathbb{Q}(X)$ and $b \in \mathbb{Z}$ and whose multiplication and inversion are defined using the same formulas as in $\mathbb{Z} \wr \mathbb{Z}$.

3 Overview of proof

The main result of this paper is the decidability of the Identity Problem and the Group Problem in $\mathbb{Z} \wr \mathbb{Z}$. In view of Lemma 2.2, it suffices to prove decidability of the Group Problem. In this section we give an overview of its proof.

Our proof proceeds in three steps. As a first step we reduce the Group Problem in $\mathbb{Z} \wr \mathbb{Z}$ to deciding whether a system of linear equations in $\mathbb{R}[X^{\pm}]$ has solution in $\mathbb{R}_{\geq 0}[X^{\pm}]^*$ with two additional degree constraints (see Proposition 3.2 and Corollary 3.3). As the second step we prove a local-global principle that further reduces solving linear equations over $\mathbb{R}_{\geq 0}[X^{\pm}]^*$ to solving a family of "local" equations over $\mathbb{R}_{>0}$ (see Proposition 3.4). As the third step, we show that solving these "local" equations can be done using the first order theory of the reals as well as Gröbner basis techniques (see Proposition 3.5 and 3.6).

Let $\mathcal{G}$ be a finite subset of $\mathbb{Z} \wr \mathbb{Z}$. Write $\mathcal{G} = \{(y_a, b_a) \mid a \in A\}$ where A is a finite set of indices. Divide A into three subsets of indices $A = I \cup J \cup K$ where

$$I \coloneqq \{i \mid b_i > 0\}, \quad J \coloneqq \{j \mid b_j < 0\}, \quad K \coloneqq \{k \mid b_k = 0\}. \tag{2}$$

First, we exclude the easy case where I or J is empty.

▶ **Proposition 3.1.** *Suppose $I = \emptyset$ or $J = \emptyset$. The semigroup $\langle \mathcal{G} \rangle$ is a group if and only if $I = J = \emptyset$ and $\sum_{k \in K} n_k y_k = 0$ for some positive integers $n_k \in \mathbb{Z}_{>0}$. In particular, this is decidable by integer programming.*

A simple proof of Proposition 3.1 is given in the Appendix A. For the rest of this paper, we will suppose $I \neq \emptyset$ and $J \neq \emptyset$.

Define

$$d \coloneqq \gcd\left(\{b_a \mid a \in I \cup J\}\right).$$

For each pair $(i, j) \in I \times J$, define the rational function

$$h_{(i,j)} \coloneqq \frac{y_i}{1 + X^d + \cdots + X^{b_i - d}} + \frac{y_j}{X^{-d} + X^{-2d} + \cdots + X^{-|b_j|}} \in \mathbb{Q}(X). \tag{3}$$

By direct computation we have

$$(y_i, b_i)^{|b_j|} \cdot (y_j, b_j)^{b_i} = \left(h_{(i,j)} \cdot (1 + X^d + \cdots + X^{b_i |b_j| - d}), 0\right). \tag{4}$$

One can also take (4) as the definition of $h_{(i,j)}$.

A subset S of $I \times J$ is called *double-full* if for every $i \in I$ there exists $j_i \in J$ such that $(i, j_i) \in S$, and for every $j \in J$ there exists $i_j \in I$ such that $(i_j, j) \in S$. The following proposition reduces the Group Problem in $\mathbb{Z} \wr \mathbb{Z}$ to solving linear equations over $\mathbb{N}[X^{\pm d}]^*$.

▶ **Proposition 3.2.** *The semigroup $\langle \mathcal{G} \rangle$ is a group if and only if there exist a double-full set $S \subset I \times J$ and polynomials $f_{(i,j)}, f_k \in \mathbb{N}[X^{\pm d}]^*$ for $(i, j) \in S, k \in K$ that satisfy the following three conditions.*

(i) ***(Single linear equation)*** *The following equation over $\mathbb{Q}(X)$ is satisfied:*

$$\sum_{(i,j) \in S} f_{(i,j)} \cdot h_{(i,j)} + \sum_{k \in K} f_k \cdot y_k = 0. \tag{5}$$

(ii) ***(Positive degree bound)*** *We have:*

$$\deg_+\left(\sum\nolimits_{(i,j) \in S} f_{(i,j)}\right) + d \geq \deg_+\left(\sum\nolimits_{k \in K} f_k\right). \tag{6}$$

(iii) ***(Negative degree bound)*** *We have:*

$$\deg_- \left(\sum\nolimits_{(i,j)\in S} f_{(i,j)} \right) \leq \deg_- \left(\sum\nolimits_{k\in K} f_k \right). \tag{7}$$

The proof of Proposition 3.2 will be given in Section 4. The idea is roughly as follows. By Lemma 2.1, $\langle \mathcal{G} \rangle$ is a group if and only if there is a full-image word $w \in \mathcal{G}^*$ that represents the neutral element. For the "only if" statement of Proposition 3.2, we will represent w as a walk over $\mathbb{Z}$, and decompose the walk into "primitive circuits". Each primitive circuit contributes a multiple of $h_{(i,j)}$ or y_k to the element represented by w. Since w represents the neutral element, this results in the linear equation in Condition (i). Conditions (ii) and (iii) will stem from the connectedness of the walk. The "if" statement is significantly harder. Given the polynomials $f_{(i,j)}, f_k$, we will construct an Eulerian graph G by attaching long "elementary circuits" in a way that corresponds to the coefficients of $f_{(i,j)}, f_k$. We then read a word w from an Euler circuit of G. Condition (i) will make sure w represents the neutral element. However, making sure G is connected is highly non-trivial and will be the main difficulty of the proof. In particular, Conditions (ii)-(iii) will be crucial. This concludes the idea of the proof for Proposition 3.2.

Note that Proposition 3.2 involves finding solutions over $\mathbb{N}[X^{\pm d}]^*$ for linear equations with coefficients in $\mathbb{Q}(X)$. When $d > 1$, this is inconvenient, so we now further reduce Proposition 3.2 to finding solutions over $\mathbb{R}_{\geq 0}[X^{\pm}]^*$ for a *system* of linear equations. For each $(i,j) \in I \times J$, since the denominator of $h_{(i,j)}$ is an element in $\mathbb{Q}[X^{\pm d}]$, there exist $h_{(i,j),0}, \ldots, h_{(i,j),d-1} \in \mathbb{Q}(X)$ such that $h_{(i,j)}$ can be written as

$$h_{(i,j)} = h_{(i,j),0}(X^d) + h_{(i,j),1}(X^d) \cdot X + \cdots + h_{(i,j),d-1}(X^d) \cdot X^{d-1}. \tag{8}$$

Similarly, for each $k \in K$, there exist $y_{k,0}, \ldots, y_{k,d-1} \in \mathbb{Q}[X^{\pm}]$ so that y_k can be written as

$$y_k = y_{k,0}(X^d) + y_{k,1}(X^d) \cdot X + \cdots + y_{k,d-1}(X^d) \cdot X^{d-1}. \tag{9}$$

The following corollary shows that, using the elements $h_{(i,j),m}, y_{k,m}$ defined in (8) and (9), we can rewrite the conditions in Proposition 3.2 using only variables in $\mathbb{R}_{\geq 0}[X^{\pm}]^*$ instead of $\mathbb{N}[X^{\pm d}]^*$. See Appendix A for a simple proof of Corollary 3.3.

▶ **Corollary 3.3.** *The semigroup $\langle \mathcal{G} \rangle$ is a group if and only if there exist a double-full set $S \subset I \times J$ and polynomials $f_S, f_K, f_{(i,j)}, f_k \in \mathbb{R}_{\geq 0}[X^{\pm}]^*$ for $(i,j) \in S, k \in K$ that satisfy the following three conditions.*

(i) ***(System of linear equations)*** *The following linear equations over $\mathbb{R}(X)$ are satisfied:*

$$\sum_{(i,j)\in S} f_{(i,j)} h_{(i,j),m} + \sum_{k\in K} f_k y_{k,m} = 0, \quad m = 0, \ldots, d-1, \tag{10}$$

$$f_S = \sum_{(i,j)\in S} f_{(i,j)}, \qquad f_K = \sum_{k\in K} f_k. \tag{11}$$

(ii) ***(Positive degree bound)*** *We have:*

$$\deg_+ (f_S) + 1 \geq \deg_+ (f_K). \tag{12}$$

(iii) ***(Negative degree bound)*** *We have:*

$$\deg_- (f_S) \leq \deg_- (f_K). \tag{13}$$

For brevity, from now on we denote $\mathbb{A} \coloneqq \mathbb{R}[X^{\pm}]$ and $\mathbb{A}^+ \coloneqq \mathbb{R}_{\geq 0}[X^{\pm}]^*$. Denote also $n \coloneqq 2 + |S| + |K|$. Define the following subset of $\mathbb{A}^n$:

$$\mathcal{M} \coloneqq \left\{ \boldsymbol{f} = \left(f_S, f_K, (f_{(i,j)})_{(i,j)\in S}, (f_k)_{k\in K}\right) \in \mathbb{A}^n \;\middle|\; \boldsymbol{f} \text{ satisfies (10) and (11)} \right\}. \tag{14}$$

That is, $\mathcal{M}$ is the set of solutions of the linear equations (10)-(11). Using linear algebra over the polynomial ring $\mathbb{A}$ (see [1]), one can effectively compute a set of vectors $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m \in \mathbb{A}^n$ such that

$$\mathcal{M} = \{\phi_1 \boldsymbol{g}_1 + \cdots + \phi_m \boldsymbol{g}_m \mid \phi_1, \ldots, \phi_m \in \mathbb{A}\}. \tag{15}$$

One can even suppose $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m \in (\mathbb{Q}[X^{\pm}])^n$ since $h_{(i,j)}$ and y_k all have rational coefficients. A set $\mathcal{M}$ of the form (15) will be called a $\mathbb{A}$-*submodule* of $\mathbb{A}^n$, and the elements $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m$ will be called a *basis* of $\mathcal{M}$.

Corollary 3.3 actually states the following: the semigroup $\langle \mathcal{G} \rangle$ is a group if and only if $\mathcal{M}$ contains an element $\boldsymbol{f} = (f_S, f_K, \ldots) \in (\mathbb{A}^+)^n$ such that $\deg_+(f_S) + 1 \geq \deg_+(f_K)$ and $\deg_-(f_S) \leq \deg_-(f_K)$. The key to deciding the existence of $\boldsymbol{f}$ is the following proposition, which can be considered as a local-global principle that generalizes a result of Einsiedler, Mouat and Tuncel [13].

▶ **Proposition 3.4.** *Let $\mathcal{M}$ be a $\mathbb{A}$-submodule of $\mathbb{A}^n$. Then $\mathcal{M}$ contains an element $\boldsymbol{f} = (f_S, f_K, \ldots) \in (\mathbb{A}^+)^n$ with $\deg_+(f_S) + 1 \geq \deg_+(f_K)$ and $\deg_-(f_S) \leq \deg_-(f_K)$, if and only if the following three conditions are all satisfied.*

(i) ***(Existence of $\boldsymbol{f}_r$ for all $r \in \mathbb{R}_{>0}$)*** *For each $r \in \mathbb{R}_{>0}$, there exists $\boldsymbol{f}_r \in \mathcal{M}$ such that*

$$\boldsymbol{f}_r(r) \in \mathbb{R}^n_{>0}. \tag{16}$$

(ii) ***(Existence of $\boldsymbol{f}_\infty$)*** *There exists $\boldsymbol{f}_\infty = (f_{\infty,S}, f_{\infty,K}, \ldots) \in \mathcal{M}$ such that*

$$\mathrm{lc}_+(\boldsymbol{f}_\infty) \in \mathbb{R}^n_{>0} \quad \text{and} \quad \deg_+(f_{\infty,S}) + 1 \geq \deg_+(f_{\infty,K}). \tag{17}$$

(iii) ***(Existence of $\boldsymbol{f}_0$)*** *There exists $\boldsymbol{f}_0 = (f_{0,S}, f_{0,K}, \ldots) \in \mathcal{M}$ such that*

$$\mathrm{lc}_-(\boldsymbol{f}_0) \in \mathbb{R}^n_{>0} \quad \text{and} \quad \deg_-(f_{0,S}) \leq \deg_-(f_{0,K}). \tag{18}$$

The proof of Proposition 3.4 will be given in Section 5. The original result of Einsiedler et al. [13, Theorem 1.3] gives a similar local-global principle without the degree constraints on f_S and f_K. While our proof follows the main steps of the original proof, we need to introduce new arguments in order for the degree constraint to stay compatible with the local-global principle.

One direction of the implication in Proposition 3.4 is clear. In fact, if $f \in \mathbb{A}^+$ and $r \in \mathbb{R}_{>0}$, then we have $f(r) \in \mathbb{R}_{>0}$ and $\mathrm{lc}_+(f), \mathrm{lc}_-(f) \in \mathbb{R}_{>0}$. Therefore, if $\mathcal{M}$ contains an element $\boldsymbol{f} = (f_S, f_K, \cdots) \in (\mathbb{A}^+)^n$ with $\deg_+(f_S) + 1 \geq \deg_+(f_K)$ and $\deg_-(f_S) \leq \deg_-(f_K)$, then simply take $\boldsymbol{f}_r = \boldsymbol{f}_\infty = \boldsymbol{f}_0 = \boldsymbol{f}$ for all r: Equation (16) is satisfied for all r as well as (17) and (18); hence all three conditions are satisfied.

On the other hand, if the Equation (16) as well as (17) and (18) can be satisfied *individually* by different $\boldsymbol{f}_r, \boldsymbol{f}_\infty, \boldsymbol{f}_0$, we cannot *a priori* find an element $\boldsymbol{f} \in \mathcal{M}$ in $(\mathbb{A}^+)^n$. Such an element $\boldsymbol{f}$ would *simultaneously* satisfy Equations (16) for all $r \in \mathbb{R}_{>0}$ as well as (17) and (18). The key idea of proving this non-trivial direction is that if all three conditions (i)-(iii) are satisfied, then we can "glue" these different $\boldsymbol{f}_r, \boldsymbol{f}_\infty$ and $\boldsymbol{f}_0$ together to obtain a single $\boldsymbol{f}$ that satisfies Equation (16) for *all* r as well as (17) and (18). While this idea comes from the original

proof, the difficult part in our generalization is to make sure the degree constraints are still satisfied after the gluing procedure. In the end, we multiply this $\boldsymbol{f}$ by a "large enough" polynomial to obtain an element in $(\mathbb{A}^+)^n$, using a theorem of Handelman (Theorem 5.3).

The following two propositions show that Conditions (i), (ii) and (iii) of Proposition 3.4 are all decidable.

▶ **Proposition 3.5.** *Let $\mathcal{M}$ be an $\mathbb{A}$-submodule of $\mathbb{A}^n$. Given as input a finite basis of $\mathcal{M}$, it is decidable whether for every $r \in \mathbb{R}_{>0}$ there exists $\boldsymbol{f}_r \in \mathcal{M}$ with $\boldsymbol{f}_r(r) \in \mathbb{R}^n_{>0}$.*

▶ **Proposition 3.6.** *Let $* \in \{+, -\}$, $a \in \mathbb{Z}$ and $\mathcal{M}$ be an $\mathbb{A}$-submodule of $\mathbb{A}^n$. Given as input a finite basis of $\mathcal{M}$, it is decidable whether there exists $\boldsymbol{f} = (f_S, f_K, \cdots) \in \mathcal{M}$ such that*

$$\mathrm{lc}_*(\boldsymbol{f}) \in \mathbb{R}^n_{>0} \quad \text{and} \quad \deg_*(f_S) + a \geq \deg_*(f_K)\,. \tag{19}$$

Proposition 3.5 and 3.6 will be proven in Section 6. The idea of proving Proposition 3.5 is to reduce the statement to the first order theory of the reals; while for proving Proposition 3.6 we will use the *super Gröbner basis* introduced in the original proof of Einsiedler et al. [13].

We are now ready to prove our main theorem by bridging the remaining gaps.

▶ **Theorem 3.7.** *The Identity Problem and the Group Problem in $\mathbb{Z} \wr \mathbb{Z}$ are decidable.*

Proof. First we show decidability of the Group Problem. Given a finite set $\mathcal{G} = \{(y_a, b_a) \mid a \in A\}$ in $\mathbb{Z} \wr \mathbb{Z}$, define the index sets I, J, K as in (2). If I or J is empty, then Proposition 3.1 shows that the Group Problem for $\mathcal{G}$ is decidable. If I and J are not empty, we enumerate all double-full sets $S \subset I \times J$. For each S we compute a finite basis of $\mathcal{M}$ defined in (14). Corollary 3.3 together with Proposition 3.4 shows that the Group Problem for $\mathcal{G}$ has a positive answer if and only if for some S, the three conditions in Proposition 3.4 are all satisfied. For each S, Condition (i) can be decided using Proposition 3.5; Condition (ii) can be decided using Proposition 3.6 by taking $* = +, a = 1$; Condition (iii) can be decided using Proposition 3.6 by swapping the coordinates S and K and taking $* = -, a = 0$. Therefore the Group Problem in $\mathbb{Z} \wr \mathbb{Z}$ is decidable. By Lemma 2.2, the Identity Problem in $\mathbb{Z} \wr \mathbb{Z}$ is also decidable. ◀

4 From semigroup to polynomial equations

4.1 Definition of $\mathcal{G}$-graphs

Section 4 is dedicated to the proof of Proposition 3.2. In this subsection, we will define the notion of a *$\mathcal{G}$-graph*. Let A be a finite set of indices. Let $\mathcal{G} = \{(y_a, b_a) \mid a \in A\}$ be a finite set of elements in the group $\mathbb{Z} \wr \mathbb{Z}$ or $\mathbb{Q}(X) \rtimes \mathbb{Z}$. We define the following notion of a $\mathcal{G}$-graph.

▶ **Definition 4.1** ($\mathcal{G}$-graphs)**.** A *$\mathcal{G}$-graph* is a directed multigraph G, whose set of vertices $V(G)$ is a finite subset of $\mathbb{Z}$, and its edges are each labeled with an index in A. Furthermore, if an edge from vertex d_1 to vertex d_2 has label a, then $d_2 = d_1 + b_a$.

For a word w over the alphabet $\mathcal{G}$, we associate to it a unique $\mathcal{G}$-graph $G(w)$, defined as follows. Write $w = (y_{a_1}, b_{a_1}) \cdots (y_{a_p}, b_{a_p})$. For each $i = 0, \ldots, p-1$, we add an edge starting at the vertex $b_{a_1} + \cdots + b_{a_i}$, ending at the vertex $b_{a_1} + \cdots + b_{a_{i+1}}$, with the label a_i. (If $i = 0$ then the edge starts at 0 and ends at b_{a_1}.) The graph $G(w)$ is then obtained by taking the connected component of the vertex 0. See Figure 1 for the illustration of an example.

By reading the letters in w one by one and tracing the corresponding edges of $G(w)$, we obtain an Euler path of $G(w)$. Furthermore, if the word w represents the neutral element (or any element of the form $(y, 0)$), then this Euler path is an Euler circuit.

We point out that the element which w represents is uniquely determined by $G(w)$:

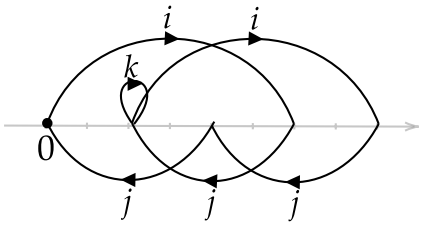


Figure 1 Illustration of $G(w)$. Here, $A = \{i, j, k\}$, $\mathcal{G} = \{(y_i, 6), (y_j, -4), (y_k, 0)\}$, $w = (y_i, 6)(y_j, -4)(y_k, 0)(y_i, 6)(y_j, -4)(y_j, -4)$.

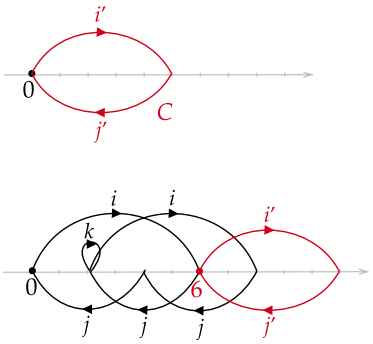


Figure 2 Attaching the circuit C to $G(w)$ at vertex 6.

▶ **Fact 4.2** (Product of associated graph). Let w be a word over the alphabet $\mathcal{G}$, and let $G = G(w)$ be its associated $\mathcal{G}$-graph. For an edge $e \in E(G)$, denote by $\ell(e)$ the label of e, denote by $s(e) \in \mathbb{Z}$ the starting vertex of e, then w represents the element

$$\left(\sum_{e \in E(G)} X^{s(e)} \cdot y_{\ell(e)}, \sum_{e \in E(G)} b_{\ell(e)} \right). \tag{20}$$

For an arbitrary $\mathcal{G}$-graph G, the element in Expression (20) will be called the *product* of the graph G. It is easy to see that, if G contains an Eulerian path, then by following this path we obtain a word w that represents the product of G.

Let C be a another Eulerian $\mathcal{G}$-graph (seen as a circuit). We define the following action of *attaching C to G at vertex v*: For each edge e of C, starting at vertex $s(e)$ with label $\ell(e)$, we add an edge e' to G, starting at vertex $s(e) + v$ with label $\ell(e)$. See Figure 2 for the illustration of an example. The product of the resulting graph is uniquely determined by G, C as well as v:

▶ **Fact 4.3** (Effect of attaching circuit to a graph). Let G be an arbitrary $\mathcal{G}$-graph and C be an Eulerian $\mathcal{G}$-graph. Denote by (y_G, b_G) the product of G and by $(y_C, 0)$ the product of C. Then attaching C to G at vertex v results in a graph with product $(y_G + X^v \cdot y_C, b_G)$.

4.2 Group Problem implies polynomial equations

In this subsection, we prove the "only if" part of Proposition 3.2. We will show that if $\langle \mathcal{G} \rangle$ is a group then there exists a double-full set $S \subset I \times J$ and polynomials $f_{(i,j)}, f_k \in \mathbb{N}[X^{\pm d}]^*$ for $(i, j) \in S, k \in K$ satisfying (i)-(iii) of Proposition 3.2.

Recall that $d \coloneqq \gcd(\{b_a \mid (y_a, b_a) \in \mathcal{G}\})$. Define the alphabet $\widehat{\mathcal{G}}$ of *radical* elements:

$$\widehat{\mathcal{G}} \coloneqq \left\{ (\widehat{y}_a, \widehat{b}_a) \;\middle|\; (y_a, b_a) \in \mathcal{G} \right\},$$

where

$$(\widehat{y}_a, \widehat{b}_a) \coloneqq \begin{cases} \left(\frac{y_a}{1 + X^d + \cdots + X^{b_a - d}}, d \right) & a \in I \quad \text{(or equivalently, } b_a > 0\text{)}, \\ \left(\frac{y_a}{1 + X^{-d} + \cdots + X^{-(|b_a| - d)}}, -d \right) & a \in J \quad \text{(or equivalently, } b_a < 0\text{)}, \\ (y_a, 0) & a \in K \quad \text{(or equivalently, } b_a = 0\text{)}. \end{cases}$$

Note that these elements are in $\mathbb{Q}(X) \rtimes \mathbb{Z}$ instead of $\mathbb{Z} \wr \mathbb{Z}$. Direct computation shows that

$$(\widehat{y}_a, \widehat{b}_a)^{|b_a|/d} = (y_a, b_a) \tag{21}$$

for $a \in I \cup J$. Equation (21) can also be taken as the definition of $(\widehat{y}_a, \widehat{b}_a)$.

Since $\langle \mathcal{G} \rangle$ is a group, by Lemma 2.1 there exists a full-image word $w \in \mathcal{G}^*$ that represents the neutral element. Replacing the letters (y_a, b_a) in w by the words $\underbrace{(\widehat{y}_a, \widehat{b}_a) \cdots (\widehat{y}_a, \widehat{b}_a)}_{|b_a|/d \text{ times}}$ for every $a \in I \cup J$, we obtain a word $\widehat{w} \in \widehat{\mathcal{G}}^*$. By Equation (21), $\widehat{w}$ also represents the neutral element. To the word $\widehat{w}$ we associate a $\widehat{\mathcal{G}}$-graph $G(\widehat{w})$. See Figure 3 for the illustration of an example of $G(\widehat{w})$; one can compare it with Figure 1 which illustrates $G(w)$ for the same w.

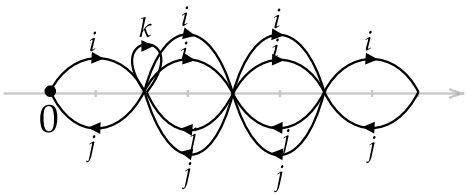


Figure 3 Decomposition of $G(\widehat{w})$. Here $\mathcal{G}$ and w are the same as in Figure 1.

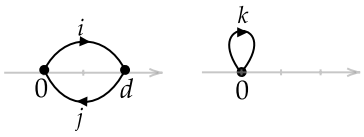


Figure 4 Primitive circuits of type (i, j) and k.

Since $\widehat{w}$ represents the neutral element, the graph $G(\widehat{w})$ is Eulerian. Since $\widehat{b}_a = \pm d$ or 0 for all indices $a \in I \cup J \cup K$, the graph $G(\widehat{w})$ can be decomposed into two classes of smaller circuits. The first class of circuits is an edge with some label $i \in I$ (edge directed to the right) followed by an edge with some label $j \in J$ (edge directed to the left); we call such a circuit *of the type* (i, j). The second class is a loop with label $k \in K$; we call such a circuit *of the type* k. We call these two classes of circuits *primitive*. See Figure 4 for an illustration.

▶ **Lemma 4.4.** *The graph $G(\widehat{w})$ can be constructed by starting with an edgeless graph with a single vertex 0 and gradually attaching primitive circuits.*

Recall the definition of $h_{(i,j)}$ in Equation (3). In fact, $h_{(i,j)}$ is the product of a primitive circuit of type (i, j), meaning $(\widehat{y}_i, \widehat{b}_i) \cdot (\widehat{y}_j, \widehat{b}_j) = (h_{(i,j)}, 0)$. Similarly, the product of a circuit of type k is $(h_k, 0) \coloneqq (y_k, 0)$. By Lemma 4.4, $G(\widehat{w})$ can be decomposed into primitive circuits. For each primitive circuit C in the decomposition, denote by $s(C) \in d\mathbb{Z}$ the vertex where C is attached, and by $type(C)$ the type of C. Denote by $\mathcal{C}$ the set of circuits in the decomposition of $G(\widehat{w})$ into primitive circuits. By Fact 4.3, the product of $G(\widehat{w})$ can be written as

$$\left(\sum_{C \in \mathcal{C}} X^{s(C)} \cdot h_{type(C)}, 0 \right) = (0, 0). \tag{22}$$

For each $(i, j) \in I \times J$ and $k \in K$, define the following polynomials in $\mathbb{N}[X^{\pm d}]$:

$$f_{(i,j)} \coloneqq \sum_{C \in \mathcal{C} \text{ of type } (i,j)} \left(X^d\right)^{\frac{s(C)}{d}}, \quad f_k \coloneqq \sum_{C \in \mathcal{C} \text{ of type } k} \left(X^d\right)^{\frac{s(C)}{d}}. \tag{23}$$

Let $S \coloneqq \{(i, j) \in I \times J \mid f_{(i,j)} \neq 0\}$. We point out that $f_k \neq 0$ for all $k \in K$, because $\widehat{w}$ is full-image, meaning $G(\widehat{w})$ contains a loop of label k for each $k \in K$. Equation (22) becomes

$$\sum_{(i,j) \in S} f_{(i,j)} \cdot h_{(i,j)} + \sum_{k \in K} f_k \cdot y_k = 0. \tag{24}$$

This is exactly Condition (i) of Proposition 3.2. It suffices to show the following to complete the proof of the first implication of Proposition 3.2.

▶ **Lemma 4.5.** *Let $f_{(i,j)}, f_k \in \mathbb{N}[X^{\pm d}]$ and $S \subset I \times J$ be defined as above, then:*

(i) *S is double-full.*

(ii) $\deg_+\left(\sum_{(i,j)\in S} f_{(i,j)}\right) + d \geq \deg_+\left(\sum_{k\in K} f_k\right)$.

(iii) $\deg_-\left(\sum_{(i,j)\in S} f_{(i,j)}\right) \leq \deg_-\left(\sum_{k\in K} f_k\right)$.

Sketch of proof. For (i), S is double full since $G(\widehat{w})$ contains edges of each type of labels. For (ii) and (iii), it suffices to notice that $G(\widehat{w})$ must be connected while $\max(V(G(\widehat{w}))) = \deg_+\left(\sum_{(i,j)\in S} f_{(i,j)}\right) + d$ and $\min(V(G(\widehat{w}))) = \deg_-\left(\sum_{(i,j)\in S} f_{(i,j)}\right)$. ◀

Proof of "only if" part of Proposition 3.2. If $\langle \mathcal{G} \rangle$ is a group, then there exists a full-image word $w \in \mathcal{G}^*$ that represents the neutral element. Consider $G(\widehat{w})$ and let $f_{(i,j)}, f_k \in \mathbb{N}[X^{\pm d}]^*, (i,j) \in S, k \in K$, be as defined in (23). The Conditions (i)-(iii) of Proposition 3.2 follow directly from Equation (24) and Lemma 4.5. ◀

4.3 Polynomial equation implies Group Problem

In this subsection, we prove the "if" part of Proposition 3.2. Given a double-full set $S \subset I \times J$ and positive polynomials $f_{(i,j)}, f_k \in \mathbb{N}[X^{\pm d}]^*$ for $(i,j) \in S, k \in K$ that satisfy Conditions (i)-(iii) of Proposition 3.2, we will construct an Eulerian $\mathcal{G}$-graph G with product zero. The main difficulty here is that the length of the edges of a $\mathcal{G}$-graph are no longer identical, as opposed to $\widehat{\mathcal{G}}$-graphs. Therefore one can no longer decompose $\mathcal{G}$-graphs into primitive circuits. The key idea is a work-around that simulates primitive circuits using longer circuits.

For $(i,j) \in I \times J$, we define an *elementary circuit of the type* (i,j) to be a circuit that starts with $|b_j|$ edges of label i, followed by b_i edges of label j. See Figure 5 for an example.

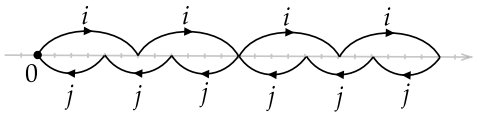


Figure 5 An elementary circuit of type (i,j). Here, $b_i = 6, b_j = -4$.

▶ **Lemma 4.6.** *Suppose $S \subset I \times J$ be double-full. There exists an Eulerian $\mathcal{G}$-graph A with $d \in V(A)$, obtained by attaching together elementary circuits of types in S.*

The idea of constructing A is illustrated in Figure 6, with a detailed proof given in Appendix A. We now characterize the product of A. An elementary circuit of type (i,j) attached at vertex dv contributes $X^{dv}(1 + X^d + \cdots + X^{b_i|b_j|-d}) \cdot h_{(i,j)}$ to the product (see Equation (4)). Since A is a combination of elementary circuits, the product of A can written as $\sum_{(i,j)\in S} a_{(i,j)} h_{(i,j)}$ for some $a_{(i,j)} \in \mathbb{N}[X^{\pm d}]$.

Let $N \coloneqq \prod_{i\in I\cup J} |b_i|$. Note that simultaneously multiplying all $f_{(i,j)}$ and f_k by any polynomial $g \in \mathbb{N}[X^{\pm d}]^*$ does not change the fact that $f_{(i,j)}, f_k \in \mathbb{N}[X^{\pm d}]^*$, and they still satisfy Conditions (i)-(iii) of Proposition 3.2. Also note that $1 + X^d + \cdots + X^{b_i|b_j|-d} \mid 1 + X^d + \cdots + X^{N-d}$. Therefore, by simultaneously multiplying all $f_{(i,j)}$ and f_k by $p \cdot (X^{-d} + 1 + \cdots + X^{N-2d})^q$ for large enough $p, q \in \mathbb{N}$, we can suppose that for all $(i,j) \in S$,

$$g_{(i,j)} \coloneqq \frac{f_{(i,j)} - a_{(i,j)} \cdot (1 + X^d + \cdots + X^{N-d})}{1 + X^d + \cdots + X^{b_i|b_j|-d}} \in \mathbb{N}[X^{\pm d}]^* \quad \text{is gap-free,} \tag{25}$$

and

$$\deg_+\left(f_{(i,j)}\right) > \deg_+(a_{(i,j)}) + N - d, \qquad \deg_-\left(f_{(i,j)}\right) < \deg_-(a_{(i,j)}). \tag{26}$$

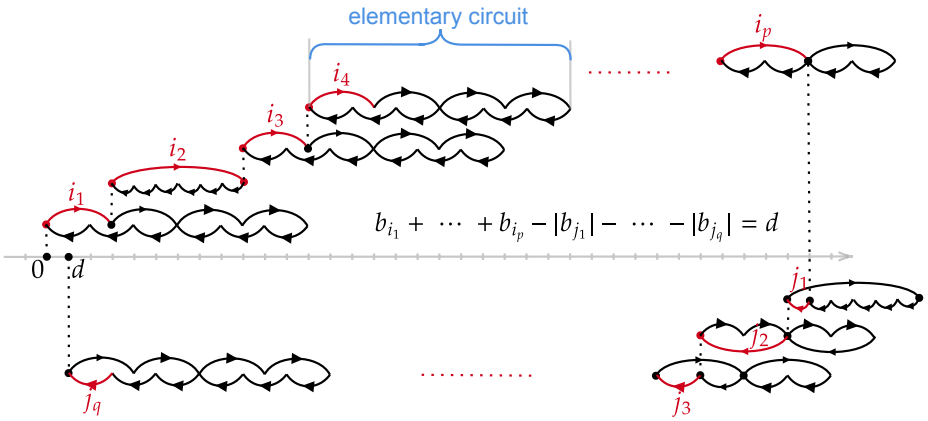


Figure 6 Graph A from Lemma 4.6.

▶ **Proposition 4.7.** *Suppose S is double-full and $f_{(i,j)}, f_k \in \mathbb{N}[X^{\pm d}]^*, (i,j) \in S, k \in K$, satisfy Equations (25), (26) and Conditions (ii)-(iii) of Proposition 3.2, then there exists an Eulerian $\mathcal{G}$-graph G whose product is $\sum_{(i,j)\in S} f_{(i,j)} \cdot h_{(i,j)} + \sum_{k\in K} f_k \cdot y_k$.*

Proof. We construct G in three steps. See Figure 7 for an illustration.

Step 1: Constructing the foundation A'. We start with the $\mathcal{G}$-graph A constructed in Lemma 4.6. We then attach to it another $N/d - 1$ copies of A, where the k-th copy is attached at vertex dk. The resulting graph is still Eulerian because $d \in V(A)$. We denote by A' the $\mathcal{G}$-graph obtained by this attachment. Then we have $0, d, \cdots, N - d \in V(A')$.

Step 2: Attaching elementary circuits of type $(i,j) \in S$. For each pair $(i,j) \in S$, we want to attach elementary circuits of type (i,j) to the graph A', such that the total contribution of these circuits to the product is $g_{(i,j)} \cdot \left(1 + X^d + \cdots + X^{b_i|b_j|-d}\right) \cdot h_{(i,j)}$, where $g_{(i,j)}$ is defined in (25). Write $g_{(i,j)} = \sum_{t=p}^{q} \gamma_t X^{dt}$. We attach a total of $\sum_{t=p}^{q} \gamma_t$ elementary circuits of type (i,j) to A', where for $t = p, p+1, \ldots, q$, exactly γ_t of these circuits are attached at the vertex dt. The resulting graph is connected (and Eulerian) because $g_{(i,j)}$ is gap-free. In fact, for each $t \in [\deg_-(g_{(i,j)})/d, \left(\deg_+(g_{(i,j)}) + b_i|b_j|\right)/d] \cap \mathbb{Z}$ and $u \in [0, N)$, such that $dt \equiv du \mod b_i|b_j|$, the vertex dt is connected to $du \in V(A')$ by a chain of circuits of type (i,j). Denote by A'' the resulting graph after doing the above attachments for all $(i,j) \in S$. Then

$$dt \in V(A'') \text{ for all } t \in [\deg_-(g_{(i,j)})/d, \left(\deg_+(g_{(i,j)}) + b_i|b_j|\right)/d] \cap \mathbb{Z},\ (i,j) \in S. \tag{27}$$

Step 3: Attaching loops of type $k \in K$. For each $k \in K$, we want to attach loops of label $k \in K$ to A'', such that the total contribution of these loops to the product is $f_k \cdot y_k$. Write $f_k = \sum_{t=p}^{q} \beta_t X^{dt}$, we attach a total of $\sum_{t=p}^{q} \beta_t$ loops of label k to A'', where for $t = p, p+1, \ldots, q$, exactly β_t of these loops are attached at the vertex dt.

We need to prove that the resulting graph G is still connected (and hence Eulerian). In view of Property (27) of the graph A'', it suffices to prove $\deg_-(f_k) \geq \min_{(i,j)\in S}\left\{\deg_-(g_{(i,j)})\right\}$ and $\deg_+(f_k) \leq \max_{(i,j)\in S}\left\{\deg_+(g_{(i,j)}) + b_i|b_j|\right\}$ for all $k \in K$. By Equations (25) and (26), we have $\deg_-(g_{(i,j)}) = \deg_-(f_{(i,j)})$ and $\deg_+(g_{(i,j)}) + b_i|b_j| - d = \deg_+(f_{(i,j)})$ for all $(i,j) \in S$. Then, by Conditions (ii) and (iii) of Proposition 3.2, we have

$$\deg_-(f_k) \geq \deg_-\left(\sum\nolimits_{k\in K} f_k\right) \geq \min_{(i,j)\in S}\{\deg_-(f_{(i,j)})\} = \min_{(i,j)\in S}\{\deg_-(g_{(i,j)})\},$$
$$\deg_+(f_k) \leq \deg_+\left(\sum\nolimits_{k\in K} f_k\right) \leq \max_{(i,j)\in S}\{\deg_+(f_{(i,j)})\} + d = \max_{(i,j)\in S}\{\deg_-(g_{(i,j)}) + b_i|b_j|\},$$

for all $k \in K$. Therefore, the resulting graph G is still connected.

Finally, we count the product of G. The product of A' is $(1 + X^d + \cdots + X^{N-d}) \cdot \sum_{(i,j)\in S} a_{(i,j)}h_{(i,j)}$. The total contribution of elementary circuits in Step 2 is $\sum_{(i,j)\in S} g_{(i,j)} \cdot \left(1 + X^d + \cdots + X^{b_i|b_j|-d}\right) \cdot h_{(i,j)}$. The total contribution of loops in Step 3 is $\sum_{k\in K} f_k \cdot y_k$. Thus, by Equation (25), the product of G is $\sum_{(i,j)\in S} f_{(i,j)} \cdot h_{(i,j)} + \sum_{k\in K} f_k \cdot y_k$. ◀

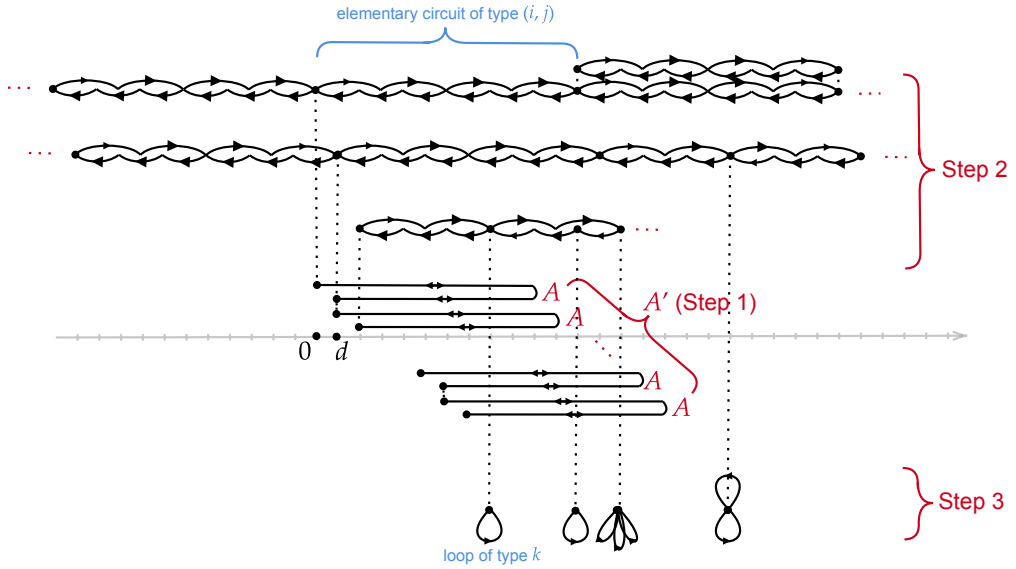


Figure 7 Graph G from Proposition 4.7.

Proof of "if" part of Proposition 3.2. We use Proposition 4.7 to prove the "if" part of Proposition 3.2. Suppose there exist a double-full set $S \subset I \times J$ and polynomials $f_{(i,j)}, f_k \in \mathbb{N}[X^{\pm d}]^*$ for $(i,j) \in S, k \in K$ that satisfy Conditions (i)-(iii). Recall that by simultaneously multiplying all $f_{(i,j)}$ and f_k by $p \cdot (X^{-d} + 1 + X^d + \cdots + X^{N-2d})^q$ for large enough $p, q \in \mathbb{N}$, we can suppose Equations (25) and (26) to be satisfied. Then Proposition 4.7 gives an Eulerian $\mathcal{G}$-graph G whose product is $\left(\sum_{(i,j)\in S} f_{(i,j)} \cdot h_{(i,j)} + \sum_{k\in K} f_k \cdot y_k, 0\right)$. This product is equal to the neutral element due to Conditions (i) of Proposition 3.2. By following an Eulerian cycle of G, we obtain a word w representing the neutral element. The word w is full-image because G contains elementary circuits of all types $(i,j) \in S$ and loops of all types $k \in K$, and because S is double-full. Therefore $\langle \mathcal{G} \rangle$ is a group by Lemma 2.1. ◀

5 A local-global principle for polynomial equations

In this section we prove Proposition 3.4. We follow the line of proof for the original result of Einsiedler et al. [13], while introducing new elements concerning the degree constraints. See Figure 8 in Appendix A for an illustration of the proof.

▶ **Lemma 5.1.** *Suppose Conditions (ii) and (iii) of Proposition 3.4 hold. Then there exists* $\boldsymbol{f}_{end} = (f_{end,S}, f_{end,K}, \cdots) \in \mathcal{M}$ *such that*

$$\mathrm{lc}_+(\boldsymbol{f}_{end}) \in \mathbb{R}^n_{>0}, \quad \deg_+(f_{end,S}) + 1 \geq \deg_+(f_{end,K}), \text{ and} \tag{28}$$

$$\mathrm{lc}_-(\boldsymbol{f}_{end}) \in \mathbb{R}^n_{>0}, \quad \deg_-(f_{end,S}) \leq \deg_-(f_{end,K}). \tag{29}$$

Since $\mathrm{lc}_\pm(\boldsymbol{f}_{end}) \in \mathbb{R}^n_{>0}$, there exists $c > 1$, such that $\boldsymbol{f}_{end}(x) \in \mathbb{R}^n_{>0}$ for all $x \in \mathbb{R}_{>0} \backslash [1/c, c]$. Define the compact set $C := [1/4c, 4c]$.

▶ **Lemma 5.2.** *Suppose Condition (i) of Proposition 3.4 hold. Let* $C \subset \mathbb{R}_{>0}$ *be a compact set, then there exists* $\boldsymbol{f}_C = (f_{C,S}, f_{C,K}, \cdots) \in \mathcal{M}$ *such that* $\boldsymbol{f}_C(x) \in \mathbb{R}^n_{>0}$ *for all* $x \in C$.

The key ingredient for finding an element in $(\mathbb{A}^+)^n$ is the following corollary of Handelman's Theorem.

▶ **Theorem 5.3** (Corollary of Handelman's Theorem [10, 15])**.** *Let* $\boldsymbol{f} \in \mathbb{A}^n$. *There exists* $g \in \mathbb{A}^+$ *such that* $g \cdot \boldsymbol{f} \in (\mathbb{A}^+)^n$ *if and only if the two following conditions are satisfied:*

(i) *For all* $r \in \mathbb{R}_{>0}$, *we have* $\boldsymbol{f}(r) \in \mathbb{R}^n_{>0}$.

(ii) *We have* $\mathrm{lc}_+(\boldsymbol{f}) \in \mathbb{R}^n_{>0}$ *and* $\mathrm{lc}_-(\boldsymbol{f}) \in \mathbb{R}^n_{>0}$.

We now sketch a proof of Proposition 3.4 by "gluing" the two elements $\boldsymbol{f}_{end}$ and $\boldsymbol{f}_C$ obtained respectively in Lemma 5.1 and 5.2, then applying Theorem 5.3. For an illustration of the proof, see Figure 8.

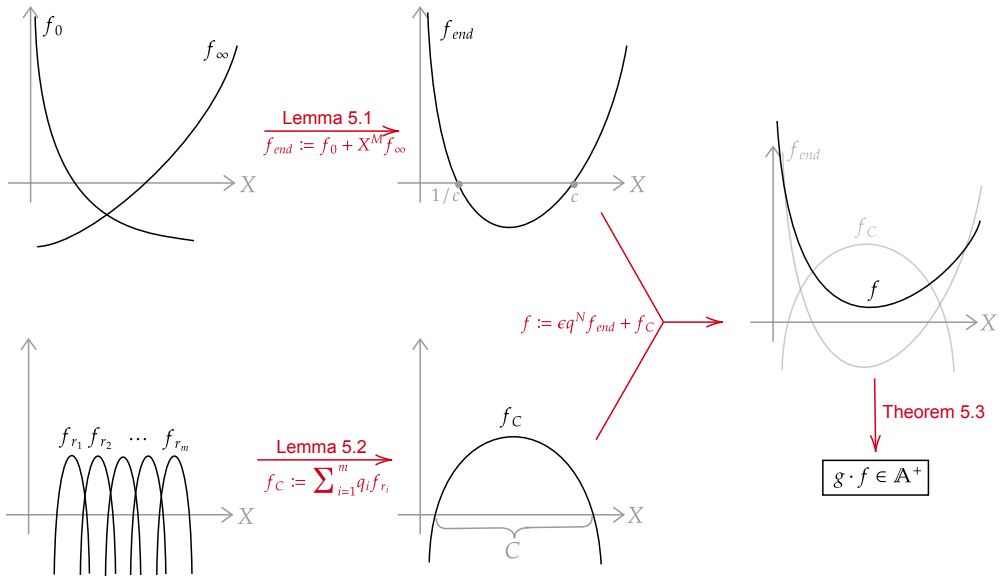


Figure 8 Proof of Proposition 3.4, illustrated at dimension $n = 1$.

Sketch of proof of Proposition 3.4. If $\mathcal{M}$ contains an element $\boldsymbol{f} = (f_S, f_K, \cdots) \in (\mathbb{A}^+)^n$ with $\deg_+(f_S) + 1 \geq \deg_+(f_K)$ and $\deg_-(f_S) \leq \deg_-(f_K)$, then simply take $\boldsymbol{f}_r = \boldsymbol{f}_\infty = \boldsymbol{f}_0 = \boldsymbol{f}$ for all $r \in \mathbb{R}_{>0}$: Equation (16) is satisfied for all r as well as (17) and (18).

Consider the non-trivial direction of implication. Let $\boldsymbol{f}_{end}, \boldsymbol{f}_C \in \mathcal{M}$ be the elements obtained respectively in Lemma 5.1 and 5.2. Define the polynomial $q := \frac{1}{2c}(X + X^{-1})$. Let $\epsilon > 0$ be such that $\epsilon \cdot \boldsymbol{f}_{end}(x) + \boldsymbol{f}_C(x) \in \mathbb{R}^n_{>0}$ for all $x \in C$. Such an ϵ exists by the compactness of C. We claim that there exists $N \in \mathbb{N}$ such that $\boldsymbol{f} := \epsilon q^N \cdot \boldsymbol{f}_{end} + \boldsymbol{f}_C$ satisfies Conditions (i) and (ii) in Theorem 5.3 simultaneously.

Let $M \in \mathbb{N}$ be such that $\deg_+(f_{end,i}) + M > \deg_+(f_{C,i})$ and $\deg_-(f_{end,i}) - M < \deg_-(f_{C,i})$ for every coordinate $i = S, K, \cdots, n$. Let $\boldsymbol{g} \coloneqq \epsilon q^M \cdot \boldsymbol{f}_{end} + \boldsymbol{f}_C$. Then we have $\mathrm{lc}_+(\boldsymbol{g}) = \mathrm{lc}_+(\boldsymbol{f}_{end}) \in \mathbb{R}^n_{>0}$ and $\mathrm{lc}_-(\boldsymbol{g}) = \mathrm{lc}_-(\boldsymbol{f}_{end}) \in \mathbb{R}^n_{>0}$, as well as

$$\deg_+(g_S) + 1 = \deg_+(f_{end,S}) + M + 1 \geq \deg_+(f_{end,K}) + M = \deg_+(g_K),$$
$$\deg_-(g_S) = \deg_-(f_{end,S}) - M \leq \deg_-(f_{end,K}) - M = \deg_-(g_K).$$

Therefore, there exists a compact set $[1/d, d] \supset C$ such that $\boldsymbol{g}(x) \in \mathbb{R}^n_{>0}$ for all $x \in \mathbb{R}_{>0} \setminus [1/d, d]$. Since $[1/d, d]$ is compact, there exists $N > M$ such that $\epsilon f_{end,i}(x) \cdot 2^N + f_{C,i}(x) > 0$ for all $i = S, K, \cdots, n$, and all $x \in [1/d, d]$.

For this N, the vector $\boldsymbol{f} \coloneqq \epsilon q^N \cdot \boldsymbol{f}_{end} + \boldsymbol{f}_C$ satisfies both Conditions (i) and (ii) in Theorem 5.3 (see the full proof in Appendix A). Therefore, we can find $g \in \mathbb{A}^+$ such that $g\boldsymbol{f} \in (\mathbb{A}^+)^n$. We have at the same time $g\boldsymbol{f} \in \mathcal{M}$ as well as $\deg_+(gf_S) + 1 \geq \deg_+(gf_K)$ and $\deg_-(gf_S) \leq \deg_-(gf_K)$. We have thus found the required element $g\boldsymbol{f}$. ◀

6 Decidability of local conditions

In this section we prove Proposition 3.5 and 3.6. Let $\mathcal{M}$ be an $\mathbb{A}$-submodule of $\mathbb{A}^n$.

▶ **Lemma 6.1.** *Let $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m$ be a basis of $\mathcal{M}$ and $r \in \mathbb{R}_{>0}$. There exists $\boldsymbol{f} \in \mathcal{M}$ with $\boldsymbol{f}(r) \in \mathbb{R}^n_{>0}$ if and only if there exist $r_1, \ldots, r_m \in \mathbb{R}$ such that $r_1\boldsymbol{g}_1(r) + \cdots + r_m\boldsymbol{g}_m(r) \in \mathbb{R}^n_{>0}$.*

▶ **Proposition 3.5.** *Let $\mathcal{M}$ be an $\mathbb{A}$-submodule of $\mathbb{A}^n$. Given as input a finite basis of $\mathcal{M}$, it is decidable whether for every $r \in \mathbb{R}_{>0}$ there exists $\boldsymbol{f}_r \in \mathcal{M}$ with $\boldsymbol{f}_r(r) \in \mathbb{R}^n_{>0}$.*

Proof. By Lemma 6.1, the statement to be decided is equivalent to the following sentence in the first order theory of the reals:

$$\forall r, r > 0 \implies (\exists r_1 \exists r_2 \cdots \exists r_m, r_1\boldsymbol{g}_1(r) + \cdots + r_m\boldsymbol{g}_m(r) \in \mathbb{R}^n_{>0}). \tag{30}$$

Its truth is decidable by Tarski's Theorem [28]. ◀

For Proposition 3.6, we start by the following definitions.

▶ **Definition 6.2.**

(i) Suppose $f = \sum_{i=p}^{q} a_i X^i \in \mathbb{A} \setminus \{0\}$, where $a_p a_q \neq 0$. Define $\mathrm{in}_+(f) \coloneqq a_q X^q$ and $\mathrm{in}_-(f) \coloneqq a_p X^p$. Additionally define $\mathrm{in}_+(0) = \mathrm{in}_-(0) = 0$.

(ii) Given $* \in \{+, -\}$, $\boldsymbol{\alpha} = (\alpha_S, \alpha_K, \ldots) \in \mathbb{Z}^n$ and $\boldsymbol{f} = (f_S, f_K, \ldots) \in \mathbb{A}^n$. Define $\mathrm{sgn}(*) = 1$ when $* = +$, and $\mathrm{sgn}(*) = -1$ when $* = -$. Let

$$m_{*,\boldsymbol{\alpha}}(\boldsymbol{f}) \coloneqq \max\{\mathrm{sgn}(*) \cdot \deg_*(f_S) + \alpha_S, \mathrm{sgn}(*) \cdot \deg_*(f_K) + \alpha_K, \ldots\}.$$

Define $\mathrm{in}_{*,\boldsymbol{\alpha}}(\boldsymbol{f}) \coloneqq (g_S, g_K, \ldots)$, where for $j = S, K, \ldots$,

$$g_j \coloneqq \begin{cases} \mathrm{in}_*(f_j) & \mathrm{sgn}(*) \cdot \deg_*(f_j) + \alpha_j = m_{*,\boldsymbol{\alpha}}(\boldsymbol{f}), \\ 0 & \deg_*(f_j) + \alpha_j < m_{*,\boldsymbol{\alpha}}(\boldsymbol{f}). \end{cases}$$

For a monomial $f = a_i X^i \in \mathbb{A}$, define $\mathrm{coef}(f) \coloneqq a_i$. In particular, $\mathrm{coef}(0) = 0$. Note that for any $* \in \{+, -\}$, $\boldsymbol{f} \in \mathbb{A}^n$ and $\boldsymbol{\alpha} \in \mathbb{Z}^n$, the above defined $\mathrm{in}_{*,\boldsymbol{\alpha}}(\boldsymbol{f})$ is an n-tuple of monomials. Writing $\mathrm{in}_{*,\boldsymbol{\alpha}}(\boldsymbol{f}) = (g_S, g_K, \ldots)$, we then extend the definition of coef() to

$$\mathrm{coef}(\mathrm{in}_{*,\boldsymbol{\alpha}}(\boldsymbol{f})) \coloneqq (\mathrm{coef}(g_S), \mathrm{coef}(g_K), \ldots) \in \mathbb{R}^n.$$

▶ **Lemma 6.3.** *Fix $* \in \{+,-\}$ and $a \in \mathbb{Z}$. The two following conditions are equivalent:*

(i) *There exists $\boldsymbol{f} = (f_S, f_K, \cdots) \in \mathcal{M}$ such that*

$$\mathrm{lc}_*(\boldsymbol{f}) \in \mathbb{R}^n_{>0} \quad \text{and} \quad \deg_*(f_S) + a \geq \deg_*(f_K). \tag{31}$$

(ii) *There exists $\boldsymbol{\alpha} = (\alpha_S, \alpha_K, \cdots) \in \mathbb{Z}^n$ with $\alpha_S - \alpha_K \leq a$, as well as $\boldsymbol{f} = (f_S, f_K, \cdots) \in \mathcal{M}$, such that $\mathrm{coef}\left(\mathrm{in}_{*,\boldsymbol{\alpha}}(\boldsymbol{f})\right) \in \mathbb{R}^n_{>0}$.*

We use the notion of a *super Gröbner basis* for $\mathcal{M}$: see [13, Chapter 2] for its exact definition. Readers can simply take the following Lemma 6.4 as its definition, since this will be the only property of the super Gröbner basis that we use in this paper.

▶ **Lemma 6.4** ([13, Lemma 2.1])**.** *Let $* \in \{+,-\}$ and $\boldsymbol{\alpha} \in \mathbb{Z}^n$. Let $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m$ be a super Gröbner basis for $\mathcal{M}$. For every $\boldsymbol{g} \in \mathcal{M}$, we have $\mathrm{in}_{*,\boldsymbol{\alpha}}(\boldsymbol{g}) = \sum_{i=1}^m p_i \cdot \mathrm{in}_{*,\boldsymbol{\alpha}}(\boldsymbol{g}_i)$ for some $p_1, \ldots, p_m \in \mathbb{A}$.*

By [13, Chapter 2], given a basis for $\mathcal{M}$, a set of super Gröbner basis exists and can be effectively computed. We now fix a set of a super Gröbner basis $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m$ for $\mathcal{M}$.

▶ **Corollary 6.5.** *Let $* \in \{+,-\}$ and $\boldsymbol{\alpha} \in \mathbb{Z}^n$. Then there exists $\boldsymbol{f} \in \mathcal{M}$ with $\mathrm{coef}\left(\mathrm{in}_{*,\boldsymbol{\alpha}}(\boldsymbol{f})\right) \in \mathbb{R}^n_{>0}$ if and only if there exist $r_1, \ldots, r_m \in \mathbb{R}$ with $\sum_{i=1}^m r_i \cdot \mathrm{coef}(\mathrm{in}_{*,\boldsymbol{\alpha}}(\boldsymbol{g}_i)) \in \mathbb{R}^n_{>0}$.*

▶ **Lemma 6.6** (Generalization of [13, Lemma 6.1])**.** *Fix $* \in \{+,-\}$. The initial tuples $\mathrm{in}_{*,\boldsymbol{\alpha}}(\boldsymbol{g}_1), \ldots, \mathrm{in}_{*,\boldsymbol{\alpha}}(\boldsymbol{g}_m)$ can take only a finite number of possible values when $\boldsymbol{\alpha}$ varies in the set $\{\boldsymbol{\alpha} = (\alpha_S, \alpha_K, \cdots) \in \mathbb{Z}^n \mid \alpha_S - \alpha_K \leq a\}$. Furthermore, one can effectively compute representatives $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_p \in \mathbb{Z}^n$, such that the tuples $\left(\mathrm{in}_{*,\boldsymbol{\alpha}_1}(\boldsymbol{g}_i)\right)_{i=1,\ldots,m}, \ldots, \left(\mathrm{in}_{*,\boldsymbol{\alpha}_p}(\boldsymbol{g}_i)\right)_{i=1,\ldots,m}$ are all the possible tuples $\left(\mathrm{in}_{*,\boldsymbol{\alpha}}(\boldsymbol{g}_i)\right)_{i=1,\ldots,m}$ when $\boldsymbol{\alpha}$ varies.*

▶ **Proposition 3.6.** *Let $* \in \{+,-\}$, $a \in \mathbb{Z}$ and $\mathcal{M}$ be an $\mathbb{A}$-submodule of $\mathbb{A}^n$. Given as input a finite basis of $\mathcal{M}$, it is decidable whether there exists $\boldsymbol{f} = (f_S, f_K, \cdots) \in \mathcal{M}$ such that*

$$\mathrm{lc}_*(\boldsymbol{f}) \in \mathbb{R}^n_{>0} \quad \text{and} \quad \deg_*(f_S) + a \geq \deg_*(f_K). \tag{19}$$

Proof. First we compute a set of super Gröbner basis $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m$ for $\mathcal{M}$. Then for each $* \in \{+,-\}$, by Lemma 6.6 we compute $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_p \in \mathbb{Z}^n$ such that the tuples $\left(\mathrm{in}_{*,\boldsymbol{\alpha}_1}(\boldsymbol{g}_i)\right)_{i=1,\ldots,m}$, $\ldots$, $\left(\mathrm{in}_{*,\boldsymbol{\alpha}_p}(\boldsymbol{g}_i)\right)_{i=1,\ldots,m}$ are all the possible tuples when $\alpha_S - \alpha_K \leq a$. For each of these $\boldsymbol{\alpha} \in \{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_p\}$, use linear programming to decide whether there exist real numbers $r_1, \ldots, r_m \in \mathbb{R}$ such that $\sum_{i=1}^m r_i \cdot \mathrm{coef}\left(\mathrm{in}_{*,\boldsymbol{\alpha}}(\boldsymbol{g}_i)\right) \in \mathbb{R}^n_{>0}$. By Corollary 6.5, such $r_1, \ldots, r_m \in \mathbb{R}$ exist if and only if there exists $\boldsymbol{f} \in \mathcal{M}$ with $\mathrm{coef}(\mathrm{in}_{*,\boldsymbol{\alpha}}(\boldsymbol{f})) \in \mathbb{R}^n_{>0}$. By Lemma 6.3, this is true if and only if there exists $\boldsymbol{f} = (f_S, f_K, \cdots) \in \mathcal{M}$ satisfying condition (19). ◀

References

1. Erwin H. Bareiss. Sylvester's identity and multistep integer-preserving gaussian elimination. *Mathematics of computation*, 22(103):565–578, 1968.
2. Gilbert Baumslag, Frank B. Cannonito, and Derek J. S. Robinson. The algorithmic theory of finitely generated metabelian groups. *Transactions of the American Mathematical Society*, 344(2):629–648, 1994.
3. Paul C. Bell, Mika Hirvensalo, and Igor Potapov. The Identity Problem for matrix semigroups in $\mathsf{SL}_2(\mathbb{Z})$ is NP-complete. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 187–206. SIAM, 2017.
4. Paul C. Bell and Igor Potapov. On the undecidability of the identity correspondence problem and its applications for word and matrix semigroups. *International Journal of Foundations of Computer Science*, 21(06):963–978, 2010.

5 Vincent D. Blondel, Emmanuel Jeandel, Pascal Koiran, and Natacha Portier. Decidable and undecidable problems about quantum automata. *SIAM Journal on Computing*, 34(6):1464–1473, 2005.

6 J. A. Bondy and U. S. R. Murty. *Graph theory with applications*, volume 290. Macmillan London, 1976.

7 Michaël Cadilhac, Dmitry Chistikov, and Georg Zetzsche. Rational subsets of baumslag-solitar groups. In *47th International Colloquium on Automata, Languages, and Programming, ICALP*, volume 168 of *LIPIcs*, pages 116:1–116:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

8 Olivier Chapuis. ∀-free metabelian groups. *The Journal of Symbolic Logic*, 62(1):159–174, 1997.

9 Christian Choffrut and Juhani Karhumäki. Some decision problems on integer matrices. *RAIRO-Theoretical Informatics and Applications-Informatique Théorique et Applications*, 39(1):125–131, 2005.

10 Valerio De Angelis and Selim Tuncel. Handelman's theorem on polynomials with positive multiples. *Codes, systems, and graphical models (Minneapolis, MN, 1999)*, pages 439–445, 2001.

11 Harm Derksen, Emmanuel Jeandel, and Pascal Koiran. Quantum automata and algebraic groups. *Journal of Symbolic Computation*, 39(3-4):357–371, 2005.

12 Ruiwen Dong. Solving homogeneous linear equations over polynomial semirings. *arXiv preprint*, 2022. To appear in STACS 2023. `arXiv:2209.13347`.

13 Manfred Einsiedler, Robert Mouat, and Selim Tuncel. When does a submodule of $(\mathbb{R}[x_1, \ldots, x_k])^n$ contain a positive element? *Monatshefte für Mathematik*, 140(4):267–283, 2003.

14 Moses Ganardi, Daniel König, Markus Lohrey, and Georg Zetzsche. Knapsack problems for wreath products. In *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, volume 96 of *LIPIcs*, pages 32:1–32:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

15 David Handelman. *Positive polynomials and product type actions of compact groups*, volume 320. American Mathematical Soc., 1985.

16 Ehud Hrushovski, Joël Ouaknine, Amaury Pouly, and James Worrell. Polynomial invariants for affine programs. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 530–539, 2018.

17 Olga Kharlampovich, Laura López, and Alexei Myasnikov. The diophantine problem in some metabelian groups. *Mathematics of Computation*, 89(325):2507–2519, 2020.

18 Olga G. Kharlampovich. A finitely presented solvable group with unsolvable word problem. *Izvestiya Rossiiskoi Akademii Nauk. Seriya Matematicheskaya*, 45(4):852–873, 1981.

19 Olga G. Kharlampovich and Mark V. Sapir. Algorithmic problems in varieties. *International Journal of Algebra and Computation*, 5(04n05):379–602, 1995.

20 Kenneth Krohn and John Rhodes. Algebraic theory of machines. I. Prime decomposition theorem for finite semigroups and machines. *Transactions of the American Mathematical Society*, 116:450–464, 1965.

21 Markus Lohrey. Subgroup membership in GL(2, Z). In *38th International Symposium on Theoretical Aspects of Computer Science (STACS 2021)*, volume 187 of *LIPIcs*, pages 51:1–51:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

22 Markus Lohrey, Benjamin Steinberg, and Georg Zetzsche. Rational subsets and submonoids of wreath products. *Information and Computation*, 243:191–204, 2015.

23 Wilhelm Magnus. On a theorem of Marshall Hall. *Annals of Mathematics*, pages 764–768, 1939.

24 A. Markov. On certain insoluble problems concerning matrices. *Doklady Akad. Nauk SSSR*, 57(6):539–542, 1947.

25 K. A. Mikhailova. The occurrence problem for direct products of groups. *Matematicheskii Sbornik*, 112(2):241–251, 1966.

26 Vitalii Anatol'evich Roman'kov. Equations in free metabelian groups. *Siberian Mathematical Journal*, 20(3):469–471, 1979.

27 N. S. Romanovskii. Some algorithmic problems for solvable groups. *Algebra and Logic*, 13(1):13–16, 1974.

28 Alfred Tarski. *A Decision Method for Elementary Algebra and Geometry*. second ed., rev., Univ. of California Press, Berkeley, 1951.

A Some omitted proofs

This appendix contains some omitted proofs from the main article. Other omitted proofs can be found in the full version of this paper.

▶ **Lemma 2.1.** *Let $\mathcal{G} = \{g_1, \dots, g_a\}$ be a set of elements in a group G. The semigroup $\langle \mathcal{G} \rangle$ is a group if and only if the neutral element I of G is represented by a full-image word over $\mathcal{G}$.*

Proof. Let $w \in \mathcal{G}^*$ be a full-image word with $\pi(w) = I$. Then for every i, the word w can be written as $w = vg_iv'$, so $g_i^{-1} = \pi(v')\pi(v) \in \langle \mathcal{G} \rangle$. Therefore, the semigroup $\langle \mathcal{G} \rangle$ contains all the inverse g_i^{-1}, and is thus a group.

If $\langle \mathcal{G} \rangle$ is a group, then for all i, the inverse g_i^{-1} can be written as $\pi(w_i)$ for some word $w_i \in \mathcal{G}^*$. Then the word $w \coloneqq g_1w_1g_2w_2 \cdots g_aw_a$ is a full-image word with $\pi(w) = \pi(g_1w_1) \cdots \pi(g_aw_a) = I$. ◀

▶ **Proposition 3.1.** *Suppose $I = \emptyset$ or $J = \emptyset$. The semigroup $\langle \mathcal{G} \rangle$ is a group if and only if $I = J = \emptyset$ and $\sum_{k \in K} n_k y_k = 0$ for some positive integers $n_k \in \mathbb{Z}_{>0}$. In particular, this is decidable by integer programming.*

Proof. First suppose $\langle \mathcal{G} \rangle$ is a group. Without loss of generality suppose $I = \emptyset$. Then every element $(y, b) \in \langle \mathcal{G} \rangle$ must satisfy $b \leq 0$. Therefore, for any $(y_j, b_j) \in \mathcal{G}$ with $b_j < 0$, we have $(y_j, b_j)^{-1} = (X^{-b_j} \cdot y_j, -b_j) \notin \langle \mathcal{G} \rangle$ by the positivity of $-b_j$. Hence $J = \emptyset$.

Since $\langle \mathcal{G} \rangle$ is a group, by Lemma 2.1 there exists a full-image word $w \in \mathcal{G}^*$ that represents $(0, 0)$. Let n_k be the number of times the word $(y_k, 0)$ appears in w, then $n_k > 0$ and w represents $\left(\sum_{k \in K} n_k y_k, 0\right) = (0, 0)$. This finishes the first implication.

For the converse implication, suppose $I = J = \emptyset$ and $\sum_{k \in K} n_k y_k = 0$ for some integers $n_k \in \mathbb{Z}_{>0}$. Then the word $\prod_{k \in K} (y_k, 0)^{n_k} = (0, 0)$. Therefore there exists a full-image word that represents $(0, 0)$. Hence $\langle \mathcal{G} \rangle$ is a group.

Let $p = \min_{k \in K}\{\deg_-(y_k)\}$ and $q = \max_{k \in K}\{\deg_+(y_k)\}$. For each $k \in K$, write $y_k = \sum_{t=p}^{q} \beta_{k,t} X^t$, then $\sum_{k \in K} n_k y_k = 0$ is equivalent to the system of linear equations

$$\sum_{k \in K} n_k \beta_{k,t} = 0, \quad t = p, p+1, \dots, q. \tag{32}$$

Deciding whether the system (32) has solution over $\mathbb{Z}_{>0}$ can be decided using integer programming. ◀

▶ **Corollary 3.3.** *The semigroup $\langle \mathcal{G} \rangle$ is a group if and only if there exist a double-full set $S \subset I \times J$ and polynomials $f_S, f_K, f_{(i,j)}, f_k \in \mathbb{R}_{\geq 0}[X^{\pm}]^*$ for $(i, j) \in S, k \in K$ that satisfy the following three conditions.*

(i) ***(System of linear equations)*** *The following linear equations over* $\mathbb{R}(X)$ *are satisfied:*

$$\sum_{(i,j)\in S} f_{(i,j)} h_{(i,j),m} + \sum_{k\in K} f_k y_{k,m} = 0, \quad m = 0,\dots,d-1, \tag{10}$$

$$f_S = \sum_{(i,j)\in S} f_{(i,j)}, \qquad f_K = \sum_{k\in K} f_k. \tag{11}$$

(ii) ***(Positive degree bound)*** *We have:*

$$\deg_+(f_S) + 1 \geq \deg_+(f_K). \tag{12}$$

(iii) ***(Negative degree bound)*** *We have:*

$$\deg_-(f_S) \leq \deg_-(f_K). \tag{13}$$

Proof. We show that Corollary 3.3 is equivalent to Proposition 3.2.

By the definition of $h_{(i,j),m}, y_{k,m}, (i,j) \in S, k \in K, m = 0,\dots,d-1$ in Equation (8) and (9), the Equation (5) in Condition (i) of Proposition 3.2 is equivalent to the following system:

$$\sum_{(i,j)\in S} f'_{(i,j)} h_{(i,j),m} + \sum_{k\in K} f'_k y_{k,m} = 0, \quad m = 0,\dots,d-1. \tag{33}$$

Where $f'_{(i,j)}, f'_k$ are polynomials in $\mathbb{N}[X^{\pm}]^*$ such that $f_{(i,j)} = f'_{(i,j)}(X^d), f_k = f'_k(X^d)$.

On one hand, suppose there exist polynomials $f_{(i,j)}, f_k \in \mathbb{N}[X^{\pm d}]^*$ that satisfy Conditions (i)-(iii) of Proposition 3.2. Then the polynomials $f'_{(i,j)}, f'_k$ satisfying the system (33) are also in $\mathbb{R}_{\geq 0}[X^{\pm}]$. Then let $f'_S = \sum_{(i,j)\in S} f'_{(i,j)}$ and $f'_K = \sum_{k\in K} f'_k$, so Conditions (i)-(iii) of Corollary 3.3 are satisfied for $f'_S, f'_K, f'_{(i,j)}, f'_k \in \mathbb{R}_{>0}[X^{\pm}]^*$.

On the other hand, suppose there exist polynomials $f'_S, f'_K, f'_{(i,j)}, f'_k \in \mathbb{R}_{>0}[X^{\pm}]^*$ that satisfy Conditions (i)-(iii) of Corollary 3.3. We show that there exist $f_S, f_K, f_{(i,j)}, f_k \in \mathbb{N}[X^{\pm}]^*, (i,j) \in S, k \in K$ that satisfy the same Equations (10)-(11), and such that $\deg_\pm f'_{(i,j)} = \deg_\pm f_{(i,j)}$, $\deg_\pm f'_k = \deg_\pm f_k$ for all $(i,j) \in S, k \in K$.

In fact, by the homogeneity of Equations (10)-(11), one can multiply all $h_{(i,j),m}$ and $y_{k,m}$ simultaneously by their common denominator, and suppose $h_{(i,j),m}, y_{k,m} \in \mathbb{Q}[X^{\pm}]$. Then, *fixing the degrees of* $f_{(i,j)}$ *and* f_k, one can rewrite Equations (10)-(11) as a system of homogeneous linear equation where the variables are the coefficients of $f_{(i,j)}$ and f_k. We then add to this system of homogeneous linear equations a boolean combination of homogeneous linear inequalities to guarantee $f_{(i,j)}, f_k \in \mathbb{R}_{>0}[X^{\pm}]^*$ (this can be expressed using inequalities for the coefficients of $f_{(i,j)}, f_k$), as well as to guarantee the degree of $f_{(i,j)}, f_k$. Since this system of homogeneous linear equations plus boolean combination of homogeneous linear inequalities has a solution over $\mathbb{R}$, it also has a solution over $\mathbb{Q}$, and even over $\mathbb{Z}$ by the homogeneity. Therefore, we obtain a solution over $\mathbb{Z}$ for the coefficients of $f_{(i,j)}, f_k$. This gives us a solution $f_{(i,j)}, f_k \in \mathbb{N}[X^{\pm}]^*$ with the same fixed degrees.

Consequently, $f_{(i,j)}, f_k, (i,j) \in S, k \in K$, satisfy the system (33). Thus, $f_{(i,j)}(X^d), f_k(X^d), (i,j) \in S, k \in K$. satisfy Conditions (i)-(iii) of Proposition 3.2. ◀

▶ **Lemma 4.4.** *The graph* $G(\widehat{w})$ *can be constructed by starting with an edgeless graph with a single vertex* 0 *and gradually attaching primitive circuits.*

Proof. Denote by G_0 the edgeless graph with a single vertex 0. We show that every Eulerian $\widehat{\mathcal{G}}$-graph G can be constructed by attaching primitive circuits to G_0. We use induction on the number of edges in G. When there is no edge in G, it is G_0, and we are done.

When there are loops in G, these are loops of some label $k \in K$, and are therefore primitive circuits themselves. Removing them results in another Eulerian graph G' and decreases the number of edges. By the induction hypothesis G' can be constructed by attaching primitive circuits to G_0. Then attaching the loops (primitive circuits of type in K) to G' results in G.

When there are no loops in G, denote $m \coloneqq \max(V(G))$. Then since G is Eulerian, there must be an edge e of label $\ell(e)$ starting from the vertex m, and an edge e' of label $\ell(e')$ ending at the vertex m. Since all the edges in G are of length d, the edge e must end at vertex $m - d$, and e' must start at $m - d$. (The length of an edge of label a is $\widehat{b}_a$.) Therefore the circuit consisting of e and e' is primitive of type $(\ell(e), \ell(e'))$. Removing the circuit (and the vertex m if necessary) results in a graph G'. G' is connected (and Eulerian) since there is no loop at m, and the only possible neighbour of the vertex m is $m - d$. By the induction hypothesis G' can be constructed by attaching primitive circuits to G_0. Then attaching to G' the primitive circuits of type $(\ell(e), \ell(e'))$ at vertex $m - d$ results in G. ◀

Indiscernibles and Flatness in Monadically Stable and Monadically NIP Classes

Jan Dreier
TU Wien, Austria

Nikolas Mählmann
Universität Bremen, Germany

Sebastian Siebertz
Universität Bremen, Germany

Szymon Toruńczyk
University of Warsaw, Poland

Abstract

Monadically stable and monadically NIP classes of structures were initially studied in the context of model theory and defined in logical terms. They have recently attracted attention in the area of structural graph theory, as they generalize notions such as nowhere denseness, bounded cliquewidth, and bounded twinwidth.

Our main result is the – to the best of our knowledge first – purely combinatorial characterization of monadically stable classes of graphs, in terms of a property dubbed *flip-flatness*. A class $\mathscr{C}$ of graphs is flip-flat if for every fixed radius r, every sufficiently large set of vertices of a graph $G \in \mathscr{C}$ contains a large subset of vertices with mutual distance larger than r, where the distance is measured in some graph G' that can be obtained from G by performing a bounded number of flips that swap edges and non-edges within a subset of vertices. Flip-flatness generalizes the notion of uniform quasi-wideness, which characterizes nowhere dense classes and had a key impact on the combinatorial and algorithmic treatment of nowhere dense classes. To obtain this result, we develop tools that also apply to the more general monadically NIP classes, based on the notion of indiscernible sequences from model theory. We show that in monadically stable and monadically NIP classes indiscernible sequences impose a strong combinatorial structure on their definable neighborhoods. All our proofs are constructive and yield efficient algorithms.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Finite Model Theory

Keywords and phrases stability, NIP, combinatorial characterization, first-order model checking

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.125

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `https://arxiv.org/abs/2206.13765` [15]

Funding *Nikolas Mählmann*: supported by the German Research Foundation (DFG) with grant greement No. 444419611.
Szymon Toruńczyk: supported by the project BOBR that is funded from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme with grant agreement No. 948057.

Acknowledgements We thank Édouard Bonnet, Jakub Gajarský, Stephan Kreutzer, Amer E. Mouawad and Alexandre Vigny for their valuable contributions to this paper. In particular, we thank Jakub Gajarský and Stephan Kreutzer for suggesting the notion of flip-flatness and providing a proof for the cases $r = 1, 2$.

1 Introduction

An important open problem in structural and algorithmic graph theory is to characterize those hereditary graph classes for which the model checking problem for first-order logic is tractable[1] [23, Section 8.2]. A result of Grohe, Kreutzer, and Siebertz [24] states that for monotone graph classes (that is, classes closed under removing vertices and edges), the limit of tractability is precisely captured by the notion of nowhere denseness, introduced by Nešetřil and Ossona de Mendez [29]. Examples of nowhere dense classes include the class of planar graphs, all classes that exclude a fixed minor, and classes with bounded expansion. Whereas these classes are sparse (for instance, they exclude some fixed biclique as a subgraph), the aforementioned problem seeks to generalize the result of Grohe, Kreutzer, and Siebertz to classes that are not necessarily sparse. Indeed, there are known hereditary graph classes that are not sparse, and for which the model checking problem is tractable, such as transductions of classes of bounded local cliquewidth [5], transductions of nowhere dense classes [16], or classes of ordered graphs (that is, graphs equipped with a total order) of bounded twinwidth [6].

So far, a complete picture is understood in two contexts: for monotone graph classes, where tractability coincides with nowhere denseness, and for hereditary classes of ordered graphs, where tractability coincides with bounded twinwidth. Despite the apparent dissimilarity of the combinatorial definitions of nowhere denseness and bounded twinwidth, those notions can be alternatively characterized in a uniform way in logical terms by the following notion, originating in model theory. Unless mentioned otherwise, all formulas are first-order formulas.

Say that a class $\mathscr{C}$ of graphs *transduces* a class $\mathscr{D}$ of graphs if for every $H \in \mathscr{D}$ there is some $G \in \mathscr{C}$ from which H can be obtained by performing the following steps: (1) coloring the vertices of G arbitrarily (2) interpreting a fixed formula $\varphi(x, y)$ (involving the edge relation and unary relations for the colors), thus yielding a new graph $\varphi(G)$ with the same vertices as G and edges uv such that $\varphi(u, v)$ holds, and finally (3) taking an induced subgraph of $\varphi(G)$. The transducability relation on graph classes is transitive, and classes that *do not* transduce the class of all graphs are called *monadically NIP*. For instance, the class of all bipartite graphs transduces the class of all graphs: to obtain an arbitrary graph G, consider its 1-subdivision, obtained by placing one vertex on each edge of G, thus yielding a bipartite graph H; then the formula $\varphi(x, y)$ expressing that x and y have a common neighbor defines a graph on $V(H)$ containing G as an induced subgraph. Hence, the class of bipartite graphs is not monadically NIP. On the other hand, all the graph classes mentioned earlier – nowhere dense classes and transductions thereof, classes of bounded twinwidth, or transductions of classes with bounded local cliquewidth – are monadically NIP. This suggests that monadic NIP might constitute the limit of tractability of the model checking problem. More precisely, the following has been conjectured[2].

▶ **Conjecture 1** ([1]). *Let $\mathscr{C}$ be a hereditary class of graphs. Then the model checking problem for first-order logic is fixed parameter tractable on $\mathscr{C}$ if and only if $\mathscr{C}$ is monadically NIP.*

Quite remarkably, among monotone graph classes, classes that are monadically NIP correspond precisely to nowhere dense classes [2], and among hereditary graph classes of ordered graphs, classes that are monadically NIP correspond precisely to classes of bounded

[1] more precisely, *fixed parameter-tractable*, that is, solvable in time $f(|\varphi|) \cdot |G|^c$, where φ is the input formula and G is the input graph, for some function $f : \mathbb{N} \to \mathbb{N}$ and constant c

[2] To the best of our knowledge the conjecture was first explicitly discussed during the open problem session of the Algorithms, Logic and Structure Workshop in Warwick, in 2016, see [1].

twinwidth [6]. One may tweak the definition of monadic NIP classes by considering other logics than first-order logic. For instance, for the counting extension CMSO_2 of monadic second-order logic, one recovers precisely the notion of classes of bounded cliquewidth [9], or classes of bounded treewidth if only monotone classes are considered. Thus, variations on the definition of monadic NIP recover important notions from graph theory: nowhere denseness, bounded twinwidth, bounded treewidth, and bounded cliquewidth.

Note that both implications in Conjecture 1 remain open. The conjecture is so far confirmed for monotone graph classes [24] (where monadically NIP classes are exactly the nowhere dense classes) and for hereditary classes of ordered graphs [6], tournaments [22], interval graphs and permutation graphs [7], (where monadically NIP classes are exactly the classes of bounded twinwidth). As a special important case, the conjecture predicts that all *monadically stable* graph classes are tractable. A class $\mathscr{C}$ is monadically stable if it does not transduce the class of all half-graphs, that is, graphs with vertices $a_1, b_1, \ldots, a_n, b_n$ such that a_i is adjacent to b_j if and only if $i \leqslant j$. Although much more restrictive than monadically NIP classes, monadically stable classes include all nowhere dense classes [2], and hence also all classes $\mathscr{D}$ that transduce in a nowhere dense class $\mathscr{C}$ (called structurally nowhere dense classes [19]). Those include dense graph classes, such as for instance squares of planar graphs. In fact, it is conjectured [30] that every monadically stable class of graphs is structurally nowhere dense.

These outlined connections between structural graph theory and model theory have recently triggered the interest to generalize combinatorial and algorithmic results from nowhere dense classes to structurally nowhere dense, monadically stable and monadically NIP classes, and ultimately to efficiently solve the model checking problem for first-order logic on these classes [5, 6, 19, 13, 21, 27, 30, 31, 32]. Logical results on monadically stable and monadically NIP classes in model theory include [3, 34, 8, 4].

Contribution

As discussed above, many central graph classes such as those with bounded cliquewidth, twinwidth or nowhere dense classes can be characterized both from a structural (i.e., graph theoretic) and a logical perspective. While monadically stable and monadically NIP graph classes are naturally defined via logic, structural characterizations have so far been elusive. In this work we take a step towards a structure theory for monadically stable and monadically NIP classes of graphs, which is the basis for their future algorithmic and combinatorial treatment, in particular, a tool for approaching Conjecture 1, as we discuss later.

Flatness. Our main result, Theorem 3, provides a purely combinatorial characterization of monadically stable graph classes in terms of *flip-flatness*. Flip-flatness generalizes *uniform quasi-wideness*[3], introduced by Dawar in [10] in his study of homomorphism preservation properties. Uniform quasi-wideness was proved by Nešetřil and Ossona de Mendez [29] to characterize nowhere dense graph classes and is a key tool in the combinatorial and algorithmic treatment of these classes. To foster the further discussion, let us formally define this notion.

[3] Another reasonable name for flip-flatness would be *flip-wideness*. We avoided this name to prevent confusion with the recently introduced graph parameter *flip-width* [37], which is studied in the same context.

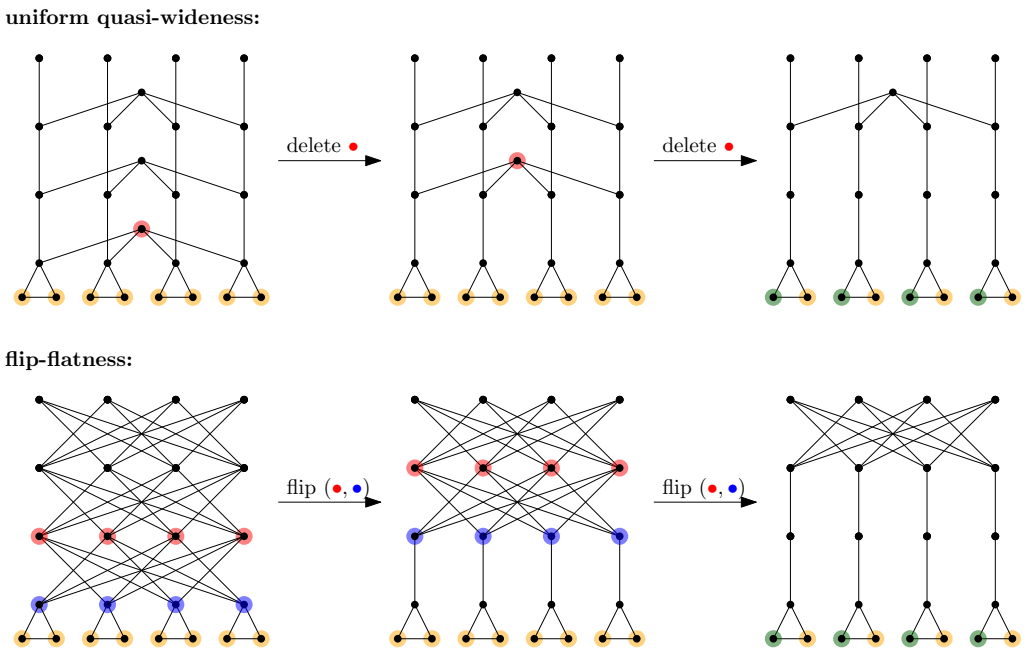


Figure 1 An example of uniform quasi-wideness (flip-flatness). Among the yellow vertices, we find a still large set of green vertices, that is distance-7 independent after deleting a bounded number of red vertices (performing a bounded number of flips between sets of red and blue vertices). Two key properties are illustrated: 1. the higher the desired independence distance, the more operations have to be performed; 2. we cannot hope to make all the yellow vertices distance-r independent with a bounded number of operations.

A set of vertices is *distance-r independent* in a graph if any two vertices in the set are at distance greater than r. A class of graphs $\mathscr{C}$ is *uniformly quasi-wide* if for every $r \in \mathbb{N}$ there exists a function $N_r : \mathbb{N} \to \mathbb{N}$ and a constant $s \in \mathbb{N}$ with the following property. For all $m \in \mathbb{N}$, $G \in \mathscr{C}$, and $A \subseteq V(G)$ with $|A| \geqslant N_r(m)$, there exists $S \subseteq V(G)$ with $|S| \leqslant s$ and $B \subseteq A \setminus S$ with $|B| \geqslant m$ such that B is distance-r independent in $G - S$ (see the top of Figure 1). Intuitively, for uniformly quasi-wide classes, in sufficiently large sets one can find large subsets that are distance-r independent after the deletion of a constant number of vertices. Uniform quasi-wideness is only suitable for the treatment of sparse graphs. Already very simple dense graph classes, such as the class of all cliques, are not uniformly quasi-wide.

Inspired by the notion of uniform quasi-wideness, Jakub Gajarský and Stephan Kreutzer proposed the new notion of *flip-flatness*. Roughly speaking, flip-flatness generalizes uniform quasi-wideness by replacing in its definition the deletion of a bounded size set of vertices by the inversion of the edge relation between a bounded number of (arbitrarily large) vertex sets. Let us make this definition more precise. A *flip* in a graph G is specified by a pair of sets $\mathsf{F} = (A, B)$ with $A, B \subseteq V(G)$. We write $G \oplus \mathsf{F}$ for the graph with the same vertices as G, and edges uv such that $uv \in E(G)$ xor $(u, v) \in (A \times B) \cup (B \times A)$. For a set $F = \{\mathsf{F}_1, \ldots, \mathsf{F}_n\}$ of flips, we write $G \oplus F$ for the graph $G \oplus \mathsf{F}_1 \oplus \cdots \oplus \mathsf{F}_n$.

▶ **Definition 2.** *A class of graphs $\mathscr{C}$ is* flip-flat *if for every $r \in \mathbb{N}$ there exists a function $N_r : \mathbb{N} \to \mathbb{N}$ and a constant $s_r \in \mathbb{N}$ such that for all $m \in \mathbb{N}$, $G \in \mathscr{C}$, and $A \subseteq V(G)$ with $|A| \geqslant N_r(m)$, there exists a set F of at most s_r flips and $B \subseteq A$ with $|B| \geqslant m$ such that B is distance-r independent in $G \oplus F$.*

Intuitively, for flip-flat classes in sufficiently large sets one can find large subsets that are distance-r independent after a constant number of flips (see bottom of Figure 1). For example the class of all cliques is flip-flat, requiring only a single flip to make the whole vertex set distance-∞ independent. Our main result is the following purely combinatorial characterization of monadic stability.

▶ **Theorem 3.** *A class of graphs is monadically stable if and only if it is flip-flat.*

Notably, our proof is algorithmic and yields polynomial bounds in the following sense.

▶ **Theorem 4.** *Every monadically stable class $\mathscr{C}$ of graphs is flip-flat, where for every r, the function N_r is polynomial. Moreover, given an n-vertex graph $G \in \mathscr{C}$, $A \subseteq V(G)$, and $r \in \mathbb{N}$, we can compute a subset $B \subseteq A$ and a set of flips F that makes B distance-r independent in $G \oplus F$ in time $\mathcal{O}(f_{\mathscr{C}}(r) \cdot n^3)$ for some function $f_{\mathscr{C}}$.*

Just as uniform quasi-wideness provided a key tool for the algorithmic treatment of nowhere dense graph classes, in particular for first-order model checking [24], we believe that the characterization by flip-flatness will provide an important step towards the algorithmic treatment of monadically stable classes. We leave as an open question whether a similar characterization of monadically NIP classes exists.

Indiscernible sequences. Our study is based on *indiscernible sequences*, which are a fundamental tool in model theory. An indiscernible sequence is a (finite or infinite) sequence $I = (\bar{a}_1, \bar{a}_2, \ldots)$ of tuples of equal length of elements of a fixed (finite or infinite) structure, such that any two finite subsequences of I that have equal length, satisfy the same formulas. More generally, for a set of formulas Δ, the sequence I is Δ-*indiscernible* if for each formula $\varphi(\bar{x}_1, \ldots, \bar{x}_k)$ from Δ either all subsequences of I of length k satisfy φ, or no subsequence of length k satisfies φ. For example, any enumeration of vertices forming a clique or an independent set in a graph is Δ-indiscernible for $\Delta = \{E(x_1, x_2)\}$ containing only the edge relation. Also, any increasing sequence of rationals in the structure $(\mathbb{Q}, <)$ is Δ-indiscernible for Δ being the set of all formulas. We use indiscernible sequences to obtain new insights about monadically stable and monadically NIP classes.

It was shown by Blumensath [4] that in monadically NIP classes, any fixed element interacts with the tuples of an indiscernible sequence in a very regular way. We give a new finitary proof of Blumensath's result. Building on this, we develop our main technical tool, Theorem 11, where we show that the regular properties of the indiscernible sequences extend to their disjoint definable neighborhoods (see Section 3.2 for details). A result similar to Theorem 11, also for disjoint definable neighborhoods in monadically NIP classes, plays a key role in [6, 35].

Apart from powering our algorithmic proof of flip-flatness, Theorem 11 has already found further applications in monadically stable classes of graphs: it was recently used to obtain improved bounds for Ramsey numbers [14] and to prove an algorithmic game-characterization of these classes, called *Flipper game* [20]. The paper [20] provides two proofs for this game characterization. One is constructive and algorithmic and builds on our work. The other builds on model theoretic tools; it is non-constructive but self-contained and highlights additional properties of monadically stable graph classes, including a second (though non-constructive) proof of Theorem 3. The algorithmic version of the Flipper game plays a crucial role for proving that the first-order model checking problem is fixed parameter tractable for structurally nowhere dense classes of graphs [16]. This suggests that our developed techniques may be an important tool towards resolving Conjecture 1.

2 Preliminaries

We use standard notation from graph theory and model theory and refer to [11] and [25] for extensive background. We write $[m]$ for the set of integers $\{1, \dots, m\}$.

Relational structures and graphs. A *(relational) signature* Σ is a set of relation symbols, each with an associated non-negative integer, called its *arity*. A Σ-*structure* $\mathbb{A}$ consists of a *universe*, which is a non-empty, possibly infinite set, and *interpretations* of the symbols from the signature: each relation symbol of arity k is interpreted as a k-ary relation over the universe. By a slight abuse of notation, we do not differentiate between structures and their universes and between relation symbols and their interpretations. By $\mathscr{C}$ we denote classes of structures over a fixed signature. Unless indicated otherwise, $\mathscr{C}$ may contain finite and infinite structures.

A *monadic extension* Σ^+ of a signature Σ is any extension of Σ by unary predicates. A unary predicate will also be called a *color*. A *monadic expansion* or *coloring* of a Σ-structure $\mathbb{A}$ is a Σ^+-structure $\mathbb{A}^+$, where Σ^+ is a monadic extension of Σ, such that $\mathbb{A}$ is the Σ-*reduct* of $\mathbb{A}^+$, that is, the Σ-structure obtained from $\mathbb{A}^+$ by removing all relations with symbols in $\Sigma^+ \setminus \Sigma$. When $\mathscr{C}$ is a class of Σ-structures and Σ^+ is a monadic extension of Σ, we write $\mathscr{C}[\Sigma^+]$ for the class of all possible Σ^+-expansions of structures from $\mathscr{C}$.

A *graph* is a finite structure over the signature consisting of a binary relation symbol E, interpreted as the symmetric and irreflexive edge relation.

First-order logic. We say that two tuples $\bar{a}, \bar{b}$ of elements are φ-*connected* in a structure $\mathbb{A}$ if $\mathbb{A} \models \varphi(\bar{a}, \bar{b})$. We call the set $N_\varphi^{\mathbb{A}}(\bar{a}) = \{\bar{b} \in \mathbb{A}^{|\bar{y}|} \ : \ \mathbb{A} \models \varphi(\bar{a}, \bar{b})\}$ the φ-*neighborhood* of $\bar{a}$. We simply write $N_\varphi(\bar{a})$ when $\mathbb{A}$ is clear from the context.

Let $\Phi(\bar{x})$ be a finite set of formulas with free variables $\bar{x}$. A Φ-*type* is a conjunction $\tau(\bar{x})$ of formulas in Φ or their negations, such that every formula in Φ occurs in τ either positively or negatively. More precisely, $\tau(\bar{x})$ is a formula of the following form, for some subset $A \subseteq \Phi$:

$$\bigwedge_{\varphi \in A} \varphi(\bar{x}) \wedge \bigwedge_{\varphi \in \Phi \setminus A} \neg\varphi(\bar{x}).$$

Note that for every $|\bar{x}|$-tuple $\bar{a}$ of elements of $\mathbb{A}$, there is exactly one Φ-type $\tau(\bar{x})$ such that $\mathbb{A} \models \tau(\bar{a})$.

Stability and NIP. While we already defined monadic stability and NIP in terms of transductions in the introduction, let us also give the equivalent original definition. A formula $\varphi(\bar{x}, \bar{y})$ has the k-*order property* on a class $\mathscr{C}$ of structures if there are $\mathbb{A} \in \mathscr{C}$ and two sequences $(\bar{a}_i)_{1 \leqslant i \leqslant k}$, $(\bar{b}_i)_{1 \leqslant i \leqslant k}$ of tuples of elements of $\mathbb{A}$, such that for all $i, j \in [k]$

$$\mathbb{A} \models \varphi(\bar{a}_i, \bar{b}_j) \quad \iff \quad i \leqslant j.$$

The formula φ is said to have the *order property* on $\mathscr{C}$ if it has the k-order property for all $k \in \mathbb{N}$. The class $\mathscr{C}$ is called *stable* if no formula has the order-property on $\mathscr{C}$. A class $\mathscr{C}$ of Σ-structures is *monadically stable* if for every monadic extension Σ^+ of Σ the class $\mathscr{C}[\Sigma^+]$ is stable.

Similarly, a formula $\varphi(\bar{x}, \bar{y})$ has the k-*independence property* on a class $\mathscr{C}$ if there are $\mathbb{A} \in \mathscr{C}$, a size k set $A \subseteq \mathbb{A}^{|\bar{x}|}$ and a sequence $(\bar{b}_J)_{J \subseteq A}$ of tuples of elements of $\mathbb{A}$ such that for all $J \subseteq A$ and for all $\bar{a} \in A$

$$\mathbb{A} \models \varphi(\bar{a}, \bar{b}_J) \quad \iff \quad \bar{a} \in J.$$

We then say that A is *shattered* by φ. We define the *independence property* (IP), classes with the *non-independence property* (NIP classes), and *monadically NIP* classes as expected. Note that every (monadically) stable class is (monadically) NIP. Baldwin and Shelah proved that in the definitions of monadic stability and monadic NIP, one can alternatively rely on formulas $\varphi(x,y)$ with just a pair of singleton variables, instead of a pair of tuples of variables [3, Lemma 8.1.3, Theorem 8.1.8].

We call a relation R *definable* on a structure $\mathbb{A}$ if $R = \{\bar{a} \in \mathbb{A}^{|\bar{x}|} : \mathbb{A} \models \varphi(\bar{a})\}$ for some formula $\varphi(\bar{x})$. The following is immediate from the previous definitions.

▶ **Lemma 5.** *Let $\mathscr{C}$ be a monadically stable (monadically NIP) class of Σ-structures, let Σ^+ be a monadic extension of Σ and let $\mathscr{D}$ be the hereditary closure of any expansion of $\mathscr{C}[\Sigma^+]$ by definable relations. Then also $\mathscr{D}$ is monadically stable (monadically NIP).*

A formula $\varphi(x,\bar{y})$ has *pairing index* k on a class $\mathscr{C}$ of structures if there is $\mathbb{A} \in \mathscr{C}$ and two sequences $(a_{ij})_{1 \leqslant i < j \leqslant k}$ and $(\bar{b}_i)_{1 \leqslant i \leqslant k}$ such that for all $1 \leqslant i < j \leqslant k$ and $\ell \in [k]$

$$\mathbb{A} \models \varphi(a_{ij}, \bar{b}_\ell) \quad \iff \quad \ell \in \{i,j\}.$$

We require that x is a single free variable, while $\bar{y}$ is allowed to be a tuple of variables. Intuitively, from a graph theoretic perspective, if a formula has unbounded pairing index on a class $\mathscr{C}$, then it can encode arbitrarily large 1-subdivided cliques in $\mathscr{C}$, where the principal vertices are represented by the tuples $\bar{b}_\ell$ and the subdivision vertices are represented by single elements a_{ij}. As discussed in the introduction, this is sufficient to encode arbitrary graphs in $\mathscr{C}$ by using an additional color predicate. In this case, $\mathscr{C}$ cannot be monadically NIP. The above reasoning is formalized in the following characterization.

▶ **Lemma 6.** *A class $\mathscr{C}$ of Σ-structures is monadically NIP if and only if for every monadic extension Σ^+ of Σ every Σ^+-formula $\varphi(x,\bar{y})$ has bounded pairing index on $\mathscr{C}[\Sigma^+]$.*

Indiscernible sequences. Let $\mathbb{A}$ be a Σ-structure and $\varphi(\bar{x}_1, \ldots, \bar{x}_m)$ a formula. We say a sequence $(\bar{a}_i)_{1 \leqslant i \leqslant n}$ of tuples from $\mathbb{A}$ (where all $\bar{x}_i$ and $\bar{a}_j$ have the same length) is a *φ-indiscernible sequence* of length n, if for every two sequences of indices $i_1 < \cdots < i_m$ and $j_1 < \cdots < j_m$ from $[n]$ we have

$$\mathbb{A} \models \varphi(\bar{a}_{i_1}, \ldots, \bar{a}_{i_m}) \quad \iff \quad \mathbb{A} \models \varphi(\bar{a}_{j_1}, \ldots, \bar{a}_{j_m}).$$

For a set of formulas Δ we call a sequence *Δ-indiscernible* if it is φ-indiscernible for all $\varphi \in \Delta$. For finite Δ, by (iteratively applying) Ramsey's theorem we can extract a Δ-indiscernible sequence from any sequence. In general structures however, in order to extract a large Δ-indiscernible sequence we must initially start with an enormously large sequence. To the best of our knowledge the following theorem goes back to Ehrenfeucht and Mostowski [17].

▶ **Theorem 7.** *Let Δ be a finite set of formulas. There exists a function f such that every sequence of elements of length at least $f(m)$ (in a structure with a signature matching Δ) contains a Δ-indiscernible subsequence of length m.*

In stable classes we can efficiently find polynomially large (that is, $f(m) = m^{O(1)}$) indiscernible sequences of elements (see also [28, Theorem 3.5]). In the full version of the paper we observe that the run time is essentially bounded by the time it takes to evaluate the formulas from Δ on a small polynomial part of the input structure.

3 Technical overview

Our work is organized as follows. Section 3.1 introduces useful combinatorial properties of indiscernibles in monadically NIP classes. We extend those properties in Section 3.2, where we prove our main tool, Theorem 11. We use this tool in Section 3.3 to prove flip-flatness. Due to space constraints, we mostly provide proof sketches, which should convey the key ideas. All missing proofs can be found in the full version of the paper [15].

3.1 Indiscernibles in monadically stable and monadically NIP classes

In classical model theory, instead of classes of finite structures, usually infinite structures are studied. For a single infinite structure $\mathbb{A}$, we say $\mathbb{A}$ is (monadically) stable/NIP, if the class $\{\mathbb{A}\}$ is (monadically) stable/NIP. In this context, an indiscernible sequence is usually assumed to be of infinite length and indiscernible over the set of all first-order formulas. Using Ω to denote the set of all first-order formulas, we will denote the latter property as Ω-*indiscernibility*.

It is well-known that indiscernible sequences can be used to characterize stability and NIP for infinite structures. To characterize NIP, define the *alternation rank* of a formula $\varphi(\bar{x}, \bar{y})$ over a sequence I in a structure $\mathbb{A}$ as the maximum k such that there exists a (possibly non-contiguous) subsequence $(\bar{a}_1, \ldots, \bar{a}_{k+1})$ of I and a tuple $\bar{b} \in \mathbb{A}^{|\bar{x}|}$ such that for all $i \in [k]$ we have $\mathbb{A} \models \varphi(\bar{b}, \bar{a}_i) \iff \mathbb{A} \models \neg\varphi(\bar{b}, \bar{a}_{i+1})$. A structure $\mathbb{A}$ is NIP if and only if every formula has finite alternation rank over every Ω-indiscernible sequence in $\mathbb{A}$ [33, Theorem 12.17]. Stable classes can be characterized in a similar way. We say the *exception rank* of a formula $\varphi(\bar{x}, \bar{y})$ over a sequence I in a structure $\mathbb{A}$ is the maximum k such that there exists a tuple $\bar{b} \in \mathbb{A}^{|\bar{x}|}$ such that for k distinct tuples $\bar{a}_i \in I$ we have $\mathbb{A} \models \varphi(\bar{b}, \bar{a}_i)$ and for k other distinct tuples $\bar{a}_i \in I$ we have $\mathbb{A} \models \neg\varphi(\bar{b}, \bar{a}_i)$. A structure $\mathbb{A}$ is stable if and only if every formula has finite exception rank over every Ω-indiscernible sequence in $\mathbb{A}$ [33, Corollary 12.24].

Hence, NIP and stablility can be characterized by the interaction of tuples of elements with indiscernible sequences. For *monadically* NIP structures, Blumensath [4] shows that the interaction of single elements with indiscernible sequences is even more restricted.

▶ **Theorem 8** ([4, Corollary 4.13]). *In every monadically NIP structure $\mathbb{A}$, for every formula $\varphi(x, \bar{y})$, where x is a single free variable, and Ω-indiscernible sequence $(\bar{a}_i)_{i\in\mathbb{N}}$ of $|\bar{y}|$-tuples the following holds: for every element $b \in \mathbb{A}$ there exists an exceptional index $\mathrm{ex}(b) \in \mathbb{N}$ and two truth values $t_<(b), t_>(b) \in \{0, 1\}$ such that*

$$\begin{aligned} &\text{for all } i < \mathrm{ex(b)} : \mathbb{A} \models \varphi(b, \bar{a}_i) \iff t_<(b) = 1, \text{ and} \\ &\text{for all } i > \mathrm{ex(b)} : \mathbb{A} \models \varphi(b, \bar{a}_i) \iff t_>(b) = 1. \end{aligned}$$

See Figure 2 for examples. In particular, the alternation rank of the formulas $\varphi(x, \bar{y})$ with a single free variable x over every Ω-indiscernible sequence in a monadically NIP structure is at most 2.

Theorem 8 will be a crucial tool for proving flip-flatness. However, as we strive for algorithmic applications, we have to develop an effective, computable version that is suitable for handling classes of finite structures. Instead of requiring Ω-indiscernibility, we will specifically consider Δ-indiscernible sequences with respect to special sets $\Delta = \Delta_k^\Phi$ that we define soon. These sets Δ_k^Φ will strike the right balance of being on the one hand sufficiently rich to allow us to derive structure from them, but on the other hand sufficiently simple such that we can efficiently evaluate formulas from Δ_k^Φ, making our flip-flatness result algorithmic.

Figure 2 Two monadically NIP structures. On the left: the infinite half-graph, where Theorem 8 applies for the edge relation with $t_<(b) = 0$ and $t_>(b) = 1$. On the right: the infinite matching, where we have $t_<(b) = t_>(b) = 0$ but a differing truth value at index $\mathrm{ex}(b) = 3$.

Fix a finite set $\Phi(x, \bar{y})$ of formulas of the form $\varphi(x, \bar{y})$ where x is a single variable. A Φ-*pattern* is a finite sequence $(\varphi_i)_{1 \leqslant i \leqslant k'}$, where each formula $\varphi_i(x, \bar{y})$ is a boolean combination of formulas $\varphi(x, \bar{y}) \in \Phi(x, \bar{y})$. Given a Φ-pattern $(\varphi_i)_{i \in [k']}$, the following formula expresses that, for a given sequence of k' tuples, each of length $|\bar{y}|$, there is some element that realizes that pattern (see Figure 3 for an example):

$$\gamma_{(\varphi_1,\dots,\varphi_{k'})}(\bar{y}_1, \dots, \bar{y}_{k'}) = \exists x. \bigwedge_{i \in [k']} \varphi_i(x, \bar{y}_i).$$

For a finite set of formulas $\Phi(x, \bar{y})$ and an integer k we define Δ_k^Φ to be the set of all formulas $\gamma_{(\varphi_1,\dots,\varphi_{k'})}$, where $(\varphi_1, \dots, \varphi_{k'})$ is a Φ-pattern of length $k' \leqslant k$. Note that the set Δ_k^Φ is finite, as (up to equivalence) there are only finitely many boolean combinations of formulas in Φ. We write Δ_k^φ for $\Delta_k^{\{\varphi\}}$.

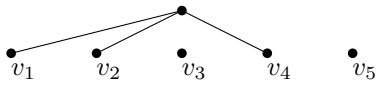


Figure 3 Example of an $\{E\}$-pattern: a graph satisfying $G \models \gamma_{(E,E,\neg E,E,\neg E)}(v_1, v_2, v_3, v_4, v_5)$.

We can now state a finitary version of Theorem 8 as follows. For the convenient use later on, we state the result not for single formulas φ but for Φ-types.

▶ **Theorem 9.** *For every monadically NIP class $\mathscr{C}$ of structures and finite set of formulas $\Phi(x, \bar{y})$ there exist integers n_0 and k, such that for every $\mathbb{A} \in \mathscr{C}$ the following holds. If $I = (\bar{a}_i)_{1 \leqslant i \leqslant n}$ is a Δ_k^Φ-indiscernible sequence of length $n \geqslant n_0$ in $\mathbb{A}$, and $b \in \mathbb{A}$, then there exists an exceptional index $\mathrm{ex}(b) \in [n]$ and Φ-types $\tau_-(x, \bar{y})$ and $\tau_+(x, \bar{y})$ such that*

$\mathbb{A} \models \tau_-(b, \bar{a}_i)$ *holds for all* $1 \leqslant i < \mathrm{ex}(b)$, *and*
$\mathbb{A} \models \tau_+(b, \bar{a}_i)$ *holds for all* $\mathrm{ex}(b) < i \leqslant n$.

Our proof uses different tools than [4]. It is combinatorial, fully constructive, and gives explicit bounds on n_0 and k as well as the required set of formulas Δ_k^Φ. One may alternatively finitize Theorem 8 via compactness, but then we do not obtain these crucial properties.

For the more restricted monadically stable classes we can give even stronger guarantees: for every element $b \in \mathbb{A}$, the types do not alternate and we have $\tau_- = \tau_+$.

▶ **Theorem 10.** *For every monadically stable class $\mathscr{C}$ of structures and finite set of formulas $\Phi(x, \bar{y})$ there exist integers n_0 and k, such that for every $\mathbb{A} \in \mathscr{C}$ the following holds. If $I = (\bar{a}_i)_{1 \leqslant i \leqslant n}$ is a Δ_k^Φ-indiscernible sequence of length $n \geqslant n_0$ in $\mathbb{A}$, and $b \in \mathbb{A}$, then there exists an exceptional index $\mathrm{ex}(b) \in [n]$ and a Φ-type τ, such that*

$\mathbb{A} \models \tau(b, \bar{a}_i)$ *for all* $i \in [n]$ *with* $i \neq \mathrm{ex}(b)$.

Building on Theorem 9, we give a full proof of Theorem 10. In order to give an intuition about the interaction of monadic stability and Δ_k^Φ-indiscernibility, we sketch a standalone proof here.

Proof sketch of Theorem 10. For simplicity, we will focus on the case were Φ is a singleton set $\{\varphi(x,\bar{y})\}$. Let k be the smallest number such that the bound for the pairing index of φ and $\neg\varphi$ on $\mathscr{C}$ is less than k. Such a bound exists since $\mathscr{C}$ is in particular monadically NIP. Assume towards a contradiction that there exists a sufficiently long Δ_k^φ-indiscernible sequence I in a structure $\mathbb{A} \in \mathscr{C}$ and an element $b \in \mathbb{A}$ that is φ-connected to at least two tuples in I and not φ-connected to at least two other tuples in I. By symmetry, we can assume that b is not φ-connected to the majority of I. We therefore find a length-k subsequence $\bar{a}_1, \dots, \bar{a}_k$ of I and two distinct indices $i_0, j_0 \in [k]$, such that b is φ-connected exactly to the i_0th and j_0th element of $\bar{a}_1, \dots, \bar{a}_k$. We say that the i_0th and j_0th element are *φ-paired* by b. Our goal is to derive a contradiction by finding also for every other pair i, j of indices an element that φ-pairs them, witnessing that φ has pairing index at least k in $\mathscr{C}$.

In stable classes, every sufficiently long Δ_k^φ-indiscernible sequence I is also *totally* Δ_k^φ-*indiscernible*: This means that every permutation I' of I is again Δ_k^φ-indiscernible, with the formulas from Δ_k^φ taking the same truth values on I' as on I. Intuitively speaking, if permuting two elements in an indiscernible sequence would change the truth value of a formula, then this formula orders these two elements and by indiscernibility also orders a large part of the sequence, contradicting stability. A proof of the statement for Ω-indiscernibles can be found in [36, Lemma 9.1.1].

The formula $\gamma(\bar{y}_1, \dots, \bar{y}_k) :=$ "$\bar{y}_{i_0}$ and $\bar{y}_{j_0}$ are φ-paired by some element" is contained in Δ_k^φ and holds on $\bar{a}_1, \dots, \bar{a}_k$, as witnessed by b. By total indiscernibility, we may permute $\bar{a}_1, \dots, \bar{a}_k$ and the formula still holds. By swapping a_{i_0} with a_i and a_{j_0} with a_j, we obtain for arbitrary i, j that "$\bar{a}_i$ and $\bar{a}_j$ are φ-paired by some element". This witnesses that φ has pairing index at least k, a contradiction. ◀

For the more general monadically NIP case, we cannot rely on total indiscernibility. Instead, we bound the alternation rank by pairing tuples located around alternation points, leading to a similar but more involved reasoning.

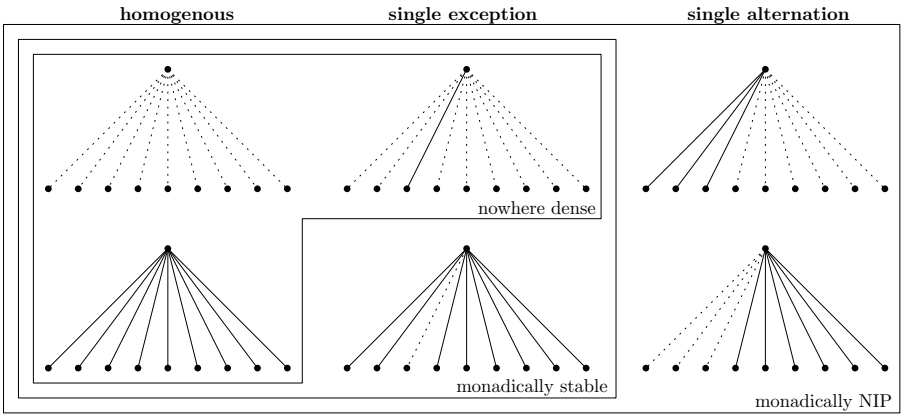


Figure 4 All types of neighborhoods a vertex (top) can have in indiscernible sequences (bottom) in different graph classes.

We conclude this section by mentioning that a behavior similar to Theorem 10 was already observed in [26] for the edge relation in nowhere dense classes. Considering only the edge relation, we depict the different possible behavior of vertices towards indiscernible sequences in different classes of graphs in Figure 4. In monadically NIP (or stable) classes, the same six (or four) patterns apply for connections with respect to any formula $\varphi(x, y)$.

3.2 Disjoint definable neighborhoods

In the previous section we have seen that in monadically stable and monadically NIP classes, every element is very homogeneously connected to all but at most one element of an indiscernible sequence. At a first glance, this exceptional behavior towards one element seems to be erratic and standing in the way of combinatorial and algorithmic applications. However, it turns out that it can be exploited to obtain additional structural properties.

The key observation is that elements that are "exceptionally connected" towards a single element of an indiscernible sequence, inherit some of the good properties of that sequence. We will give a simple example to demonstrate this idea. Let G be a graph containing certain red vertices R and blue vertices B, such that the edges between R and B describe a matching (see Figure 5, left).

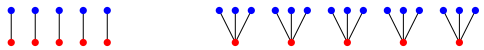


Figure 5 Examples of one-to-one and many-to-one connections in a graph. On the left: a matching. On the right: a star forest.

Given a formula $\varphi(x_1, \dots, x_q)$, define

$$\hat{\varphi}(x_1, \dots, x_q) := \exists b_1, \dots, b_q \in B. \varphi(b_1, \dots, b_q) \wedge \bigwedge_{i \in [q]} E(x_i, b_i).$$

Take a $\hat{\varphi}$-indiscernible sequence R' among R. Obviously, every vertex in the blue neighborhood B' of R' has one exceptional connection towards R', that is, towards its unique matching neighbor in R'. It is now easy to see that B' is a φ-indiscernible sequence in G: for every sequence of red vertices $a_1, \dots, a_q \in R'$ and their unique blue matching neighbors $b_1, \dots, b_q \in B'$ we have $G \models \hat{\varphi}(a_1, \dots, a_q)$ if and only if $G \models \varphi(b_1, \dots, b_q)$.

This example sketches how first-order definable one-to-one connections towards elements of an indiscernible sequence preserve indiscernibility. The more general case however is the many-to-one case, where we have a set of elements, each of which is exceptionally connected to a single element of an indiscernible sequence I, while allowing multiple elements to be exceptionally connected to the same element of I (think, for example, of R and B being the centers and leaves of a star-forest as depicted in Figure 5, right). Our notion of such many-to-one connections will be that of *disjoint α-neighborhoods*. Recall, that for a formula $\alpha(\bar{x}, y)$, the α-neighborhood $N_\alpha(\bar{a})$ of a tuple $\bar{a}$ is defined as the set of all b satisfying $\alpha(\bar{a}, b)$. We say a sequence J of $|\bar{x}|$-tuples has *disjoint α-neighborhoods* if $N_\alpha(\bar{a}_1) \cap N_\alpha(\bar{a}_2) = \varnothing$ for all distinct $\bar{a}_1, \bar{a}_2 \in J$.

As the main technical tool of this paper, we prove for monadically NIP classes that every sequence of disjoint α-neighborhoods contains a large subsequence that exerts strong control over its neighborhood. This lifts the strongly regular behaviour of idiscernible sequences to sequences of disjoint α-neighborhoods. The main result of this section, Theorem 11, is the backbone of our flip-flatness proof and states the following. In every large sequence of disjoint α-neighborhoods, we will find a still-large subsequence such that the φ-connections

of every element a towards all α-neighborhoods in the subsequence can be described by a bounded set of sample elements in the following sense. After possibly omitting one exceptional neighborhood at index ex(a) depending on a, the φ-connections are completely homogeneous (in the stable case) or alternate at most once (in the NIP case). A related (but orthogonal) result was also proved in [38, Lemma 64] using tools from model theory.

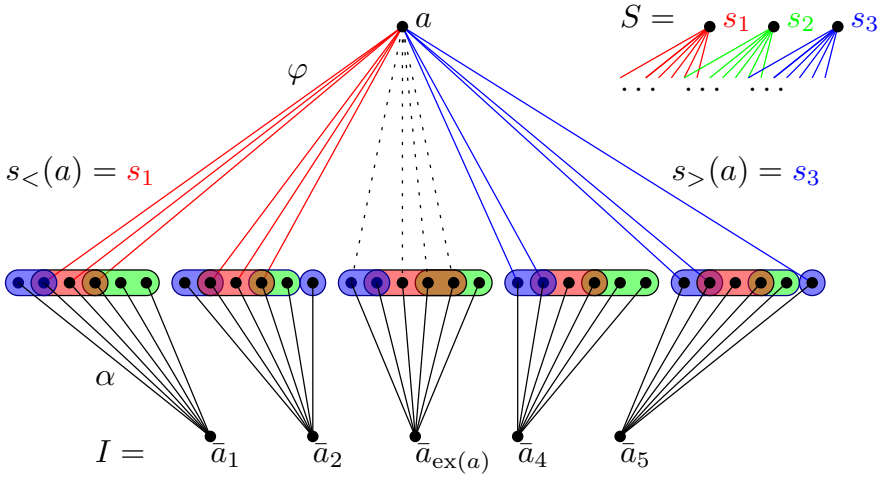


Figure 6 A visualization of Theorem 11. On the bottom: a sequence I with disjoint α-neighborhoods. On the top right: a small set of sample elements S. The φ-neighborhoods of the elements in S in the α-neighborhood of I are colored accordingly. Note that their φ-neighborhoods can overlap. The φ-neighborhood of a in the α-neighborhood of I is equal to the φ-neighborhood of $s_<(a) = s_1$ before the exceptional index $\mathrm{ex}(a)$. After it, it is equal to the φ-neighborhood of $s_>(a) = s_3$. For the α-neighborhood of $\bar{a}_{\mathrm{ex}(a)}$ we make no claims.

▶ **Theorem 11.** *For every monadically NIP class of structures $\mathscr{C}$, and formulas $\varphi(x, y)$ and $\alpha(\bar{x}, y)$, there exists a function $N : \mathbb{N} \to \mathbb{N}$ and an integer k such that for every $m \in \mathbb{N}$ every structure $\mathbb{A} \in \mathscr{C}$ and every finite sequence of tuples $J \subseteq \mathbb{A}^{|\bar{x}|}$ of length $N(m)$ whose α-neighborhoods are disjoint the following holds. There exists a subsequence $I = (\bar{a}_i)_{1 \leqslant i \leqslant m} \subseteq J$ of length m and a set $S \subseteq \mathbb{A}$ of at most k sample elements such that for every element $a \in \mathbb{A}$ there exists an exceptional index $\mathrm{ex}(a) \in [m]$ and a pair $s_<(a), s_>(a) \in S$ such that*

$$\text{for all } 1 \leqslant i < \mathrm{ex(a)} : N_\alpha(\bar{a}_i) \cap N_\varphi(a) = N_\alpha(\bar{a}_i) \cap N_\varphi(s_<(a)), \text{ and}$$
$$\text{for all } \mathrm{ex(a)} < i \leqslant m : N_\alpha(\bar{a}_i) \cap N_\varphi(a) = N_\alpha(\bar{a}_i) \cap N_\varphi(s_>(a)).$$

If $a \in N_\alpha(\bar{a}_i)$ for some $\bar{a}_i \in I$, then $i = \mathrm{ex}(a)$.

If $\mathscr{C}$ is monadically stable, then $s_<(a) = s_>(a)$ for every $a \in \mathbb{A}$.

A visualization of the NIP case of Theorem 11 is provided in Figure 6. An important ingredient of the proof is the following Ramsey-type result due to Ding et al. [12, Corollary 2.4]. We say a bipartite graph with sides $a_1, \ldots, a_\ell$ and $b_1, \ldots, b_\ell$ forms

- a *matching* of order ℓ if a_i and b_j are adjacent if and only if $i = j$ for all $i, j \in [\ell]$,
- a *co-matching* of order ℓ if a_i and b_j are adjacent if and only if $i \neq j$ for all $i, j \in [\ell]$,
- a *ladder* of order ℓ if a_i and b_j are adjacent if and only if $i \leqslant j$ for all $i, j \in [\ell]$.

We call two distinct vertices u and v *twins* in a graph G they have the same neighborhood with regard to the edge relation, i.e. $N_E^G(u) \setminus \{v\} = N_E^G(v) \setminus \{u\}$.

▶ **Theorem 12** ([12, Corollary 2.4]). *There exists a function $Q : \mathbb{N} \to \mathbb{N}$ such that for every $\ell \in \mathbb{N}$ and for every bipartite graph $G = (L, R, E)$ without twins, where L has size at least $Q(\ell)$, contains a matching, co-matching, or ladder of order ℓ as an induced subgraph.*

Equipped with Theorem 10 from the previous subsection and the above Theorem 12, we will now give a proof sketch of the monadically stable case of Theorem 11.

Proof sketch of Theorem 11 (monadically stable case). We will build the sequence I by inductively extracting indiscernible subsequences I_i of J and growing a set of sample elements $S_i = \{s_1, \dots, s_i\}$. During the induction, we maintain the following invariant: every two distinct sample elements from S_i have pairwise different φ-neighborhoods in the α-neighborhood of every tuple of I_i. We start with $I_0 = J$ and $S_0 = \varnothing$ where this trivially holds. Let us now describe the inductive construction of I_{i+1} and S_{i+1}. We first add a constant symbol for every element in S_i to the signature of $\mathbb{A}$. This extended signature will allow us to express for all $j \in [i]$ the formula $\psi_j(x, \bar{y})$ stating that x has the same φ-neighborhood as s_j in the α-neighborhood of $\bar{y}$, i.e., $N_\alpha(\bar{y}) \cap N_\varphi(x) = N_\alpha(\bar{y}) \cap N_\varphi(s_j)$. We collect all these formulas into the set $\Phi = \{\psi_j \ : \ j \in [i]\}$ and build I_{i+1} by extracting a Δ_k^Φ-indiscernible sequence from I_i, for the appropriate value of k given by Theorem 10. Now by Theorem 10, for every element $b \in \mathbb{A}$, there exists a Φ-type $\tau(x, \bar{y})$ such that b is τ-connected to all but at most one tuple from I_{i+1}. Since the Φ-types capture information about the connections relative to vertices in S_i, one of the following two cases applies for every $b \in \mathbb{A}$.

1. The φ-neighborhood of b is different to the φ-neighborhood of every element of S_i in the α-neighborhood around all but at most one tuple of I_{i+1}.
2. For some $j \in [i]$, the φ-neighborhood of b is equal to the φ neighborhood of s_j in the α-neighborhood around all but at most one tuple of I_{i+1} (whose index will $\mathrm{ex}(b)$).

If the first case applies for some $b \in \mathbb{A}$, then we can build S_{i+1} by setting $s_{i+1} := b$, which satisfies our invariant (after we possibly drop one more element from I_{i+1}). If the second case applies for every $b \in \mathbb{A}$, every element is represented by S_i, and the construction stops with $I := I_{i+1}$ and $S := S_i$.

It now remains to show that the construction stops after less than k steps for some k depending only on $\mathscr{C}$, φ, and α. To this end, we show that for every $\ell \in \mathbb{N}$, there exists $k \in \mathbb{N}$ with the following property. If there exists I_k and S_k satisfying the invariant of our construction, then we can derive a formula ψ from φ and α which has pairing index at least ℓ in an expansion of $\mathbb{A}$ with a constant number of colors (in the full proof we use 3 colors). As $\mathscr{C}$ is monadically NIP, the pairing index of ψ is bounded, which also yields a bound for k.

Assume we are given I_k and S_k, such that all the elements from S_k have different φ-neighborhoods, in the α-neighborhoods around the tuples of I_k. By choosing k large enough, repeated application of Theorem 12 and the pigeonhole principle, we find a subset $S' = \{s_1, \dots, s_{\ell^2}\}$ of S_k and a subsequence $I' = \{\bar{a}_1, \dots, \bar{a}_\ell\}$ of I_k with the following property. For each $j \in [\ell]$, there exists a subset N'_j of the α-neighborhood of $\bar{a}_j$, such that the φ-connections from S' to the N'_j all form a matching, all form a co-matching, or all form a ladder. Assume the φ-connections form a matching. Let us now show that the formula

$$\psi(x, \bar{y}) := \exists z \in N_\varphi(x) \cap N_\alpha(\bar{y}).\ R(z)$$

has pairing index at least ℓ in the class $\mathscr{C}$ extended with an additional unary predicate R. The situation is depicted in Figure 7. The formulas φ and α interpret a large 1-subdivided biclique in $\mathbb{A}$ as follows. The tuples from I' and the elements from S' form the principle vertices. The subdivision vertices are formed by $\bigcup_{i \in [\ell]} N'_i$. Due to the assumed matchings, each subdivision element has exactly one incoming φ-connection from S'. Due to the tuples

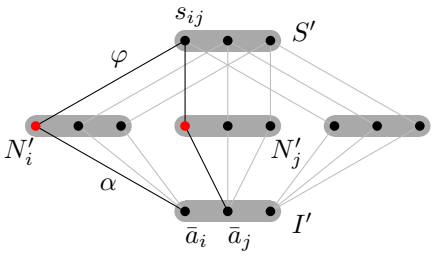


Figure 7 Pairing the elements of I'.

in I' having disjoint α-neighborhoods, each subdivision element has exactly one incoming α-connection from the elements of I'. Since I' has size ℓ and S' has size ℓ^2 we can assign to every pair $\bar{a}_i, \bar{a}_j \in I'$ a unique element $s_{ij} \in S'$. By marking with the predicate R the two unique subdivision elements from $N_\varphi(s_{ij}) \cap N'_i$ and $N_\varphi(s_{ij}) \cap N'_j$, we get that s_{ij} is ψ-connected only to $\bar{a}_i$ and $\bar{a}_j$ among I'. As desired, S' and I' witness that ψ has pairing index at least ℓ.

In case the φ-connections form a co-matching or a ladder, we can replace φ with a derived formula φ', such that the φ'-connections form a matching in a coloring of $\mathbb{A}$, which completes the proof sketch for the size bound of S.

Lastly, in order to ensure that $a \in N_\alpha(\bar{a}_i)$ implies $i = \mathrm{ex}(a)$, we can augment the formula $\psi_j(x, \bar{y})$ in the construction to return false whenever x is contained in $N_\alpha(\bar{y})$. This way, we enforce the single exception given by Theorem 10 to be on a_i. ◀

To prove the more general NIP case, we replace Theorem 10 with Theorem 9, which introduces the possibility for a vertex a to be sampled by two vertices $s_<(a)$ and $s_>(a)$, alternating around $\mathrm{ex}(a)$. We also have to slightly adapt the search for the next sample element s_j using an additional indiscernibility argument.

3.3 Flatness in monadically stable classes of graphs

In this section we use Theorem 11 to characterize monadically stable graph classes in terms of flip-flatness. We start with the forward direction, restated for convenience.

▶ **Theorem 4.** *Every monadically stable class $\mathscr{C}$ of graphs is flip-flat, where for every r, the function N_r is polynomial. Moreover, given an n-vertex graph $G \in \mathscr{C}$, $A \subseteq V(G)$, and $r \in \mathbb{N}$, we can compute a subset $B \subseteq A$ and a set of flips F that makes B distance-r independent in $G \oplus F$ in time $\mathcal{O}(f_{\mathscr{C}}(r) \cdot n^3)$ for some function $f_{\mathscr{C}}$.*

Partial proof. Let $\mathscr{C}$ be a monadically stable class of graphs and $r \in \mathbb{N}$. We want to show that in every graph $G \in \mathscr{C}$, in every set $A \subseteq V(G)$ of size $N_r(m)$ we find a subset $B \subseteq A$ of size at least m and a set F of at most s_r flips such that B is distance-r independent in $G \oplus F$. We will first inductively describe how to obtain the set of vertices B and the set of flips F and bound the runtime and values for N_r and s_r later. In the base case we have $r = 0$. We can pick $B := A$ and $F := \varnothing$ and there is nothing to show.

In the inductive case we assume the result is proved for r and extend it to $r + 1$. Let $r = 2i + p$ for some $i \in \mathbb{N}$ and parity $p \in \{0, 1\}$. Apply the induction hypothesis to obtain s_r flips F_r and a set A_r that is distance-r independent in $G_r := G \oplus F_r$. Note that for every fixed number t of flips, since flips are definable by coloring and a quantifier-free formula, the class $\mathscr{C}_t$ of all graphs obtainable from graphs of $\mathscr{C}$ by at most t flips is monadically stable by Lemma 5. Hence, G_r comes from the monadically stable class $\mathscr{C}_{s_r}$. Our goal is to

find a set of flips F'_{r+1} (of fixed finite size which will determine the number s_{r+1}) together with a set $A_{r+1} \subseteq A_r$, that is distance-$(r+1)$ independent in $G_{r+1} := G_r \oplus F'_{r+1}$. Then $G_r = G \oplus F_{r+1}$ with $F_{r+1} = F_r \cup F'_{r+1}$.

Since the elements in A_r have pairwise disjoint distance-i neighborhoods, we can apply Theorem 11 to A_r with $\varphi(x,y) = E(x,y)$ and $\alpha(x,y) = \mathrm{dist}_{\leqslant i}(x,y)$. Since we are in the monadically stable case, this yields a subset $A_{r+1} \subseteq A_r$ and a small set of sample vertices S such that for every vertex $a \in V(G)$ there exists $s(a) \in S$ and $\mathrm{ex}(a) \in A_{r+1}$ such that a has the same edge-neighborhood as $s(a)$ in the distance-i neighborhoods of all elements from $A_{r+1} \setminus \{\mathrm{ex}(a)\}$. We now do a case distinction depending on whether r is even or odd.

The odd case: $r = 2i+1$. For every $s \in S$ let C_s be the set containing every vertex a for which we have $s(a) = s$ and that is at distance at least $i+1$ from every vertex in A_{r+1}. Let D_s be the set containing every edge-neighbor of s that is at distance exactly i from one of the vertices in A_{r+1}. We now build the set of flips F'_{r+1} by adding for every sample vertex $s \in S$ the flip (C_s, D_s).

Let us now argue that A_{r+1} is distance-$(r+1)$ independent in $G_{r+1} := G_r$. We only flip edges in G_r between pairs of vertices a, b such that a has distance (in G_r) at least $i+1$ and b has distance exactly i to A_{r+1}. It follows that A_{r+1} remains distance-$(2i+1)$ independent in G_{r+1}. Assume towards a contradiction that there exists a path $P = (a, a_1, \ldots, a_i, u, b_i, \ldots, b_1, b)$ of length $r+1 = 2i+2$ between two vertices $a, b \in A_{r+1}$ in G_{r+1}. The distance between a and a_i (resp. b and b_i) is not affected by the flips. Only the connection between a_i (resp. b_i) and u can possibly be impacted. Additionally, note that $u \in C_{s(u)}$.

Since a and b are distinct we have that either $\mathrm{ex}(u) \neq a$ or $\mathrm{ex}(u) \neq b$. By symmetry we can assume the former case. As a_i is in the distance-i neighborhood of a, we have $G_r \models E(u, a_i) \iff G_r \models E(s(u), a_i)$. Observe that if $E(s(u), a_i)$ holds in G_r, then $a_i \in D_{s(u)}$ and therefore the edge (u, a_i) was removed by the flips and is not in G_{r+1}. Similarly, if $E(s(u), a_i)$ does not hold in G_r, then $a_i \notin D_{s(u)}$ and the edge (u, a_i) was not introduced by any flip. We can conclude that P is not a path in G_{r+1}, and that A_{r+1} is distance-$(r+1)$ independent in G_{r+1}.

The even case and runtime analysis. The even distance creates a symmetry that requires additional care to handle, but otherwise proceeds similarly to the odd case. The arguments concerning size bounds and runtimes can be found there as well. The runtime bound crucially uses the fact that we take Δ_k^Φ-indiscernible sequences only with respect to formulas Φ that can be evaluated in polynomial time. ◀

We have shown that for graph classes, monadic stability implies flip-flatness. We now show that the reverse holds as well. We will use the following statement, which is an immediate consequence of Gaifman's locality theorem [18]. For an introduction of the locality theorem see for example [23, Section 4.1].

▶ **Corollary 13** (of [18, Main Theorem]). *Let $\varphi(x,y)$ be a formula. Then there are numbers $r, t \in \mathbb{N}$, where r depends only on the quantifier-rank of φ and t depends only on the signature and quantifier-rank of φ, such that every (colored) graph G can be vertex-colored using t colors in such a way that for any two vertices $u, v \in V(G)$ with distance greater than r in G, $G \models \varphi(u,v)$ depends only on the colors of u and v. We call r the* Gaifman radius *of φ.*

▶ **Lemma 14.** *Every flip-flat class of graphs is monadically stable.*

Proof sketch of Lemma 14. Assume towards a contradiction that there exists a class $\mathscr{C}$ that is not monadically stable but flip-flat. Then there exists a formula $\sigma(x, y)$ and a graph $G \in \mathscr{C}$, such that σ defines an order on a large vertex set A in a coloring G^+. Let r be the Gaifman radius of σ. By flip-flatness there exists a (still large) subset B of A and a bounded size set of flips F, such that B is distance-r independent in $H := G^+ \oplus F$. Let H^+ be the graph where we have marked the flips F with colors in H. We can rewrite σ to a formula σ' of the same quantifier rank such that for all $u, v \in V(G)$ we have $H^+ \models \sigma'(u, v)$ if and only if $G^+ \models \sigma(u, v)$. In particular, σ' orders B in H^+. By Corollary 13 and the pidgeonhole principle, there must be two distinct vertices $u, v \in B$ such that $H^+ \models \sigma'(u, v)$ if and only if $H^+ \models \sigma'(v, u)$. However as σ' orders B, we also have $H^+ \models \sigma'(u, v)$ if and only if $H^+ \not\models \sigma'(v, u)$; a contradiction. ◀

From Theorem 4 and Lemma 14 we conclude the following.

▶ **Theorem 3.** *A class of graphs is monadically stable if and only if it is flip-flat.*

References

1 Algorithms, Logic and Structure Workshop in Warwick – Open Problem Session. URL: `https://warwick.ac.uk/fac/sci/maths/people/staff/daniel_kral/alglogstr/openproblems.pdf`, 2016. [Online; accessed 23-Jan-2023].

2 Hans Adler and Isolde Adler. Interpreting nowhere dense graph classes as a classical notion of model theory. *European Journal of Combinatorics*, 36:322–330, 2014.

3 John T Baldwin and Saharon Shelah. Second-order quantifiers and the complexity of theories. *Notre Dame Journal of Formal Logic*, 26(3):229–303, 1985.

4 Achim Blumensath. Simple monadic theories and indiscernibles. *Mathematical Logic Quarterly*, 57(1):65–86, 2011. `doi:10.1002/malq.200910121`.

5 Édouard Bonnet, Jan Dreier, Jakub Gajarský, Stephan Kreutzer, Nikolas Mählmann, Pierre Simon, and Szymon Torunczyk. Model checking on interpretations of classes of bounded local cliquewidth. In Christel Baier and Dana Fisman, editors, *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, pages 54:1–54:13. ACM, 2022. `doi:10.1145/3531130.3533367`.

6 Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, Pierre Simon, Stéphan Thomassé, and Szymon Toruńczyk. Twin-width IV: Ordered graphs and matrices. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 924–937, New York, NY, USA, 2022. Association for Computing Machinery. `doi:10.1145/3519935.3520037`.

7 Édouard Bonnet, Dibyayan Chakraborty, Eun Jung Kim, Noleen Köhler, Raul Lopes, and Stéphan Thomassé. Twin-width viii: delineation and win-wins, 2022. `doi:10.48550/arXiv.2204.00722`.

8 Samuel Braunfeld and Michael Laskowski. Characterizations of monadic NIP. *Transactions of the American Mathematical Society, Series B*, 8(30):948–970, 2021.

9 Bruno Courcelle and Sang-il Oum. Vertex-minors, monadic second-order logic, and a conjecture by seese. *Journal of Combinatorial Theory, Series B*, 97(1):91–126, 2007. `doi:10.1016/j.jctb.2006.04.003`.

10 Anuj Dawar. Homomorphism preservation on quasi-wide classes. *Journal of Computer and System Sciences*, 76(5):324–332, 2010.

11 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

12 Guoli Ding, Bogdan Oporowski, James G. Oxley, and Dirk Vertigan. Unavoidable minors of large 3-connected binary matroids. *J. Comb. Theory, Ser. B*, 66(2):334–360, 1996. `doi:10.1006/jctb.1996.0026`.

13 Jan Dreier, Jakub Gajarský, Sandra Kiefer, Michał Pilipczuk, and Szymon Toruńczyk. Tree-like decompositions for transductions of sparse graphs. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2022)*, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3531130.3533349.

14 Jan Dreier, Nikolas Mählmann, Amer E. Mouawad, Sebastian Siebertz, and Alexandre Vigny. Combinatorial and Algorithmic Aspects of Monadic Stability. In Sang Won Bae and Heejin Park, editors, *33rd International Symposium on Algorithms and Computation (ISAAC 2022)*, volume 248 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:17, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ISAAC.2022.11.

15 Jan Dreier, Nikolas Mählmann, Sebastian Siebertz, and Szymon Toruńczyk. Indiscernibles and flatness in monadically stable and monadically nip classes. *arXiv preprint*, 2022. arXiv:2206.13765.

16 Jan Dreier, Nikolas Mählmann, and Sebastian Siebertz. First-order model checking on structurally sparse graph classes, 2023. doi:10.48550/arXiv.2302.03527.

17 Andrzej Ehrenfeucht and Andrzej Mostowski. Models of axiomatic theories admitting automorphisms. *Fundamenta mathematicae*, 1(43):50–68, 1956.

18 Haim Gaifman. On local and non-local properties. In *Proceedings of the Herbrand Symposium*, volume 107 of *Stud. Logic Found. Math.*, pages 105–135. Elsevier, 1982.

19 Jakub Gajarskỳ, Stephan Kreutzer, Jaroslav Nešetřil, Patrice Ossona De Mendez, Michał Pilipczuk, Sebastian Siebertz, and Szymon Toruńczyk. First-order interpretations of bounded expansion classes. *ACM Transactions on Computational Logic (TOCL)*, 21(4):1–41, 2020.

20 Jakub Gajarský, Nikolas Mählmann, Rose McCarty, Pierre Ohlmann, Michał Pilipczuk, Wojciech Przybyszewski, Sebastian Siebertz, Marek Sokołowski, and Szymon Toruńczyk. Flipper games for monadically stable graph classes. *arXiv preprint*, 2023. arXiv:2301.13735.

21 Jakub Gajarský, Michał Pilipczuk, and Szymon Toruńczyk. Stable graphs of bounded twin-width. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '22, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3531130.3533356.

22 Colin Geniet and Stéphan Thomassé. First order logic and twin-width in tournaments and dense oriented graphs, 2022. doi:10.48550/arXiv.2207.07683.

23 Martin Grohe. Logic, graphs, and algorithms. *Logic and Automata*, 2:357–422, 2008.

24 Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding first-order properties of nowhere dense graphs. *Journal of the ACM (JACM)*, 64(3):1–32, 2017.

25 Wilfrid Hodges. *A Shorter Model Theory*. Cambridge University Press, 1997.

26 Stephan Kreutzer, Roman Rabinovich, and Sebastian Siebertz. Polynomial kernels and wideness properties of nowhere dense graph classes. *ACM Transactions on Algorithms (TALG)*, 15(2):1–19, 2018.

27 O-joung Kwon, Michał Pilipczuk, and Sebastian Siebertz. On low rank-width colorings. *European Journal of Combinatorics*, 83:103002, 2020.

28 M. Malliaris and S. Shelah. Regularity lemmas for stable graphs. *Transactions of the American Mathematical Society*, 366(3):1551–1585, 2014. URL: http://www.jstor.org/stable/23813167.

29 Jaroslav Nešetřil and Patrice Ossona De Mendez. On nowhere dense graphs. *European Journal of Combinatorics*, 32(4):600–617, 2011.

30 Jaroslav Nešetřil, Patrice Ossona de Mendez, Michał Pilipczuk, Roman Rabinovich, and Sebastian Siebertz. Rankwidth meets stability. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2014–2033. SIAM, 2021.

31 Jaroslav Nešetřil, Patrice Ossona de Mendez, and Sebastian Siebertz. Structural properties of the first-order transduction quasiorder. In *30th EACSL Annual Conference on Computer Science Logic (CSL 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

32 Jaroslav Nešetřil, Roman Rabinovich, Patrice Ossona de Mendez, and Sebastian Siebertz. Linear rankwidth meets stability. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1180–1199. SIAM, 2020.

33 Bruno Poizat. *A Course in Model Theory: An Introduction to Contemporary Mathematical Logic*. Springer New York, 2000.

34 Saharon Shelah. Monadic logic: Hanf numbers. In *Around classification theory of models*, pages 203–223. Springer, 1986.

35 Pierre Simon and Szymon Toruńczyk. Ordered graphs of bounded twin-width, 2021. doi:10.48550/arXiv.2102.06881.

36 Katrin Tent and Martin Ziegler. *A course in model theory*, volume 40 of *Lecture Notes in Logic*. Cambridge University Press, 2012.

37 Symon Toruńczyk. Flip-width: Cops and robber on dense graphs, 2023. arXiv:2302.00352.

38 Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, Pierre Simon, Stéphan Thomassé, and Szymon Toruńczyk. Twin-width iv: ordered graphs and matrices, 2021. arXiv:2102.03117.

Black-Box Testing Liveness Properties of Partially Observable Stochastic Systems

Javier Esparza
Technische Universiät München, Germany

Vincent P. Grande
RWTH Aachen University, Germany

Abstract

We study black-box testing for stochastic systems and arbitrary ω-regular specifications, explicitly including liveness properties. We are given a finite-state probabilistic system that we can only execute from the initial state. We have no information on the number of reachable states, or on the probabilities; further, we can only partially observe the states. The only action we can take is to restart the system. We design restart strategies guaranteeing that, if the specification is violated with non-zero probability, then w.p.1 the number of restarts is finite, and the infinite run executed after the last restart violates the specification. This improves on previous work that required full observability. We obtain asymptotically optimal upper bounds on the expected number of steps until the last restart. We conduct experiments on a number of benchmarks, and show that our strategies allow one to find violations in Markov chains much larger than the ones considered in previous work.

2012 ACM Subject Classification Theory of computation → Verification by model checking

Keywords and phrases Partially observable Markov chains, ω-regular properties, black-box testing

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.126

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `https://arxiv.org/abs/2303.03292`

Supplementary Material *Software (Source Code)*:
`https://git.rwth-aachen.de/netsci/restarting-markov-chains-experiments/`

Funding *Javier Esparza*: Partially funded by the DFG within Research Training Group 2428 CONVEY. *Vincent P. Grande*: Funded by the DFG (German Research Foundation) under Research Training Group 2236/2 UnRAVeL.

Acknowledgements The authors thank the anonymous reviewers for helpful feedback that improved the paper.

1 Introduction

Black-box testing is a fundamental analysis technique when the user does not have access to the design or the internal structure of a system [12, 15]. Since it only examines one run of the system at a time, it is computationally cheap, which makes it often the only applicable method for large systems.

We study the black-box testing problem for finite-state probabilistic systems and ω-regular specifications: Given an ω-regular specification, the problem consists of finding a run of the program that violates the property, assuming that such runs have nonzero probability.

Let us describe our assumptions in more detail. We do not have access to the code of the system or its internal structure, and we do not know any upper bound on the size of its state space. We can repeatedly execute the system, restarting it at any time. W.l.o.g. we assume that all runs of the system are infinite. We do not assume full observability of the states of the system, only that we can observe whether the atomic propositions of the property

are currently true or false. For example, if the property states that a system variable, say x, should have a positive value infinitely often, then we only assume that at each state we can observe the sign of x; letting Σ denote the set of possible observations, we have $\Sigma = \{+, -\}$, standing for a positive and a zero or negative value, respectively (in the rest of the introduction we shorten "zero or negative" to "negative"). Every system execution induces an observation, that is, an element of Σ^ω. The violations of the property are the ω-words $V \subseteq \Sigma^\omega$ containing only finitely many occurrences of $+$.

Our goal is to find a *strategy* that decides after each step whether to abort the current run and restart the system, or continue the execution of the current run. The strategy must ensure that some run that violates the property, that is, a run whose observation belongs to V, is eventually executed. The strategy decides depending on the observations made so far. Formally, given Σ and the set of actions $A = \{\mathsf{r}, \mathsf{c}\}$ (for "restart" and "continue") a strategy for V is a mapping from $(\Sigma \times A)^*\Sigma$, the sequence of observations and actions executed so far, to A, the next decision. Our goal is to find a strategy σ satisfying the following property:

> For every finite-state program P over Σ, if $V \subseteq \Sigma^\omega$ has positive probability and the runs of P are restarted according to σ, then w.p.1 the number of restarts is finite, and the observation of the run executed after the last restart belongs to V.

Observe that it is not clear that such strategies exist. They are easy to find for safety properties, where the fact that a run violates the property is witnessed by a *finite* prefix[1], but for liveness properties there is no such prefix in general. We show that these strategies exist for every ω-regular language V. Moreover, the strategies only need to maintain a number of counters that depends only on V, and not on the program. So in order to restart P according to σ one only needs logarithmic memory in the length of the current sequence.

▶ **Example 1.** To give a first idea of why these strategies also exist for liveness properties, consider the property over $\Sigma = \{+, -\}$ stating that a variable x should have a positive value only finitely often. The runs violating the property are those that visit $+$-states infinitely often. Our results show that the following strategy works in detecting a run violating the property (among others):

> After the n-th restart, repeatedly execute blocks of $2n$ steps. If at some point after executing the first block *the second half* of the concatenation of the blocks executed so far contains only negative states, then restart.

For example, assume there have been 4 restarts. Then the strategy repeatedly executes blocks of 8 steps. If after executing $1, 2, 3, \ldots$ of these blocks the last $4, 8, 16, \ldots$ states are negative, then the strategy restarts for the 5th time. If that is never the case, then there are only 4 restarts. Figure 1 shows a family of Markov chains for which naive strategies do not work, but the above strategy does: almost surely the number of restarts is finite and the run after the last restart visits the rightmost state infinitely often. Observe that for every $n \geq 0$ the family exhibits executions that visit $+$ states at least n times, and executions that visit a $+$ state at most once every n steps.

[1] One can choose for σ the strategy "after the n-th reset, execute n steps; if this finite execution is not a witness, restart, otherwise continue forever." Indeed, if the shortest witness has length k, then for every $n \geq k$, after the n-th restart the strategy executes a witness with positive probability, and so it eventually executes one w.p.1.

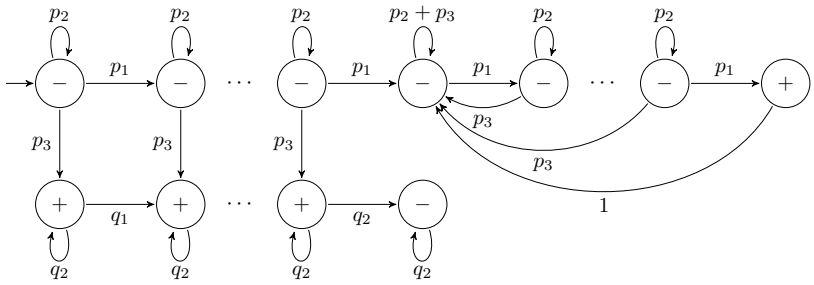


Figure 1 A family of partially observable Markov chains.

We also obtain asymptotically optimal upper bounds on the expected time until the last restart, that is, on the time until the execution of the run violating the property starts. The bounds depend on two parameters of the Markov chain associated to the program, called the *progress radius* and the *progress probability*. An important part of our contribution is the identification of these parameters as the key ones to analyze.

While our results are stated in an abstract setting, they easily translate into practice. In a practical scenario, on top of the values of the atomic propositions, we can also observe useful debugging information, like the values of some variables. We let a computer execute runs of the system for some fixed time t according to the strategy σ. If at time t we observe that the last restart took place a long time ago, then we stop testing and return the run executed since the last restart as candidate for a violation of the property. In the experimental section of our paper we use this scenario to detect errors in *population protocols*, a model of distributed computation, whose state space is too large to find them by other means.

Related work. There is a wealth of literature on black-box testing and black-box checking [12, 15], but the underlying models are not probabilistic and the methods require to know an upper bound on the number of states. Work on probabilistic model-checking assumes that (a model of) the system is known [2]. There are also works on black-box verification of probabilistic systems using statistical model checking of statistical hypothesis testing [22, 17, 18, 20, 21] (see also [11, 13] for surveys on statistical model checking). They consider a different problem: we focus on producing a counterexample run, while the goal of black-box verification is to accept or reject a hypothesis on the probability of the runs that satisfy a property. Our work is also related to the *runtime enforcement problem* [16, 3, 14, 7, 8], which also focus on identifying violations of a property. However, in these works either the setting is not probabilistic, or only a subset of the ω-regular properties close to saftey properties is considered. Finally, the paper closest to ours is [6], which considers the same problem, but for fully observable systems. In particular, in the worst case the strategies introduced in [6] require to store the full sequence of states visited along a run, and so they use linear memory in the length of the current sequence, instead of logarithmic memory, as is the case for our strategy.

Structure of the paper. The paper is organized as follows. Section 2 contains preliminaries. Section 3 introduces the black-box testing problem for arbitrary ω-regular languages with partial observability, and shows that it can be reduced to the problem for canonical languages called the Rabin languages. Section 4 presents our black-box strategies for the Rabin languages, and proves them correct. Section 5 obtains asymptotically optimal upper bounds on the time to the last restart. Section 6 reports some experimental results.

2 Preliminaries

Directed graphs. A directed graph is a pair $G = (V, E)$, where V is the set of nodes and $E \subseteq V \times V$ is the set of edges. A path (infinite path) of G is a finite (infinite) sequence $\pi = v_0, v_1, \ldots$ of nodes such that $(v_i, v_{i+1}) \in E$ for every $i = 0, 1, \ldots$. A path consisting only of one node is *empty*. Given two vertices $v, v' \in V$, the *distance* from v to v' is the length of a shortest path from v to v', and the distance from v to a set $V' \subseteq V$ is the minimum over all $v' \in V'$ of the distance from v to v'.

A graph G is strongly connected if for every two vertices v, v' there is a path leading from v to v'. A graph $G' = (V', E')$ is a subgraph of G, denoted $G' \preceq G$, if $V' \subseteq V$ and $E' \subseteq E \cap (V' \times V')$; we write $G' \prec G$ if $G' \preceq G$ and $G' \neq G$. A graph $G' \preceq G$ is a strongly connected component (SCC) of G if it is strongly connected and no graph G'' satisfying $G' \prec G'' \preceq G$ is strongly connected. An SCC $G' = (V', E')$ of G is a bottom SCC (BSCC) if $v \in V'$ and $(v, v') \in E$ imply $v' \in V'$.

Partially observable Markov chains. Fix a finite set Σ of *observations*. A *partially observable Markov chain* is a tuple $\mathcal{M} = (S, s_{in}, \Sigma, Obs, \mathbf{P})$, where

- Σ is a set of *observations*;
- S is a finite set of *states* and $s_{in} \in S$ is the *initial* state;
- $Obs\colon S \to \Sigma$ is an *observation function* that assigns to every state an observation; and
- $\mathbf{P} : S \times S \to [0, 1]$ is the transition probability matrix, such that for every $s \in S$ it holds $\sum_{s' \in S} \mathbf{P}(s, s') = 1$,

Intuitively, $Obs(s)$ models the information we can observe when the chain visits s. For example, if s is the state of a program, consisting of the value of the program counter and the values of all variables, $Obs(s)$ could be just the values of the program counter, or the values of a subset of public variables. The graph of $\mathcal{M}$ has S as set of nodes and $\{(s, s') \mid \mathbf{P}(s, s') > 0\}$ as set of edges. Abusing language, we also use $\mathcal{M}$ to denote the graph of $\mathcal{M}$. A *run* of $\mathcal{M}$ is an infinite path $\rho = s_0 s_1 \cdots$ of $\mathcal{M}$; we let $\rho[i]$ denote the state s_i. The sequence $Obs(\rho) \coloneqq Obs(s_0)\,Obs(s_1) \cdots$ is the *observation* associated to ρ. Each path π in $\mathcal{M}$ determines the set of runs $\mathsf{Cone}(\pi)$ consisting of all runs that start with π. To $\mathcal{M}$ we assign the probability space $(\mathsf{Runs}, \mathcal{F}, \mathbb{P})$, where Runs is the set of all runs in $\mathcal{M}$, $\mathcal{F}$ is the σ-algebra generated by all $\mathsf{Cone}(\pi)$, and $\mathbb{P}$ is the unique probability measure such that $\mathbb{P}[\mathsf{Cone}(s_0 s_1 \cdots s_k)] = \mu(s_0) \cdot \prod_{i=1}^{k} \mathbf{P}(s_{i-1}, s_i)$, where the empty product equals 1. The expected value of a random variable $f\colon \mathsf{Runs} \to \mathbb{R}$ is $\mathbb{E}[f] = \int_{\mathsf{Runs}} f \; d\mathbb{P}$.

Partially Observable Markov Decision Processes. A Σ-*observable Markov Decision Process* (Σ-MDP) is a tuple $\mathsf{M} = (S, s_{in}, \Sigma, Obs, A, \Delta)$, where S, s_{in}, Σ, Obs are as for Markov chains, A is a finite set of *actions*, and $\Delta\colon S \times A \to \mathcal{D}(S)$ is a *transition function* that for each state s and action $a \in A(s)$ yields a probability distribution over successor states. The probability of state s' in this distribution is denoted $\Delta(s, a, s')$.

Strategies. A *strategy* on Σ-MDPs with A as set of actions is a function $\sigma\colon (\Sigma \times A)^*\Sigma \to A$, which given a finite path $\pi = \ell_0 a_0\, \ell_1\, a_1 \ldots a_{n-1}\, \ell_n \in (\Sigma \times A)^*\Sigma$, yields the action $\sigma(\pi) \in A$ to be taken next. Notice that σ only "observes" $Obs(s)$, not the state s itself. Therefore, it can be applied to any Σ-MDP $\mathsf{M} = (S, s_{in}, \Sigma, Obs, A, \Delta)$, inducing the Markov chain $\mathsf{M}^\sigma = (S^\sigma, s_{in}, \Sigma, Obs, A, \mathbf{P}^\sigma)$ defined as follows: $S^\sigma = (S \times A)^* \times S$; and for every state $\pi \in S^\sigma$ of M^σ ending at a state $s \in S$ of M, the successor distribution is defined by $\mathbf{P}^\sigma(\pi, \pi\, a\, s') \coloneqq \Delta(s, a, s')$ if $\sigma(\pi) = a$ and 0 otherwise.

3 The black-box testing problem

Fix a set Σ of observations, and let r, c (for **r**estart and **c**ontinue) be two actions. We associate to a Σ-observable Markov chain $\mathcal{M} = (S, s_{in}, \Sigma, Obs, \mathbf{P})$ a *restart MDP* $\mathsf{M}_\mathsf{r} = (S, s_{in}, Obs, \{\mathsf{r}, \mathsf{c}\}, \Delta)$, where for every two states $s, s' \in S$ the transition function is given by: $\Delta(s, \mathsf{r}, s') = 1$ if $s' = s_{in}$ and 0 otherwise, and $\Delta(s, \boldsymbol{c}, s') = \mathbf{P}(s, s')$. Intuitively, at every state of M_r we have the choice between restarting the chain $\mathcal{M}$ or continuing.

We consider black-box strategies on Σ and $\{\mathsf{r}, \mathsf{c}\}$. Observe that if a run π of $\mathsf{M}_\mathsf{r}^\sigma$ contains finitely many occurrences of r, then the suffix of π after the last occurrence of r is a run of $\mathcal{M}$ (after dropping the occurrences of the continue action c). More precisely, if $\pi = \pi_0\pi'$, where π' is the longest suffix of π not containing r, then $\pi' = (\pi_0\, s_{in})\,(\pi_0\, s_{in}\, \mathsf{c}\, s_1)\,(\pi_0\, s_{in}\, \mathsf{c}\, s_1\, \mathsf{c}\, s_2)\ldots$, where $s_{in}s_1s_2\ldots$ is a run of $\mathcal{M}$. The sequence of observations of $s_{in}s_1s_2\ldots$ is an infinite word over Σ, called the *tail* of π; formally $tail(\pi) \coloneqq Obs(s_{in})Obs(s_1)Obs(s_2)\cdots$.

▶ **Definition 2** (Black-box testing strategies). *Let $L \subseteq \Sigma^\omega$ be an ω-regular language. A* black-box strategy *σ on Σ and $\{\mathsf{r}, \mathsf{c}\}$ is a* testing strategy for *L if it satisfies the following property: for every Σ-observable Markov chain $\mathcal{M}$, if $\Pr_\mathcal{M}(L) > 0$ then w.p.1 a run of $\mathsf{M}_\mathsf{r}^\sigma$ has a finite number of restarts, and its tail belongs to L. The* black-box testing problem *for L consists of finding a black-box testing strategy for L.*

We denote by $\#\mathsf{r}(\rho) \in \mathbb{N} \cup \{\infty\}$ the number of appearances of the restart action r in ρ. Intuitively, the language L models the set of potential violations of a given liveness specifications. If we sample any finite-state Σ-observable Markov chain $\mathcal{M}$ according to a testing strategy for L, then w.p.1 we eventually stop restarting, and the tail of the run is a violation, or there exist no violations.

3.1 Canonical black-box testing problems

Using standard automata-theoretic techniques, the black-box testing problem for an arbitrary ω-regular language L can be reduced to the black-box testing problem for a canonical language. For this, we need to introduce some standard notions of the theory of automata on infinite words.

A deterministic Rabin automaton (DRA) $\mathcal{A}$ over an alphabet Σ is a tuple $(Q, \Sigma, \gamma, q_0, Acc)$, where Q is a finite set of states, $\gamma\colon Q \times \Sigma \to Q$ is a transition function, $q_0 \in Q$ is the initial state, and $Acc \subseteq 2^Q \times 2^Q$ is the acceptance condition. The elements of Acc are called *Rabin pairs*. A word $w = a_0a_1a_2\ldots \in \Sigma^\omega$ is accepted by $\mathcal{A}$ if the unique run $q_0q_1q_2\ldots$ of $\mathcal{A}$ on w satisfies the following condition: there exists a Rabin pair $(E, F) \in Acc$ such that $a_i \in E$ for infinitely many $i \in \mathbb{N}$ and $a_i \in F$ for finitely many $i \in \mathbb{N}$. It is well known that DRAs recognize exactly the ω-regular languages (see e.g. [2]). The *Rabin index* of an ω-regular language L is the minimal number of Rabin pairs of the DRAs that recognize L.

▶ **Definition 3.** *Let $k \geq 1$, and let $M_k = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_k, \boldsymbol{f}_1, \ldots, \boldsymbol{f}_k\}$ be a set of* markers. *The* Rabin language *$\mathcal{R}_k \subseteq (2^{M_k})^\omega$ is the language of all words $w = \alpha_0\alpha_1\cdots \in (2^{M_k})^\omega$ satisfying the following property: there exists $1 \leq j \leq k$ such that $\boldsymbol{e}_j \in \alpha_i$ for infinitely many $i \geq 0$, and $\boldsymbol{f}_j \in \alpha_i$ for at most finitely many $i \geq 0$.*

We show that the black-box testing problem for languages of Rabin index k can be reduced to the black-box testing problem for $\mathcal{R}_k$.

▶ **Lemma 4.** *There is an algorithm that, given an ω-regular language $L \subseteq \Sigma^\omega$ of index k and given a testing strategy σ_k for $\mathcal{R}_k$, effectively constructs a testing strategy σ_L for L.*

Proof. (Sketch, full proof in the Appendix.) Let $\mathcal{A} = (Q, \Sigma, \gamma, q_0, Acc)$ be a DRA recognizing $L \subseteq \Sigma^\omega$ with accepting condition $Acc = \{(E_1, F_1), \dots, (E_k, F_k)\}$, i.e., Acc contains k Rabin pairs. Let σ_k be a black-box strategy for the Rabin language $\mathcal{R}_k$. We construct a black-box strategy σ_L for L.

Let $w = \ell_1 a_1 \ell_2 \cdots \ell_{n-1} a_n \ell_n \in (\Sigma \times \{\mathsf{r}, \mathsf{c}\})^* \Sigma$. We define the action $\sigma_L(w)$ as follows. Let $q_0 q_1 \dots q_n$ be the unique run of $\mathcal{A}$ on the word $\ell_1 \ell_2 \dots \ell_n \in \Sigma^*$. We then define $v = \ell'_1 a_1 \ell'_2 \cdots \ell'_{n-1} a_n \ell'_n \in (2^{M_k} \times \{\mathsf{r}, \mathsf{c}\})^* 2^{M_k}$ as the word given by: $\boldsymbol{e}_j \in \ell'_i$ iff $q_i \in E_j$, and $\boldsymbol{f}_j \in \ell'_i$ iff $q_i \in F_j$. (Intuitively, we mark with $\boldsymbol{e}_j$ the positions in the run at which the DRA visits E_j, and with $\boldsymbol{f}_j$ the positions at which the DRA visits F_j.) We set $\sigma_L(w) \coloneqq \sigma_k(v)$. We show in the Appendix that σ_L is a black-box strategy for L. ◀

4 Black-box strategies for Rabin languages

We describe a family of testing strategies for the Rabin languages $\{\mathcal{R}_k \mid k \geq 1\}$. In Section 4.1 we describe our strategy in detail. In Section 4.2 we introduce the progress radius and the progress probability, two parameters of a chain needed to prove correctness and necessary for quantitative analysis in Section 5. In Section 4.3 we formally prove that our strategy works.

4.1 The strategy

Let $\mathcal{M}$ be a Markov chain with observations in 2^{M_k}, and let $\pi = s_0 s_1 s_2 \cdots s_m$ be a finite path of $\mathcal{M}$. The *length* of π is m, and its last state, denoted $last(\pi)$, is s_m. The *second half* of π is the path $SecondHalf(\pi) \coloneqq s_{\lceil m/2 \rceil} \dots s_m$. The *concatenation* of π and a finite path $\rho = r_0 r_1 \cdots r_l$ of $\mathcal{M}$ such that $s_m = r_0$ is the path $\pi \odot \rho \coloneqq s_0 s_1 s_2 \cdots s_m r_1 \cdots r_l$. A path π is *i-good* if it has length 0 or there are markers $\boldsymbol{e}_i, \boldsymbol{f}_i \in M_k$ such that some state s of π satisfies $\boldsymbol{e}_i \in Obs(s)$ and no state s of π satisfies $\boldsymbol{f}_i \in Obs(s)$. Further, π is *good* if it is i-good for some $1 \leq i \leq k$.

The strategy $\mathfrak{S}[f]$, described in Figure 2, is parametrized by a function $f \colon \mathbb{N} \to \mathbb{N}$. The only requirement on f is $\limsup_{n \to \infty} f(n) = \infty$. In words, after the n-th restart the strategy

$n \coloneqq 0$ ▷ number of restarts
while true **do**
  $\pi \leftarrow s_{in}$ ▷ initial state of the chain
  while $SecondHalf(\pi)$ is good **do**
    sample path ρ from state $last(\pi)$
    of length $2 \cdot f(n)$ ▷ even length for convenience
    $\pi \leftarrow \pi \odot \rho$
  end while ▷ restart
  $n \leftarrow n + 1$
end while

Figure 2 Strategy $\mathfrak{S}[f]$ for the Rabin language $\mathcal{R}_k$ and a function $f \colon \mathbb{N} \to \mathbb{N}$.

keeps sampling in blocks of $2 \cdot f(n)$ steps until the *second half* of the complete path sampled so far is bad, in which case it restarts. For example, after the n-th restart the strategy samples a block $\pi_0 = \pi_{01} \odot \pi_{02}$, where $|\pi_{01}| = |\pi_{02}| = f(n)$, and checks whether π_{02} is good; if not, it restarts, otherwise it samples a block $\pi_1 = \pi_{11} \odot \pi_{12}$ starting from $last(\pi_{02})$, and checks whether $\pi_{11} \odot \pi_{12}$ is good; if not, it restarts; if so it samples a block $\pi_2 = \pi_{21} \odot \pi_{22}$ starting from $last(\pi_{12})$, and checks whether $\pi_{12} \odot \pi_{21} \odot \pi_{22}$ is good, etc. Intuitively, the growth of f controls how the strategy prioritizes deep runs into the chain over quick restarts while the number of restarts increases.

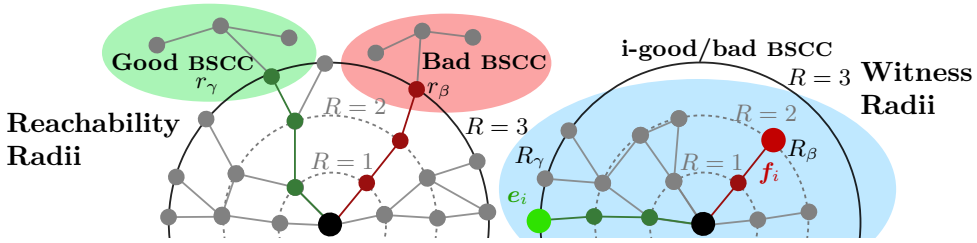


Figure 3 *Left:* Intuitively, r_γ and r_β denote an upper bound of how hard it is to reach a BSCC. *Right:* R_γ and R_β measure how hard it is to reach a state with label $\boldsymbol{e}_i/\boldsymbol{f}_i$ inside the BSCCs.

In the rest of the paper we prove that our strategy is correct, and obtain optimal upper bounds on the number of steps to the last reset. These bounds are given in terms of two parameters of the chain: the progress radius and the progress probability. We introduce the parameters in section 4.2.

4.2 Progress radius and progress probability

We define the notion of progress radius and progress probability for a Markov chain $\mathcal{M}$ with 2^{M_k} as set of observations and such that $\Pr(\mathcal{R}_k) > 0$. Intuitively, the progress radius is the smallest number of steps such that, for any state of the chain, conducting only this number of steps one can "make progress" toward producing a good run or a bad run. The progress probability gives a lower bound for the probability of the paths that make progress.

We define the notions only for the case $k = 1$, which already contains all the important features. The definition for arbitrary k is more technical, and is given in the Appendix.

Good runs and good BSCCs. We extend the definition of good paths to good runs and good BSCCs of a Markov chain. A run $\rho = s_0 s_1 s_2 \dots$ is *good* if $\boldsymbol{e}_1$ appears infinitely often in ρ and $\boldsymbol{f}_1$ finitely often, and *bad* otherwise. So a run ρ is good iff there exists a decomposition of ρ into an infinite concatenation $\rho \coloneqq \pi_0 \odot \pi_1 \odot \pi_2 \odot \cdots$ of non-empty paths such that $\pi_1, \pi_2, \dots$ are good. We let P_{good} denote the probability of the good runs of $\mathcal{M}$.

A BSCC of $\mathcal{M}$ is *good* if it contains at least one state labeled by $\boldsymbol{e}_1$ and no state labeled by $\boldsymbol{f}_1$, and bad otherwise. It is well-known that the runs of any finite-state Markov chain reach a BSCC and visit all its states infinitely often w.p.1 [2, Thm. 10.27]. It follows that good (resp. bad) runs eventually reach a good (resp. bad) BSCC w.p.1.

Progress radius. Intuitively, the progress radius R_{m} is the smallest number of steps such that, for any state s, by conducting R_{m} steps one can "make progress" toward producing a good run – by reaching a good BSCC or, if already in one, by reaching a state with observation $\boldsymbol{e}_1$ – or a bad run.

▶ **Definition 5** (Good-reachability and good-witness radii). *Let B_γ be the set of states of $\mathcal{M}$ that belong to good BSCCs and let S_γ be the set of states from which it is possible to reach B_γ, and let $s \in S_\gamma$. A non-empty path π starting at s is a* good progress path *if*

- *$s \in S_\gamma \setminus B_\gamma$, and π ends at a state of B_γ; or*
- *$s \in B_\gamma$, and π ends at a state with observation $\boldsymbol{e}_1$.*

The good-reachability radius *r_γ is the maximum, taken over every $s \in S_\gamma \setminus B_\gamma$, of the length of a shortest progress path for s. The* good-witness radius *R_γ is the same maximum, but taken over every $s \in B_\gamma$.*

The bad-reachability and bad-witness radii, denoted r_β and R_β are defined analogously. Only the notion of progress path of a state $s \in B_\beta$ needs to be adapted. Loosely speaking, a bad BSCC either contains no states with observation $\boldsymbol{e}_1$, or it contains some state with observation $\boldsymbol{f}_1$. Accordingly, if no state of the BSCC of s has observation $\boldsymbol{e}_1$, then any non-empty path starting at s is a progress path, and otherwise a progress path of s is a non-empty path starting at s and ending at a state with observation $\boldsymbol{f}_1$. We illustrate the definition of the reachability and witness radii in Figure 3. We leave r_β, R_β, p_β, and P_β undefined if the chain does not contain a bad BSCC, and hence runs are good w.p.1.

▶ **Definition 6** (Progress radius). *The* progress radius R_{m} *of* $\mathcal{M}$ *is the maximum of* r_γ, R_γ, r_β, *and* R_β.

Progress probability. From any state of the Markov chain it is possible to "make progress" by executing a progress path of length R_{m}. However, the probability of such paths varies from state to state. Intuitively, the progress probability gives a lower bound on the probability of making progress.

▶ **Definition 7.** *Let* B_γ *be the set of states of* $\mathcal{M}$ *that belong to good BSCCs, let* S_γ *be the set of states from which it is possible to reach* B_γ, *and let* $s \in S_\gamma$. *The* good-reachability probability p_γ *is the minimum, taken over every* $s \in S_\gamma \setminus B_\gamma$, *of the probability that a path with length* r_γ *starting at* s *contains a good progress path. The* good-witness probability P_γ *is the same mininum, but taken over every* $s \in B_\gamma$ *with paths of length* R_γ. *The corresponding bad probabilities are defined analogously. The* progress probability P_{m} *is the minimum of* $p_\gamma, P_\gamma, p_\beta, P_\beta$.

4.3 Correctness proof

We prove that the strategy $\mathfrak{S}[f]$ of section 4.1 is a valid testing strategy $\mathfrak{S}[f]$ for arbitrary Markov chains $\mathcal{M}$. First, we will give an upper bound on the probability that $\mathfrak{S}[f]$ restarts "incorrectly", i.e. at a state $s \in S_\gamma$ from which a good BSCC could still be reached.

▶ **Lemma 8.** *Let* $\mathcal{M}$ *be a Markov chain, and let* $\mathsf{M}_{\mathsf{r}}^{\mathfrak{S}[f]}$ *be its associated Markov chain with* $\mathfrak{S}[f]$ *as restart strategy. Let* NB_n *be the set of paths of* $\mathsf{M}_{\mathsf{r}}^{\mathfrak{S}[f]}$ *that have at least* $n-1$ *restarts and only visit states in* S_γ *after the* $(n-1)$*-th restart. We have:*

$$\Pr[\#\mathsf{r} \geq n \mid NB_n] \leq 3(1 - P_{\mathrm{m}})^{\lfloor f(n)/R_{\mathrm{m}} \rfloor - 1}$$

The technical proof of this lemma is in the Appendix. We give here the proof for a special case that illustrates most ideas. Consider the Markov chain with labels in $2^{\{\boldsymbol{e}_1, \boldsymbol{f}_1\}}$ at the top of Figure 4. The labeling function is $Obs(s_{\mathrm{goal}}) = \{\boldsymbol{e}_1\}$ and $Obs(s) = \emptyset$ for all other states, and $Obs(\rho) \in \mathcal{R}_1$ iff ρ visits s_{goal} infinitely often. The set S_γ contains all states because s_{goal} is reachable from every state. The only BSCC is $\mathcal{B} = \{s_1, s_{\mathrm{goal}}\}$, and it is a good BSCC. From the definitions of the parameters we obtain $r_\gamma = R_\gamma = 1$, $p_\gamma = p$ and $P_\gamma = q$. Further, since there are no bad BSCCs, r_β and R_β are undefined, and so $R_{\mathrm{m}} = 1$. So for this Markov chain Lemma 8 states $\Pr[\#\mathsf{r} \geq n \mid NB_n] \leq 3(1 - P_{\mathrm{m}})^{f(n)-1}$. Let us see why this is the case.

Let ρ be a run of $\mathsf{M}_{\mathsf{r}}^{\mathfrak{S}[f]}$ such that $\#\mathsf{r}(\rho) \geq n$, i.e., ρ has at least n restarts. Since S_γ contains all states, we have $\rho \in NB_n$ iff $\#\mathsf{r}(\rho) \geq n-1$. We consider three cases. In the definition of the cases we start counting steps immediately after the $(n-1)$-th restart, and denote by $\rho[a, b]$ the fragment of ρ that starts immediately before step a, and ends immediately after step b.

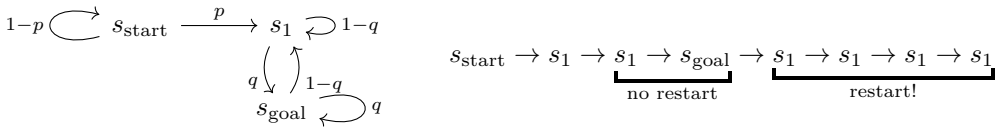


Figure 4 A Markov chain (left), and (a finite prefix of) one of its runs for $n = 2$ and $f(n) = n$ (right). After 2 steps, the run has reached a good BSCC $\mathcal{B}$. After 4 steps, σ checks whether to restart, but decides against it because of s_{goal} in step 4. After 8 steps it checks again, restarting this time. This restart is covered by the third case of the case distinction, with $k = 4$. The run must have visited s_{goal} in step 4, because otherwise the minimal k would be 3 or less.

(a) After $f(n)$ steps, ρ has not yet reached $\mathcal{B}$.
Then ρ has stayed in s_{start} for $f(n)$ consecutive steps, which, since $p = p_\gamma$, happens with probability at most $(1 - p_\gamma)^{f(n)}$.

(b) After $f(n)$ steps, ρ has already reached $\mathcal{B}$. Further, the n-th restart happens immediately after step $2f(n)$.
In this case, by the definition of the strategy, ρ does not visit s_{goal} during the interval $\rho[f(n) + 1, 2f(n)]$ (the second half of $[0, 2f(n)]$). So ρ stays in s_1 during the interval $\rho[f(n) + 1, 2f(n)]$ which, since ρ has already reached $\mathcal{B}$ by step $f(n)$, occurs with probability $(1 - P_\gamma)^{f(n)}$.

(c) After $f(n)$ steps, ρ has already reached $\mathcal{B}$. Further, the n-th restart does not happen before step $2f(n) + 1$.
Since the n-th restart happens at some point, and not before step $2f(n) + 1$, by the definition of the strategy there is a smallest number $k \geq f(n)$ such that ρ does not visit s_{goal} during the interval $\rho[k + 1, 2k]$. Because we assume that the n-th restart happens after step $2f(n)$, we even have $k > f(n)$. By the minimality of k, the run ρ does visit s_{goal} during the interval $\rho[k, 2k - 2]$. So ρ moves to s_{goal} at step k, and then stays in s_1 for k steps. The probability of the runs that eventually move to s_{goal} and then move to stay in s_1 for k steps is $P_\gamma(1 - P_\gamma)^k$.

Figure 4 shows at the bottom an example of a run, and how the stratgy handles it. Since (a)-(c) are mutually exclusive events, $\Pr[\#\mathsf{r} \geq n \mid NB_n]$ is bounded by the sum of their probabilities, where in case (c) we sum over all possible values of k. This yields:

$$Pr[\#\mathsf{r} \geq n \mid NB_n] \leq (1 - p_\gamma)^{f(n)} + (1 - P_\gamma)^{f(n)} + \sum_{k=f(n)+1}^{\infty} P_\gamma(1 - P_\gamma)^k \leq 3(1 - P_{\text{m}})^{f(n)}.$$

The proof for arbitrary Markov chains given in the Appendix has the same structure, and in particular the same split into three different events. Applying Lemma 8 we now easily obtain (see the Appendix for a detailed proof) an upper bound for the probability to restart an n-th time. Note that this bound captures the "correct" as well as the "incorrect" restarts:

▶ **Lemma 9** (Restarting probability). *Let $\mathcal{M}$ be a Markov chain, and let $\mathsf{M}_{\mathsf{r}}^{\mathfrak{S}[f]}$ be its associated Markov chain with $\mathfrak{S}[f]$ as restart strategy. The probability that a run restarts again after $n - 1$ restarts satisfies:*

$$\Pr[\#\mathsf{r} \geq n \mid \#\mathsf{r} \geq n - 1] \leq 1 - P_{good}\left(1 - 3(1 - P_{\text{m}})^{\lfloor f(n)/R_{\text{m}} \rfloor - 1}\right)$$

Proof. Let NB_n be the set of paths of $\mathsf{M}_{\mathsf{r}}^{\mathfrak{S}[f]}$ that have at least $n - 1$ restarts and only visit states in S_γ after the $(n - 1)$-th restart and $\overline{NB_n}$ its complement. We have

$$\Pr[\#\mathsf{r} \geq n \mid \#\mathsf{r} \geq n-1] = \Pr[\#\mathsf{r} \geq n \mid NB_n] \cdot \Pr[NB_n \mid \#\mathsf{r} \geq n-1] + \Pr[\#\mathsf{r} \geq n \mid \overline{NB}_n] \cdot \Pr[\overline{NB}_n \mid \#\mathsf{r} \geq n-1].$$

Applying Lemma 8 and $\Pr[\#\mathsf{r} \geq n \mid \overline{NB}_n] \leq 1$, we get

$$\Pr[\#\mathsf{r} \geq n \mid \#\mathsf{r} \geq n-1] \leq \left(3(1-P_{\mathrm{m}})^{\lfloor f(n)/R_{\mathrm{m}} \rfloor - 1}\right) \Pr[NB_n \mid \#\mathsf{r} \geq n-1] + \Pr[\overline{NB}_n \mid \#\mathsf{r} \geq n-1]$$

W.p.1, good runs of $\mathcal{M}$ only visit states of S_γ and so $\Pr[NB_n \mid \#\mathsf{r} \geq n-1] \geq P_{\mathrm{good}}$ and thus $\Pr[\overline{NB}_n \mid \#\mathsf{r} \geq n-1] \leq 1 - P_{\mathrm{good}}$, which completes the proof. ◀

Finally, we show that $\mathfrak{S}[f]$ is a correct testing strategy. Further, we show that the condition $\limsup_{n\to\infty} f(n) = \infty$ is not ony sufficient, but also necessary. The previous lemma gives an upper bound on the probability for a restart that, for increasing $f(n)$, drops below 1. If $f(n)$ is above that threshold for infinitely many n, it suffices to show that the strategy $\mathfrak{S}[f]$ restarts every bad run:

▶ **Theorem 10.** *$\mathfrak{S}[f]$ is a testing strategy for the Rabin language $\mathcal{R}_k$ iff the function f satisfies $\limsup_{n\to\infty} f(n) = \infty$.*

Proof.
($\Rightarrow$): We prove the contrapositive. If $\limsup_{n\to\infty} f(n) < \infty$ then there is a bound b such that $f(n) \leq b$ for every $n \geq 0$. Consider a Markov chain over 2^{M_1} consisting of a path of $2b+1$ states, with the last state leading to itself with probability 1; the last state is labeled with $\boldsymbol{e}_1$, and no state is labeled with $\boldsymbol{f}_1$. Then the chain has a unique run that goes from the initial to the last state of the path and stays there forever, and its observation is a word of $\mathcal{R}_1$; therefore, $\Pr(\mathcal{R}_1) = 1$. However, since $2f(n) \leq 2b+1$, $\mathfrak{S}[f]$ always restarts the chain before reaching the last state.
($\Leftarrow$): By the previous lemma, we can bound the restart probability after $n-1$ restarts by $1 - \left(1 - 3(1-P_{\mathrm{m}})^{\lfloor f(n)/R_{\mathrm{m}} \rfloor - 1}\right) P_{\mathrm{good}}$. Because $0 < P_{\mathrm{m}} \leq 1$ and and $P_{\mathrm{good}} > 0$, for large enough $f(n)$ this is smaller than $1 - P_{\mathrm{good}}/2 < 1$. Because of $\limsup_{n\to\infty} f(n) = \infty$, we have that the probability to restart the run another time is at most $1 - P_{\mathrm{good}}/2$ for infinitely many n, and hence the total number of restarts is finite with probability 1. A bad run would enter a bad BSCC B w.p.1 and would then go on to visit a set consisting of all the f_i corresponding to B infinitely often. Thus, $\mathfrak{S}[f]$ would restart this run and hence reached a good run when it does not restart. ◀

5 Quantitative analysis

The quality of a testing strategy is given by the expected number of steps until the last restart, because this is the overhead spent until a violation starts to be executed. As in [6], given a labeled Markov chain $\mathcal{M}$ and a testing strategy σ, we define the number of steps to the last restart as random variables over the Markov chain $\mathsf{M}_{\mathsf{r}}^{\sigma}$:

▶ **Definition 11** ($S(\rho)$ and $S_n(\rho)$)**.** *Let ρ be a run of $\mathsf{M}_{\mathsf{r}}^{\sigma}$. We define: $S(\rho)$ is equal to 0 if r does not occur in ρ; it is equal to the length of the longest prefix of ρ ending in r, if r occurs at least once and finitely often in ρ; and it is equal to ∞ otherwise. Further, for every $n \geq 1$ we define $S_n(\rho)$ to be equal to 0 if r occurs less than n times in ρ; and equal to the length of the segment between the $(n-1)$-th (or the beginning of ρ) and the n-th occurrence of r.*

In this section we investigate the dependence of $\mathbb{E}[S]$ on the function $f(n)$. A priori it is unclear whether $f(n)$ should grow fast or slow. Consider the case in which all BSCCs of the chain, good or bad, have size 1, and a run eventually reaches a good BSCC with probability p. In this case the strategy restarts the chain until a sample reaches a good BSCC for the first time. If $f(n)$ grows fast, then after a few restarts, say r, every subsequent run reaches a BSCC of the chain with large probability, and so the expected number of restarts is small, at most $r + (1/p)$. However, the number of steps executed during these few restarts is large, because $f(n)$ grows fast; indeed, only the run after the penultimate restart executes already at least $2f(r + (1/p) - 1)$ steps.

In a first step we show that $\mathbb{E}[S] = \infty$ holds for every function $f(n) \in 2^{\Omega(n)}$.

▶ **Proposition 12.** *Let $f \in 2^{\Omega(n)}$. Then there exists a Markov chain such that the testing strategy of Figure 2 satisfies $\mathbb{E}(S) = \infty$.*

Proof. Let f be in $2^{\Omega(n)}$. Then there exists some $k > 0$ such that we have $\limsup_{n\to\infty} f(n) \cdot (1/2)^{n/k} > 0$. Consider a Markov chain with $P_{\text{good}} = 1 - (1/2)^{1/k}$. Then we have $\mathbb{E}(S) = \sum_{n=0}^{\infty} \mathbb{E}(S_n \mid \#\mathsf{r} \geq n-1)P(\#\mathsf{r} \geq n-1)$. We have that $P(\#\mathsf{r} \geq n-1 \mid \#\mathsf{r} \geq n-2) \geq 1-P_{\text{good}}$ because only good runs will not be restarted. We also have that $\mathbb{E}(S_n \mid \#\mathsf{r} \geq n-1) \geq f(n)(1-P_{\text{good}})$ because of the same reason. Thus

$$\mathbb{E}[S] = \sum_{n=0}^{\infty} \mathbb{E}(S_n \mid \#\mathsf{r} \geq n-1)P(\#\mathsf{r} \geq n-1) \geq \sum_{n=0}^{\infty} f(n)(1-P_{\text{good}}) \cdot (1-P_{\text{good}})^n$$

and hence $\mathbb{E}[S] \geq \sum_{n=0}^{\infty} f(n)(1/2)^{n/k} = \infty$. ◀

It follows that (if we limit ourselves to monotonic functions, which is no restriction in practice), we only need to consider functions $f(n)$ satisfying $f(n) \in \omega(1) \cap 2^{o(n)}$. In the rest of the section we study the strategies corresponding to polynomial functions $f(n) = n^c$ for $c \in \mathbb{N}_+$, and obtain an upper bound as a function of the parameters $R_{\text{m}}/P_{\text{m}}$, P_{good}, and c. The study of subexponential but superpolynomial functions is beyond the scope of this paper.

5.1 Quantitative analysis of strategies with $f(n) = n^c$

We give an upper bound on $\mathbb{E}(S)$, the expected total number of steps before the last restart. Our starting point is Lemma 9, which bounds the probability to restart for the n-th time, if $(n-1)$ restarts have already happened. When the number n of restarts is small, the value of the right-hand-side is above 1, and so the bound is not useful. We first obtain a value X such that after X restarts the right-hand-side drops below 1.

▶ **Lemma 13.** *Let $X = \sqrt[c]{R_{\text{m}}\,(2 + \ln(1/6)/\ln(1-P_{\text{m}}))}$. For all $n \geq X$, we have*

$$\Pr[\#\mathsf{r} \geq n \mid \#\mathsf{r} \geq n-1] \leq 1 - P_{good}/2$$

when restarting according to $\mathfrak{S}[n \mapsto n^c]$.

Proof. Follows immediately from Lemma 9, the fact that the restart probability decreases with n, the definition of X, and some calculations. We recall the statement of Lemma 9:

$$\Pr[\#\mathsf{r} \geq n \mid \#\mathsf{r} \geq n-1] \leq 1 - P_{\text{good}}\left(1 - 3(1-P_{\text{m}})^{\lfloor f(n)/R_{\text{m}} \rfloor - 1}\right)$$

Plugging in an $n \geq X$ validates the claim. ◀

We now try to find a bound for $\mathbb{E}[S]$: By linearity of expectation, we have $\mathbb{E}[S] = \sum_{i=1}^{\infty} \mathbb{E}[S_n]$. We split the sum into two parts: for $n < X$, and for $n \geq X$. For $n < X$ we just approximate $\Pr[\#\mathsf{r} \geq n-1]$ by 1. For $n > X$ we can say more thanks to Lemma 13:

$$\begin{aligned}\Pr[\#\mathsf{r} \geq n-1] &= \Pr[\#\mathsf{r} \geq n-1 | \#\mathsf{r} \geq n-2] \cdots \Pr[\#\mathsf{r} \geq X+1 | \#\mathsf{r} \geq X] \cdot \Pr[\#\mathsf{r} \geq X] \\ &\leq \prod_{k=\lceil X \rceil}^{n} \Pr[\#\mathsf{r} \geq k | R \geq k-1] \leq (1 - P_{\text{good}}/2)^{n-X}\end{aligned}$$

This yields:

$$\begin{aligned}\mathbb{E}[S] &= \sum_{n=0}^{\infty} \mathbb{E}[S_n \mid \#\mathsf{r} \geq n-1] \Pr[\#\mathsf{r} \geq n-1] \\ &\leq \sum_{n=0}^{X} \mathbb{E}[S_n \mid \#\mathsf{r} \geq n-1] + \sum_{n=X}^{\infty} \mathbb{E}[S_n \mid \#\mathsf{r} \geq n-1] \cdot (1 - P_{\text{good}}/2)^{n-X} \qquad (1)\end{aligned}$$

It remains to bound the expected number of steps between two restarts $\mathbb{E}[S_n \mid \#\mathsf{r} \geq n-1]$, which is done in Lemma 14 below. The proof can be found in the Appendix. The proof first observes that the expected number of steps it takes to reach a good or a bad BSCC is r_γ/p_γ resp. r_β/p_β. Then we give a bound on the expected number of steps it takes to perform a progress path inside a bad BSCC for the first time, or to not perform a progress path inside a good BSCC for an entire second half of a run at some point after the $(n-1)$-st restart; the bound is also in terms of $R_{\text{m}}/P_{\text{m}}$ and $R_{\text{m}}/P_{\text{m}}(1-P_\gamma)$. The term $2f(n)$ comes from the fact that the strategy always executes at least $2f(n)$ steps. The term $2R_{\text{m}}$ is an artifact due to the "granularity" of the analysis, where we divide runs in blocks of R_{m} steps.

▶ **Lemma 14** (Expected number of steps in a fragment). *For the strategy $\mathfrak{S}[n \mapsto n^c]$ we have:*

$$\mathbb{E}[S_n \mid \#\mathsf{r} \geq n-1] \leq 2(R_{\text{m}} + f(n)) + 9\left(\frac{R_{\text{m}}}{P_{\text{m}}(1-P_\gamma)}\right). \qquad (1)$$

Plugging Lemma 14 into (1), we finally obtain (see the Appendix):

▶ **Theorem 15** (Expected number of total steps). *For the strategy $\mathfrak{S}[n \mapsto n^c]$ we have:*

$$\mathbb{E}[S] \in \mathbb{O}\left((c+1)! \cdot 2^c \cdot \left(\frac{R_{\text{m}}}{P_{\text{m}}}\right)^{1+1/c} + \frac{2^c(c+1)!}{P_{good}^{c+1}} + (c+1)!(2c)^{c+1}\right).$$

If we fix a value c, we obtain a much simpler statement:

▶ **Corollary 16.** *For a fixed c, the strategy $\mathfrak{S}[n \mapsto n^c]$ satisfies:*

$$\mathbb{E}[S] \in \mathbb{O}\left(\left(\frac{R_{\text{m}}}{P_{\text{m}}}\right)^{1+1/c} + \frac{1}{P_{good}^{c+1}}\right).$$

Thus the bound on the total number of steps depends on two quantities, $R_{\text{m}}/P_{\text{m}}$ and P_{good}. A small c favours the effect of $R_{\text{m}}/P_{\text{m}}$ on the bound, a larger c the effect of P_{good}. In Section 6 we will see that this closely matches the performance of the algorithms for different values of c on synthetic Markov chains and on Markov chains from the PRISM benchmark set.

5.2 Optimality of the Strategy $f(n) = n^c$

We will prove the following optimality guarantee for our strategies.

▶ **Theorem 17.** *For every $c \in \mathbb{N}_+$ there is a family of Markov chains such that our bound of Corollary 16 on $\mathfrak{S}[n \mapsto n^c]$ is asymptotically optimal, i.e., no other black-box testing strategy is in a better asymptotic complexity class.*

This proves two points: first, our bounds cannot be substantially improved. Second, one necessarily needs information on $\frac{R_m}{P_m}$ and P_{good} to pick an optimal value for c; without any information every value is equally good.

Proof. Consider the family of Markov chains at the top of Figure 5. We take an arbitrary $k > 1$ and set $M = k^{c-1}$ and $p = q = 1/k$. With this choice we have $P_{good} = P_m = 1/k$, and $R_m = k^{c-1}$. By Lemma 15, the strategy $\mathfrak{S}[n \mapsto n^c]$. satisfies $\mathbb{E}[S] \in \mathbb{O}((R_m/P_m)^{1+1/c} + (1/P_{good})^{c+1}) = \mathbb{O}(k^{c+1})$

We compare this with the optimal number of expected steps before the final restart. Since runs that visit s_{goal} at least once are good w.p.1, any optimal strategy stops restarting exactly after the visit to s_{goal}. We claim that every such strategy satisfies $\mathbb{E}[S] \geq R_m/(P_{good}P_m)(1 - P_{good})$. For this, we make four observations. First, the probability of a good run is P_{good}. Second, the expected number of steps of a good run until the first visit to s_{goal} is R_m/P_m. Third, the smallest number of steps required to distinguish a bad run, i.e. being in the left BSCC, from a good run is equal to R_m, because until R_m steps are executed, all states visited carry the same label. Hence, every strategy takes $R_m/(P_{good}P_m)$ steps on average before reaching the state s_{goal} for the first time. Fourth, on average $1/P_{good}$ tries are required to have one try result in a good run. Hence, on average at least $\frac{1/P_{good}-1}{1/P_{good}}$ of the $R_m/(P_{good}P_m)$ steps happen before the last restart. Since $\frac{1/P_{good}-1}{1/P_{good}} = (1 - P_{good})$, this proves the claim. Now $R_m/(P_{good}P_m)(1 - P_{good}) = k^{c+1} - k^c \in \Theta(k^{c+1})$ and we are done. ◀

6 Experiments

We report on experiments on three kinds of systems. First, we conduct experiments on two synthetic families of Markov Chains. Second, we repeat the experiments of [6] on models from the standard PRISM Benchmark Suite [10] using our black-box strategies. Finally, we conduct experiments on population protocols from the benchmark suite of the Peregrine tool [4, 5].

Synthetic Experiments. Consider the two (families of) labeled Markov chains at the top of Figure 5. The labels are a and b. In the top chain, state s_{goal} is labeled by a, all others by b. In the bottom chain, the states s_2 to s_M and s_{goal} are labeled by $\{a, s_{start}\}$ and s_{sink} by b. The language L is the set of words containing infinitely many occurrences of a. In the top chain at the initial state we go right or left with probability q and $(1 - q)$, respectively. Runs that go left are bad, and runs that go right are good w.p.1. It follows $P_{good} = q$, $R_m = M$, and $P_m = \min(p, q)$. In our experiments we fix $q = 1/2$. By controlling M and p, we obtain chains with different values of R_m and P_m for fixed $P_{good} = 1/2$. In the bottom chain, $R_\beta = R_\gamma = 1$, $R_m = r_\gamma = M$, $p_\gamma = p^M$, $p_\beta = (1 - p)$ and $P_m = \min(p^M, 1 - p)$ and $P_{good} = p^M$.

Recall that the bound obtained in the last section is $\mathbb{E}[S] \leq f(c)(R_m/P_m)^{1+1/c} + g(c)(1/P_{good})^{c+1}$ where $f(c)$ and $g(c)$ are fast-growing functions of c. If P_{good} and R_m/P_m are small, then $f(c)$ and $g(c)$ dominate the number of steps, and hence strategies with small c should perform better. The data confirms this prediction. Further, for fixed P_{good}, the

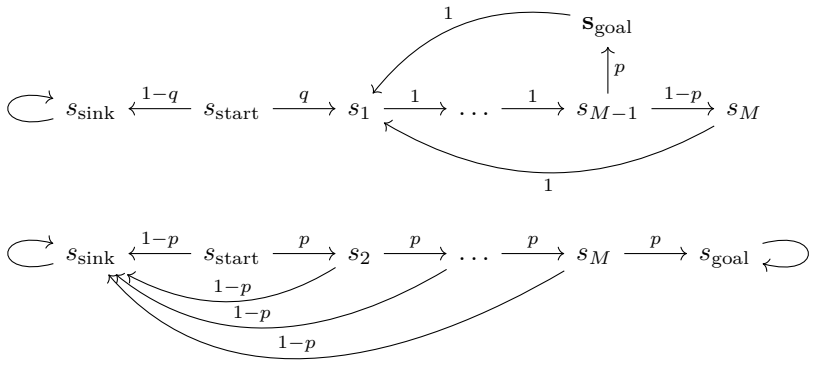


Figure 5 Two families of Markov chains. The initial state is s_{start}. The good runs are those that visit s_{goal} infinitely often. For the top chains, $P_{\text{good}} = q$, $R_{\text{m}} = M$, and $P_{\text{m}} = p$. For the bottom chains, $P_{\text{good}} = p^M$, $R_{\text{m}} = M$, $P_{\text{m}} = p^M$.

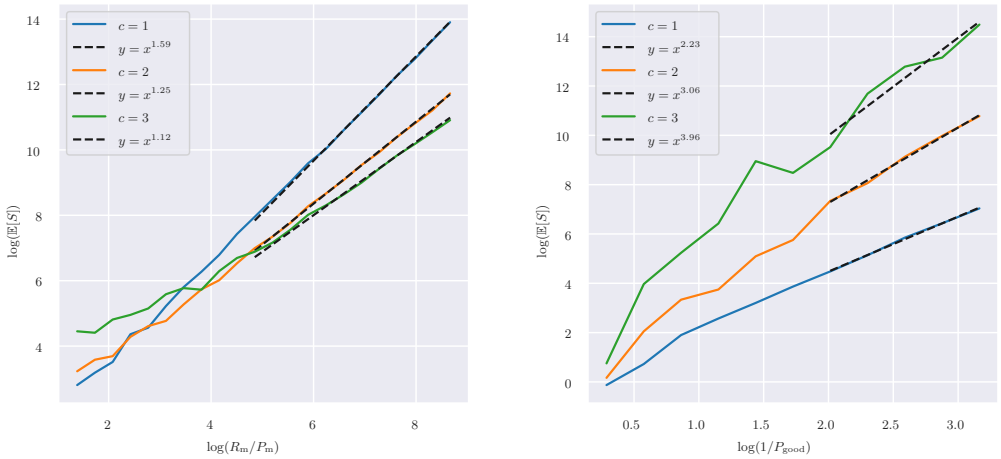


Figure 6 On the left, double-logarithmic plot of the expected total number of steps before the last restart $\mathbb{E}(S)$ for the chain at the top of Figure 5 as a function of $R_{\text{m}}/P_{\text{m}}$ for strategies (2) with $f(n) = n^c$ for varying c. On the right, same for the bottom chain as a function of $1/P_{\text{good}}$. The plots also show linear regressions. The leading exponent can be taken from the legend.

bound predicts $\mathbb{E}[S] \in O((R_{\text{m}}/P_{\text{m}})^{1+1/c})$, and so for growing $R_{\text{m}}/P_{\text{m}}$ strategies with large c should perform better. The left diagram confirms this. Also, the graphs become straight lines in the double logarithmic plot, confirming the predicted polynomial growth. Finally, for $R_{\text{m}}/P_{\text{m}}$ and $1/P_{\text{good}}$ growing roughly at the same speed as in the lower Markov chain, the bound predicts $\mathbb{E}[S] \in O(1/P_{\text{good}}^{c+1})$ for $c = 2, 3$ and $\mathbb{E}[S] \in O(M^2/P_{\text{good}}^{c+1})$ for $c = 1$, and hence for growing P_{good} and $R_{\text{m}}/P_{\text{m}}$, strategies with small c perform better. Again, the right diagram confirms this.

Experiments on the PRISM data set. We evaluate the performance of our black-box testing strategies for different values of c on discrete time Markov chain benchmarks from the PRISM Benchmark suite [10], and compare them with the strategies of [6] for fully observable systems. Table 1 shows the results. The properties checked are of the form $\mathbf{GF}$, $(\mathbf{GF} \to \mathbf{FG})$, or their negations. We add a gridworld example[2] denoted $\overline{\texttt{GW}}$, with larger values of the parameters, to increase the number of states to $\sim 5 \cdot 10^8$. When trying to construct the corresponding Markov chain, Storm experienced a timeout. Runs are sampled using the simulator of the Storm model checker [9] and the python extension Stormpy. We abort a run after 10^6 (Up to $3 \cdot 10^7$ for the gridworld examples gw, $\overline{\texttt{gw}}$, and $\overline{\texttt{GW}}$) steps without a restart. The probability of another restart is negligibly small.

The Cautious$_{10}$- and the Bold$_{0.1}$-strategy of [6] store the complete sequence of states observed, and so need linear memory in the length of the sample. Our strategies use at most a logarithmic amount of memory, at none or little cost in the number of steps to the last restart. Our strategies never timeout and, surprisingly, often require *fewer* steps than fully-observable ones. In particular, the strategies for fully observable systems cannot handle gridworlds, and only the bold strategy handles gridworld. One reason for this difference is our strategies' ability to adapt to the size of the chain automatically by increasing values of $f(n)$ as n grows. In two cases (nand and bluetooth) the fully observing strategies perform better by a factor of ~ 2 to ~ 3. In comparison to the improvement by a factor of ~ 50 in scale$_{10}$ and a factor of ~ 90 in gridworld of the newly presented black-box strategies over the whitebox strategies, this is negligible.

Table 1 Average number of steps before the final restart, averaged over 300 (100 for Herman and $\overline{\texttt{GW}}$) runs. Results for our strategies for $c = 1, 2, 3$, and the bold and cautious strategies of [6].

	nand	bluetooth	scale$_{10}$	crowds	herman	gw	$\overline{\texttt{gw}}$	$\overline{\texttt{GW}}$
# states	$7{\cdot}10^7$	143 291	121	$1{\cdot}10^7$	$5{\cdot}10^5$	309 327	309 327	$5{\cdot}10^8$
$c = 1$	31 246	4 428	116	**44**	**2**	486	171 219	8 082 659
$c = 2$	18 827	4 548	**75**	61	**1**	404	**152 127**	4 883 449
$c = 3$	32 777	7 615	179	99	**1**	**293**	579 896	**4 252 263**
Bold$_{0.1}$	10 583	4 637	14 528	199	**0**	TO	TO	
Cautious$_{10}$	**6 900**	**2 425**	3 670	101	TO	26 361	TO	

Table 2 Testing population protocols with the strategies $\mathfrak{S}[n \mapsto n^c]$. Experiments were run 100 times, averaging the number of steps to the last restart with a restart threshold of 250 for Average and Conquer (AvC) and 10 000 for the Majority Protocol.

	AvC$_{17,8}$(faulty)		Maj$_{\leq 12}$(faulty)		AvC$_{17,8}$		Maj$_{5,6}$	
$c = 1$	13 645	ce	872	ce	126 294	true	4 264 508	ge
$c = 2$	181 746	ce	4 763	ce	10 485 163	true	11 878 533	ge
Peregrine		TO		ce		TO		true

Experiments on population Protocols. Population protocols are consensus protocols in which a crowd of indistinguishable agents decide a property of their initial configuration by reaching a stable consensus [1, 4]. The specification states that for each initial configuration

[2] Unfortunately, the experimental setup of [6] cannot be applied to this example [19].

the agents eventually reach the right consensus (property holds/does not hold). We have tested our strategies on several protocols from the benchmark suite of Peregrine, the state-of-the-art model checker for population protocols [4, 5]. The first protocol of Table 2 is faulty, but Peregrine cannot prove it; our strategy finds initial configurations for which the protocol exhibits a fault. For the second protocol both our strategies and Peregrine find faulty configurations. The third protocol is correct; Peregrine fails to prove it, and our strategies correctly fail to find counterexamples. The last protocol is correct, but in expectation consensus is reached only after an exponential number of steps in the parameters; we complement the specification, and search for a run that achieves consensus. Thanks to the logarithmic memory requirements, our strategies can run deep into the Markov chain and find the run.

7 Conclusions

We have studied the problem of testing partially observable stochastic systems against ω-regular specifications in a black-box setting where testers can only restart the system, have no information on size or probabilities, and cannot observe the states of the system, only its outputs. We have shown that, despite these limitations, black-box testing strategies exist. We have obtained asymptotically optimal bounds on the number of steps to the last restart. Surprisingly, our strategies never require many more steps than the strategies for fully observable systems of [6], and often even less. Sometimes, the improvement is by a large factor (up to ~ 90 in our experiments) or the black-box strategies are able to solve instances where the strategies of [6] time out.

References

1 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Comput.*, 18(4):235–253, 2006.

2 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, Cambridge, Massachusetts, 2008.

3 David A. Basin, Vincent Jugé, Felix Klaedtke, and Eugen Zalinescu. Enforceable security policies revisited. *ACM Trans. Inf. Syst. Secur.*, 16(1):3:1–3:26, 2013.

4 Michael Blondin, Javier Esparza, and Stefan Jaax. Peregrine: A tool for the analysis of population protocols. In *CAV (1)*, volume 10981 of *Lecture Notes in Computer Science*, pages 604–611. Springer, 2018.

5 Javier Esparza, Martin Helfrich, Stefan Jaax, and Philipp J. Meyer. Peregrine 2.0: Explaining correctness of population protocols through stage graphs. In *ATVA*, volume 12302 of *Lecture Notes in Computer Science*, pages 550–556. Springer, 2020.

6 Javier Esparza, Stefan Kiefer, Jan Kretínský, and Maximilian Weininger. Enforcing ω-regular properties in markov chains by restarting. In *CONCUR*, volume 203 of *LIPIcs*, pages 5:1–5:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

7 Yliès Falcone, Laurent Mounier, Jean-Claude Fernandez, and Jean-Luc Richier. Runtime enforcement monitors: composition, synthesis, and enforcement abilities. *Formal Methods Syst. Des.*, 38(3):223–262, 2011.

8 Yliès Falcone and Srinivas Pinisetty. On the runtime enforcement of timed properties. In *RV*, volume 11757 of *Lecture Notes in Computer Science*, pages 48–69. Springer, 2019.

9 Christian Hensel, Sebastian Junges, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. The probabilistic model checker storm. *CoRR*, abs/2002.07080, 2020. `arXiv:2002.07080`.

10 Marta Z. Kwiatkowska, Gethin Norman, and David Parker. The PRISM benchmark suite. In *QEST*, pages 203–204. IEEE Computer Society, 2012.

11 Kim G. Larsen and Axel Legay. On the power of statistical model checking. In *ISoLA (2)*, volume 9953 of *Lecture Notes in Computer Science*, pages 843–862, 2016.

12 David Lee and Mihalis Yannakakis. Principles and methods of testing finite state machines-a survey. *Proc. IEEE*, 84(8):1090–1123, 1996.

13 Axel Legay, Benoît Delahaye, and Saddek Bensalem. Statistical model checking: An overview. In *RV*, volume 6418 of *Lecture Notes in Computer Science*, pages 122–135. Springer, 2010.

14 Jay Ligatti, Lujo Bauer, and David Walker. Run-time enforcement of nonsafety policies. *ACM Trans. Inf. Syst. Secur.*, 12(3):19:1–19:41, 2009.

15 Doron A. Peled, Moshe Y. Vardi, and Mihalis Yannakakis. Black box checking. *J. Autom. Lang. Comb.*, 7(2):225–246, 2002.

16 Fred B. Schneider. Enforceable security policies. *ACM Trans. Inf. Syst. Secur.*, 3(1):30–50, 2000.

17 Koushik Sen, Mahesh Viswanathan, and Gul Agha. Statistical model checking of black-box probabilistic systems. In *CAV*, pages 202–215, 2004. doi:10.1007/978-3-540-27813-9_16.

18 Koushik Sen, Mahesh Viswanathan, and Gul Agha. On statistical model checking of stochastic systems. In *CAV*, volume 3576 of *Lecture Notes in Computer Science*, pages 266–280. Springer, 2005.

19 Maximilian Weininger. Personal communication, 2022.

20 Håkan L. S. Younes. Probabilistic verification for "black-box" systems. In *CAV*, volume 3576 of *Lecture Notes in Computer Science*, pages 253–265. Springer, 2005.

21 Håkan L. S. Younes, Edmund M. Clarke, and Paolo Zuliani. Statistical verification of probabilistic properties with unbounded until. In *SBMF*, volume 6527 of *Lecture Notes in Computer Science*, pages 144–160. Springer, 2010.

22 Håkan L. S. Younes and Reid G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *CAV*, volume 2404 of *Lecture Notes in Computer Science*, pages 223–235. Springer, 2002.

A Notes

The appendix can be found in the extended version at https://arxiv.org/abs/2303.03292.

The Fine-Grained Complexity of Boolean Conjunctive Queries and Sum-Product Problems

Austen Z. Fan
Department of Computer Sciences, University of Wisconsin-Madison, WI, USA

Paraschos Koutris
Department of Computer Sciences, University of Wisconsin-Madison, WI, USA

Hangdong Zhao
Department of Computer Sciences, University of Wisconsin-Madison, WI, USA

Abstract

We study the fine-grained complexity of evaluating Boolean Conjunctive Queries and their generalization to sum-of-product problems over an arbitrary semiring. For these problems, we present a general *semiring-oblivious* reduction from the k-clique problem to any query structure (hypergraph). Our reduction uses the notion of *embedding* a graph to a hypergraph, first introduced by Marx [20]. As a consequence of our reduction, we can show tight conditional lower bounds for many classes of hypergraphs, including cycles, Loomis-Whitney joins, some bipartite graphs, and chordal graphs. These lower bounds have a dependence on what we call the *clique embedding power* of a hypergraph H, which we believe is a quantity of independent interest. We show that the clique embedding power is always less than the submodular width of the hypergraph, and present a decidable algorithm for computing it. We conclude with many open problems for future research.

2012 ACM Subject Classification Theory of computation → Database theory; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Fine-grained complexity, conjunctive queries, semiring-oblivious reduction

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.127

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `http://arxiv.org/abs/2304.14557` [9]

Funding NSF IIS-1910014.

1 Introduction

In a seminal paper, Marx proved the celebrated result that CSP($\mathcal{H}$) is fixed-parameter tractable (FPT) if and only if the hypergraph $\mathcal{H}$ has a bounded submodular width [20]. In the language of database theory, a Boolean Conjunctive Query (BCQ) can be identified as the problem of CSP($\mathcal{H}$) where $\mathcal{H}$ is the hypergraph associated with the query [11]. Thus, Marx's result implies that a class of Boolean Conjunctive Queries is FPT if and only if its submodular width is bounded above by some universal constant. Built on this result, Khamis, Ngo, and Suciu introduced in [17] the PANDA (Proof-Assisted eNtropic Degree-Aware) algorithm, which can evaluate a BCQ[1] in time $\tilde{O}(|I|^{\mathsf{subw}(H)})$, where $|I|$ is the input size and $\mathsf{subw}(\mathcal{H})$ is the submodular width of $\mathcal{H}$ (here $\tilde{O}$ hides polylogarithmic factors). Remarkably, the running time of PANDA achieves the best known running time of *combinatorial algorithm*[2] for *all* BCQs. It is thus an important open question whether there exists a faster combinatorial algorithm than PANDA for some Boolean CQ.

[1] Technically, the PANDA algorithm works for Boolean or full CQs.
[2] Informally speaking, this requires the algorithm does not leverage fast matrix multiplication techniques.

To show that large submodular width implies not being FPT, Marx introduced the notion of an *embedding*, which essentially describes a reduction from one CSP problem to another. Our key insight in this work is that we can apply the notion of an embedding to measure how well *cliques* of different sizes can be embedded to a hypergraph $\mathcal{H}$. By taking the supremum over all possible clique sizes, we arrive at the definition of *clique embedding power*, denoted $\mathsf{emb}(\mathcal{H})$. The use of cliques as the starting problem means that we can use popular lower bound conjectures in fine-grained complexity (the Boolean k-Clique conjecture, the Min-Weight k-Clique conjecture) to obtain (conditional) lower bounds for the evaluation of BCQs that depend on $\mathsf{emb}(\mathcal{H})$.

Equipped with the new notion of the clique embedding power, we can show tight lower bounds for several classes of queries. That is, assuming the Boolean k-Clique Conjecture, we derive (conditional) lower bounds for many queries that meet their submodular width, and therefore the current best algorithm, up to polylogarithmic factors. In particular, we show that for cycles [2], Loomis-Whitney joins [22], and chordal graphs, among others, the current combinatorial algorithms are optimal.

We further extend the embedding reduction to be independent of the underlying (commutative) semiring[3]. It was observed by Green, Karvounarakis, and Tannen [10] that the semantics of CQs can be naturally generalized to sum-of-product operations over a semiring. This point of view unifies a number of database query semantics that seem unrelated. For example, evaluation over set semantics corresponds to evaluation over the Boolean semiring $\sigma_{\mathbb{B}} = (\{0,1\}, \vee, \wedge, 0, 1)$, while bag semantics corresponds to the semiring $(\mathbb{N}, +, \times, 0, 1)$. Interestingly, following this framework, the decision problem of finding a k-clique in a graph can be interpreted as the following sum-of-product operation: consider the input graph $G = (V, E)$ as the edge-weighted graph of the complete graph with $|V|$ vertices where $\mathsf{weight}(e) = \mathbb{1}_{e \in E}$; then the problem is to compute $\bigvee_{V' \subseteq V : |V'| = k} \bigwedge \mathsf{w}(\{v, w\})$. Observe that by changing the underlying semiring to be the tropical semiring $\mathsf{trop} = (\mathbb{R}^{\infty}, \min, +, \infty, 0)$, this formulation computes the min-weight k-clique problem. Indeed, given an edge-weighted graph (where the weight of non-existence edges is 0), the minimum weight of its k-clique is exactly $\min_{V' \subseteq V : |V'| = k} \sum \mathsf{w}(\{v, w\})$. We prove that the clique embedding reduction is *semiring-oblivious*, i.e., the reduction holds for arbitrary underlying semirings. This enables one to transfer the lower bound result independent of the underlying semiring and should be of independent interest.

Recent years have witnessed emerging interests in proving lower bounds for the runtime of database queries (see Durand [8] for a wonderful survey). Casel and Schmid consider the fine-grained complexity of regular path queries over graph databases [7]. Joglekar and Ré prove a full dichotomy for whether a 1-series-parallel graph admits a subquadratic algorithm [13]. Their proof is based on the hardness hypothesis that 3-XOR cannot be solved in subquadratic time. Perhaps the line of work in spirit closest to ours is the characterization of queries which can be enumerated by linear preprocessing time and constant delay [3, 6, 4]. However, their results focus on the enumeration problem and therefore are different from the main subject of our paper. Furthermore, their characterization mainly classifies queries based on the existence of a linear preprocessing time and constant delay algorithm. In contrast, our method can provide a lower bound for *every* query.

[3] A triple $(\mathbf{D}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ is a commutative semiring if $\oplus$ and $\otimes$ are commutative binary operators over $\mathbf{D}$ with the following properties: (*i*) $(\mathbf{D}, \oplus)$ is a commutative monoid with an additive identity $\mathbf{0}$. (*ii*) $(\mathbf{D}, \otimes)$ is a commutative monoid with a multiplicative identity $\mathbf{1}$. (*iii*) $\otimes$ distributes over $\oplus$. (*iv*) For any element $e \in \mathbf{D}$, we have $e \otimes \mathbf{0} = \mathbf{0} \otimes e = \mathbf{0}$.

Our Contributions. We summarize our contributions as follows:

- We introduce the notion of the *clique embedding power* $\mathsf{emb}(\mathcal{H})$ of a hypergraph $\mathcal{H}$ (Section 3). We show several interesting properties of this notion; most importantly, we show that it is always upper-bounded by the submodular width, $\mathsf{subw}(\mathcal{H})$. This connection can be seen as additional evidence of the plausibility of the lower bound conjectures for the k-clique problem.
- We show how to construct a reduction from the k-clique problem to any hypergraph $\mathcal{H}$ for any semiring, and discuss how the clique embedding power provides a lower bound for its running time (Section 4).
- We study how to compute $\mathsf{emb}(\mathcal{H})$ (Section 5). In particular, we prove that it is a decidable problem, and give a Mixed Integer Linear Program formulation. One interesting consequence of this formulation is that to achieve $\mathsf{emb}(\mathcal{H})$ it suffices to consider clique sizes that depend on the hypergraph size.
- We identify several classes of hypergraphs for which $\mathsf{emb}(\mathcal{H}) = \mathsf{subw}(\mathcal{H})$ (Section 6). For these classes of queries, our lower bounds match the best-known upper bounds if we consider the Boolean semiring with combinatorial algorithms or the tropical semiring. The most interesting class of hypergraphs we consider is the class of *chordal hypergraphs* (which captures chordal graphs).
- Finally, we identify a hypergraph with six vertices for which there is a gap between its clique embedding power and submodular width (Section 7). We believe that the existence of this gap leaves many open questions.

2 Background

In this section, we define the central problem, and notions necessary for our results.

The SumProduct Problem. We define this general problem following the notation in [16, 14]. Consider ℓ variables $x_1, x_2, \ldots, x_\ell$, where each variable x_i takes values in some discrete domain $\mathsf{Dom}(x_i)$. A *valuation* v is a function that maps each x_i to $\mathsf{Dom}(x_i)$. For a subset $S \subseteq [\ell]$, we define the tuple $\mathbf{x}_S = (x_i)_{i \in S}$ and $v(\mathbf{x}_S) = (v(x_i))_{i \in S}$.

The SumProduct Problem is parameterized by:

1. a commutative semiring $\sigma = (\mathbf{D}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, where $\mathbf{D}$ is a fixed domain.
2. a hypergraph $\mathcal{H} = (V, E)$ where $V = [\ell]$.

The input I specifies for every hyperedge $e \in E$ a function $R_e : \prod_{i \in e} \mathsf{Dom}(x_i) \to \mathbf{D}$. This function is represented in the input as a table of all tuples of the form $(\mathbf{a}_e, R_e(\mathbf{a}_e))$, such that $R_e(\mathbf{a}_e) \neq \mathbf{0}$. This input representation is standard in the CSP and database settings. We use $|I|$ to denote the input size, which is simply the sum of sizes of all tables in the input.

The SumProduct Problem then asks to compute the following function:

$$\bigoplus_{v:\text{valuation}} \bigotimes_{e \in E} R_e(v(\mathbf{x}_e)).$$

We will say that v is a *solution* for the above problem if $\bigotimes_{e \in E} R_e(v(\mathbf{x}_e)) \neq \mathbf{0}$.

Within this framework, we can capture several important problems depending on the choice of the semiring and the hypergraph. If we consider the Boolean semiring $\sigma_{\mathbb{B}} = (\{0, 1\}, \vee, \wedge, 0, 1)$, then each R_e behaves as a relational instance (R_e is 1 if the tuple is in the instance, otherwise 0) and the SumProduct function captures Boolean Conjunctive Query evaluation. If $\sigma = (\mathbb{N}, +, \times, 0, 1)$ and R_e is defined as above, then the SumProduct function computes the number of solutions to a Conjunctive Query. Another important class of

problems is captured when we consider the min-tropical semiring $\mathsf{trop} = (\mathbb{R}^{\infty}, \min, +, \infty, 0)$ and we assign each tuple to a non-negative weight; this computes a minimum weight solution that satisfies the structural properties.

The Complexity for SumProduct Problems. We adopt the *random-access machine (RAM)* as our computation model with $O(\log n)$-bit words, which is standard in fine-grained complexity. The machine has read-only input registers and it contains the database and the query, read-write work memory registers, and write-only output registers. It is assumed that each register can store any tuple, and each tuple is stored in one register. The machine can perform all "standard" [4] operations on one or two registers in constant time.

In this paper, we are interested in the computational complexity of a SumProduct problem for a fixed hypergraph $\mathcal{H}$. (This is typically called *data complexity*). We will consider two different ways of treating semirings when we think about algorithms.

In the first variant, we fix the semiring σ along with the hypergraph $\mathcal{H}$. This means that the representation of the semiring is not part of the input and is known a priori to the algorithm. We denote this problem as $\mathsf{SumProd}\langle\sigma, \mathcal{H}\rangle$. In the second variant, we consider algorithms that access the semiring only via an oracle. In particular, the algorithm does not know the semiring a priori and can only access it during runtime by providing the values for the $\oplus, \otimes$ operations. We assume that each of these operations takes a constant amount of time. We denote this problem as $\mathsf{SumProd}\langle\mathcal{H}\rangle$.

Our goal in this paper is to specify the exact exponent of $|I|$ in the polynomial-time runtime cost of an algorithm that computes $\mathsf{SumProd}\langle\sigma, \mathcal{H}\rangle$ or $\mathsf{SumProd}\langle\mathcal{H}\rangle$.

Tree Decompositions. A *tree decomposition* of a hypergraph $\mathcal{H}$ is a pair $(\mathcal{T}, \chi)$, where $\mathcal{T}$ is a tree and χ maps each node $t \in V(\mathcal{T})$ of the tree to a subset $\chi(t)$ of $V(\mathcal{H})$ such that:

1. every hyperedge $e \in E(\mathcal{H})$ is a subset of $\chi(t)$ for some $t \in V(\mathcal{T})$; and
2. for every vertex $v \in V(\mathcal{H})$, the set $\{t \mid v \in \chi(t)\}$ is a non-empty connected subtree of $\mathcal{T}$.

We say that a hypergraph $\mathcal{H}$ is *acyclic* if it has a tree decomposition such that each bag corresponds to a hyperedge.

Notions of Width. Let $\mathcal{H}$ be a hypergraph and F be a set function over $V(H)$. The F-width of a tree decomposition $(\mathcal{T}, \chi)$ is defined as $\max_t F(\chi(t))$. The F-width of $\mathcal{H}$ is the minimum F-width over all possible tree decompositions of $\mathcal{H}$.

A *fractional independent set* of a hypergraph $\mathcal{H}$ is a mapping $\mu : V(\mathcal{H}) \to [0, 1]$ such that $\sum_{v \in e} \mu(v) \leq 1$ for every $e \in E(\mathcal{H})$. We naturally extend functions on the vertices of $\mathcal{H}$ to subsets of vertices of $\mathcal{H}$ by setting $\mu(X) = \sum_{v \in X} \mu(v)$.

The *adaptive width* $\mathsf{adw}(\mathcal{H})$ of a hypergraph $\mathcal{H}$ is defined as the supreme of F-width$(\mathcal{H})$, where F goes over all fractional independent sets of $\mathcal{H}$. Hence if $\mathsf{adw}(\mathcal{H}) \leq w$, then for every μ, there exists a tree decomposition of $\mathcal{H}$ with μ-width at most w.

A set function F is *submodular* if for any two sets A, B we have $F(A \cup B) + F(A \cap B) \leq F(A) + F(B)$. It is monotone if whenever $A \subseteq B$, then $F(A) \leq F(B)$. The *submodular width* $\mathsf{subw}(\mathcal{H})$ of a hypergraph $\mathcal{H}$ is defined as the supreme of F-width$(\mathcal{H})$, where F now ranges over all non-negative, monotone, and submodular set functions over $V(\mathcal{H})$ such that for every hyperedge $e \in E(\mathcal{H})$, we have $F(e) \leq 1$. A non-negative, monotone, and submodular set function F is *edge-dominated* if $F(e) \leq 1$, for every $e \in E$.

[4] This includes all arithmetic (e.g. $+, -, \div, *$) and logical operations.

The *fractional hypertree width* of a hypergraph $\mathcal{H}$ is $\mathsf{fhw}(\mathcal{H}) = \min_{(\mathcal{T},\chi)} \max_{t \in V(\mathcal{T})} \rho^*(\chi(t))$, where ρ^* is the minimum fractional edge cover number of the set $\chi(t)$. It holds that $\mathsf{adw}(\mathcal{H}) \leq \mathsf{subw}(\mathcal{H}) \leq \mathsf{fhw}(\mathcal{H})$.

It is known that $\mathsf{SumProd}\langle\sigma_{\mathbb{B}}, \mathcal{H}\rangle$ can be computed in time $\tilde{O}(|I|^{\mathsf{subw}(\mathcal{H})})$ using the PANDA algorithm [17]. However, we do not know of a way to achieve the same runtime for the general $\mathsf{SumProd}\langle\mathcal{H}\rangle$ problem. For this, the best known runtime is $\tilde{O}(|I|^{\#\mathsf{subw}(\mathcal{H})})$, where $\mathsf{subw}(\mathcal{H}) \leq \#\mathsf{subw}(\mathcal{H}) \leq \mathsf{fhw}(\mathcal{H})$ [14]. On the other hand, there are hypergraphs for which we can compute $\mathsf{SumProd}\langle\sigma_{\mathbb{B}}, \mathcal{H}\rangle$ with runtime better than $\tilde{O}(|I|^{\mathsf{subw}(\mathcal{H})})$ using non-combinatorial algorithms. For example, if $\mathcal{H}$ is a triangle we can obtain a runtime $\tilde{O}(|I|^{2\omega/(\omega+1)})$, where ω is the matrix multiplication exponent (the submodular width of the triangle is $3/2$).

Conjectures in Fine-Grained Complexity. Our lower bounds will be based on the following popular conjectures in fine-grained complexity. To state the conjectures, it will be helpful to define the *k-clique problem over a semiring σ*: given an undirected graph $G = (V, E)$ where each edge has a weight in the domain of the semiring, we are asked to compute the semiring-product over all the k-cliques in G, where the weight of each clique is the semiring-sum of clique edge weights.

▶ **Definition 1** (Boolean k-Clique Conjecture)**.** *There is no real $\epsilon > 0$ such that computing the k-clique problem (with $k \geq 3$) over the Boolean semiring in an (undirected) n-node graph requires time $O(n^{k-\epsilon})$ using a combinatorial algorithm.*

▶ **Definition 2** (Min-Weight k-Clique Conjecture)**.** *There is no real $\epsilon > 0$ such that computing the k-clique problem (with $k \geq 3$) over the tropical semiring in an (undirected) n-node graph with integer edge weights can be done in time $O(n^{k-\epsilon})$.*

When $k = 3$, min-weight 3-clique is equivalent to the All-Pairs Shortest Path (APSP) problem under subcubic reductions. The Min-Weight Clique Conjecture assumes the Min-Weight k-Clique conjecture for every integer $k \geq 3$ (similarly for the Boolean Clique Conjecture).

3 The Clique Embedding Power

In this section, we define the clique embedding power, a quantity central to this paper.

3.1 Graph Embeddings

We introduce first the definition of embedding a graph G to a hypergraph $\mathcal{H}$, first defined by Marx [20, 19]. We say that two sets of vertices $X, Y \subseteq V(\mathcal{H})$ *touch* in $\mathcal{H}$ if either $X \cap Y \neq \emptyset$ or there is a hyperedge $e \in E(\mathcal{H})$ that intersects both X and Y. We say a hypergraph is connected if its underlying clique graph is connected.

▶ **Definition 3** (Graph Embedding)**.** *Let G be an undirected graph, and $\mathcal{H}$ be a hypergraph. An* embedding *from G to $\mathcal{H}$, denoted $G \mapsto \mathcal{H}$, is a mapping ψ that maps every vertex $v \in V(G)$ to a non-empty subset $\psi(v) \subseteq V(\mathcal{H})$ such that the following hold:*

1. *$\psi(v)$ induces a connected subhypergraph;*
2. *if $u, v \in V(G)$ are adjacent in G, then $\psi(u), \psi(v)$* touch *in $\mathcal{H}$.*

It will often be convenient to describe an embedding ψ by the reverse mapping $\psi^{-1}(x) = \{i \mid x \in \psi(i)\}$, where x is a vertex in $V(\mathcal{H})$. Given an embedding ψ and a vertex $v \in V(\mathcal{H})$, we define its *vertex depth* as $d_\psi(v) = |\psi^{-1}(v)|$. For a hyperedge $e \in E(\mathcal{H})$, we define its *weak edge depth* as $d_\psi(e) = |\{v \in V(G) \mid \psi(v) \cap e \neq \emptyset\}|$, i.e., the number of vertices of G that map to some variable in e. Moreover, we define the *edge depth* of e as $d^+_\psi(e) = \sum_{v \in e} d_\psi(v)$.

The *weak edge depth* of an embedding ψ can then be defined as $\mathsf{wed}(\psi) = \max_e d_\psi(e)$, and the edge depth as $\mathsf{ed}(\psi) = \max_e d^+_\psi(e)$. Additionally, we define as $\mathsf{wed}(G \mapsto \mathcal{H})$ the minimum weak edge depth of any embedding ψ from G to $\mathcal{H}$. Similarly for $\mathsf{ed}(G \mapsto \mathcal{H})$. It is easy to see that $\mathsf{wed}(G \mapsto \mathcal{H}) \leq \mathsf{ed}(G \mapsto \mathcal{H})$.

It will be particularly important for our purposes to think about embedding the k-clique graph C_k to an arbitrary hypergraph $\mathcal{H}$. In this case, it will be simpler to think of the vertices of G as the numbers $1, \ldots, k$ and the embedding ψ as a mapping from the set $\{1, \ldots, k\}$ to a subset of $V(\mathcal{H})$. We can now define the following quantity, which captures how well we can embed a k-clique to H for an integer $k \geq 3$:

$$\mathsf{emb}_k(\mathcal{H}) := \frac{k}{\mathsf{wed}(C_k \mapsto \mathcal{H})}.$$

▶ **Example 4.** Consider the hypergraph $\mathcal{H}$ with the following hyperedges:

$$\{x_1, x_2, x_3\}, \{x_1, y\}, \{x_2, y\}, \{x_3, y\}$$

We can embed the 5-clique into $\mathcal{H}$ as follows:

$$1 \to \{x_1\}, 2 \to \{x_2\}, 3 \to \{x_3\}, 4, 5 \to \{y\}.$$

It is easy to check that this is a valid embedding, since, for example, $1, 4$ touch at the edge $\{x_1, y\}$. Moreover, $\mathsf{wed}(C_5 \mapsto G) = 3$, hence $\mathsf{emb}_5(G) = 5/3$.

3.2 Embedding Properties

In this part, we will explore how $\mathsf{wed}(C_k \mapsto \mathcal{H})$ and $\mathsf{emb}_k(\mathcal{H})$ behave as a function of the size of the clique k. We start with some basic observations.

▶ **Proposition 5.** *For any hypergraph $\mathcal{H}$ and integer $k \geq 3$:*
1. $\mathsf{wed}(C_k \mapsto \mathcal{H}) \leq k$.
2. $\mathsf{wed}(C_k \mapsto \mathcal{H}) \leq \mathsf{wed}(C_{k+1} \mapsto H) \leq \mathsf{wed}(C_k \mapsto \mathcal{H}) + 1$.
3. *If $k = m \cdot n$, where $k, m, n \in \mathbb{Z}_{\geq 0}$, then* $\mathsf{emb}_k(\mathcal{H}) \geq \mathsf{emb}_m(\mathcal{H})$.

Proof. (1) We define an embedding ψ from a k-clique where $\psi(i) = V(\mathcal{H})$ for every $i = 1, \ldots, k$. It is easy to see that ψ is an embedding with weak edge depth k.

(2) For the first inequality, take any ψ_{k+1}, we can construct a ψ_k by only preserving the mapping ψ_{k+1} for $[k]$. Then, for any $e \in E(\mathcal{H})$, we have

$$\{y \in V(C_k) \mid \psi_k(y) \cap e \neq \emptyset\} \subseteq \{y \in V(C_{k+1}) \mid \psi_{k+1}(y) \cap e \neq \emptyset\}$$

Thus,

$$\mathsf{wed}(C_k \mapsto \mathcal{H}) \leq \mathsf{wed}(\psi_k) := \max_{e \in E(\mathcal{H})} d_{\psi_k}(e) \leq \mathsf{wed}(\psi_{k+1}).$$

For the second inequality, take any ψ_k, we construct a ψ_{k+1} by preserving the mapping ψ_k and ψ_{k+1} maps $k+1$ to $V(\mathcal{H})$. Then, for any $e \in E(\mathcal{H})$, we have

$$d_{\psi_{k+1}}(e) = d_{\psi_k}(e) + 1$$

so $\mathsf{wed}(\psi_{k+1}) = \mathsf{wed}(\psi_k) + 1$ and in particular, we can take ψ_k such that

$$\mathsf{wed}(\psi_{k+1}) \leq \mathsf{wed}(C_k \mapsto \mathcal{H}) + 1$$

which implies that

$$\mathsf{wed}(C_{k+1} \mapsto \mathcal{H}) \leq \mathsf{wed}(C_k \mapsto \mathcal{H}) + 1$$

(3) Suppose ψ is the embedding that achieves $\mathsf{emb}_m(\mathcal{H})$ for C_m. It suffices to construct an embedding ψ' for C_k which achieves the same quantity $\mathsf{emb}_m(\mathcal{H})$. To do so, we simply bundle every n vertices in C_k to be a "hypernode". That is, label the bundles as $b_1, \ldots, b_n$. and $\psi'(v) = \psi(i)$ if and only if $v \in B_i$. The embedding power given by ψ' is then

$$\frac{k}{\mathsf{wed}(\psi')} = \frac{mn}{\mathsf{wed}(\psi)n} = \frac{m}{\mathsf{wed}(\psi)} = \mathsf{emb}_m(\mathcal{H}).$$ ◀

The first item of the above proposition tells us that $\mathsf{emb}_k(\mathcal{H}) \geq 1$ for any k. But how does $\mathsf{emb}_k(\mathcal{H})$ behave as k grows? We next show that $\mathsf{emb}_k(\mathcal{H})$ is always upper bounded by the submodular width of $\mathcal{H}$.

▶ **Lemma 6.** *Let $\mathcal{H}$ be a hypergraph. Take an embedding $\psi : C_k \mapsto \mathcal{H}$. Let $(\mathcal{T}, \chi)$ be a tree decomposition of $\mathcal{H}$. Then, there exists a node $t \in T$ such that for every $i = 1, \ldots, k$, $\psi(i) \cap \chi(t) \neq \emptyset$.*

Proof. For $i = 1, \ldots, k$, let $\mathcal{T}_i$ be the subgraph of $\mathcal{T}$ that includes all nodes $t \in V(\mathcal{T})$ such that $\psi(i) \cap \chi(t) \neq \emptyset$.

We first claim that $\mathcal{T}_i$ *forms a tree.* To show this, it suffices to show that $\mathcal{T}_i$ is connected. Indeed, take any two nodes t_1, t_2 in $\mathcal{T}_i$. This means that there exists $x_1 \in \chi(t_1) \cap \psi(i)$ and $x_2 \in \chi(t_2) \cap \psi(i)$. Since $x_1, x_2 \in \psi(i)$ and $\psi(i)$ induces a connected subgraph in $\mathcal{H}$, there exists a sequence of vertices $x_1 = z_1, \ldots, z_k = x_2$, all in $\psi(i)$, such that every two consecutive vertices belong to an edge of $\mathcal{H}$. Let $S_1, \ldots, S_k$ be the trees in $\mathcal{T}$ that contain $z_1, \ldots, z_k$ respectively. Take any two consecutive z_i, z_{i+1}: since they belong to the same edge, there exists a bag that contains both of them, hence S_i, S_{i+1} intersect. This means that there exists a path between t_1, t_2 in $\mathcal{T}$ such that every node is in T_i.

Second, we claim that *any two $\mathcal{T}_i, \mathcal{T}_j$ have at least one common vertex.* Indeed, $\psi(i)$, $\psi(j)$ must touch in $\mathcal{H}$. If there exists a variable $x \in \psi(i) \cap \psi(j)$, then any vertex that contains x is a common vertex between $\mathcal{T}_i, \mathcal{T}_j$. Otherwise, there exists $x \in \psi(i)$, $y \in \psi(j)$ such that x, y occur together in a hyperedge $e \in E(\mathcal{H})$. But this means that some node $t \in \mathcal{T}$ contains both x, y, hence $\mathcal{T}_i, \mathcal{T}_j$ intersect at t.

Finally, we apply the fact that a family of subtrees of a tree satisfies the *Helly property* [12], i.e. *a collection of subtrees of a tree has at least one common node if and only if every pair of subtrees has at least one common node.* Indeed, the trees $\mathcal{T}_1, \ldots, \mathcal{T}_k$ satisfy the latter property, so there is a vertex t common to all of them. Such t has the desired property of the lemma. ◀

We can now state the following Theorem 7 on the embedding power of a hypergraph.

▶ **Theorem 7.** *For any hypergraph $\mathcal{H}$ and integer $k \geq 3$, the following holds:*

$$\mathsf{wed}(C_k \mapsto \mathcal{H}) \geq \frac{k}{\mathsf{subw}(\mathcal{H})}$$

Proof. Let $\mathsf{wed}(C_k \mapsto \mathcal{H}) = \alpha$. Then, there is an embedding $\psi : C_k \mapsto \mathcal{H}$ with weak edge depth α. We will show that $\mathsf{subw}(\mathcal{H}) \geq k/\alpha$.

First, we define the following set function over subsets of $V(\mathcal{H})$: for any $S \subseteq V(\mathcal{H})$, let $\mu(S) = |\{i \mid \psi(i) \cap S \neq \emptyset\}|/\alpha$. This is a coverage function, and hence it is a submodular function. It is also edge-dominated, since for any hyperedge e, we have $\mu(e) = |\{i \mid \psi(i) \cap e \neq \emptyset\}|/\alpha \leq 1$.

Now, consider any decomposition $(\mathcal{T}, B_t)$ of $\mathcal{H}$. From Lemma 6, there is a node $t \in \mathcal{T}$ such that or every $i = 1, \ldots, k$, $\psi(i) \cap B_t \neq \emptyset$. Hence, $\mu(B_t) = |\{i \mid \psi(i) \cap B_t \neq \emptyset\}|/\alpha = k/\alpha$. Thus, the submodular width of the decomposition is at least k/α. ◀

The above result tells us that $\mathsf{emb}_k(\mathcal{H}) \leq \mathsf{subw}(\mathcal{H})$ for any $k \geq 3$. Hence, taking the supremum of $\mathsf{emb}_k(\mathcal{H})$ for $k \geq 3$ is well-defined since the set is bounded. This leads us to the following definition:

▶ **Definition 8** (Clique Embedding Power). *Given a hypergraph $\mathcal{H}$, define the* clique embedding power *of $\mathcal{H}$ as*

$$\mathsf{emb}(\mathcal{H}) := \sup_{k \geq 3} \mathsf{emb}_k(\mathcal{H}) = \sup_{k \geq 3} \frac{k}{\mathsf{wed}(C_k \mapsto \mathcal{H})}.$$

The following is immediate:

▶ **Corollary 9.** *For any hypergraph $\mathcal{H}$, $1 \leq \mathsf{emb}(\mathcal{H}) \leq \mathsf{subw}(\mathcal{H})$.*

For the connection between edge depth width and adaptive width, we have the following theorem analogous to Theorem 7. The proof can be found in [9].

▶ **Theorem 10.** *For any hypergraph $\mathcal{H}$, the following holds:*

$$\mathsf{ed}(C_k \mapsto \mathcal{H}) \geq \frac{k}{\mathsf{adw}(\mathcal{H})}$$

4 Lower Bounds

In this section, we show how to use the clique embedding power to obtain conditional lower bounds for SumProduct problems. Our main reduction follows the reduction used in [20], but also has to account for constructing the appropriate values for the semiring computations.

▶ **Theorem 11.** *For any hypergraph $\mathcal{H}$ and semiring σ, if $\mathsf{SumProd}\langle\sigma, \mathcal{H}\rangle$ can be solved in time $O(|I|^c)$ with input I, then k-Clique over σ can be solved in time $O(n^{c \cdot \mathsf{wed}(C_k \mapsto \mathcal{H})})$ where n is the number of vertices.*

Proof. We will show a reduction from the k-clique problem with n vertices over a semiring σ to $\mathsf{SumProd}\langle\sigma, \mathcal{H}\rangle$. Without loss of generality, we will assume that the input graph G to the k-clique problem is k-partite, with partitions $V_1, \ldots, V_k$. Indeed, given any graph $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_n\}$, consider the k-partite graph $G^k = (V^k, E^k)$ where $V^k = \{v_i^j \mid 1 \leq i \leq n, 1 \leq j \leq k\}$ and for any two vertices $v_i^j, v_p^q \in V^k$, $\{v_i^j, v_p^q\} \in E^k$ iff $\{v_i, v_p\} \in E$ and $j \neq q$. Then there is a one-to-one mapping from a k-clique in G to a k-clique in G^k.

Let ψ be an embedding from C_k to $\mathcal{H}$ that achieves a weak edge depth $\lambda = k/\mathsf{emb}_k(\mathcal{H})$. As we mentioned before, it is convenient to take $V(C_k) = \{1, \ldots, k\}$. We now construct the input instance I for $\mathsf{SumProd}\langle\sigma, \mathcal{H}\rangle$. More explicitly, the task is to construct the function R_e for each hyperedge $e \in E$.

To this end, we first assign to each pair $\{i, j\} : i \neq j, i, j, \in \{1, 2, \ldots, k\}$ a hyperedge $\theta(\{i, j\}) = e \in E(\mathcal{H})$ satisfying the following conditions: $\psi(i) \cap e \neq \emptyset$ and $\psi(j) \cap e \neq \emptyset$. Such an e must exist by the definition of an embedding. If there is more than one hyperedge satisfying the condition, we arbitrarily choose one.

For every variable $x \in V(\mathcal{H})$, let $\psi^{-1}(x)$ be the subset of $\{1, \ldots, k\}$ mapping to x. Then, we define the domain $\mathsf{Dom}(x_i)$ of each variable x_i in the input instance as vectors over $[n]^{|\psi^{-1}(x_i)|}$. Let $S_e = \{i \in [k] \mid \psi(i) \cap e \neq \emptyset\}$. Note that $|S_e| = d_\psi(e) \leq \lambda$. Also, note

that $\psi^{-1}(x) \subseteq S_e$ for all $x \in e$. Then, we compute all cliques in the graph G between the partitions $V_i, i \in S_e$; these cliques will be of size $|S_e|$ and can be computed in running time $O(n^\lambda)$ by brute force.

For every clique $\{a_i \in V_i \mid i \in S_e\}$, let t be the tuple over $\prod_{i \in S_e} \mathsf{Dom}(x_i)$ such that its value at position x is $\langle a_i \mid i \in \psi^{-1}(x)\rangle$. Then, we set

$$R_e(t) = \mathbf{1} \otimes \bigotimes_{\{i,j\}:\theta(\{i,j\})=e} w(\{i,j\}).$$

In other words, we set the value as the semiring product of all the weights between the edges $\{a_i, a_j\}$ in the clique whenever the pair $\{i,j\}$ is assigned to the hyperedge e. All the other tuples are mapped to $\mathbf{0}$. By construction, the size of the input is $|I| = O(n^\lambda)$.

We now show that the two problems will return exactly the same output. To show this, we first show that there is a bijection between k-cliques in G and the solutions of the SumProduct instance.

$\Leftarrow$ Take a clique $\{a_1, \ldots, a_k\}$ in G. We map the clique to the valuation $v(x) = \langle a_i \mid i \in \psi^{-1}(x)\rangle$. This valuation is a solution to the SumProduct problem, since any subset of $\{a_1, \ldots, a_k\}$ forms a sub-clique. Hence for any hyperedge e, $R_e(v(\mathbf{x}_e)) \neq \mathbf{0}$.

$\Rightarrow$ Take a valuation v. Consider any $i \in \{1, \ldots, k\}$ and consider any two variables $x, y \in \psi(i)$ (recall that $\psi(i)$ must be nonempty). Recall that $x, y \in V(\mathcal{H})$. We claim that the i-th index in the valuation $v(x), v(y)$ must take the same value, which we will denote as a_i; this follows from the connectivity condition of the embedding. Indeed, since $x, y \in \psi(i)$, there exists a sequence of hyperedges $e_1, e_2, \ldots, e_m$ where $m \geq 1$ such that $e_j \cap e_{j+1} \neq \emptyset$ for $1 \leq j \leq m-1$ and $x \in e_1$, $y \in e_m$. By the construction, the i-th index in $v(x)$ will then "propagate" to that in $v(y)$. This proves the claim. It then suffices to show that $\{a_1, \ldots, a_k\}$ is a clique in G. Indeed take any $i, j \in \{1, \ldots, k\}$. Since i and j are adjacent as two vertices in C_k, we know $\psi(i)$ and $\psi(j)$ touch. Therefore, there exists a hyperedge e that contains some $x \in \psi(i)$ and $y \in \psi(j)$. But this means that $\{a_i, a_j\}$ must form an edge in G.

We next show that the semiring product of the weights in the clique has the same value as the semiring product of the corresponding solution. Indeed:

$$\bigotimes_{e \in E} R_e(v(\mathbf{x}_e)) = \mathbf{1} \otimes \bigotimes_{e \in E} \bigotimes_{\{i,j\}:\theta(\{i,j\})=e} w(\{i,j\}) = \bigotimes_{\{i,j\}:i \neq j} w(\{i,j\})$$

where the last equality holds because each edge of the k-clique is assigned to exactly one hyperedge of $\mathcal{H}$.

The above claim together with the bijection show that the output will be the same; indeed, each the semiring sums will go over exactly the same elements with the same values.

To conclude the proof, suppose that $\mathsf{SumProd}\langle\sigma, \mathcal{H}\rangle$ could be answered in time $O(|I|^c)$ for some $c \geq 1$. This means that we can solve the k-clique problem over σ in time $O(n^\lambda + n^{c\lambda}) = O(n^{c \cdot \mathsf{wed}(C_k \mapsto \mathcal{H})})$. ◀

As an immediate consequence of Theorem 11, we can show the following lower bound.

▶ **Proposition 12.** *Under the Min-Weight k-Clique conjecture,* $\mathsf{SumProd}\langle\mathsf{trop}, \mathcal{H}\rangle$ *(and thus* $\mathsf{SumProd}\langle\mathcal{H}\rangle$*) cannot be computed in time $O(|I|^{\mathsf{emb}_k(\mathcal{H})-\epsilon})$ for any constant $\epsilon > 0$.*

Proof. Indeed, if $\mathsf{SumProd}\langle\mathsf{trop}, \mathcal{H}\rangle$ can be computed in time $O(|I|^{\mathsf{emb}_k(\mathcal{H})-\epsilon})$ for some constant $\epsilon > 0$, then by Theorem 11 the k-Clique problem over the tropical semiring can be solved in time $O(n^{(\mathsf{emb}_k(\mathcal{H})-\epsilon)\cdot\mathsf{wed}(C_k \mapsto \mathcal{H})}) = O(n^{k-\delta})$ for some $\delta > 0$. However, this violates the Min-Weight k-Clique conjecture. ◀

Similarly, we can show the following:

▶ **Proposition 13.** *Under the Boolean k-Clique conjecture,* $\mathsf{SumProd}\langle\sigma_{\mathbb{B}}, \mathcal{H}\rangle$ *(and thus* $\mathsf{SumProd}\langle\mathcal{H}\rangle$*) cannot be computed via a combinatorial algorithm in time* $O(|I|^{\mathsf{emb}_k(\mathcal{H})-\epsilon})$ *for any constant* $\epsilon > 0$.

The above two results imply that to obtain the best lower bound, we need to find the clique size with the largest $\mathsf{emb}_k(\mathcal{H})$. However, the function $k \mapsto \mathsf{emb}_k(\mathcal{H})$ is really intriguing. It is not clear whether in the definition supremum is ever needed, i.e., whether there exists a hypergraph where the embedding power is achieved in the limit.

In Section 5, we show that for every hypergraph $\mathcal{H}$, there exists a natural number k such that $\mathsf{emb}(\mathcal{H}) = \mathsf{emb}_k(\mathcal{H})$. We also demonstrate how to compute $\mathsf{emb}(\mathcal{H})$ through a MILP and locate the complexity of computing the embedding power within the class 2-EXPTIME (double exponential time). The insight of our method is that, instead of computing the "integral" embedding power, one can consider the "fractional" embedding power and then recover the "integral" one by letting the clique size k to be sufficiently large.

5 Decidability of the Clique Embedding Power

To illustrate the algorithm for computing $\mathsf{emb}(\mathcal{H})$, it is instructive to first show how to compute $\mathsf{emb}_k(\mathcal{H})$ for a fixed clique size k.

5.1 An Integer Linear Program for $\mathsf{wed}(C_k \mapsto \mathcal{H})$

The following ILP formulation computes the minimum weak edge depth $w = \mathsf{wed}(C_k \mapsto \mathcal{H})$.

$$
\begin{array}{lllll}
\min & w & & & \\
\text{s.t.} & \sum_{S\subseteq V} x_S = k & & & \\
& x_S = 0 & \forall S \subseteq V & \text{where } S \text{ is not connected} & \\
& \min\{x_S, x_T\} = 0 & \forall S, T \subseteq V & \text{where } S, T \text{ do not touch} & \\
& \sum_{S\subseteq V : e\cap S\neq\emptyset} x_S \leq w & \forall e \in \mathcal{E} & & \\
& x_S \in \mathbb{Z}_{\geq 0} & \forall S \subseteq V & &
\end{array}
\tag{1}
$$

Each integer variable x_S, $S \subseteq V$, indicates how many vertices in C_k are assigned to the subset S. For example, if $x_{\{1,2\}} = 3$, this means in the embedding ψ, three vertices are mapped to the subset $\{1, 2\} \subseteq V$. It is sufficient to record only the number of vertices in C_k because of the symmetry of the clique. That is, since any two vertices are connected in C_k, one can arbitrarily permute the vertices in C_k so that the resulting map ψ' is still an embedding (given ψ is). Moreover, since the clique size k is fixed, to compute $\mathsf{emb}_k(\mathcal{H})$ it suffices to minimize w.

Observe that the condition $\min\{x_S, x_T\} = 0$ is not a linear condition. To encode it as such, we perform a standard transformation. We introduce a binary variable y_S for every set $S \subseteq V$. Then, we can write it as

$$
\begin{cases}
x_S + k \cdot y_S \leq k \\
x_T + k \cdot y_T \leq k \\
y_S + y_T \geq 1
\end{cases}
$$

Indeed, since y_S and y_T are binary variables, at least one of them is 1. Without loss of generality assume $y_S = 1$. Then $x_S = 0$ since $x_S \in \mathbb{Z}_{\geq 0}$. Therefore two subsets that do not touch cannot both be chosen in the embedding.

5.2 A Mixed Integer Linear Program for emb($\mathcal{H}$)

The above ILP construction does not directly yield a way to compute the clique embedding power, since the latter is defined to be the supremum for all k.

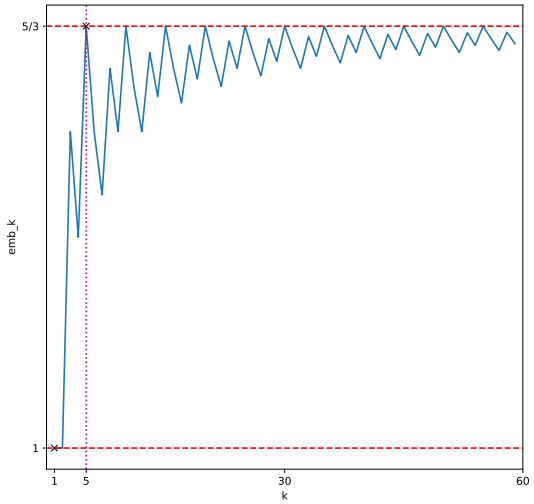


Figure 1 $\mathsf{emb}_k(\mathcal{H})$ for the 6-cycle.

As alluded before, the behavior of $\mathsf{emb}_k(\mathcal{H})$ as a function of k is non-trivial (and certainly not monotone). Figure 1 depicts how the clique embedding power changes with respect to different clique sizes for the 6-cycle, where the horizontal line represents the clique size.

To compute the supremum, the key idea is to change the integer variables x_S to be continuous (so they behave as fractions) and "normalize" the clique size k to 1. Specifically, we can write the following mixed integer linear program (MILP).

$$
\begin{array}{lllll}
\min & w & & & \\
\text{s.t.} & \sum_{S\subseteq V} x_S = 1 & & & \\
& x_S = 0 & \forall S \subseteq V & \text{where } S \text{ is not connected} & \\
& \min\{x_S, x_T\} = 0 & \forall S,T \subseteq V & \text{where } S,T \text{ do not touch} & \\
& \sum_{S\subseteq V: e\cap S\neq\emptyset} x_S \leq w & \forall e \in \mathcal{E} & & \\
& x_S \in \mathbb{R}_{\geq 0} & \forall S \subseteq V & &
\end{array}
\tag{2}
$$

▶ **Proposition 14.** *Let w^* be the optimal solution of MILP (2). Then, $\mathsf{emb}(\mathcal{H}) = 1/w^*$. Additionally, there exists an integer $K \geq 3$ such that $\mathsf{emb}(\mathcal{H}) = \mathsf{emb}_K(\mathcal{H})$.*

Proof. We first show for any k, $\mathsf{emb}_k(\mathcal{H}) \geq 1/w^*$. Indeed, any embedding $\psi : C_k \to \mathcal{H}$ determines the values of the variables x_S in MILP (1). Let $\hat{x_S} = \frac{x_S}{k}$ and $\hat{w} = \frac{w}{k}$ be an assignment of the variables in MILP (2). It is easy to see that this is a feasible assignment. Thus, $w^* \leq \mathsf{wed}(C_k \mapsto \mathcal{H})/k$. Therefore $\mathsf{emb}_k(\mathcal{H}) = k/\mathsf{wed}(C_k \mapsto \mathcal{H}) \leq 1/w^*$.

Next, observe that $\mathsf{emb}(\mathcal{H})$ is a rational number. In fact, the solution w^* for MILP (2) is a rational number, since every constant is a rational number [24]. Let K be the least common multiplier of their denominators of the fractions in the set $\{x_S\}$. Then, the assignment $K \cdot x_S, K \cdot w$ is a feasible solution for MILP (1) for $k = K$. This implies that $K \cdot w^* \geq \mathsf{wed}(C_K \mapsto \mathcal{H})$, so $\mathsf{emb}_K(\mathcal{H}) \geq 1/w^*$.

Table 1 Summary of emb and subw for some classes of queries.

	emb		subw	
Acyclic	1	[Theorem 29]	1	[27]
Chordal	=	[Theorem 24]	=	[22]
ℓ-cycle	$2-1/\lceil \ell/2 \rceil$	[Proposition 18]	$2-1/\lceil \ell/2 \rceil$	[2]
$K_{2,\ell}$	$2-1/\ell$	[Proposition 19]	$2-1/\ell$	[17]
$K_{3,3}$	2	[Proposition 20]	2	[17]
A_ℓ	$(\ell-1)/2$	[Proposition 26]	$(\ell-1)/2$	[22]
$\mathcal{H}_{\ell,k}$	ℓ/k	[Proposition 27]	ℓ/k	[22]
Q_b	17/9		2	[13]
Q_{hb}	7/4		2	[Proposition 30]

Thus, we have shown that $1/w^*$ is an upper bound for $\{\mathsf{emb}_k(\mathcal{H})\}_k$, but also $\mathsf{emb}_K(\mathcal{H}) = 1/w^*$. Hence, $\mathsf{emb}(\mathcal{H}) = \mathsf{emb}_K(\mathcal{H}) = 1/w^*$. ◀

This leads to the following theorem (whose proof can be found in [9]).

▶ **Theorem 15.** *The problem of computing* $\mathsf{emb}(\mathcal{H})$ *for a hypergraph* $\mathcal{H}$ *is in 2-*EXPTIME *and, in particular, is decidable.*

Unfortunately, our method does not yield an upper bound on how large the K in Proposition 14 might be. There is no reason that K cannot be very large, e.g. doubly exponential to the size of $\mathcal{H}$. Some knowledge of that could be very useful in computing the clique embedding power. For example, one can compute all the embeddings from C_k, for k not greater than the upper bound, and output the one with the largest embedding power. The best-known upper bound we have so far is the following. The proof can be found in [9].

▶ **Proposition 16.** *For any hypergraph* $\mathcal{H}$*, there is a constant* $K = O((2^{|V|})!)$ *such that* $\mathsf{emb}(\mathcal{H}) = \mathsf{emb}_K(\mathcal{H})$.

6 Examples of Tightness

In this section, we identify several classes of queries where the clique embedding power coincides with the submodular width. For brevity, we write emb, subw, fhw, and adw when the underlying hypergraph is clear under context. Table 1 summarizes our results.

6.1 Cycles

For the cycle query of length $\ell \geq 3$, we show that $\mathsf{emb} = \mathsf{subw} = 2 - 1/\lceil \ell/2 \rceil$. The best-known algorithm for ℓ-cycle detection (and counting) of Alon, Yuster, and Zwick [2] runs in time $O(|I|^{\mathsf{subw}})$. First, we show the following lemma.

▶ **Lemma 17.** *Consider the cycle query of length* $\ell \geq 3$*. Then* $\mathsf{emb} \geq 2 - 1/\lceil \frac{\ell}{2} \rceil$.

Proof. We start with the case where ℓ is odd and name the variables of the cycle query as $x_1, \ldots, x_\ell$. Then, we define $\lambda = (\ell+1)/2$ and an embedding from a ℓ-clique as follows:

$$\begin{aligned} \psi^{-1}(x_1) &= \{1, 2, \ldots, \lambda - 1\} \\ \psi^{-1}(x_2) &= \{2, 3, \ldots, \lambda\} \\ &\cdots \\ \psi^{-1}(x_\ell) &= \{2\lambda - 1, 1, \ldots, \lambda - 2\} \end{aligned} \tag{3}$$

In other words, ψ maps each $i \in [\ell]$ into a consecutive segment consisting of $\lambda-1$ vertices in the cycle. To see why ψ is an embedding, we observe that for any $i, j \in [\ell]$ such that $\psi(i) \cap \psi(j) = \emptyset$, $|\psi(i) \cup \psi(j)| = 2\lambda - 2 = \ell - 1$, so there is an edge that intersects both $\psi(i)$ and $\psi(j)$. It is easy to see that $\mathsf{wed}(\psi) = \lambda = (\ell+1)/2$. Thus, $\mathsf{emb} \geq \ell/\lambda = 2\ell/(\ell+1) = 2 - 2/(\ell+1)$.

If ℓ is even, we define $\lambda = \ell/2$ and an embedding from a $(\ell-1)$-clique as follows:

$$\begin{aligned}
\psi^{-1}(x_1) &= \{1, 2, \ldots, \lambda - 1\} \\
\psi^{-1}(x_2) &= \{2, 3, \ldots, \lambda\} \\
&\cdots \\
\psi^{-1}(x_{\ell-1}) &= \{2\lambda - 2, 2\lambda - 1, 1, \ldots, \lambda - 3\} \\
\psi^{-1}(x_\ell) &= \psi^{-1}(x_{\ell-1})
\end{aligned}$$

where $\psi^{-1}(x_i), i \in [\ell-1]$ is exactly the embedding we constructed for $(\ell-1)$-cycle. We show that this is a valid embedding. Let $i, j \in [\ell]$ such that $\psi(i) \cap \psi(j) = \emptyset$.

1. If $i \in \psi^{-1}(x_{\ell-1})$ (or $j \in \psi^{-1}(x_{\ell-1})$), then $|\psi(i) \cup \psi(j)| = \ell - 1$, so there is an edge that intersects both $\psi(i)$ and $\psi(j)$.
2. If $i, j \notin \psi^{-1}(x_{\ell-1})$, then $\psi(i)$ and $\psi(j)$ do not contain $x_{\ell-1}$ and x_ℓ. There is an edge that intersects both $\psi(i)$ and $\psi(j)$ since $|\psi(i) \cup \psi(j)| = \ell - 2$.

For this embedding, we have $\mathsf{wed}(\psi) = \lambda = \ell/2$, so $\mathsf{emb} \geq (\ell-1)/\lambda = 2 - 2/\ell$. ◀

Thus, we have the following proposition.

▶ **Proposition 18.** *Consider the cycle query of length $\ell \geq 3$. Then we have*

$$\mathsf{emb} = \mathsf{subw} = 2 - \frac{1}{\lceil \frac{\ell}{2} \rceil}$$

Proof. It can be shown using Example 7.4 in [17] (setting $m = 1$) that $\mathsf{subw} \leq 2 - 1/\lceil \frac{\ell}{2} \rceil$ (technically the Example 7.4 in [17] only deals with cycles of even length, but their argument can be easily adapted to cycles of odd length). We thus conclude by applying Lemma 17 and Theorem 7. ◀

6.2 Complete Bipartite Graphs

We consider a complete bipartite graph $K_{m,n}$ where the two partitions of its vertices are $\{x_1, \ldots, x_m\}$ and $\{y_1, \ldots, y_n\}$. We study two of its special cases, $K_{2,\ell}$ and $K_{3,3}$. The proofs of the following two propositions can be found in [9].

▶ **Proposition 19.** *For the bipartite graph $K_{2,\ell}$, $\mathsf{emb}(K_{2,\ell}) = \mathsf{subw}(K_{2,\ell}) = 2 - 1/\ell$.*

▶ **Proposition 20.** *For $K_{3,3}$, we have $\mathsf{emb}(K_{3,3}) = \mathsf{subw}(K_{3,3}) = 2$.*

Finding $\mathsf{emb}(K_{m,n})$ and $\mathsf{subw}(K_{m,n})$ in the most general case is still an open question.

6.3 Chordal Queries

In this section, we identify a special class of queries, called *choral queries*. We introduce necessary definitions and lemmas to prove that for a chordal query, emb, subw, fhw, and adw all coincide, as stated in Theorem 24.

Let G be a graph. A chord of a cycle C of G is an edge that connects two non-adjacent nodes in C. We say that G is *chordal* if any cycle in G of length greater than 3 has a chord. We can extend chordality to hypergraphs by considering the clique-graph of a hypergraph $\mathcal{H}$, where edges are added between all pairs of vertices contained in the same hyperedge.

Let $(\mathcal{T}, \chi)$ be a tree decomposition of a hypergraph $\mathcal{H}$ and $\mathsf{bags}(\mathcal{T}) \stackrel{\text{def}}{=} \{\chi(t) \mid t \in V(\mathcal{T})\}$. We say that $(\mathcal{T}, \chi)$ is *proper* if there is no tree decomposition $(\mathcal{T}', \chi')$ such that

1. for every bag $b_1 \in \mathsf{bags}(\mathcal{T}')$, there is a bag $b_2 \in \mathsf{bags}(\mathcal{T})$ such that $b_1 \subseteq b_2$;
2. $\mathsf{bags}(\mathcal{T}') \not\supseteq \mathsf{bags}(\mathcal{T})$,

The following important properties hold for chordal graphs.

▶ **Lemma 21** ([5]). *If G is a chordal graph and $(\mathcal{T}, \chi)$ is a proper tree decomposition of G, then the bags of $(\mathcal{T}, \chi)$, i.e. $\mathsf{bags}(\mathcal{T})$, are the maximal cliques in G.*

For chordal hypergraphs, we can show the following lemma:

▶ **Lemma 22.** *Let $\mathcal{H}$ be a hypergraph. Then, $(\mathcal{T}, \chi)$ is a (proper) tree decomposition of $\mathcal{H}$ if and only if it is a (proper) tree decomposition of the clique-graph of $\mathcal{H}$.*

Proof. We first show that $(\mathcal{T}, \chi)$ is a tree decomposition of $\mathcal{H}$ if and only if it is also a tree decomposition of the clique-graph of $\mathcal{H}$. The forward direction is straightforward. For the backward direction, let $(\mathcal{T}, \chi)$ be a decomposition of the clique-graph of $\mathcal{H}$. Then for any hyperedge $e \in \mathcal{H}$ and any pair of vertices $u, v \in e$, we know that $\{t \mid u \in \chi(t)\} \cap \{t \mid v \in \chi(t)\} \neq \emptyset$. By the *Helly Property*, there is a bag that contains all vertices in the hyperedge e. Therefore, $(\mathcal{T}, \chi)$ is a tree decomposition for $\mathcal{H}$. It is easy to extend the proof for proper tree decompositions. ◀

The following corollary is immediate from both Lemma 21 and Lemma 22:

▶ **Corollary 23.** *If $\mathcal{H}$ is a chordal hypergraph and $(\mathcal{T}, \chi)$ is a proper tree decomposition of $\mathcal{H}$, then the bags of $(\mathcal{T}, \chi)$ are the maximal cliques in the clique-graph of $\mathcal{H}$.*

The above corollary tells us that every proper tree decomposition has the same set of bags, with the only difference being the way the bags are connected in the tree. From this, we can easily obtain that $\mathsf{subw} = \mathsf{fhw}$. However, we have an even stronger result:

▶ **Theorem 24.** *If $\mathcal{H}$ is a chordal hypergprah, then* $\mathsf{emb} = \mathsf{adw} = \mathsf{subw} = \mathsf{fhw}$.

Proof. Since $\mathcal{H}$ is chordal, by Corollary 23, the bags of any proper tree decomposition $(\mathcal{T}, \chi)$ of $\mathcal{H}$ are the maximal cliques in the clique-graph of $\mathcal{H}$. Then, there is a node $t \in V(\mathcal{T})$ such that the minimum fractional edge cover (also the maximum fractional vertex packing) of $\chi(t)$ is fhw. In particular, let $\{u_i \mid i \in \chi(t)\}$ be the optimal weights assigned to each vertex in $\chi(t)$ that obtain the maximum fractional vertex packing, so $\sum_{i \in \chi(t)} u_i = \mathsf{fhw}$. We let $\hat{u}_i = u_i / \sum_{i \in \chi(t)} u_i$ and k be the smallest integer such that $k \cdot \hat{u}_i$ is an integer for every $i \in \chi(t)$. Now we construct an embedding ψ from C_k to $\mathcal{H}$ so that every $\psi(j)$, for $j \in [k]$ is a singleton and for each $i \in \chi(t)$, let $d_\psi^{-1}(i) \stackrel{\text{def}}{=} k \cdot \hat{u}_i$. This assignment uses up all k vertices in C_k, since $\sum_{i \in \chi(t)} k \cdot \hat{u}_i = k$. Then,

$$\mathsf{wed}(\psi) = \max_{e \in E(\mathcal{H})} \sum_{i \in e} k \cdot \hat{u}_i = \frac{k}{\sum_{i \in \chi(t)} u_i} \cdot \max_{e \in E(\mathcal{H})} \sum_{i \in e} u_i \leq \frac{k}{\sum_{i \in \chi(t)} u_i}$$

and thus, we get $\mathsf{emb} = \mathsf{fhw}$ since

$$\mathsf{emb} \geq \mathsf{emb}(C_k \mapsto \mathcal{H}) \geq \frac{k}{\mathsf{wed}(\psi)} \geq \sum_{i \in \chi(t)} u_i = \mathsf{fhw}.$$

For adw, we define the following modular function over subsets of $V(\mathcal{H})$: for any $S \subseteq V(\mathcal{H})$, let $\mu(S) \stackrel{\text{def}}{=} \sum_{i \in S} u_i$. It is edge-dominated since for every hyperedge e, $\mu(e) = \sum_{i \in e} u_i \leq 1$. Moreover, we have that $\mu(\chi(t)) = \sum_{i \in \chi(t)} u_i = \mathsf{fhw}$. That is, $\mathsf{adw} \geq \mathsf{fhw}$, so $\mathsf{adw} = \mathsf{fhw}$. As a remark, it is also viable to use Lemma 3.1 in [17] to prove the claim for adaptive width. ◀

Recall that Corollary 23 implies if $\mathcal{H}$ is chordal, then every proper tree decomposition of $\mathcal{H}$ has the same set of bags. We show the converse is also true, which could be of independent interest. The proof is in [9].

▶ **Lemma 25.** *Let $\mathcal{H}$ be a hypergraph. If every proper tree decomposition of $\mathcal{H}$ has the same set of bags, then $\mathcal{H}$ is chordal.*

We identify three classes of hypergraphs (almost-cliques, hypercliques, and acyclic hypergraphs) that are chordal and find their clique embedding powers and submodular widths.

Almost-cliques. Consider the ℓ-clique where one vertex, say x_1, connects to k vertices only, where $1 \leq k < \ell - 1$ (hence it is the missing edges from being a ℓ-clique). We denote such a hypergraph as A_ℓ. To show that A_ℓ is chordal, we observe that for any cycle of length ≥ 4 that contains x_1, the two adjacent vertices of x_1 in the cycle must be connected by an edge in A_ℓ and that edge is a chord to the given cycle. We also show the following proposition.

▶ **Proposition 26.** *For an almost-cliques A_ℓ, $\mathsf{emb} = \mathsf{subw} = \mathsf{fhw} = (\ell - 1)/2$.*

Proof. To prove the claim, suppose WLOG x_i connects only to x_i, where $i \in 2, 3, \ldots, k$. Then, we take the decomposition with two bags: $\{x_1, x_2, \ldots, x_k\}$, $\{x_2, x_3, \ldots, x_\ell\}$, where each bag has an edge cover of at most $(\ell - 1)/2$ since the first bag induces an k-clique and the second bag induces an $(\ell - 1)$-clique. Hence, $\mathsf{fhw} \leq (\ell - 1)/2$.

On the other hand, consider the embedding ψ from the $(\ell - 1)$-clique, where $\psi(i) = x_{i+1}$, $1 \leq i \leq \ell - 1$; it is easy to verify that this is a valid embedding such that $\mathsf{wed}(\psi) = 2$, hence $\mathsf{emb} \geq (\ell - 1)/2$. Therefore, we have shown that $\mathsf{emb} = \mathsf{subw} = (\ell - 1)/2$ by Theorem 7. ◀

Hypercliques. Next, we consider the (ℓ, k)-hyperclique query $\mathcal{H}_{\ell,k}$, where $1 < k \leq \ell$. This query has ℓ variables, and includes as atoms all possible subsets of $\{x_1, \ldots, x_\ell\}$ of size exactly k. When $k = \ell - 1$, the query simply becomes a Loomis-Whitney join [22]. It is easy to see that $\mathcal{H}_{\ell,k}$ is chordal since the clique-graph of $\mathcal{H}_{\ell,k}$ is a ℓ-clique.

▶ **Proposition 27.** *For $\mathcal{H}_{\ell,k}$, we have $\mathsf{emb} = \mathsf{subw} = \mathsf{fhw} = \ell/k$.*

Proof. First, we show that $\mathsf{fhw} \leq \ell/k$. Indeed, there is a fractional edge cover that assigns a weight of $1/k$ to each hyperedge that contains k consecutive vertices in $\{x_1, \ldots, x_\ell\}$ (let the successor of x_ℓ be x_1). The fractional edge cover is then ℓ/k. We show next that this bound coincides with emb.

We simply define the embedding ψ from a ℓ-clique as $\psi(i) = x_i, i \in [\ell]$. Then, $\mathsf{wed}(\psi) = k$ since every hyperedge has exactly k vertices. Therefore, $\mathsf{emb} \geq \ell/k \geq \mathsf{fhw}$ and we conclude by applying Lemma 17 and Theorem 7. ◀

Acyclic Hypergraphs. First, we claim that acyclic queries are indeed chordal queries.

▶ **Lemma 28.** *An acyclic hypergraph $\mathcal{H}$ is chordal.*

Proof. We prove by induction on the number of hyperedges in the hypergraph $\mathcal{H} = (V, E)$. If $|E| = 1$, it is clique-graph is a clique, thus it is chordal. The induction hypothesis assumes the claim for acyclic hypergraphs with $|E| \leq k$ hyperedges. Let $\mathcal{H}$ be an acyclic hypergraph such that $|E| = k+1$. Since $\mathcal{H}$ is acyclic, it has a join forest whose vertices are the hyperedges of $\mathcal{H}$. Let $e_\ell \in E$ be a leaf of the join forest and it is easy to show that $\mathcal{H}' = (V, E \setminus \{e_\ell\})$ is an acyclic hypergraph with k hyperedges. For any cycle in the clique-graph of $\mathcal{H}$ having length ≥ 4, we discuss the following two cases.

If every edge of the cycle is in the clique-graph of $\mathcal{H}'$: by the induction hypothesis, there is a chord for this cycle in the clique-graph of $\mathcal{H}'$ (thus also in $\mathcal{H}$).

Otherwise, there is an edge e in the cycle that is in the clique-graph of $\mathcal{H}$, but not in the clique-graph of $\mathcal{H}'$: therefore, the edge e is only contained by e_ℓ. This implies that there is a vertex u that is only contained by e_ℓ, not by any other edges in E. Let $\{u, v\}$ and $\{u, w\}$ be the edges connecting u in the given cycle, we know that $\{u, v, w\} \subseteq e_\ell$ and thus, $\{v, w\}$ is a chord for the given cycle. ◀

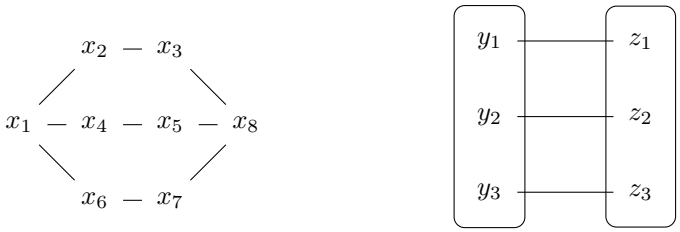
Now we prove the following theorem for acyclic hypergraphs:

▶ **Theorem 29.** *For an acyclic hypergprah $\mathcal{H}$,* $\mathsf{emb} = \mathsf{adw} = \mathsf{subw} = \mathsf{fhw} = 1$.

Proof. From Proposition 5, we know that $\mathsf{emb} \geq 1$. Since it is known that $\mathsf{subw} = \mathsf{fhw} = 1$, the theorem is then a direct result from Lemma 28 and Theorem 24. ◀

7 Gap Between Clique Embedding Power and Submodular Width

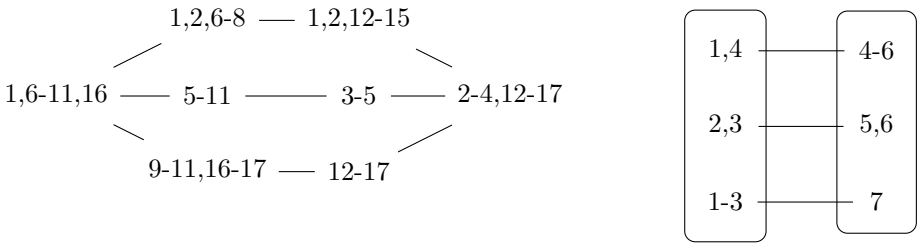
In this section, we discuss the boat query and its variant depicted in Figure 2, where gaps between the clique embedding power and submodular width can be shown.



(a) Boat Query Q_b. **(b)** Hyper-boat Query Q_{hb}.

Figure 2 The Boat query and its variant, the Hyper-boat Query.

7.1 Clique Embedding Power, Submodular Width and Adaptive Width



(a) emb for Q_b. **(b)** emb for Q_{hb}.

Figure 3 Optimal embedding for the boat query and its variants.

Using MILP (2), we find the optimal clique embedding for Q_b and Q_{hb}, as illustrated in Figure 3. The numbers represent the vertices from the clique, and we adopt the shorthand notation, say, 6-8 to refer to the set {6,7,8}. The clique embedding powers for Q_b and Q_{hb} are $\frac{17}{9}$ and $\frac{7}{4}$, respectively. However, [13] proves that for the boat query, $\mathsf{subw}(Q_b) = 2$. This

implies a gap since $\mathsf{emb}(Q_b) = 17/9 < \mathsf{subw}(Q_b) = 2$. We show that for the hyper-boat query Q_{hb}, there is also a gap between the optimal clique embedding power and submodular width. In particular, we show that $\mathsf{subw}(Q_{hb}) = 2$ in the following proposition, which implies the following gap: $\mathsf{emb}(Q_{hb}) = \frac{7}{4} < \mathsf{subw}(Q_{hb}) = 2$. Its proof can be found in [9].

▶ **Proposition 30.** *For Q_{hb}, we have $\mathsf{subw}(Q_{hb}) = 2$.*

7.2 Subquadratic Equivalence Between Boat Queries

In this section, we demonstrate an interesting connection between the two boat queries. To start, let's consider Q_b and Q_{hb}. Both queries admit an algorithm that runs in time $O(|I|^2)$. Informally, we are going to show that either both queries can be executed significantly faster, or neither can. Following the seminal paper by Williams and Williams [25], we define *truly subquadratic algorithm* and *subquadratic equivalence*.

▶ **Definition 31.** *An algorithm is said to be* truly subquadratic *if it runs in time $O(m^{2-\epsilon})$ for some $\epsilon > 0$ (m is the input size).*

Two problems A and B are *subquadratic equivalent* if A admits a truly subquadratic algorithm iff B admits a truly subquadratic algorithm. We show that the two boat queries are subquadratic equivalent.

▶ **Theorem 32.** *Q_b is subquadratic equivalent to Q_{hb}.*

Proof. It's easy to see that a truly subquadratic algorithm for Q_b gives a truly subquadratic algorithm for Q_{hb}. Indeed, given an input instance I_{hb} of Q_{hb}, we can form an input instance I_b of Q_b where the table (x_1, x_2) is the projection of the table (y_1, y_2, y_3) in I_{hb}, and similar for the tables $(x_1, x_4), (x_1, x_6), (x_3, x_8), (x_5, x_8)$ and (x_7, x_8). We then solve I_b by the algorithm for Q_b. It is easy to see that this algorithm is correct and runs in truly quadratic time.

The converse direction needs more work, since if we were to simply create the table (y_1, y_2, y_3) for Q_{hb} by joining the tables $(x_1, x_2), (x_1, x_4)$ and (x_1, x_6) for Q_b, the size of the result might be significantly greater than all previous tables. For example, if the sizes of the tables $(x_1, x_2), (x_1, x_4)$ and (x_1, x_6) are all m, then joining them could result in a table of size $m^{\frac{3}{2}}$ and therefore calling the algorithm for Q_{hb} on this instance does not necessarily yield a truly subquadratic algorithm for Q_b.

We perform our fine-grained reduction based on *heavy-light split*. Our goal is to give a subquadratic algorithm for Q_b assuming there is one such algorithm for Q_{hb}. Suppose the subquadratic algorithm for Q_b runs in time $O(m^{2-\delta})$ for some $\delta > 0$, where m is the size of all tables. Our algorithm for Q_b runs as follows. First, it checks whether there are entries of attribute x_1 that has degree more than $\Delta := m^\epsilon$ in tables $(x_1, x_2), (x_1, x_4)$ and (x_1, x_6) for some $\epsilon > 0$ to be specified later. Those are called *heavy* and there are at most $\frac{m}{\Delta}$ many of them. For those entries, we fix each one so that the remaining query becomes acyclic, and thus can be solved in linear time by Yannakakis algorithm [27]. We do the same procedure for heavy entries of attribute x_8. Therefore, any result of Q_b that contains a heavy entry in attributes x_1 or x_8 will be detected in time $O(m^{2-\epsilon})$. It remains to consider the case where the entries of attributes x_1 and x_8 have degrees less than Δ, which are called *light*. In this case, we loop over all light entries of x_1 in the table (x_1, x_2) and directly join them with the tables (x_1, x_4) and (x_1, x_6) and project the result to build a table (x_2, x_4, x_6). We then do the same procedure for joining x_8. This will cost time $O(m \cdot \Delta \cdot \Delta) = O(m^{1+2\epsilon})$. We then call the $O(m^{2-\delta})$ algorithm for Q_{hb}, which cost time $O(m^{(1+2\epsilon)(2-\delta)})$. By choosing $0 < \epsilon < \frac{\delta}{4-2\delta}$ (note that $\delta < 2$), we observe that the whole algorithm for Q_b takes time $O(m^{2-\epsilon}) + O(m^{(1+2\epsilon)(2-\delta)}) = O(m^{2-\epsilon'})$ for some $\epsilon' > 0$. ◀

We remark that the reduction from Q_{hb} to Q_b is parametrized by the running time of the algorithm for Q_{hb}. That is, the reduction is not uniform in the sense that only after given $\delta > 0$ can we specify a suitable ϵ. Theorem 32 implies that either both boat queries admit a truly subquadratic algorithm or none of them does.

The fact that there is a gap between $\mathsf{subw}(Q_{hb}) = 2$ and $\mathsf{emb}(Q_{hb}) = \frac{7}{4}$ suggests currently our lower bound does not match with the best upper bound, i.e., PANDA. This implies either that PANDA is not universally optimal, or that we are missing the best possible lower bound. We leave this as an open question.

Finally, we note that Theorem 4 in [13], which proves there does not exist a $\tilde{O}(m^{2-\epsilon} + |\text{OUT}|^5)$ algorithm for the boat query unless 3-XOR can be solved in time $\tilde{O}(m^{2-t})$ for a $t > 0$, does not directly translate into the quadratic hardness for the boat query in our case. This is because their reduction uses the output of the boat query in an essential way to "hack back the collision" which is not available in the Boolean case.

8 Related Work

Fine-Grained Complexity. The study of fine-grained complexity aims to show the (conditional) hardness of easy problems. Recent years have witnessed a bloom of development into this fascinating subject, resulting in many tight lower bounds which match exactly, or up to poly log factors, the running time of best-known algorithms [18, 26, 25, 1]. Among many others, popular hardness assumptions include the Strong Exponential Time Hypothesis (SETH), Boolean Matrix Multiplication (BMM), and All-Pairs Shortest Paths (APSP). Our work can be seen as a particular instance under this framework, i.e. using Boolean or Min-Weight k-Clique Conjecture to show conditional lower bounds for BCQs. Interestingly, our reduction of k-cycles essentially mirrors the construction in the proof of Theorem 3.1 in [18].

Conjunctive Queries (CQs) Evaluation. The efficient evaluation of CQs constitutes the core theme of database theory. Khamis, Ngo, and Suciu introduced in [17] the PANDA algorithm that runs in time as predicted by the submodular width of the query hypergraph. This groundbreaking result establishes a profound connection between various lines of work on tree decompositions [19, 20], worst-case optimal join algorithms [23, 22], and the interplay between CQ evaluation and information theory [15, 28, 14].

Functional Aggregate Queries (FAQs). FAQs [16] provides a Sum-of-Product framework that captures the semantics of conjunctive queries over arbitrary semirings. The semiring point-of-view originated from the seminal paper [10]. Khamis, Ngo, and Rudra [16] initiate the study of the efficient evaluation of FAQs. [14] introduces the FAQ version of the submodular width #subw and the #PANDA algorithm (as the FAQ version of the PANDA algorithm) that achieves the runtime as predicated by #subw. We show that the embedding from a k-clique into a hypergraph holds for arbitrary semirings, which enables one to transfer the hardness of k-clique to FAQ independent of the underlying semiring. To the best of our knowledge, this is the first *semiring-oblivious* reduction.

Enumeration and Preprocessing. Bagan, Durand and Grandjean characterized in [3] when a constant delay and linear preprocessing algorithm for self-join-free conjunctive queries is possible. A recent paper [6] makes an initial foray towards the characterization of conjunctive

[5] |OUT| is the size of the output.

queries with self-joins. Also recently, [4] identifies new queries which can be solved with linear preprocessing time and constant delay. Their hardness results are based on the Hyperclique conjecture, the Boolean Matrix Multiplication conjecture, and the 3SUM conjecture.

9 Conclusion

In this paper, we study the fine-grained complexity of BCQs. We give a semiring-oblivious reduction from the k-clique problem to an arbitrary hypergraph. Assuming the Boolean k-Clique Conjecture, we obtain conditional lower bounds for many queries that match the combinatorial upper bound achieved by the best-known algorithms, possibly up to a poly-logarithmic factor.

One attractive future direction is to fully unravel the gap between the clique embedding power and submodular width, where improved lower bounds or upper bounds are possible. The Boolean k-Clique Conjecture states that there is no $O(n^{k-\epsilon})$ *combinatorial* algorithm for detecting k-cliques. One future direction is to base the hardness assumption over Nešetřil and Poljak's algorithm [21], which solves the k-clique problem in $O\left(n^{(\omega/3)k}\right)$ by leveraging fast matrix multiplication techniques and show lower bounds for any algorithm.

References

1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is Valiant's parser. In *FOCS*, pages 98–117. IEEE Computer Society, 2015.

2 Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.

3 Guillaume Bagan, Arnaud Durand, and Etienne Grandjean. On acyclic conjunctive queries and constant delay enumeration. In *CSL*, volume 4646 of *Lecture Notes in Computer Science*, pages 208–222. Springer, 2007.

4 Karl Bringmann and Nofar Carmeli. Unbalanced triangle detection and enumeration hardness for unions of conjunctive queries. *CoRR*, abs/2210.11996, 2022. `arXiv:2210.11996`.

5 Nofar Carmeli, Batya Kenig, Benny Kimelfeld, and Markus Kröll. Efficiently enumerating minimal triangulations. *Discret. Appl. Math.*, 303:216–236, 2021.

6 Nofar Carmeli and Luc Segoufin. Conjunctive queries with self-joins, towards a fine-grained complexity analysis. *CoRR*, abs/2206.04988, 2022. `arXiv:2206.04988`.

7 Katrin Casel and Markus L. Schmid. Fine-grained complexity of regular path queries. In *ICDT*, volume 186 of *LIPIcs*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

8 Arnaud Durand. Fine-grained complexity analysis of queries: From decision to counting and enumeration. In *PODS*, pages 331–346. ACM, 2020.

9 Austen Z. Fan, Paraschos Koutris, and Hangdong Zhao. The fine-grained complexity of boolean conjunctive queries and sum-product problems. *CoRR*, 2023. `arXiv:2304.14557`.

10 Todd J. Green, Gregory Karvounarakis, and Val Tannen. Provenance semirings. In *PODS*, pages 31–40. ACM, 2007.

11 Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *Journal of the ACM (JACM)*, 54(1):1–24, 2007.

12 Pinar Heggernes. Treewidth, partial k-trees, and chordal graphs. *Partial curriculum in INF334-Advanced algorithmical techniques, Department of Informatics, University of Bergen, Norway*, 2005.

13 Manas Joglekar and Christopher Ré. It's all a matter of degree - using degree information to optimize multiway joins. *Theory Comput. Syst.*, 62(4):810–853, 2018.

14 Mahmoud Abo Khamis, Ryan R. Curtin, Benjamin Moseley, Hung Q. Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. On functional aggregate queries with additive inequalities. In *PODS*, pages 414–431. ACM, 2019.

15 Mahmoud Abo Khamis, Phokion G. Kolaitis, Hung Q. Ngo, and Dan Suciu. Bag query containment and information theory. In *PODS*, pages 95–112. ACM, 2020.

16 Mahmoud Abo Khamis, Hung Q. Ngo, and Atri Rudra. FAQ: questions asked frequently. In *PODS*, pages 13–28. ACM, 2016.

17 Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu. What do shannon-type inequalities, submodular width, and disjunctive datalog have to do with one another? In *PODS*, pages 429–444. ACM, 2017.

18 Andrea Lincoln, Virginia Vassilevska Williams, and R. Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In *SODA*, pages 1236–1252. SIAM, 2018.

19 Dániel Marx. Can you beat treewidth? *Theory Comput.*, 6(1):85–112, 2010.

20 Dániel Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. *J. ACM*, 60(6):42:1–42:51, 2013.

21 Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

22 Hung Q. Ngo, Ely Porat, Christopher Ré, and Atri Rudra. Worst-case optimal join algorithms. *J. ACM*, 65(3):16:1–16:40, 2018.

23 Hung Q. Ngo, Christopher Ré, and Atri Rudra. Skew strikes back: new developments in the theory of join algorithms. *SIGMOD Rec.*, 42(4):5–16, 2013.

24 Alexander Schrijver. *Theory of linear and integer programming*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1999.

25 Virginia Vassilevska Williams and R. Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM*, 65(5):27:1–27:38, 2018.

26 Virginia Vassilevska Williams and Yinzhan Xu. Monochromatic triangles, triangle listing and APSP. In *FOCS*, pages 786–797. IEEE, 2020.

27 Mihalis Yannakakis. Algorithms for acyclic database schemes. In *VLDB*, pages 82–94. IEEE Computer Society, 1981.

28 Hangdong Zhao, Shaleen Deep, and Paraschos Koutris. Space-time tradeoffs for conjunctive queries with access patterns. *CoRR*, 2023. `arXiv:2304.06221`.

Flipper Games for Monadically Stable Graph Classes

Jakub Gajarský
University of Warsaw, Poland
Rose McCarty
Princeton University, NJ, USA
Michał Pilipczuk
University of Warsaw, Poland
Sebastian Siebertz
Universität Bremen, Germany
Szymon Toruńczyk
University of Warsaw, Poland
Nikolas Mählmann
Universität Bremen, Germany
Pierre Ohlmann
University of Warsaw, Poland
Wojciech Przybyszewski
University of Warsaw, Poland
Marek Sokołowski
University of Warsaw, Poland

Abstract

A class of graphs $\mathcal{C}$ is *monadically stable* if for every unary expansion $\widehat{\mathcal{C}}$ of $\mathcal{C}$, one cannot encode – using first-order transductions – arbitrarily long linear orders in graphs from $\widehat{\mathcal{C}}$. It is known that nowhere dense graph classes are monadically stable; these include classes of bounded maximum degree and classes that exclude a fixed topological minor. On the other hand, monadic stability is a property expressed in purely model-theoretic terms that is also suited for capturing structure in dense graphs.

In this work we provide a characterization of monadic stability in terms of the *Flipper game*: a game on a graph played by *Flipper*, who in each round can complement the edge relation between any pair of vertex subsets, and *Localizer*, who in each round is forced to restrict the game to a ball of bounded radius. This is an analog of the *Splitter game*, which characterizes nowhere dense classes of graphs (Grohe, Kreutzer, and Siebertz, J. ACM '17).

We give two different proofs of our main result. The first proof is based on tools borrowed from model theory, and it exposes an additional property of monadically stable graph classes that is close in spirit to definability of types. Also, as a byproduct, we show that monadic stability for graph classes coincides with monadic stability of existential formulas with two free variables, and we provide another combinatorial characterization of monadic stability via forbidden patterns. The second proof relies on the recently introduced notion of *flip-flatness* (Dreier, Mählmann, Siebertz, and Toruńczyk, arXiv 2206.13765) and provides an efficient algorithm to compute Flipper's moves in a winning strategy.

2012 ACM Subject Classification Theory of computation → Finite Model Theory; Mathematics of computing → Graph theory

Keywords and phrases Stability theory, structural graph theory, games

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.128

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `https://arxiv.org/abs/2301.13735` [11]

Funding This work is a part of projects CUTACOMBS (R. McCarty) and BOBR (J. Gajarský, P. Ohlmann, M. Pilipczuk, W. Przybyszewski, M. Sokołowski and S. Toruńczyk) that have received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreements No. 714704 and 948057, respectively). Nikolas Mählmann and Sebastian Siebertz are supported by the German Research Foundation (DFG) with grant agreement No. 444419611. Rose McCarty is also supported by National Science Foundation (NSF) grant No DMS-2202961.

1 Introduction

Monadic stability is a notion of logical tameness for classes of structures. Introduced by Baldwin and Shelah [3] in the context of model theory[1], it has recently attracted attention in the field of structural graph theory. We recall the definition below. One of the main contributions of this paper is to provide purely combinatorial characterizations of monadically stable classes of graphs via games and via forbidden patterns. Our game characterization is effective, and can be employed in algorithmic applications, as we explain later.

In this paper we focus on (undirected, simple) graphs, rather than arbitrary structures. A graph is modelled as a relational structure with one symmetric binary relation signifying adjacency. By a *class* of graphs we mean any set of graphs. For a class of graphs $\mathcal{C}$, a *unary expansion* of $\mathcal{C}$ is any class $\widehat{\mathcal{C}}$ of structures such that each $\widehat{G} \in \widehat{\mathcal{C}}$ is obtained from some graph in $G \in \mathcal{C}$ by adding some unary predicates. Thus, the elements of $\widehat{\mathcal{C}}$ can be regarded as vertex-colored graphs from $\mathcal{C}$. A class of graphs $\mathcal{C}$ is called *monadically stable* if one cannot interpret, using a fixed formula $\varphi(\bar{x}, \bar{y})$ of first-order logic, arbitrarily long linear orders in any unary expansion $\widehat{\mathcal{C}}$ of $\mathcal{C}$. More precisely, for every unary expansion $\widehat{\mathcal{C}}$ and formula $\varphi(\bar{x}, \bar{y})$ with $|\bar{x}| = |\bar{y}|$ (over the signature of $\widehat{\mathcal{C}}$) there is a bound ℓ such that there is no structure $\widehat{G} \in \widehat{\mathcal{C}}$ and tuples $\bar{a}_1, \ldots, \bar{a}_\ell \in V(\widehat{G})^{\bar{x}}$ such that $\widehat{G} \models \varphi(\bar{a}_i, \bar{a}_j)$ if and only if $i \leqslant j$. More generally, $\mathcal{C}$ is *monadically dependent* (or *monadically NIP*) if one cannot interpret, using a fixed formula $\varphi(\bar{x}, \bar{y})$ of first-order logic, all finite graphs in any unary expansion of $\mathcal{C}$. Thus, from the model-theoretic perspective, the intuition is that being monadically dependent is being non-trivially constrained: for any fixed interpretation, one cannot interpret arbitrarily complicated structures in vertex-colored graphs from the considered class. On the other hand, graphs from monadically stable classes are "orderless", in the sense that one cannot totally order any large part of them using a fixed first-order formula.

Baldwin and Shelah proved that in the definitions, one can alternatively consider only formulas $\varphi(x, y)$ with just a pair of free variables, instead of a pair of tuples of variables [3, Lemma 8.1.3, Theorem 8.1.8]. Moreover, they proved that monadically stable theories are *tree decomposable* [3, Theorem 4.2.17], providing a structure theorem for such theories, although one of a very infinitary nature. A more explicit, combinatorial structure theorem for monadically stable and monadically dependent is desirable for obtaining algorithmic results for the considered classes, as we discuss later.

On the other hand, Braunfeld and Laskowski [6] very recently proved that for *hereditary* classes of structures $\mathcal{C}$ that are not monadically stable or monadically dependent, the required obstructions (total orders or arbitrary graphs) can be exhibited by a boolean combination of existential formulas $\varphi(\bar{x}, \bar{y})$ in the signature of $\mathcal{C}$, without any additional unary predicates. Among other things, this shows that for hereditary classes of structures, the notions of monadic stability coincides with the more well-known notion of stability, and similarly, monadic dependence coincides with dependence (NIP). Furthermore, since the formulas are existential, this result can be seen as a combinatorial non-structure theorem for hereditary classes that are not monadically stable (resp. monadically dependent). Still, they do not provide explicit structural results for classes that are monadically stable or monadically dependent.

[1] Formally, Baldwin and Shelah [3], as well as Braunfeld and Laskowski [6], study monadically dependent and monadically stable *theories*, rather than classes of structures. Some of their results transfer to the more general setting of monadically dependent/stable classes of structures.

Explicit, combinatorial and algorithmic structural results for monadically dependent and monadically stable classes are not only desired, but also expected to exist, based on the known examples of such classes that have been studied in graph theory and computer science. As observed by Adler and Adler [2] based on the work of Podewski and Ziegler [16], all *nowhere dense* graph classes are monadically stable. A class $\mathcal{C}$ is nowhere dense if for every fixed $r \in \mathbb{N}$, one cannot find r-subdivisions of arbitrarily large cliques as subgraphs of graphs in $\mathcal{C}$. In particular, every class excluding a fixed topological minor (so also the class of planar graphs, or the class of subcubic graphs) is monadically stable. In fact, it follows from the results of Adler and Adler [2] and of Dvořák [10] that monadic stability and monadic dependence are both equivalent to nowhere denseness when considering only sparse classes of graphs (formally, classes of graphs that excludes a fixed biclique as a subgraph). However, monadic stability and monadic dependence are not bound to sparsity; they can be used to understand and quantify structure in dense graphs as well.

The pinnacle of the theory of nowhere dense graph classes is the result of Grohe, Kreutzer, and Siebertz [14] that the model-checking problem for first-order logic is fixed-parameter tractable on any nowhere dense class of graphs.

▶ **Theorem 1** ([14]). *For every nowhere dense graph class $\mathcal{C}$, first-order sentence φ, and $\varepsilon > 0$, there exists an algorithm that given an n-vertex graph $G \in \mathcal{C}$ decides whether $G \models \varphi$ in time $\mathcal{O}_{\mathcal{C},\varphi,\varepsilon}(n^{1+\varepsilon})$.*

Here, and in the following, the notation $\mathcal{O}_p(\cdot)$ hides multiplicative factors that depend only on the the parameter p.

Monadically dependent classes include all monadically stable classes, in particular all nowhere dense classes, but also for instance all classes of bounded twin-width [5]. An analogous result, with $1 + \varepsilon$ replaced by 3, holds for all classes $\mathcal{C}$ of *ordered* graphs[2] of bounded twin-width [4].

In light of the discussion above, monadic stability and monadic dependence seem to be well-behaved generalizations of nowhere denseness that are defined in purely model-theoretic terms; hence these concepts may be even better suited for treating the model-checking problem for first-order logic. This motivated the following conjecture [1], which has been a subject of intensive study over the last few years[3].

▶ **Conjecture 2.** *Let $\mathcal{C}$ be a monadically dependent graph class. There exists a constant $c \in \mathbb{N}$ depending only on $\mathcal{C}$ and, for every first-order sentence φ, an algorithm that, given a n-vertex graph $G \in \mathcal{C}$, decides whether $G \models \varphi$ in time $\mathcal{O}_{\mathcal{C},\varphi}(n^c)$.*

Conjecture 2 is not even resolved for monadically stable classes. To approach this conjecture, it is imperative to obtain explicit, combinatorial structure theorems for monadically stable and in monadically dependent graph classes, with a particular focus on finding analogs of the tools used in the proof of Theorem 1. Our work contributes in this direction. We provide certain recursive tree-like decompositions for graphs in monadically stable graph classes, which can be most intuitively explained in terms of games. On the one hand, our decompositions generalize a similar result for nowhere dense classes, recalled below. On the other hand, they are remininiscent of the tree decomposability property proved by Baldwin

[2] *Ordered graphs* are graphs equipped with a total order.
[3] To the best of our knowledge the conjecture was first explicitly discussed during the open problem session of the Algorithms, Logic and Structure Workshop in Warwick, in 2016, see [1].

and Shelah, but are more explicit and finitary in nature. Furthermore, we provide a characterization of monadic stability via forbidden patterns, similar to the known characterization of nowhere denseness.

Splitter game. The cornerstone of the proof of Theorem 1 is a game-theoretic characterization of nowhere denseness, through the *Splitter game*. This game has a fixed radius parameter $r \in \mathbb{N}$ and is played on a graph G between two players, *Splitter* and *Localizer*, who make moves in rounds alternately. In each round, first Splitter chooses any vertex u and removes it from the graph. Next, Localizer selects any other vertex v, and the game gets restricted to the subgraph induced by the ball of radius r with center at v. The game ends with Splitter's victory when there are no vertices left in the graph.

▶ **Theorem 3** ([14]). *A class $\mathcal{C}$ of graphs is nowhere dense if and only if for every $r \in \mathbb{N}$ there exists $k \in \mathbb{N}$ such that for every $G \in \mathcal{C}$, Splitter can win the radius-r Splitter game on G within k rounds.*

Very roughly speaking, Theorem 3 shows that any graph from a nowhere dense class can be hierarchically decomposed into smaller and smaller parts so that the decomposition has height bounded by a constant k depending only on the class and the locality parameter r. This decomposition is used in the algorithm of Theorem 1 to guide model-checking.

Flipper game. In this work we introduce an analog of the Splitter game for monadically stable graph classes: the *Flipper game*. Similarly to before, the game is played on a graph G and there is a fixed radius parameter $r \in \mathbb{N}$. There are two players, *Flipper* and *Localizer*, which make moves in rounds alternately. In each round, first Flipper selects any pair of vertex subsets A, B (possibly non-disjoint) and applies the *flip* between A and B: inverts the adjacency between any pair (a, b) of vertices with $a \in A$ and $b \in B$. Then Localizer, just as in the Splitter game, selects a ball of radius r, and the game is restricted to the subgraph induced by this ball. The game is won by Flipper once there is only one vertex left. See Figure 1 for an illustration.

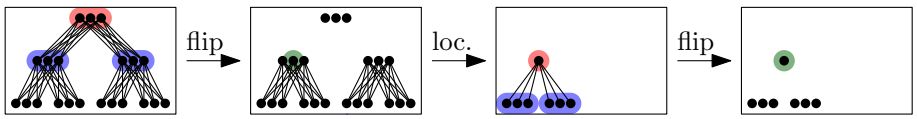


Figure 1 An example play of the radius-1 Flipper game. Taking turns, Flipper flips the red set with the blue set and Localizer restricts to the radius-1 ball centered at the green vertex.

We remark that the Flipper game is a radius-constrained variant of the natural game for graph parameter *SC-depth*, which is functionally equivalent to *shrubdepth*, in the same way that the Splitter game is a radius-constrained variant of the natural game for treedepth. SC-depth and shrubdepth were introduced and studied by Ganian et al. in [13, 12].

Our main result is the following analog of Theorem 3 for monadically stable classes.

▶ **Theorem 4.** *A class $\mathcal{C}$ of graphs is monadically stable if and only if for every $r \in \mathbb{N}$ there exists $k \in \mathbb{N}$ such that for every graph $G \in \mathcal{C}$, Flipper can win the radius-r Flipper game on G within k rounds.*

Let us compare Theorem 4 with another recent characterization of monadic stability, proposed by Gajarský and Kreutzer, and proved by Dreier, Mählmann, Siebertz, and Toruńczyk [9], through the notion of *flip-flatness*. This notion is an analog of *uniform*

quasi-wideness, introduced by Dawar [7]. Without going into technical details, a class of graphs $\mathcal{C}$ is *uniformly quasi-wide* if for any graph $G \in \mathcal{C}$ and any large enough set of vertices A in G, one can find many vertices in A that are pairwise far from each other after the removal of a constant number of vertices from G. As proved by Nešetřil and Ossona de Mendez [15], a class of graphs is uniformly quasi-wide if and only if it is nowhere dense. The definition of flip-flatness is obtained from uniform quasi-wideness similarly as the Flipper game is obtained from the Splitter game: by replacing the concept of deleting a vertex with applying a flip; see Definition 7 for a formal definition. The fact that monadic stability is equivalent to flip-flatness (as proved in [9]) and to the existence of a short winning strategy in the Flipper game (as proved in this paper) suggests the following: the structural theory of monadically stable graph classes mirrors that of nowhere dense graph classes, where the flip operation is the analog of the operation of removing a vertex.

We give two very different proofs of Theorem 4. The first proof is based on elementary model-theoretic techniques, and it provides new insight into the properties of monadically stable graph classes. As a side effect, it gives a new (though non-algorithmic) proof of the main result of [9]: equivalence of monadic stability and flip-flatness. On the other hand, the second proof relies on the combinatorial techniques developed in [9]. It has the advantage of being effective, and provides an efficient algorithm for computing Flipper's moves in a winning strategy.

Forbidden patterns. A class $\mathcal{C}$ of graphs is nowhere dense if for every fixed $r \in \mathbb{N}$ the exact r-subdivision of some clique K_n is not a subgraph of any $G \in \mathcal{C}$, which can be understood as a forbidden pattern characterization. Our model-theoretic proof of Theorem 4 uncovers a similar characterization of monadically stable classes, providing a strong combinatorial non-structure theorem. We prove that a class $\mathcal{C}$ of graphs is monadically stable if and only if there exists a fixed $\ell \in \mathbb{N}$ such that all graphs from $\mathcal{C}$ exclude a ladder of length ℓ as a semi-induced subgraph (see Section 2 for a formal definition), and $\mathcal{C}$ is *pattern-free*. A class $\mathcal{C}$ of graphs is *not* pattern-free if for some $r \geqslant 1, k \in \mathbb{N}$ the exact r-subdivision of every clique K_n can be obtained from an induced subgraph H of some $G \in \mathcal{C}$ by first partitioning $V(H)$ into k parts, and then either flipping the edges, removing all the edges, or inserting all the edges between some pairs of the partition. Equivalently $\mathcal{C}$ is *not* pattern-free if, using a quantifier-free formula $\varphi(x, y)$, one can encode (more formally *transduce*) the class of all r-subdivided cliques for a fixed $r \geqslant 1$.

Model-theoretic proof. The following statement lists properties equivalent to monadic stability uncovered in our model-theoretic proof of Theorem 4. Each condition is shortly explained below the theorem and formally defined in the full version of the paper [11].

▶ **Theorem 5.** *Let $\mathcal{C}$ be a class of graphs. Then the following conditions are equivalent:*

1. *$\mathcal{C}$ is monadically stable.*
2. *$\mathcal{C}$ has a stable edge relation and is monadically dependent with respect to existential formulas $\varphi(x, y)$ with two free variables.*
3. *$\mathcal{C}$ has a stable edge relation and is pattern-free.*
4. *For every $r \in \mathbb{N}$ every model G of the theory of $\mathcal{C}$, every elementary extension H of G, and every vertex $v \in V(H) - V(G)$, there is a finite set $S \subseteq V(G)$ that r-separates v from G.*
5. *For every $r \in \mathbb{N}$ there is $k \in \mathbb{N}$ such that Flipper wins the Flipper Game with qf-definable separation of radius r on every $G \in \mathcal{C}$ in at most k rounds.*

6. *For every $r \in \mathbb{N}$ there is $k \in \mathbb{N}$ such that Flipper wins the Flipper game of radius r on every $G \in \mathcal{C}$ in at most k rounds.*
7. *$\mathcal{C}$ is flip-flat.*

Note that Theorem 4 is the equivalence (1)↔(6). Let us give a brief overview of the presented conditions.

Conditions (1) and (2), respectively, are monadic stability and a weak form of *existential monadic stability*. Recall that Baldwin and Shelah proved that it is sufficient to consider formulas $\varphi(x, y)$ with two free variables in the definition of monadic stability (instead of formulas $\varphi(\bar{x}, \bar{y})$). Braunfeld and Laskowski proved that it is sufficient to consider boolean combinations of existential formulas $\varphi(\bar{x}, \bar{y})$ that do not involve additional unary predicates. The condition (2) lies somewhere in between: it implies that it is sufficient to consider existential formulas $\varphi(x, y)$ with two variables, possibly involving additional unary predicates. In particular, it implies the result of Baldwin and Shelah (in the case of graph classes) and is incomparable with the result of Braunfeld and Laskowski. Our proof uses different techniques.

Condition (3) concerns the combinatorial notion of pattern-freeness discussed earlier.

Condition (4) is phrased in the language of model theory and serves a key role in our proof. It resembles a fundamental property called "definability of types", and in essence it says the following: whenever working with a model G of the theory of $\mathcal{C}$, every element of any elementary extension of G can be robustly "controlled" by a finite subset of G. We believe that the new notion of r-separation used here is of independent interest. It refers to non-existence of short paths after applying some flips governed by S.

Conditions (5) and (6) assert the existence of a short winning strategy in two variants of the Flipper game.

Finally, condition (7) is the notion of flip-flatness, whose equivalence with monadic stability was proved by Dreier et al. [9].

Algorithmic proof. We also give a purely combinatorial proof of the forward implication of Theorem 4, which in particular provides a way to efficiently compute Flipper's moves in a winning strategy. Formally, we show the following.

▶ **Theorem 6.** *Let $\mathcal{C}$ be a monadically stable class of graphs. Then for every radius $r \in \mathbb{N}$ there exist $k \in \mathbb{N}$ and a Flipper strategy* flip* *such that the following holds:*

- *When playing according to* flip* *in the Flipper game of radius r on any graph $G \in \mathcal{C}$, Flipper wins within at most k rounds.*
- *Each move of* flip* *on an n-vertex graph $G \in \mathcal{C}$ can be computed in time $\mathcal{O}_{\mathcal{C},r}(n^2)$.*

The main idea behind the proof of Theorem 6 is to rely on the result of Dreier et al. that monadically stable graph classes are flip-flat [9]. Using the combinatorial tools developed in [9], we strengthen this property: we prove that the set of flips F whose application uncovers a large *scattered set* Y (a set of vertices that are pairwise far from each other) can be selected in a somewhat canonical way, so that knowing any 5-tuple of vertices in Y is enough to uniquely determine F. We can then use such strengthened flip-flatness to provide a winning strategy for Flipper; this roughly resembles the Splitter's strategy used by Grohe et al. in their proof of Theorem 3, which in turn relies on uniform quasi-wideness.

Theorem 6, the algorithmic version of Theorem 4, is the key to algorithmic applications of the Flipper game. In particular, it was very recently used by Dreier, Mählmann, and Siebertz [8] to approach the first-order model checking problem on monadically stable graph classes and prove that it is fixed-parameter tractable on structurally nowhere dense classes, an important subclass of monadically stable classes.

Organization. After introducing monadic stability and the Flipper game in the next section, we give an outline of the model theoretic proof (Section 3) and the algorithmic proof (Section 4). We refer to the appended full version for details.

2 Preliminaries

All graphs in this paper are simple and loopless but not necessarily finite. For a vertex v of a graph G, we write $N(v)$ for the (open) neighborhood of v in G; so $N(v) := \{u \in V(G) \mid uv \in E(G)\}$. For two sets $X, Y \subseteq V(G)$ the bipartite graph *semi-induced* by X and Y in G, denoted $G[X,Y]$, is the bipartite graph with parts X and Y, and edges uv for $u \in X$, $v \in Y$ with $uv \in E(G)$. For vertices $a, b \in V(G)$, an (a,b)-*path* is a path with ends a and b. Similarly, for sets $A, B \subseteq V(G)$, an (A,B)-*path* is a path where one end is in A and the other end is in B.

Model theory. We work with first-order logic over a fixed signature Σ that consists of (possibly infinitely many) constant symbols and of relation symbols. A *model* is a Σ-structure, and is typically denoted $\mathbf{M}, \mathbf{N}$, etc. We usually do not distinguish between a model and its domain, when writing, for instance, $m \in \mathbf{M}$ or $f\colon \mathbf{M} \to X$, or $X \subseteq \mathbf{M}$. A graph G is viewed as a model over the signature consisting of one binary relation denoted E, indicating adjacency between vertices.

A *theory* T (over Σ) is a set of Σ-sentences. A *model of* a theory T is a model $\mathbf{M}$ such that $\mathbf{M} \models \varphi$ for all $\varphi \in T$. When a theory has a model, it is said to be *consistent*. The *theory of* a class of Σ-structures $\mathcal{C}$ is the set of all Σ-sentences φ such that $\mathbf{M} \models \varphi$ for all $\mathbf{M} \in \mathcal{C}$. The *elementary closure* $\overline{\mathcal{C}}$ of $\mathcal{C}$ is the set of all models $\mathbf{M}$ of the theory of $\mathcal{C}$. Thus $\mathcal{C} \subseteq \overline{\mathcal{C}}$, and $\mathcal{C}$ and $\overline{\mathcal{C}}$ have equal theories.

Let $\mathbf{M}$ and $\mathbf{N}$ be two structures with $\mathbf{M} \subseteq \mathbf{N}$, that is, the domain of $\mathbf{M}$ is contained in the domain of $\mathbf{N}$. Then $\mathbf{N}$ is an *elementary extension* of $\mathbf{M}$, written $\mathbf{M} \prec \mathbf{N}$, if for every formula $\varphi(\bar{x})$ (without parameters) and tuple $\bar{m} \in \mathbf{M}^{\bar{x}}$ we have $\mathbf{M} \models \varphi(\bar{m})$ if and only if $\mathbf{N} \models \varphi(\bar{m})$. We also say that $\mathbf{M}$ is an *elementary substructure* of $\mathbf{N}$.

Stability and dependence. A formula $\varphi(\bar{x};\bar{y})$ is *stable* in a class $\mathcal{C}$ of structures if there exists $k \in \mathbb{N}$ such that for every $\mathbf{M} \in \mathcal{C}$, there are no sequences $\bar{a}_1, \dots \bar{a}_k \in \mathbf{M}^{\bar{x}}$ and $\bar{b}_1, \dots, \bar{b}_k \in \mathbf{M}^{\bar{y}}$ such that $\mathbf{M} \models \varphi(\bar{a}_i;\bar{b}_j)$ if and only if $i < j$ for $1 \leqslant i, j \leqslant k$. We say that a class $\mathcal{C}$ of graphs has a *stable edge relation* if the formula $E(x;y)$ is stable in $\mathcal{C}$. Equivalently, $\mathcal{C}$ excludes some ladder as a semi-induced subgraph, where a *ladder* (often called also *half-graph*) of order k is the graph with vertices $a_1, \dots, a_k, b_1, \dots, b_k$ and edges $a_i b_j$ for all $1 \leqslant i < j \leqslant k$. Note that replacing $<$ by $\leqslant$ in the above definitions does not change them.

A formula $\varphi(\bar{x};\bar{y})$ is *dependent*, or *NIP* (standing for "not the independence property") in a class $\mathcal{C}$ if there exists $k \in \mathbb{N}$ such that for every $\mathbf{M} \in \mathcal{C}$, there are no tuples $\bar{a}_1, \dots, \bar{a}_k \in \mathbf{M}^{\bar{x}}$ and $\bar{b}_J \in \mathbf{M}^{\bar{y}}$ for $J \subseteq \{1, \dots, k\}$ such that $\mathbf{M} \models \varphi(\bar{a}_i;\bar{b}_J)$ if and only if $i \in J$ for $1 \leqslant i \leqslant k$ and $J \subseteq \{1, \dots, k\}$. Observe that a formula which is stable is also dependent. A class $\mathcal{C}$ is *stable* (resp. *dependent*) if every formula $\varphi(\bar{x};\bar{y})$ is stable (resp. dependent) in $\mathcal{C}$.

Let Σ be a signature and let $\widehat{\Sigma}$ be a signature extending Σ by (possibly infinitely many) unary relation symbols and constant symbols. A $\widehat{\Sigma}$-structure $\widehat{\mathbf{M}}$ is a *lift* of a Σ-structure $\mathbf{M}$ if $\mathbf{M}$ is obtained from $\widehat{\mathbf{M}}$ by forgetting the symbols from $\widehat{\Sigma} - \Sigma$. A class of $\widehat{\Sigma}$-structures $\widehat{\mathcal{C}}$ is a *unary expansion* of a class of Σ-structures $\mathcal{C}$ if every structure $\widehat{\mathbf{M}} \in \widehat{\mathcal{C}}$ is a lift of some structure $\mathbf{M} \in \mathcal{C}$. A class $\mathcal{C}$ of structures is *monadically stable* if every unary expansion $\widehat{\mathcal{C}}$ of $\mathcal{C}$ is stable. Similarly, $\mathcal{C}$ is *monadically dependent* (or *monadically NIP*) if every unary expansion $\widehat{\mathcal{C}}$ of $\mathcal{C}$ is

dependent. A single structure $\mathbf{M}$ is monadically stable (resp. monadically dependent) if the class $\{\mathbf{M}\}$ is. Note that a class which is monadically stable (resp. monadically dependent) is stable (resp. dependent).

Flips. An *atomic flip* is an operation F specified by a pair (A, B) of (possibly intersecting) vertex sets, which complements the adjacency relation between the sets A and B in a given graph G. Formally, for a graph G, the graph obtained from G by applying the atomic flip F is the graph denoted $G \oplus \mathsf{F}$ with vertex set $V(G)$, where, for distinct vertices u, v in $V(G)$,

$$uv \in E(G \oplus \mathsf{F}) \qquad \iff \qquad \begin{cases} uv \notin E(G), & \text{if } (u,v) \in (A \times B) \cup (B \times A); \\ uv \in E(G), & \text{otherwise.} \end{cases}$$

A set of flips $\{\mathsf{F}_1, \ldots, \mathsf{F}_k\}$ defines an operation F that, given a graph G, results in the graph $G \oplus F := G \oplus \mathsf{F}_1 \oplus \cdots \oplus \mathsf{F}_k$. One can easily show that the order in which we carry out the atomic flips does not matter and that it would be useless to consider multisets. Abusing terminology, we will often just say that the operation F is a set of flips, and write $F = \{\mathsf{F}_1, \ldots, \mathsf{F}_k\}$.

Let $\mathcal{F}$ be a family of vertex sets. Then an $\mathcal{F}$-flip is a set of flips of the form $\{\mathsf{F}_1, \ldots, \mathsf{F}_k\}$, where each flip F_i is a pair (A, B) with $A, B \in \mathcal{F}$. Note that there are at most $2^{|\mathcal{F}|^2}$ different $\mathcal{F}$-flips. In our context, the family $\mathcal{F}$ will usually be a partition of the vertex set of some graph G. An *$\mathcal{F}$-flip* of a graph G, where $\mathcal{F}$ is a family of subsets of $V(G)$, is a graph G' obtained from G after applying an $\mathcal{F}$-flip. Whenever we speak about an $\mathcal{F}$-flip, it will be always clear from the context whether we mean a graph or the family of flips used to obtain it.

Flipper game. Fix a radius r. The Flipper game (or the Flipper/Localizer game) of radius r is played by two players, Flipper and Localizer, on a graph G as follows. At the beginning, set $G_0 := G$. In the ith round, for $i > 0$, the game proceeds as follows.

- If $|G_{i-1}| = 1$ then Flipper wins.
- Localizer chooses a vertex v in G_{i-1} and we set G_{i-1}^{loc} to be the subgraph of G_{i-1} induced by the ball $B^r(v)$ of radius r around v in G_{i-1}.
- Flipper chooses an atomic flip F and applies it to produce G_i, i.e. $G_i = G_{i-1}^{\mathrm{loc}} \oplus \mathsf{F}$.

Variants. It will be convenient to work with different variants of the Flipper game.

- *Batched flipping:* One can consider a variant of the Flipper game where Flipper in the ith move applies a set F of flips to G_{i-1}^{loc} to obtain G_i, where $|F| \leqslant g(i)$ for some function $g\colon \mathbb{N} \to \mathbb{N}$. This does not change the game significantly – if Flipper wins this extended game in m rounds, then Flipper wins the standard Flipper game in $\sum_{i=1}^{m} g(i)$ rounds.
- *Localization modes:* In the definition above, the graph shrinks at each step, and when localizing, distances are computed by only taking into account vertices of the current (shrinked) graph. For this reason, we sometimes call it the *shrinking* variant, or we say that the localization is shrinking. In the *confining* variant, Localizer still remains confined within short distance to his past moves, however Flipper always produces flips of the original graph and distances are measured with respect to the full vertex set.
- *Definability of flips:* We also consider the variant where Flipper is restricted to choosing flips defined using quantifier-free formulas with parameters from the original graph, which we call qf-definable flips.
- *Separation:* Finally, we will also use a variant where distances are measured according to the later defined *separation metric* capturing all possible flips that can be performed over a given (definable) partition of the vertex set.

It will turn out that all these variants are equivalent; for more discussion and formal proofs, see the full version [11]. In particular we will work with the confining Flipper game with qf-definable separation, whose formal definition is deferred to Section 3.

Flip-flatness. The following notion of flip-flatness was introduced in [9] and characterizes monadic stability for graph classes. Given a graph G, a set of vertices $A \subseteq V(G)$ is called *distance-r independent* if all vertices in A are pairwise at distance greater than r in G.

▶ **Definition 7** (Flip-flatness). *A class of graphs $\mathcal{C}$ is* flip-flat *if for every $r \in \mathbb{N}$ there exists a function $N_r \colon \mathbb{N} \to \mathbb{N}$ and a constant $s_r \in \mathbb{N}$ such that for all $m \in \mathbb{N}$, $G \in \mathcal{C}$, and $A \subseteq V(G)$ with $|A| \geqslant N_r(m)$, there exists a set F of flips with $|F| \leqslant s_r$ and $B \subseteq A$ with $|B| \geqslant m$ such that B is distance-r independent in $G \oplus F$.*

▶ **Theorem 8** ([9]). *A class of graphs is monadically stable if and only if it is flip-flat.*

3 Outline of the model-theoretic proof

We prove the implications between the conditions of Theorem 5 as depicted in Figure 2.

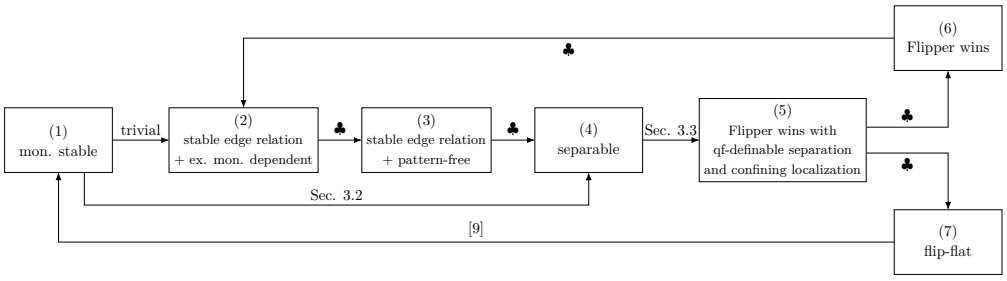


Figure 2 The implications that constitute Theorem 5. Implications marked with ♣ are proved in the full version of the paper.

The implication (1)→(2) is trivial. We prove (2)→(3) by contraposition: using the forbidden patterns, we derive the independence property for some existential formula. The implication (3)→(4) is the core part of our proof; due to space restrictions we will provide a sketch of the implication (1)→(4) (which requires fewer definitions) in Section 3.2. The implication (4)→(5) is proved by proposing a strategy with qf-definable separation in the confining game for Flipper and using compactness combined with (4) to argue that it leads to a victory within a bounded number of rounds. The proof is sketched in Section 3.3. We prove the implication (5)→(7) by (essentially) providing a strategy for Localizer in the confining game with qf-definable separation when the class is not flip-flat. Then we rely on the implication (7)→(1) from [9] to close the circle of implications; this proves the equivalence of (1)-(7) with the exception of (6). We remark that (7)→(1) is the easy implication of [9], hence our reasoning can also serve as an alternative proof of the flip-flatness characterization given in [9].

To put the Flipper game into the picture, we separately prove the implications (5)→(6) and (6)→(2). The implication (5)→(6) relies on a conceptually easy, but technically not-so-trivial translation of the strategies. In the implication (6)→(2) we use obstructions to existential monadic stability to give a strategy for Localizer in the Flipper game that enables her to endure for arbitrarily long.

3.1 Separation

The crucial new ingredient in our model-theoretic proof is the notion of *r-separation*. The definitions provided here are streamlined compared to the full version due to space restrictions.

Let G be a graph, and let $S \subseteq V(G)$ be a finite set of vertices. Consider the equivalence relation $\sim_S$ on $V(G)$, in which two vertices a, b are equivalent if either $a, b \in S$ and $a = b$, or $a, b \notin S$ and $N(a) \cap S = N(b) \cap S$. An *S-class* is an equivalence class of $\sim_S$. In other words, it is a set of vertices either of the form $\{s\}$ for some $s \in S$, or of the form $\{v \in V(G) - S \mid N(v) \cap S = T\}$ for some $T \subseteq S$. The *S-class* of a vertex $v \in V(G)$ is the unique S-class which contains v. Hence, $V(G)$ is partitioned into S-classes, and the number of S-classes is at most $|S| + 2^{|S|}$. An *S-flip* of a graph G is an $\mathcal{F}$-flip G' of G, where $\mathcal{F}$ is the partition of $V(G)$ into S-classes.

▶ **Definition 9** (r-separation). *Let G be a graph, S a finite subset of vertices of G. We say that vertices a and b of G are* r-separated over S, *denoted by*[4] $a \mathop{\downarrow}_S^r b$, *if there exists a S-flip H of G such that* $\mathrm{dist}_H(a, b) > r$.

For any $r \in \mathbb{N}$ and any graph G, finite subset S of $V(G)$, and sets $A, B \in V(G)$, we write $A \mathop{\downarrow}_S^r B$ if there exists an S-flip H of G such that H has no (A, B)-path of length at most r. Note that $A \mathop{\downarrow}_S^r B$ is a stronger condition than $a \mathop{\downarrow}_S^r b$ for all $a \in A$ and $b \in B$, since we require that the same S-flip H is used for all $a \in A$ and $b \in B$. We write $\not\downarrow_S^r$ to denote the negation of the relation $\mathop{\downarrow}_S^r$. If $A \not\downarrow_S^r B$ we say that A and B are *r-connected over S*. If A consists of a single vertex a then we write $a \mathop{\downarrow}_S^r B$ for $A \mathop{\downarrow}_S^r B$.

We now formally introduce the confining Flipper game with qf-definable separation; the most important difference is that we evaluate distances in the original graph G: the localizing ball is always defined with respect to the distance induced by $\mathop{\downarrow}_S^r$ in G, where S is the set of vertices played by Flipper.

Fix a radius $r \in \mathbb{N}$. The game of radius r is played on a graph G as follows. Let $A_0 = V(G)$ and $S_0 = \emptyset$. For $k = 1, 2, \ldots$, the kth round proceeds as follows.

- If $|A_{k-1}| = 1$, then Flipper wins.
- Otherwise, Localizer picks $c_k \in A_{k-1}$ and we set

$$A_k := A_{k-1} - \left\{ w \;\middle|\; w \mathop{\downarrow}_{S_{k-1}}^{r} c_k \right\}$$

  (where separation is evaluated in the graph G).
- Then Flipper picks $s_k \in V(G)$ and we set $S_k := S_{k-1} \cup \{s_k\}$, and proceed to the next round.

As previously, we may allow Flipper to add $g(i)$ vertices to S_{i-1} in the ith round, where $g \colon \mathbb{N} \to \mathbb{N}$ is some fixed function. Again, if Flipper can win this new game in m rounds, then Flipper can also win the original game in $\sum_{i=1}^{m} g(i)$ rounds.

In the full paper [11, Lemma 3.2], we prove that a winning strategy for Flipper in the above game can be adapted by to win in the original Flipper game. The idea is that Flipper carries out all possible S-flips and then flips back to (an induced subgraph of) the original graph.

[4] The symbol $\downarrow$ denotes forking independence in stable theories. Its use here is justified by the relationship of r-separation and forking independence in monadically stable theories, which is briefly explained in the full version of the paper.

▶ **Lemma 10.** *There exists a function $f : \mathbb{N} \to \mathbb{N}$ such that for every radius r and every graph G the following holds. If Flipper wins the confining game with qf-definable separation of radius $2r$ on G in at most k rounds, then Flipper wins the shrinking game with arbitrary flips of radius r on G in at most $f(k)$ rounds.*

Note that the converse direction, allowing to translate a winning strategy of Flipper from the shrinking to the confining variant, is not immediately clear. However, the equivalence of the two games ultimately follows from Theorem 5.

3.2 Finite separators in monadically stable models

In this section, we provide our key model-theoretic characterization of monadically stable graphs. We will use $\mathbf{M}$ to denote a graph that is typically infinite.

▶ **Definition 11.** *A graph $\mathbf{M}$ is* r-separable *if for every elementary extension $\mathbf{N}$ of $\mathbf{M}$, and every $v \in \mathbf{N} - \mathbf{M}$, there is a finite set $S \subseteq \mathbf{M}$ such that $v \downarrow_S^r \mathbf{M}$ in $\mathbf{N}$.*

The main result of this section is the following theorem.

▶ **Theorem 12.** *Every monadically stable graph $\mathbf{M}$ is r-separable, for every $r \in \mathbb{N}$.*

The proof will rely on Lemma 13 and Lemma 14 below. To state them, we will need one more definition. Let $\mathbf{M}$ be a graph and $A, B \subseteq \mathbf{M}$. We say that $a, a' \in A$ have the same *E-type* over B if $N(a) \cap B = N(a') \cap B$; this is clearly an equivalence relation. We denote the set of E-types of A over B by $\mathrm{Types}^E(A/B)$.

▶ **Lemma 13.** *Fix $r \in \mathbb{N}$. Let $\mathbf{M}$ be a monadically stable graph, let $\mathbf{N}$ be an elementary extension of $\mathbf{M}$, and let $v \in \mathbf{N}$ be such that the r-ball $B^r(v)$ around v in $\mathbf{N}$ is disjoint from $\mathbf{M}$. Then $\mathrm{Types}^E(B^r(v)/\mathbf{M})$ is finite.*

We now briefly outline the idea behind the proof of Lemma 13. It is a folklore result that in an infinite bipartite graph with sides L and R there is an infinite induced matching, or an infinite induced co-matching, or an infinite induced ladder, or $\mathrm{Types}^E(L/R)$ is finite. Assume towards a contradiction that the last option (with $L = B^r(v)$ and $R = \mathbf{M}$) does not hold. Since we work with a monadically stable graph, we cannot have an infinite ladder. Therefore, there is an infinite induced matching or co-matching between $B^r(v)$ and $\mathbf{M}$. By symmetry, we can assume the former. From this we obtain, for any k, vertices $a_1, \ldots, a_k$ in $\mathbf{M}$ and $b_1, \ldots, b_k$ in $B^r(v) \subseteq \mathbf{N} - \mathbf{M}$ such that the corresponding pairs a_i, b_i form a semi-induced matching and any b_i, b_j are connected by a path of length at most $2r$ that passes through v. Let H denote the subgraph of N induced by $b_1, \ldots, b_k$ together with the paths connecting them (we pick one such path for each pair). Using the fact that $\mathbf{M}$ is an elementary substructure of $\mathbf{N}$, we can then show that there exist as many disjoint copies of H in $\mathbf{M}$ as we want, and all these copies behave in the same way towards $a_1, \ldots a_k$ as the original H. Consequently, we can for each pair a_i, a_j use one copy of H to create a short path between a_i and a_j, and all these paths can be defined using a single first-order formula, which in turn defines a subdivided clique with k principal vertices. Since k is arbitrary, this means that $\mathbf{M}$ is not monadically dependent, as desired.

▶ **Lemma 14.** *For any graphs $\mathbf{M}$ and $\mathbf{N}$ with $\mathbf{M} \prec \mathbf{N}$ and such that $\mathbf{N}$ is monadically stable and for any set $U \subseteq \mathbf{N} - \mathbf{M}$ such that $\mathrm{Types}^E(U/\mathbf{M})$ is finite, there exists a finite set $S \subseteq \mathbf{M}$ and an S-flip which:*

- *1-separates U from $\mathbf{M}$; and*
- *does not flip the S-class $T := \{v \in \mathbf{N} : \forall s \in S. \neg E(v, s)\}$ with any other S-class (including itself), as long as $T \cap U$ is nonempty.*

The idea behind the proof of Lemma 14 can be briefly summarized as follows. Roughly speaking, we aim to find a finite $S \subseteq \mathbf{M}$ such that if $a, a' \in U$ are in the same S-class, then $N(a) \cap \mathbf{M} = N(a') \cap \mathbf{M}$, and analogously, if $b, b' \in \mathbf{M}$ are in the same S-class, then $N(a) \cap U = N(a') \cap U$. Then we can use these properties to suitably flip between S-classes to obtain the result. First we note that since $\mathrm{Types}^E(U/\mathbf{M})$ is finite, we have that $\mathrm{Types}^E(\mathbf{M}/U)$ is also finite, and so by taking one representative from each class of $\mathrm{Types}^E(\mathbf{M}/U)$ we find a finite subset $S_{\mathbf{M}} \subseteq \mathbf{M}$ such that any two elements in the same $S_{\mathbf{M}}$-class have the same neighborhood in $\mathbf{M}$. Clearly, by the same idea we could find a finite subset set S_U of U such that vertices in the same S_U-class have the same neighborhood in U. However, we need our set S_U to be contained in $\mathbf{M}$. To achieve this, we rely on a fundamental fact about stable formulas known as *definability of types*, which allows us to show that sets $N(u) \cap \mathbf{M}$ (where $u \in U$) can be defined from within $\mathbf{M}$ by looking at S_u-classes, where S_u is a finite subset of $\mathbf{M}$. Since there are only finitely many types of vertices in U with respect to the adjacency towards $\mathbf{M}$, we can list them as $u_1, \ldots, u_k$ and set $S_U := \cup_i S_u$. We can then take $S = S_{\mathbf{M}} \cup S_U$. We remark that the set S defined in the actual proof of Lemma 14 contains more vertices; we refer to the full version of the paper for details.

An inductive proof of Theorem 12 now follows by putting together Lemmas 13 and 14.

Proof sketch of Theorem 12. We proceed by induction on r. Let $\mathbf{M}$ be a monadically stable graph and let $\mathbf{N}$ be an elementary extension of $\mathbf{M}$. For every $v \in \mathbf{N} - \mathbf{M}$, we have to find a finite set $S \subseteq \mathbf{M}$ such that $v \downarrow^r_S \mathbf{M}$ in $\mathbf{N}$. The base case $r = 0$ is immediate as we may take S to be $\varnothing$ since $v \notin \mathbf{M}$. In the inductive step, assume that the result is proved for the distance $r \in \mathbb{N}$. Stated differently, assume there is a finite $S \subseteq \mathbf{M}$ and an S-flip $\mathbf{N}'$ of $\mathbf{N}$ in which the r-ball around v is disjoint from $\mathbf{M}$. It is easily checked that an S-flip of a monadically stable graph is monadically stable, and so $\mathbf{N}'$ is monadically stable. Moreover, one can also show that $\mathbf{N}'$ is an elementary extension of the subgraph of $\mathbf{N}'$ induced by the domain of $\mathbf{M}$. We can therefore apply Lemma 13 to $\mathbf{N}', \mathbf{M}$ and $B^r_{\mathbf{N}'}(v)$. By Lemma 13, $\mathrm{Types}^E(B^r(v)/\mathbf{M})$ is finite. Now Lemma 14 applied to $B^r(v)$ finishes the inductive step and the proof (we are using the fact that we obtain a set S and an S-flip, which doesn't flip the S-class that contains $B^{r-1}(v)$). ◀

Since monadic stability is preserved in the elementary closure[5], we get the following corollary, proving the implication (1)→(4) in Theorem 5.

▶ **Corollary 15.** *If $\mathcal{C}$ is a monadically stable class of graphs and $r \in \mathbb{N}$, then every $\mathbf{M} \in \overline{\mathcal{C}}$ is r-separable.*

3.3 From separability to winning the confining Flipper game

For brevity, in this section we use "Flipper game" to refer to the confining Flipper game with qf-definable separation.

▶ **Theorem 16.** *Fix $r \in \mathbb{N}$, and let $\mathcal{C}$ be a class of graphs such that every $G \in \overline{\mathcal{C}}$ is r-separable. Then there exists $k \in \mathbb{N}$ such that Flipper wins the Flipper game with radius r in k rounds on every $G \in \mathcal{C}$.*

In the proof we will use the Tarski-Vaught test, which we now recall.

[5] The preservation of monadic stability in the elementary closure is true but not obvious (follows from [6]). However, in the full version of the paper, we prove the implication (3)→(4) of Theorem 5 and only require the preservation of edge-stability and pattern-freeness, which we prove easily.

▶ **Theorem 17** (Tarski-Vaught Test). *The following conditions are equivalent for any structures* $\mathbf{M}$ *and* $\mathbf{N}$ *with* $\mathbf{M} \subseteq \mathbf{N}$.

- *The structure* $\mathbf{N}$ *is an elementary extension of* $\mathbf{M}$.
- *For every formula* $\varphi(y; \bar{x})$ *and tuple* $\bar{m} \in \mathbf{M}^{\bar{x}}$, *if* $\mathbf{N} \models \varphi(n; \bar{m})$ *holds for some* $n \in \mathbf{N}$, *then* $\mathbf{N} \models \varphi(n'; \bar{m})$ *holds for some* $n' \in \mathbf{M}$.

In the rest of this section we sketch the proof of Theorem 16. Fix an enumeration $\varphi_1, \varphi_2, \ldots$ of all formulas (in the signature of graphs) of the form $\varphi(y, x_1, \ldots, x_\ell)$, with $\ell \geqslant 0$. We define a strategy of Flipper in any graph G. In the kth round, after Localizer picks $c_k \in A_{k-1}$, Flipper first sets $S := S_{k-1} \cup \{c_k\}$ and marks c_k. Then, for every $i = 1, \ldots, k$, for the formula $\varphi_i(y, \bar{x})$, Flipper does the following.

> For each $\bar{a} \in S^{\bar{x}}$ such that $G \models \exists y.\varphi_i(y, \bar{a})$, Flipper marks any vertex $b \in V(G)$ such that $G \models \varphi_i(b, \bar{a})$.

We say that any strategy of Flipper with this property is *Localizer-complete*. The marked vertices form Flipper response in the kth round, and we set S_k to be the union of S_{k-1} and all the marked vertices. Note that there is a function $f \colon \mathbb{N} \to \mathbb{N}$ such that $|S_k| \leqslant f(k)$ for all $k \in \mathbb{N}$, regardless of which vertices Localizer picks or which of the formulas $\exists y.\varphi_i(y, \bar{a})$ hold.

We prove that there is a number $k \in \mathbb{N}$ such that when Flipper plays according to any Localizer-complete strategy on a graph $G \in \mathcal{C}$, then he wins in at most k rounds. Assume that the conclusion of the theorem does not hold. Then, there exists a sequence of graphs $G_1, G_2, \ldots \in \mathcal{C}$, where in G_n Localizer has a strategy ensuring that Flipper does not win for at least n rounds. We shall now prove that there is some graph G in the elementary closure of $\mathcal{C}$ and a vertex in the graph that survives in the arena indefinitely, when Flipper plays according to a Localizer-complete strategy. We will then use the r-separability of G to derive a contradiction.

▷ Claim 18. There exists a graph $G \in \overline{\mathcal{C}}$, a strategy of Localizer, and a Localizer-complete strategy of Flipper for which the Flipper Game on G lasts indefinitely and the intersection of the arenas $\bigcap_{n<\omega} A_n$ is nonempty.

Proof sketch. For every graph $G_n \in \mathcal{C}$, choose any Localizer-complete strategy of Flipper, and any strategy of Localizer ensuring the game continues for more than n rounds.

In each G_i, use constants to mark moves of Localizer and Flipper in a play in which they play for i moves according to the chosen strategies, and moreover mark by c_ω an arbitrary vertex that remains the arena after i rounds. We then consider, for every $i \in \mathbb{N}$, a sentence ψ_i that is true in a graph if and only if the play encoded by the introduced constants is a valid i move play in the Flipper game and c_ω is in the arena after the ith move. We then have $G_i \models \psi_i$ for each i. By a compactness argument, we can argue that there exists a graph $G \in \overline{\mathcal{C}}$ such that $G \models \psi_i$ for every i. Then we have in G that $c_\omega \in A_i$ for each i, and so $c_\omega \in \bigcap_{n<\omega} A_n$, which means that $\bigcap_{n<\omega} A_n$ is nonempty, as desired. ◁

Let $G \in \overline{\mathcal{C}}$ be the graph produced by Claim 18, along with the strategies of Localizer and Flipper. By assumption, G is r-separable. Recall that $A_0 \supseteq A_1 \supseteq \ldots$ is the sequence of arenas in the play, $c_1, c_2, \ldots$ is the sequence of moves of Localizer, and $S_0 \subseteq S_1 \subseteq \ldots$ is the sequence of sets of vertices marked by Flipper. Denote $A_\omega := \bigcap_{n<\omega} A_n$, and $S_\omega := \bigcup_{n<\omega} S_n$. We will get a contradiction with the previous claim by proving the following claim:

▷ Claim 19. A_ω is empty.

Proof. Observe that for each $k \in \mathbb{N}$, we have $c_k \notin S_{k-1}$: as soon as Localizer plays c_k in S_{k-1}, the arena A_k shrinks to a single vertex and Flipper wins in the following round. Then, A_k is disjoint from S_{k-1}: since Localizer plays c_k outside of S_{k-1}, each vertex of S_{k-1} becomes separated from c_k and thus is removed from the arena. It follows that $A_\omega \cap S_\omega = \emptyset$.

Since Flipper follows a Localizer-complete strategy, S_ω induces an elementary substructure of G by the Tarski-Vaught test (Theorem 17). We also have that $c_1, c_2, \ldots \in S_\omega$ by construction. Now suppose for a contradiction that there exists some $c_\omega \in A_\omega$. We remark that $c_\omega \notin S_\omega$.

By Theorem 12, there exists a finite set $S \subseteq S_\omega$ such that $c_\omega \downarrow^r_S S_\omega$. As S is finite, there is some $n < \omega$ such that $S \subseteq S_n$, so in particular, $c_\omega \downarrow^r_{S_n} S_\omega$. On the other hand, $c_\omega \not\downarrow^r_{S_n} c_{n+1}$, as $c_\omega \in A_{n+1}$. This is a contradiction since $c_{n+1} \in S_\omega$. ◁

However, this means that there exists a graph $G \in \overline{\mathcal{C}}$ and strategies of Localizer and Flipper, for which A_ω is simultaneously nonempty (Claim 18) and empty (Claim 19). This contradicts the existence of the graphs $G_1, G_2, \ldots \in \mathcal{C}$ and completes the proof of Theorem 16.

4 Outline of the algorithmic proof

In this part we outline the proof of Theorem 6 by sketching a winning Flipper strategy whose moves can be computed in time $\mathcal{O}_{\mathcal{C},r}(n^2)$.

Let us first sketch a natural approach to use the flip-flatness characterization of monadic stability (see Definition 7) to derive a winning strategy for Flipper. Consider the radius-r Flipper game on a graph G from a monadically stable class $\mathcal{C}$. For convenience we may assume for now that we work with an extended version of the game where at each round Flipper can apply a bounded (in term of the round's index) number of flips, instead of just one (see the discussion in the preliminaries). As making a vertex isolated requires one flip – between the vertex in question and its neighborhood – we can always assume that the flips applied by Flipper in round i make all the i vertices previously played by Localizer isolated. Hence, Localizer needs to play a new vertex in each round, thus building a growing set X of her moves.

Fix some constant $m \in \mathbb{N}$. According to flip-flatness, there exists some number $N := N_{2r}(m)$ with the property that once X has grown to the size N, we find a set of flips F – whose size is bounded independently of m – and a set Y of m vertices in X that are pairwise at distance greater than $2r$ in $G \oplus F$. It now looks reasonable that Flipper applies the flips from F within his next move. Indeed, since after applying F the vertices of Y are at distance more than $2r$ from each other, the intuition is that F robustly "disconnects" the graph so that the subsequent move of the Localizer will necessarily localize the game to a simpler setting. This intuition is, however, difficult to capture: flip-flatness a priori does not provide any guarantees on the disconnectedness of $G \oplus F$ other than that the vertices of Y are far from each other.

The main idea for circumventing this issue is to revisit the notion of flip-flatness and strengthen it with an additional *predictability* property. Intuitively, predictability says that being given any set of 5 vertices in Y as above is sufficient to uniquely reconstruct the set of flips F. Formally, we prove the following strengthening of the results of [9]. Here and later on, $O(G)$ denotes the set of linear orders on the vertices of G.

▶ **Theorem 20** (Predictable flip-flatness). *Fix a radius $r \in \mathbb{N}$ and a monadically stable class of graphs $\mathcal{C}$. Then there exist the following:*

- *An unbounded non-decreasing function $\alpha_r \colon N \to \mathbb{N}$ and a bound $\lambda_r \in \mathbb{N}$.*

- *A function* FF_r *that maps each triple* $(G \in \mathcal{C}, \preccurlyeq \in O(G), X \subseteq V(G))$ *to a pair* (Y, F) *such that:*
  - F *is a set of at most* λ_r *flips in* G*, and*
  - Y *is a set of* $\alpha_r(|X|)$ *vertices of* X *that is distance-*r *independent in* $G \oplus F$*.*
- *A function* $\mathsf{Predict}_r$ *that maps each triple* $(G \in \mathcal{C}, \preccurlyeq \in O(G), Z \subseteq V(G))$ *with* $|Z| = 5$ *to a set* F *of flips in* G *such that the following holds:*
  - *For every* $X \subseteq V(G)$*, if* $(Y, F) = \mathsf{FF}_r(G, \preccurlyeq, X)$ *and* $Z \subseteq Y$*, then* $F = \mathsf{Predict}_r(G, \preccurlyeq, Z)$*.*

Moreover, given G*,* $\preccurlyeq$*, and* Z*,* $\mathsf{Predict}_r(G, \preccurlyeq, Z)$ *can be computed in time* $\mathcal{O}_{\mathcal{C},r}(|V(G)|^2)$*.*

Let us explain the intuition behind the mappings FF_r and $\mathsf{Predict}_r$ provided by Theorem 20. The existence of bounds α_r and λ_r and of the function FF_r with the properties as above is guaranteed by the standard flip-flatness, see Definition 7 and Theorem 8. However, in the proof we pick the function FF_r in a very specific way, so that the flip set F is defined in a somewhat minimal way with respect to a given vertex ordering $\preccurlyeq$. This enables us to predict what the flip set F should be given any set of 5 vertices from Y. This condition is captured by the function $\mathsf{Predict}_r$.

We remark that the predictability property implies the following condition, which we call *canonicity*, and which may be easier to think about. (We assume the notation from Theorem 20.)

- For every $G \in \mathcal{C}$, $\preccurlyeq \in O(G)$, and $X, X' \subseteq V(G)$, if we denote $(Y, F) = \mathsf{FF}_r(G, \preccurlyeq, X)$ and $(Y', F') = \mathsf{FF}_r(G, \preccurlyeq, X')$, then $|Y \cap Y'| \geqslant 5$ entails $F = F'$.

Indeed, to derive canonicity from predictability note that $F = \mathsf{Predict}_r(G, \preccurlyeq, Z) = F'$, where Z is any 5-element subset of $Y \cap Y'$. Predictability strengthens canonicity by requiring that the mapping from 5-element subsets to flip sets is governed by a single function $\mathsf{Predict}_r$, which is moreover efficiently computable. We prove Theorem 20 in the appended full version of the paper. The proof is based on the combinatorial tools from [9], which were developed to prove the standard flip-flatness. However, the generated sets of flips have to be chosen and analyzed with much greater care.

We now outline how Flipper can use predictable flip-flatness for radius $2r$ to win the radius-r Flipper game in a bounded number of rounds. Suppose the game is played on a graph G; we also fix an arbitrary ordering $\preccurlyeq$ of vertices of G. Flipper will keep track of a growing set X of vertices played by the Localizer. The game proceeds in a number of *eras*, where at the end of each era X will be augmented by one vertex. In an era, Flipper will spend $2 \cdot \binom{|X|}{5}$ rounds trying to robustly disconnect the current set X. To this end, for every 5-element subset Z of X Flipper performs a pair of rounds:

- In the first round, Flipper computes $F := \mathsf{Predict}_{2r}(G, \preccurlyeq, Z)$ and applies the flips from F. Subsequently, Localizer needs to localize the game to a ball of radius r in the F-flip of the current graph.
- In the second round, Flipper reverses the flips by applying F again, and Localizer again localizes.

Thus, after performing a pair of rounds as above, we end with an induced subgraph of the original graph, which moreover is contained in a ball of radius r in the F-flip. Having performed all the $\binom{|X|}{5}$ pairs of rounds as above, Flipper makes the last round of this era: he applies flips that isolates all vertices of X, thus forcing Localizer to play any vertex outside of X that is still available. This adds a new vertex to X and a new era begins.

Let us sketch why this strategy leads to a victory of Flipper within a bounded number of rounds. Suppose the game proceeds for N eras, where N is such that $\alpha_{2r}(N) \geqslant 7$. Then we can apply predictable flip-flatness to the set X built within those eras, thus obtaining a pair $(Y, F) := \mathsf{FF}_{2r}(G, \preccurlyeq, X)$ such that $|Y| = 7$ and F is a set of flips such that Y is

distance-$2r$ independent in $G \oplus F$. Enumerate Y as $\{v_1, \ldots, v_7\}$, according to the order in which they were added to X during the game. Let $Z := \{v_1, \ldots, v_5\}$ and note that $F = \mathsf{Predict}_{2r}(G, \preccurlyeq, Z)$. Observe that in the era following the addition of v_5 to X, Flipper considered Z as one of the 5-element subsets of the (current) set X. Consequently, within one of the pairs of rounds in this era, he applied flips from F and forced Localizer to localize the game subsequently. Since v_6 and v_7 are at distance larger than $2r$ in $G \oplus F$, this necessarily resulted in removing v_6 or v_7 from the graph. This is a contradiction with the assumption that both v_6 and v_7 were played later in the game.

References

1 Algorithms, Logic and Structure Workshop in Warwick – Open Problem Session. URL: https://warwick.ac.uk/fac/sci/maths/people/staff/daniel_kral/alglogstr/openproblems.pdf, 2016. [Online; accessed 23-Jan-2023].

2 Hans Adler and Isolde Adler. Interpreting nowhere dense graph classes as a classical notion of model theory. *Eur. J. Comb.*, 36:322–330, 2014. doi:10.1016/j.ejc.2013.06.048.

3 J.T. Baldwin and S. Shelah. Second-order quantifiers and the complexity of theories. *Notre Dame Journal of Formal Logic*, 26(3):229–303, 1985.

4 Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, Pierre Simon, Stéphan Thomassé, and Szymon Toruńczyk. Twin-width iv: Ordered graphs and matrices. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 924–937, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519935.3520037.

5 Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width I: tractable FO model checking. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 601–612. IEEE, 2020. doi:10.1109/FOCS46700.2020.00062.

6 Samuel Braunfeld and Michael C Laskowski. Existential characterizations of monadic NIP. *arXiv preprint*, 2022. arXiv:2209.05120.

7 Anuj Dawar. Homomorphism preservation on quasi-wide classes. *J. Comput. Syst. Sci.*, 76(5):324–332, 2010. doi:10.1016/j.jcss.2009.10.005.

8 Jan Dreier, Nikolas Mählmann, and Sebastian Siebertz. First-order model checking on structurally sparse graph classes. *arXiv preprint*, 2023. arXiv:2302.03527.

9 Jan Dreier, Nikolas Mählmann, Sebastian Siebertz, and Szymon Toruńczyk. Indiscernibles and flatness in monadically stable and monadically NIP classes. *arXiv preprint*, 2022. arXiv:2206.13765.

10 Zdenek Dvořák. Induced subdivisions and bounded expansion. *Eur. J. Comb.*, 69:143–148, 2018. doi:10.1016/j.ejc.2017.10.004.

11 Jakub Gajarský, Nikolas Mählmann, Rose McCarty, Pierre Ohlmann, Michał Pilipczuk, Wojciech Przybyszewski, Sebastian Siebertz, Marek Sokołowski, and Szymon Toruńczyk. Flipper games for monadically stable graph classes, 2023. doi:10.48550/ARXIV.2301.13735.

12 Robert Ganian, Petr Hliněný, Jaroslav Nešetřil, Jan Obdržálek, and Patrice Ossona de Mendez. Shrub-depth: Capturing height of dense graphs. *Log. Methods Comput. Sci.*, 15(1), 2019. doi:10.23638/LMCS-15(1:7)2019.

13 Robert Ganian, Petr Hliněný, Jaroslav Nešetřil, Jan Obdržálek, Patrice Ossona de Mendez, and Reshma Ramadurai. When trees grow low: Shrubs and fast MSO_1. In *37th International Symposium on Mathematical Foundations of Computer Science 2012, MFCS 2012*, volume 7464 of *Lecture Notes in Computer Science*, pages 419–430. Springer, 2012.

14 Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding first-order properties of nowhere dense graphs. *J. ACM*, 64(3):17:1–17:32, 2017. doi:10.1145/3051095.

15 Jaroslav Nešetřil and Patrice Ossona de Mendez. On nowhere dense graphs. *Eur. J. Comb.*, 32(4):600–617, 2011. doi:10.1016/j.ejc.2011.01.006.

16 Klaus-Peter Podewski and Martin Ziegler. Stable graphs. *Fundamenta Mathematicae*, 100(2):101–107, 1978.

Regular Methods for Operator Precedence Languages

Thomas A. Henzinger
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Pavol Kebis
University of Oxford, UK

Nicolas Mazzocchi[1]
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

N. Ege Saraç
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Abstract

The operator precedence languages (OPLs) represent the largest known subclass of the context-free languages which enjoys all desirable closure and decidability properties. This includes the decidability of language inclusion, which is the ultimate verification problem. Operator precedence grammars, automata, and logics have been investigated and used, for example, to verify programs with arithmetic expressions and exceptions (both of which are deterministic pushdown but lie outside the scope of the visibly pushdown languages). In this paper, we complete the picture and give, for the first time, an algebraic characterization of the class of OPLs in the form of a syntactic congruence that has finitely many equivalence classes exactly for the operator precedence languages. This is a generalization of the celebrated Myhill-Nerode theorem for the regular languages to OPLs. As one of the consequences, we show that universality and language inclusion for nondeterministic operator precedence automata can be solved by an antichain algorithm. Antichain algorithms avoid determinization and complementation through an explicit subset construction, by leveraging a quasi-order on words, which allows the pruning of the search space for counterexample words without sacrificing completeness. Antichain algorithms can be implemented symbolically, and these implementations are today the best-performing algorithms in practice for the inclusion of finite automata. We give a generic construction of the quasi-order needed for antichain algorithms from a finite syntactic congruence. This yields the first antichain algorithm for OPLs, an algorithm that solves the ExpTime-hard language inclusion problem for OPLs in exponential time.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases operator precedence automata, syntactic congruence, antichain algorithm

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.129

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Funding This work was supported in part by the ERC-2020-AdG 101020093.

Acknowledgements We thank Pierre Ganty for early discussions and the anonymous reviewers for their helpful comments.

1 Introduction

Pushdown automata are a fundamental model of computation and the preferred formalism to parse programs in a deterministic manner. In verification, they are used to encode the behaviors of both systems and specifications that involve, for example, nested procedure calls.

[1] Corresponding author

However, unlike for regular languages specified by finite automata, the inclusion of context-free languages given by pushdown automata is undecidable, even for deterministic machines. This is why expressive subclasses of context-free languages with decidable properties have been studied in the past decades. Prominent among those formalisms is the class of visibly pushdown languages [3], which is strictly contained in the deterministic context-free languages. A visibly pushdown language (VPL) is a context-free language where each word admits a single parse tree, which does not depend on the pushdown automaton that generates (or accepts) the word. More technically, visibly pushdown automata (VPDAs) extend finite automata with a memory stack that is restricted to "push" and "pop" operations on disjoint subsets of the input alphabet. VPDAs have become popular in verification for several reasons. First, they recognize "well-nested" words, which find applications in the analysis of HTML and XML documents. Second, their restricted stack behavior enables desirable closure and decidability properties; in particular, in contrast to deterministic context-free languages, VPDAs can be complemented and their inclusion is decidable. Third, the VPLs admit a generalization of the celebrated Myhill-Nerode theorem for the regular languages [2]: they can be characterized algebraically by a finite syntactic congruence, which not only explains the decidability results, but also leads to symbolic verification algorithms, such as antichain-based universality and inclusion checking for VPDAs [11].

There are, however, important languages that are parsable by deterministic pushdown automata, yet are not visibly pushdown. An important example are the arithmetic expressions with two binary operators, addition and multiplication, where multiplication takes precedence over addition. Most programming languages allow such expressions with implicit precedence relations between operators, instead of insisting on explicit parantheses to disambiguate. For this very purpose, Floyd introduced three elementary precedence relations between letters, namely, *equals in precedence* $\doteq$, *yields precedence* $\lessdot$, and *takes precedence* $\gtrdot$, which provide structure to words. He introduced the *operator precedence languages* (OPLs), a subclass of the context-free languages, where non-conflicting precedence relations between letters can be derived from the context-free grammar [33]. The ability to extract non-conflicting relations from the grammar provides a unique parse tree for each word. However, unlike for VPLs, a letter is not assigned to a unique stack operation, but will trigger "push" and "pop" operations depending on its precedence with respect to the adjacent letters. This allows OPLs to model not only arithmetic expressions, but also languages with exception handling capabilities, where a single closed parenthesis may close several open parentheses [1, 48].

The class of OPLs lies strictly between the VPLs and the deterministic context-free languages. Despite their extra expressive power, the OPLs enjoy the closure and decidability properties of the VPLs, and they even do so at the same cost in computational complexity: the class of OPLs is closed under all boolean and regular operations (union, intersection, complement, concatenation, reverse, and Kleene star) [20, 21]; their emptiness can be solved in PTime (it is PTime-hard for VPDAs), and universality and inclusion in ExpTime (they are ExpTime-hard for VPDAs) [43]. Moreover, OPLs admit a logical characterization in terms of a monadic second-order theory over words, as well as an operational characterization in terms of automata with a stack (called OPAs) [43]. In short, OPLs offer many of the benefits of the VPLs at no extra cost.

In this paper, we complete the picture by showing that OPLs also offer an algebraic characterization in form of a generalized Myhill-Nerode theorem. Specifically, we define a syntactic congruence relation $\equiv_L$ for languages L such that $\equiv_L$ has finitely many equivalence classes if and only if L is an OPL. Finite syntactic congruences provide a formalism-independent (i.e., grammar- and automaton-independent) definition for capturing the algebraic essence of

a class of languages. In addition to the regular languages (Myhill-Nerode) and the VPLs, such congruences have been given also for tree languages [37], for profinite languages [47], for omega-regular languages [4, 44], for sequential and rational transducers [15, 30]. Furthermore, such characterization results through syntactic congruences have been used to design determinization [2, 38], minimization [34, 41], and learning [12, 41, 46] algorithms.

Our contribution in this paper is twofold. Besides giving a finite congruence-based characterization of OPLs, we show how such a characterization can be used to obtain antichain-based verification algorithms, i.e., symbolic algorithms for checking the universality and inclusion of operator precedence automata (OPA). Checking language inclusion is the paradigmatic verification problem for any automaton-based specification formalism, but it is also computationally difficult: PSpace-hard for finite automata, ExpTime-hard for VPDAs, undecidable for pushdown automata. This is why the verification community has devised and implemented symbolic algorithms, which avoid explicit subset constructions for determinization and complementation by manipulating symbolic representations of sets of states. For finite automata, the antichain-based algorithms have proven to be particularly efficient in practice: DWINA [29] outperforms MONA [40] for deciding WS1S formulae, ATC4VPA [11] outperforms VPAchecker [50] for deciding VPDAs inclusion, and Acacia [31] outperforms Lily [39] for LTL synthesis. They leverage a quasi-order on words to prune the search for counterexamples. Intuitively, whenever two words are candidates to contradict the inclusion between two given languages, and the words are related by the quasi-order at hand, the "greater" word can be discarded without compromising the completeness of the search. During symbolic fixpoint iteration, this "quasi-order reduction" yields a succinct representation of intermediate state sets. Based on our syntactic congurence, we show how to systematically compute a quasi-order that enables the antichain approach. Then, we provide the first antichain algorithm for checking language inclusion (and as a special case, universality) between OPAs. In fact, our antichain inclusion algorithm can take any suitable syntactic congruence over structured words (more precisely, any finite equivalence relation that is monotonic for structured words and saturates its language). The instantiation of the antichain algorithm with our syntactic congruence yields an ExpTime algorithm for the inclusion of OPAs, which is optimal in terms of enumeration complexity.

In summary, we generalize two of the most appealing features of the regular languages – the finite characterization by a syntactic congruence, and the antichain inclusion algorithm – to the important context-free subclass of operator precedence languages.

Overview. In Section 2, we define operator precedence alphabets and structured words. We present operator precedence grammars as originally defined by Floyd. We then define the operator precedence languages (OPLs) together with their automaton model (OPAs). Finally, we summarize the known closure and complexity results for OPLs and OPAs. In Section 3, we introduce the syntactic congruence that characterizes the class of OPLs. Subsection 3.1 proves that the syntactic congruence of every OPLs has finitely many equivalence classes, and Subsection 3.2 proves that every language whose syntactic congruence has finitely many equivalence classes is an OPL. In Section 4, we present our antichain inclusion algorithm. First, we introduce the notion of a language abstraction and prove that our syntactic congruence is a language abstraction of OPLs. We also present a quasi-order that relaxes the syntactic congruence while preserving the property of being a language abstraction. Then, we provide an antichain algorithm that decides the inclusion between automata whose languages have finite abstractions. We prove the correctness of our algorithm and establish its complexity on OPAs. In Section 5, we conclude with future directions.

Related Work. Operator precedence grammars and their languages were introduced by Floyd [33] with the motivation to construct efficient parsers. Inspired by Floyd's work, Wirth and Weber [51] defined simple precedence grammars as the basis of an ALGOL-like language. The relation between these two models was studied in [32]. The properties of OPLs were studied in [17, 21]. Later, their relation with the class of VPLs was established in [20], their parallel parsing was explored in [5], and automata-theoretic and logical characterizations were provided in [43]. Recent contributions provide a model-checking algorithm for operator precedence automata [14], a generalization to a weighted model [27], and their application to verifying procedural programs with exceptions [48].

The OPLs form a class of structured context-free languages [45] that sits strictly between deterministic context-free languages and the VPLs [3, 19]. To the best of our knowledge, the OPLs constitute the largest known class that enjoys all desired closure and decidability properties. Several attempts have been made to move beyond this class, however, this often comes at the cost of losing some desirable property. For example, the locally chain-parsable languages are not closed under concatenation and Kleene star [18], and the higher-order OPLs with fixed order are not closed under concatenation [22]. Despite the fact that they are more powerful than the VPLs and enjoy all closure and decidability properties, the class of OPLs is not nearly as well studied. In particular, a finite syntactic congruence characterizing the VPLs was provided in [2]. An analogous result was missing for the OPLs until now.

The antichain algorithm for checking language inclusion was originally introduced for finite automata [52] and later extended to alternating finite automata [53]. The approach has been adapted to solve games with imperfect information [13], the inclusion of tree automata [8], the realizability of linear temporal logic [31], the satisfiability of quantified boolean formulas [9], the inclusion of visibly pushdown automata [11], the inclusion of ω-visibly pushdown automata [24], the satisfiability of weak monadic second-order logic [28], and the inclusion of Büchi automata [25, 26]. The antichain-based approach can be expressed as a complete abstract interpretation as it is captured by the framework introduced in [35, 36]. We provide the first antichain inclusion algorithm for OPLs, and the first generic method to construct an antichain algorithm from a finite syntactic congruence.

2 Operator Precedence Languages

We assume that the reader is familiar with formal language theory.

2.1 Operator Precedence Relations and Structured Words

Let Σ be a finite alphabet. We refer by Σ^* to the set of all words over Σ, by ε to the empty word, and we let $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. Given a word $w \in \Sigma^*$, we denote by $|w|$ its length, by $w^{\triangleleft}$ its first letter, and by $w^{\triangleright}$ its last letter. In particular $|\varepsilon| = 0$, $\varepsilon^{\triangleleft} = \varepsilon$, and $\varepsilon^{\triangleright} = \varepsilon$.

An *operator precedence alphabet* $\widehat{\Sigma}$ is an alphabet Σ equipped with the precedence relations $\lessdot$, $\gtrdot$, $\doteq$, given by a matrix (see Figure 1). Formally, for each ordered pair of letters $(a, b) \in \Sigma^2$, exactly one[1] of the following holds:

- *a yields precedence to b*, denoted $a \lessdot b$,
- *a takes precedence over b*, denoted $a \gtrdot b$,
- *a equals in precedence with b*, denoted $a \doteq b$.

[1] In the literature, operator precedence matrices are defined over sets of precedence relations, leading then to notion of precedence conflict. We use the restriction to singletons because it covers the interesting part of the theory.

	$+$	$\times$	0	1	$(\!\|$	$\|\!)$	ε
$+$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\times$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
0	$\gtrdot$	$\gtrdot$	$\cdot$	$\cdot$	$\cdot$	$\gtrdot$	$\gtrdot$
1	$\gtrdot$	$\gtrdot$	$\cdot$	$\cdot$	$\cdot$	$\gtrdot$	$\gtrdot$
$(\!\|$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	$\gtrdot$
$\|\!)$	$\gtrdot$	$\gtrdot$	$\cdot$	$\cdot$	$\cdot$	$\gtrdot$	$\gtrdot$
ε	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$

Figure 1 (left) Operator precedence matrix where parentheses take precedence over multiplication, which takes precedence over addition. The cells marked by $\cdot$ denote the irrelevant relations.

$$\varepsilon \lessdot 1 \gtrdot + \lessdot 0 \gtrdot \times \lessdot (\!| \lessdot \mathbf{1} \gtrdot + \lessdot \mathbf{1} \gtrdot |\!) \gtrdot \varepsilon$$
$$\varepsilon \lessdot 1 \gtrdot + \lessdot 0 \gtrdot \times \lessdot (\!| \lessdot \mathbf{+} \gtrdot |\!) \gtrdot \varepsilon$$
$$\varepsilon \lessdot 1 \gtrdot + \lessdot \mathbf{0} \gtrdot \times \lessdot \mathbf{(\!|} \doteq \mathbf{|\!)} \gtrdot \varepsilon$$
$$\varepsilon \lessdot \mathbf{1} \gtrdot + \lessdot \mathbf{\times} \gtrdot \varepsilon$$
$$\varepsilon \lessdot \mathbf{+} \gtrdot \varepsilon$$
$$\varepsilon \doteq \varepsilon$$

Figure 2 (center) Computation of the collapsed from of $1 + 0 \times (\!|1 + 1|\!)$.

$$1 + 0 \times (\!|\mathbf{1} + \mathbf{1}|\!)$$
$$1 + 0 \times (\!|A \mathbf{+} B|\!)$$
$$1 + \mathbf{0} \times \mathbf{(\!|}A\mathbf{|\!)}$$
$$\mathbf{1} + B \mathbf{\times} C$$
$$A \mathbf{+} B$$
$$A$$

Figure 3 (right) Derivation tree of the words $1 + 0 \times (\!|1 + 1|\!) \in L(G_{\text{arith}})$.

For $a, b \in \Sigma$, we write $a \gtrsim b$ iff $a \gtrdot b$ or $a \doteq b$, and similarly $a \lesssim b$ iff $a \lessdot b$ or $a \doteq b$. It is worth emphasizing that, despite their appearance, the operator precedence relations $\lessdot$, $\lesssim$, $\gtrdot$, $\gtrsim$ and $\doteq$ are in general neither reflexive nor transitive. We extend the precedence relations with ε such that $\varepsilon \lessdot a$, $a \gtrdot \varepsilon$, and $\varepsilon \doteq \varepsilon$ for all $a \in \Sigma$.

Every word induces a sequence of precedences. For some words, this sequence corresponds to a *chain* [43], which is a building block of structured words.

▶ **Definition 1** (chain). *Let $a_i \in \widehat{\Sigma}$ and $u_i \in \widehat{\Sigma}^*$ for all $i \in \mathbb{N}$, and let $n \geq 1$. A word $w = a_0 a_1 \dots a_{n+1}$ is a* simple chain *when $a_0, a_{n+1} \in \widehat{\Sigma} \cup \{\varepsilon\}$ and $a_0 \lessdot a_1 \doteq a_2 \doteq \dots \doteq a_n \gtrdot a_{n+1}$. A word $w = a_0 u_0 a_1 u_1 \dots a_n u_n a_{n+1}$ is a* composite chain *when $a_0 a_1 \dots a_{n+1}$ is a simple chain and for all $0 \leq i \leq n$, either $a_i u_i a_{i+1}$ is a (simple or composite) chain or $u_i = \varepsilon$. A word w is a* chain *when w is a simple or a composite chain.*

For all $x, y, z \in \widehat{\Sigma}^*$, the predicate ${}^{x}[y]^{z}$ holds iff $(x^{\triangleright})y(z^{\triangleleft})$ is a chain. Note that, if ${}^{x}[y]^{z}$ then $xyz \neq \varepsilon$.

▶ **Example 2.** Let $\widehat{\Sigma}$ be the operator precedence alphabet in Figure 1 that specifies the precedence relations for generating arithmetic expressions. The word $(\!|(\!|)\!|)$ is a simple chain because $(\!| \lessdot (\!| \doteq |\!) \gtrdot |\!)$. Moreover, the word $(\!|1 + 1|\!)$ is a composite chain because the words $(\!|1+$, $+1|\!)$, and $(\!| + |\!)$ are simple chains.

Next, we define a function that conservatively simplifies the structure of a given word.

▶ **Definition 3** (collapsing function). *For a given operator precedence alphabet $\widehat{\Sigma}$, its* collapsing function $\lambda_{\widehat{\Sigma}} \colon \widehat{\Sigma}^* \to \widehat{\Sigma}^*$ *is defined inductively as follows: $\lambda_{\widehat{\Sigma}}(w) = \lambda_{\widehat{\Sigma}}(xz)$ if $w = xyz$ and ${}^{x}[y]^{z}$ for some $x, y, z \in \widehat{\Sigma}^+$, and $\lambda_{\widehat{\Sigma}}(w) = w$ if there is no such $x, y, z \in \widehat{\Sigma}^+$. When $\widehat{\Sigma}$ is clear from the context, we denote its collapsing function by λ.*

For every $w \in \widehat{\Sigma}$, observe that $\lambda(w)$ is in the following collapsed form: there exist $1 \leq i \leq j \leq n = |\lambda(w)|$ such that $a_1 \gtrsim \dots \gtrsim a_{i-1} \gtrdot a_i \doteq a_{i+1} \doteq \dots \doteq a_j \lessdot a_{j+1} \lesssim \dots \lesssim a_n$.

▶ **Example 4.** Let $\widehat{\Sigma}$ be the operator precedence alphabet in Figure 1. Let $w = (\!|1{+}0|\!) \times (\!|1{+}1|\!)$ and observe that $\lambda(w) = (\!|\,|\!) \times (\!|\,|\!)$ since ${}^{(\!|}[1 + 0]^{|\!)}$ and ${}^{(\!|}[1 + 1]^{|\!)}$. Note also that $(\!| \doteq |\!) \gtrdot \times \lessdot (\!| \doteq |\!)$.

Note that the collapsed form is unique and allows us to generalize classical notions of well-nested words.

▶ **Definition 5** (structured words). *Let $\widehat{\Sigma}$ be an operator precedence alphabet. We define the following sets of words:*

$$\begin{aligned}
\widehat{\Sigma}^*_{\lessdot} &= \{w \in \widehat{\Sigma}^* \mid \lambda(w) = a_1 \dots a_n \text{ where } a_i \lessdot a_{i+1} \text{ for all } i, \text{ or } |\lambda(w)| \leq 1\} \\
\widehat{\Sigma}^*_{\gtrdot} &= \{w \in \widehat{\Sigma}^* \mid \lambda(w) = a_1 \dots a_n \text{ where } a_i \gtrdot a_{i+1} \text{ for all } i, \text{ or } |\lambda(w)| \leq 1\} \\
\widehat{\Sigma}^*_{\doteq} &= \{w \in \widehat{\Sigma}^* \mid \lambda(w) = a_1 \dots a_n \text{ where } a_i \doteq a_{i+1} \text{ for all } i, \text{ or } |\lambda(w)| \leq 1\} = \widehat{\Sigma}^*_{\lessdot} \cap \widehat{\Sigma}^*_{\gtrdot}
\end{aligned}$$

Looking back at the definition of collapsed form, one can verify for every word $w \in \widehat{\Sigma}^*$ that $w \in \widehat{\Sigma}^*_{\lessdot}$ iff $i = 1$, and $w \in \widehat{\Sigma}^*_{\gtrdot}$ iff $j = n$.

▶ **Example 6.** Let $\widehat{\Sigma}$ be the operator precedence alphabet in Figure 1. The word $+ \times (\!|)\!|$ is in $\widehat{\Sigma}^*_{\lessdot}$, the word $(\!|)\!| \times +$ is in $\widehat{\Sigma}^*_{\gtrdot}$, and the word $(\!|)\!|$ is in $\widehat{\Sigma}^*_{\doteq}$. Moreover, note that $+ \lessdot \times \lessdot (\!| \doteq |\!)$ and $(\!| \doteq |\!) \gtrdot \times \gtrdot +$.

2.2 Operator Precedence Grammars

A *context-free grammar* $G = (\Sigma, V, R, S)$ is tuple where Σ is a finite set of terminal symbols, V is a finite set of non-terminal symbols, $R \subseteq V \times (\Sigma \cup V)^*$ is a finite set of derivation rules, and $S \in V$ is the starting symbol. Given $\alpha, \beta \in (\Sigma \cup V)^*$, we write $\alpha \to \beta$ when β can be derived from α with one rule, i.e., when there exists $(\alpha_2, \beta_2) \in R$, $\alpha = \alpha_1\alpha_2\alpha_3$ and $\beta = \alpha_1\beta_2\alpha_3$. Derivations using a sequence of rules are denoted by $\to^*$, the transitive closure of the relation $\to$. The language of G is $L(G) = \{w \in \Sigma^* \mid S \to^* w\}$. A derivation tree for $u \in L(G)$ is a tree over $\Sigma \cup V \cup \{\varepsilon\}$ such that the root is labeled by S, the concatenation of all leaves is u, and if a node is labeled by α and its children labeled by $\beta_1, \dots, \beta_k$ then $(\alpha, \beta_1 \dots \beta_k) \in R$. A grammar is said to be *non-ambiguous* when for all $u \in L(G)$ admits a unique derivation tree.

Intuitively, an *operator precedence grammar* (OPG for short) is an unambiguous context-free grammar whose derivation trees comply with some operator precedence matrix. Formally, let $G = (\Sigma, V, R, S)$ be a context-free grammar and $A \in V$ be a non-terminal, and define the following sets of terminal symbols where $B \in V \cup \{\varepsilon\}$ and $\alpha \in (V \cup \Sigma)^*$:

$$\mathcal{L}_G(A) = \{a \in \Sigma \mid A \to^* Ba\alpha\} \qquad \mathcal{R}_G(A) = \{a \in \Sigma \mid A \to^* \alpha aB\}$$

Given $a, b \in \Sigma$, we define the following operator precedence relations where $\alpha, \beta \in (V \cup \Sigma)^*$:

- $a \lessdot_G b$ iff there exists a rule $A \to \alpha aC\beta$ where $C \in V$ and $b \in \mathcal{L}_G(C)$,
- $a \gtrdot_G b$ iff there exists a rule $A \to \alpha Cb\beta$ where $C \in V$ and $a \in \mathcal{R}_G(C)$,
- $a \doteq_G b$ iff there exists a rule $A \to \alpha aCb\beta$ where $C \in V \cup \{\varepsilon\}$.

Finally, G is an operator precedence grammar if and only if for all $a, b \in \Sigma$, we have that $|\{\odot \in \{\lessdot_G, \doteq_G, \gtrdot_G\} \mid a \odot b\}| \leq 1$.

▶ **Example 7.** Let $G_{\text{arith}} = (\Sigma, V, R, A)$ be a context-free grammar over $\widehat{\Sigma} = \{+, \times, (\!|, |\!), 0, 1\}$ as in Figure 1 where $V = \{A, B, C\}$ and R contains the following rules:

$$A \to A + B \mid B \qquad B \to B \times C \mid C \qquad C \to (\!|A|\!) \mid 0 \mid 1$$

The language $L(G_{\text{arith}})$ consists of valid arithmetic expressions with an implicit relation between terminal symbols: parentheses take precedence over multiplication, which takes precedence over addition [43]. The missing relations, replaced by $\cdot$ in the matrix of Figure 1, denote the precedence relations that cannot be encountered by the given grammar, so the chosen precedence relation does not matter. For example, 00 and $|\!)(\!|$ are not valid arithmetic expressions and cannot be generated by G_{arith}. We remark that the structures of derivation trees and chains share strong similarities as highlighted by Figure 2 and Figure 3.

2.3 Operator Precedence Automata

Intuitively, operator precedence automata are pushdown automata where stack operations are determined by the precedence relations between the next letter and the top of the stack.

▶ **Definition 8** (operator precedence automaton). *An* operator precedence automaton *(OPA for short) over $\widehat{\Sigma}$ is a tuple $\mathcal{A} = (Q, I, F, \Delta)$ where Q is a finite set of states, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is a set of accepting states, and $\Delta \subseteq \big(Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma^+ \cup \{\bot\})\big)^2$ is the $\widehat{\Sigma}$-driven transition relation where $\Gamma = \Sigma \times Q$ is the stack alphabet and $\bot$ denotes the empty stack, meaning that, when $((s, a, \alpha), (t, b, \beta)) \in \Delta$ the following holds:*

- *If $\alpha = \bot$ or $\alpha = \langle q, a'\rangle\alpha'$ with $a' \lessdot a$, then the input triggers a* push *stack-operation implying that $b = \varepsilon$ and $\beta = \langle s, a\rangle\alpha$. We write $(s, \alpha) \xrightarrow{a} (t, \beta)$.*
- *If $\alpha = \langle q, a'\rangle\alpha'$ with $a' \doteq a$, then the input triggers a* shift *stack-operation implying that $b = \varepsilon$ and $\beta = \langle q, a\rangle\alpha'$. We write $(s, \alpha) \_\xrightarrow{a} (t, \beta)$.*
- *If $\alpha = \langle q, a'\rangle\alpha'$ with $a' \gtrdot a$, then the input triggers a* pop *stack-operation implying that $b = a$ and $\beta = \alpha'$. We write $(s, \alpha) \xRightarrow{a} (t, \beta)$.*

Let $\mathcal{A}$ be an OPA. A *configuration* of $\mathcal{A}$ is a triplet (q, u, θ) where $q \in Q$ is the current state, $u \in \Sigma^*$ is the input suffix left to be read, and $\theta \in \Gamma^+ \cup \{\bot\}$ is the current stack. A *run* of $\mathcal{A}$ is a finite sequence of configurations $((q_i, u_i, \theta_i))_{1 \le i \le n}$ for some $n \in \mathbb{N}$ such that, for all $1 \le i \le n$, the automaton fires (i) a push-transition $(q_{i-1}, \theta_{i-1}) \xrightarrow{a} (q_i, \theta_i)$ where $u_{i-1} = au_i$, (ii) a shift-transition $(q_{i-1}, \theta_{i-1}) \_\xrightarrow{a} (q_i, \theta_i)$ where $u_{i-1} = au_i$, or (iii) a pop-transition $(q_{i-1}, \theta_{i-1}) \xRightarrow{a} (q_i, \theta_i)$ where $u_{i-1} = u_i \in \{au \mid u \in \Sigma^*\}$. We write $(s, u, \alpha) \rightsquigarrow (t, v, \beta)$ when $(s, u, \alpha)(t, v, \beta)$ is a run, and let $(s, u, \alpha) \rightsquigarrow^* (t, v, \beta)$ be its reflexive transitive closure. For all $n \in \mathbb{N}$, we define the predicate $(s, u, \alpha) \rightsquigarrow^n (t, v, \beta)$ inductively by $(s, u, \alpha) = (t, v, \beta)$ when $n = 0$ and by $\exists (q, w, \theta), (s, u, \alpha) \rightsquigarrow (q, w, \theta) \rightsquigarrow^{n-1} (t, v, \beta)$ otherwise. The *language* of $\mathcal{A}$ is defined by $L(\mathcal{A}) = \{w \in \Sigma^* \mid q_0 \in I, q_F \in F, (q_0, w, \bot) \rightsquigarrow^* (q_F, \varepsilon, \bot)\}$. An OPA is *deterministic* when $|I| = 1$ and Δ is a function from $Q \times \Sigma \times (\Gamma^+ \cup \{\bot\})$ to $Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma^+ \cup \{\bot\})$, and it is *complete* when from every configuration (s, u, θ) there exists a run that ends in $(t, \varepsilon, \bot)$ for some state $t \in Q$. For a given stack $\theta \in \Gamma^+ \cup \{\bot\}$, we define $\theta^\top$ as the stack symbol at the top of θ if $\theta \in \Gamma^+$, and $\theta^\top = \varepsilon$ if $\theta = \bot$.

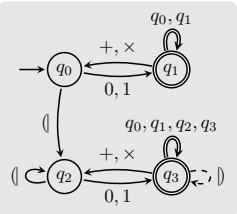
▶ **Definition 9** (operator precedence language). *An* operator precedence language *(OPL for short) is a language recognized by some operator precedence automaton.*

If L is an OPL over the operator precedence alphabet $\widehat{\Sigma}$, we say that L is a $\widehat{\Sigma}$-OPL.

▶ Remark 10. The literature on OPLs often assumes the $\doteq$-acyclicity of operator precedence relations of the alphabet, i.e., that there is no $n \ge 1$ and $a_1, \ldots, a_n \in \Sigma$ with $a_1 \doteq \ldots \doteq a_n \doteq a_1$. This assumption is used to bound the right-hand side of OPG derivation rules, and find a key application for constructing an OPG that recognizes the language of a given OPA [43]. We omit this assumption since it is not needed for establishing the results on OPAs, including the construction of an OPA that recognizes the language of a given OPG.

Now, we present an OPA that recognizes valid arithmetic expressions.

▶ **Example 11.** Recall the OPG of Example 7 generating arithmetic expressions over the operator precedence alphabet of Figure 1. In Figure 4, we show an OPA that recognizes the same language and an example of a computation.



state	input	stack	state	input	stack
q_0	$1 \times \langle 0 + 1 \rangle$	$\bot$	q_2	$1\rangle$	$\langle q_3, + \rangle \langle q_0, \langle \rangle \langle q_1, \times \rangle \bot$
q_1	$\times \langle 0 + 1 \rangle$	$\langle q_0, 1 \rangle \bot$	q_3	$\rangle$	$\langle q_2, 1 \rangle \langle q_3, + \rangle \langle q_0, \langle \rangle \langle q_1, \times \rangle \bot$
q_1	$\times \langle 0 + 1 \rangle$	$\bot$	q_3	$\rangle$	$\langle q_3, + \rangle \langle q_0, \langle \rangle \langle q_1, \times \rangle \bot$
q_0	$\langle 0 + 1 \rangle$	$\langle q_1, \times \rangle \bot$	q_3	$\rangle$	$\langle q_0, \langle \rangle \langle q_1, \times \rangle \bot$
q_2	$0 + 1\rangle$	$\langle q_0, \langle \rangle \langle q_1, \times \rangle \bot$	q_3	ε	$\langle q_0, \rangle \rangle \langle q_1, \times \rangle \bot$
q_3	$+1\rangle$	$\langle q_2, 0 \rangle \langle q_0, \langle \rangle \langle q_1, \times \rangle \bot$	q_3	ε	$\langle q_1, \times \rangle \bot$
q_3	$+1\rangle$	$\langle q_0, \langle \rangle \langle q_1, \times \rangle \bot$	q_3	ε	$\bot$

Figure 4 An OPA recognizing the arithmetic expressions generated by the OPG in Example 7 and its run on the input word $1 \times \langle 0 + 1 \rangle$. Shift-, push-, and pop-transitions are respectively denoted by dashed, normal, and double arrows.

2.4 Expressiveness and Decidability of Operator Precedence Languages

In this section, briefly summarize some known results about OPLs. First, we remark that OPLs are context-free languages as they are recognized by a subclass of pushdown automata.

▶ **Theorem 12** (from [20]). *Deterministic context-free languages strictly include OPLs.*

The language $L = \{a^n b a^n \mid n \geq 0\}$, which is a deterministic context-free language, separates the two classes. Indeed, it is not an OPL because while the first segment of a^n must push to the stack (i.e., $a \lessdot a$), the last segment must pop (i.e., $a \gtrdot a$), resulting in conflicting precedence relations. Next, we recall that OPLs enjoy the many closure properties.

▶ **Theorem 13** (from [20, 21]). *OPLs are closed under boolean operations, concatenation, Kleene star, reversal, prefixing, and suffixing.*

The class of VPLs enjoy these closure as well. In fact, every VPL can be expressed as an OPL with an operator precedence alphabet designed as follows: internal characters and returns take precedence over any character; calls equal in precedence with returns, and they yield precedence to calls and internal characters.

▶ **Theorem 14** (from [20]). *OPLs strictly include visibly pushdown languages.*

The language $L = \{a^n b^n \mid n \geq 1\} \cup \{c^n d^n \mid n \geq 1\} \cup \{e^n (bd)^n \mid n \geq 1\}$, which is an OPL due to their closure under union, separate the two classes. Indeed, for L to be a VPL, the first set requires that a is a call and b is a return. Similarly, c is a call and d is a return due to the second set. However, the last set requires that at most one of b and d is a return, resulting in a contradiction. We also note that OPAs support determinization.

▶ **Theorem 15** (from [43]). *Every OPL can be recognized by a deterministic OPA.*

Despite their expressive power, OPL remain decidable for the classical decision problems. In particular, OPAs enjoy the same order of complexity as VPDA for basic decision problems.

▶ **Theorem 16** (from [42, 43]). *The language emptiness is in PTime-C for OPAs. The language inclusion, universality, and equivalence are in PTime for deterministic OPAs and ExpTime-C for nondeterministic OPAs.*

▶ Remark 17. The membership problem is in PTime for OPAs. Determining whether a given word w is accepted by a given OPA $\mathcal{A}$ can be done in polynomial time by constructing an automaton $\mathcal{B}$ that accepts only w, constructing the intersection $\mathcal{C}$ of $\mathcal{A}$ and $\mathcal{B}$, and deciding the non-emptiness of $\mathcal{C}$.

3 A Finite Congruence for Operator Precedence Languages

This section introduces a congruence-based characterization of OPLs, similar to the Myhill-Nerode congruence for regular languages. We let $\widehat{\Sigma}$ be an operator precedence alphabet throughout the section. A relation $\bowtie$ over $\widehat{\Sigma}^*$ is monotonic when $x \bowtie y$ implies $uxv \bowtie uyv$ for all $x, y, u, v \in \widehat{\Sigma}^*$. Intuitively, monotonicity requires two words in relation to stay related while becoming embedded into some context that constructs a larger word. However, such a definition is not well suited for structured words as it does not follow how chains are constructed. Hence, we introduce a more restrictive notion than monotonicity.

▶ **Definition 18** (chain-monotonicity). *A relation $\bowtie$ over $\widehat{\Sigma}^*$ is* chain-monotonic *when $x \bowtie y$ implies $uu_0xv_0v \bowtie uu_0yv_0v$ for all $x, y, u, v, u_0, v_0 \in \widehat{\Sigma}^*$ such that $u_0z^{\lhd} \in \widehat{\Sigma}^*_{\geq}$, $z^{\rhd}v_0 \in \widehat{\Sigma}^*_{\leq}$, and ${}^{u}[u_0zv_0]^{v}$ for each $z \in \{x, y\}$.*

Chain-monotonicity requires two words in relation to stay related while being embedded into some context that construct larger structured words. This leads us to describe when two words agree on whether an embedding into a larger word forms a chain. For this, we introduce a relation that relates words that behave similarly with respect to the chain structure.

▶ **Definition 19** (chain equivalence). *We define the* chain equivalence $\approx$ *over $\widehat{\Sigma}^*$ as follows:*

$$x \approx y \iff \bigwedge \begin{cases} x^{\lhd} = y^{\lhd} \wedge x^{\rhd} = y^{\rhd} \\ \forall u, v, u_0, v_0 \in \widehat{\Sigma}^*, \left(u_0x^{\lhd} \in \widehat{\Sigma}^*_{\geq} \wedge x^{\rhd}v_0 \in \widehat{\Sigma}^*_{\leq}\right) \Rightarrow \left({}^{u}[u_0xv_0]^{v} \Leftrightarrow {}^{u}[u_0yv_0]^{v}\right) \end{cases}$$

We observe that ε is in relation with itself exclusively, i.e., $x = \varepsilon$ iff $\varepsilon \approx x$ iff $x \approx \varepsilon$. Consider a word $w \in \widehat{\Sigma}^+$ for which $\lambda(w)$ is of the form $a_1 \dots a_\ell b_1 \dots b_m c_1 \dots c_n$ for some $\ell, m, n \in \mathbb{N}$ such that $a_1 \geq \dots \geq a_\ell \gtrdot b_1 \doteq \dots \doteq b_m \lessdot c_1 \leq \dots \leq c_n$ where $a_i, b_j, c_k \in \Sigma$ for all i, j, k. We define the *profile* of w as $P_w = (w^{\lhd}, w^{\rhd}, P_w^{\lhd}, P_w^{\rhd})$, where $P_w^{\lhd} = \{a_1, b_1\} \cup \{a_{i+1} \mid a_i \gtrdot a_{i+1}, 1 \leq i < \ell\}$ and $P_w^{\rhd} = \{b_m, c_n\} \cup \{c_k \mid c_k \lessdot c_{k+1}, 1 \leq k < n\}$. There are at most $|\Sigma|^2 \times 2^{2|\Sigma|-2} + 1$ profiles. We can show that two words with the same profile are chain equivalent, leading to the following proposition.

▶ **Proposition 20.** *$\approx$ is a chain-monotonic equivalence relation with finitely many classes.*

Next, we introduce an equivalence relation that characterizes OPLs.

▶ **Definition 21** (syntactic congruence). *Given $L \subseteq \widehat{\Sigma}^*$, we define $\equiv_L$ as the following relation over $\widehat{\Sigma}^*$:*

$$x \equiv_L y \iff x \approx y \wedge \begin{cases} \forall u, v, u_0, v_0 \in \widehat{\Sigma}^*, \left(u_0x^{\lhd} \in \widehat{\Sigma}^*_{\geq} \wedge x^{\rhd}v_0 \in \widehat{\Sigma}^*_{\leq} \wedge {}^{u}[u_0xv_0]^{v}\right) \\ \qquad \Rightarrow \left(uu_0xv_0v \in L \Leftrightarrow uu_0yv_0v \in L\right) \end{cases}$$

Let us demonstrate the syntactic congruence.

▶ **Example 22.** Let $\Sigma = \{a, b\}$ and let $\widehat{\Sigma}$ be the operator precedence alphabet with the relations $a \lessdot a$, $a \doteq b$, $b \gtrdot a$, and $b \gtrdot b$. Consider the language $L = \{a^n b^n \mid n \geq 1\}$.

There are 17 potential profiles for $\widehat{\Sigma}$ in total. Although some of them cannot occur due to the precedence relations of $\widehat{\Sigma}$, the remaining ones correspond to the equivalence classes of $\approx$. For example, $(a, a, \{a\}, \{a, b\})$ cannot occur since $b \gtrdot a$, and $(a, b, \{a\}, \{b\})$ contains exactly the words in L which are of the form $a^n b^n$ for some $n \geq 1$. For brevity, we only show how the syntactic congruence $\equiv_L$ refines the class of $\approx$ corresponding to $(a, a, \{a\}, \{a\})$ by splitting it into four subclasses. The profile $(a, a, \{a\}, \{a\})$ captures exactly the words of the form $w = a$ or $w = aua$ where in each prefix of au there are no more b's than a's. Notice that for such w, $\lambda(w)$ is of the form $(ab)^*a^+$, where $a^+ = \{a^n \mid n > 0\}$.

We first argue that $a \not\equiv_L aa$ but $aa \equiv_L aa^n$ for all $n \geq 1$. Taking $u = v = u_0 = \varepsilon$ and $v_0 = b$, observe that the preconditions for the syntactic congruence are satisfied but $ab \in L$ while $aab \notin L$, therefore $a \not\equiv_L aa$. Now, let $n \geq 2$, and consider the words aa and aa^n. Intuitively, since there is no $x, y \in \widehat{\Sigma}^*$ such that $xaay \in L$ and $xaa^n y \in L$, we show that whenever the preconditions for the congruence are satisfied, both longer words are out of L. Given $u, v, u_0, v_0 \in \widehat{\Sigma}^*$ such that $u_0 a \in \widehat{\Sigma}^*_{\geq}$, $av_0 \in \widehat{\Sigma}^*_{\leq}$, and ${}^u[u_0 aav_0]^v$, we assume towards contradiction that $uu_0 aav_0 v \in L$. Since $uu_0 aav_0 v \in L$ and $u_0 a \in \widehat{\Sigma}^*_{\geq}$, we have $u_0 = \varepsilon$. Moreover, since $av_0 \in \widehat{\Sigma}^*_{\leq}$, we have that v_0 is either of the from a^* or a^*b. Consequently, $\lambda(u_0 aav_0)$ is aaa^* or aaa^*b. This contradicts that ${}^u[u_0 aav_0]^v$ because $a \lessdot a$, and therefore $uu_0 aav_0 v \notin L$. The same argument shows that $uu_0 aa^n v_0 v \notin L$, implying that $aa \equiv_L aa^n$. Similarly as above, we can show that $u \not\equiv_L v$ but $v \equiv_L w$ for all $u, v, w \in \widehat{\Sigma}^*$ such that $\lambda(u) = (ab)^i a$, $\lambda(v) = (ab)^j aa$, and $\lambda(w) = (ab)^k aa^n$, where $n, i, j, k \geq 1$.

We now show that the syntactic congruence is chain-monotonic.

▶ **Theorem 23.** *For every $L \subseteq \widehat{\Sigma}^*$, $\equiv_L$ is a chain-monotonic equivalence relation.*

The main result of this section is the characterization theorem below. We prove each direction separately in Sections 3.1 and 3.2.

▶ **Theorem 24.** *A language L is an OPL iff $\equiv_L$ admits finitely many equivalence classes.*

3.1 Finiteness of the Syntactic Congruence

Let $\widehat{\Sigma}$ be an operator precedence alphabet, $\mathcal{A} = (Q, I, F, \Delta)$ be an OPA over $\widehat{\Sigma}$, and $\star \notin \Sigma$ be a fresh letter for which we extend the precedence relation with $a \lessdot \star$ for all $a \in \Sigma$.

For every word $w \in \widehat{\Sigma}^*$, we define the functions $f_w \colon Q \times (\Gamma \cup \{\bot\}) \to 2^Q$ and $\Phi_w \colon Q \times (\Gamma \cup \{\bot\}) \to 2^{\Gamma^+ \cup \{\bot\}}$ such that for all $q \in Q$ and all $\gamma \in \Gamma \cup \{\bot\}$, we have $f_w(q, \gamma) = \{q_w \in Q \mid \exists \gamma_w \in \Gamma^+ \cup \{\bot\}, (q, w\star, \gamma) \rightsquigarrow^* (q_w, \star, \gamma_w)\}$ and $\Phi_w(q, \gamma) = \{\gamma_w \in \Gamma^+ \cup \{\bot\} \mid \exists q_w \in Q, (q, w\star, \gamma) \rightsquigarrow^* (q_w, \star, \gamma_w)\}$. Intuitively, the states in $f_w(q, \gamma)$ and the stacks in $\Phi_w(q, \gamma)$ come from the configurations that $\mathcal{A}$ can reach after reading w from an initial state in I, but before triggering any pop-transition due to reaching the end of the word w.

Furthermore, for every $w \in \widehat{\Sigma}^*$, we define the function $g_w \colon Q^2 \times (\Gamma \cup \{\bot\}) \to 2^Q$ such that for all $q_1, q_2 \in Q$ and all $\gamma \in \Gamma \cup \{\bot\}$ we have $g_w(q_1, q_2, \gamma) = \{p_w \in Q \mid \exists \gamma_w \in \Phi_w(q_1, \gamma), (q_2, \varepsilon, \gamma_w) \rightsquigarrow^* (p_w, \varepsilon, \bot)\}$. Intuitively, $g_w(q_1, q_2, \gamma)$ is the set of states that $\mathcal{A}$ can reach after triggering from q_2 the pop-transitions that empty the (unique) stack $\gamma_w \in \Phi_w(q_1, \gamma)$ that was generated by reading w while moving from the state q_1 to some state in $f_w(q_1, \gamma)$.

Recall that for a given stack $\theta \in \Gamma^+ \cup \{\bot\}$, we denote by $\theta^\top$ the stack symbol at the top of θ, which is ε when $\theta = \bot$. Moreover, for a given set of stacks $\Theta \subseteq \Gamma^+ \cup \{\bot\}$, let us define $\Theta^\top = \{\theta^\top \mid \theta \in \Theta\}$. For the sequel, we define the following equivalence relation:

▶ **Definition 25** (structural congruence). *Given an OPA $A = (Q, I, F, \Delta)$, we define the relation $\equiv_A$ over $\widehat{\Sigma}^*$ as follows:*

$$x \equiv_{\mathcal{A}} y \iff x \approx y \wedge f_x = f_y \wedge g_x = g_y \wedge \big(\forall q \in Q, \forall \gamma \in \Gamma \cup \{\bot\}, (\Phi_x(q, \gamma))^\top = (\Phi_y(q, \gamma))^\top\big)$$

First, we show that the structural congruence of any OPA has a finite index.

▶ **Lemma 26.** *For every OPA $\mathcal{A}$ with n states and m input letters, the structural congruence $\equiv_{\mathcal{A}}$ has at most $\mathcal{O}(m)^{\mathcal{O}(m \times n)^{\mathcal{O}(1)}}$ equivalence classes.*

Then, we show that for any OPA the syntactic congruence of its language is coarser than its structural congruence, therefore has a finite index as well.

▶ **Lemma 27.** *For every OPA $\mathcal{A}$, the congruence $\equiv_{L(\mathcal{A})}$ is coarser than the congruence $\equiv_{\mathcal{A}}$.*

As a direct result of Lemmas 26 and 27 above, we obtain the following.

▶ **Corollary 28.** *For every $L \subseteq \widehat{\Sigma}^*$, if L is a $\widehat{\Sigma}$-OPL then $\equiv_L$ has finite index.*

3.2 From the Syntactic Congruence to Operator Precedence Automata

Consider a language $L \subseteq \widehat{\Sigma}^*$ such that $\equiv_L$ has finitely many equivalence classes. We construct a deterministic OPA that recognizes L and whose states are based on the equivalence classes of $\equiv_L$. Given $w \in \widehat{\Sigma}^*$, we denote by $[w]$ its equivalence class with respect to $\equiv_L$. We construct $\mathcal{A} = (Q, \{q_0\}, F, \Delta)$ with the set of states $Q = \{([u], [v]) \mid u, v \in \widehat{\Sigma}^*\}$, the initial state $q_0 = ([\varepsilon], [\varepsilon])$, the set of accepting states $F = \{([\varepsilon], [w]) \mid w \in L\}$, and the $\widehat{\Sigma}$-driven transition function $\Delta \colon Q \times \Sigma \times (\Gamma^+ \cup \{\bot\}) \to Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma^+ \cup \{\bot\})$, where $\Gamma = \Sigma \times Q$, is defined as follows: Δ maps $(([u], [v]), a, \langle b, ([u'], [v'])\rangle\theta)$ to $(([a], [\varepsilon]), \varepsilon, \langle a, ([u], [v])\rangle\langle b, ([u'], [v'])\rangle\theta)$ if $b \lessdot a$, it returns $(([uva], [\varepsilon]), \varepsilon, \langle a, ([u'], [v'])\rangle\theta)$ if $b \doteq a$, and $(([u'], [v'uv]), a, \theta)$ if $b \gtrdot a$. The soundness of our construction is given by the proof of the following lemma in Appendix.

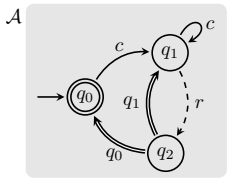
▶ **Lemma 29.** *For every $L \subseteq \widehat{\Sigma}^*$, if $\equiv_L$ has finite index then L is a $\widehat{\Sigma}$-OPL.*

4 Antichain-based Inclusion Checking

Considering two languages L_1 and L_2 given by some automata, the classical approach for deciding whether $L_1 \subseteq L_2$ holds is to first compute the complement $\overline{L}_2$ of L_2, and then decide the emptiness of $L_1 \cap \overline{L}_2$. The major drawback with this approach is that the complementation requires the determinization of the automaton denoting L_2. A way to avoid the determinization is to search among words of L_1 for a counterexample to $L_1 \subseteq L_2$. For this, a breadth-first search can be performed symbolically as a fixpoint iteration. In order to guarantee its termination, the search is equipped with a well quasi-order, and considers only words that are not subsumed, i.e., the minima of L_1 with respect to the quasi-order. It is known that well quasi-orders satisfy the finite basis property, i.e., all sets of words have finitely many minima. Our approach is inspired by [36] which, in the context of unstructured words, presents the antichain approach as a Galois connection, and observes that the upward closure of the quasi-order is a complete abstraction of concatenation according to the standard notion of completeness in abstract interpretation [16]. We identify, in the context of structured words, sufficient conditions on quasi-orders to enable the antichain approach, by defining the class of *language abstraction* quasi-orders (which satisfy the finite basis property). Further, we relax the syntactic congruence into a quasi-order that is a language abstraction of a given OPL. In particular, we prove that the syntactic congruence itself is a language abstraction for its language. Then, we design our inclusion algorithm based on a fixpoint characterization of OPLs, which allows us to iterate breadth-first over all words accepted by a given OPA. Once equipped with a language abstraction quasi-order, this fixpoint is guaranteed to terminate, thus to synthesize a finite set $T \subseteq L_1$ of membership queries for L_2 which suffices to decide whether $L_1 \subseteq L_2$ holds.

4.1 Language Abstraction by Quasi-order

Let E be a set of elements and $\preccurlyeq$ be a binary relation over E. The relation $\preccurlyeq$ is a *quasi-order* when it is reflexive and transitive. A quasi-order $\preccurlyeq$ over E is *decidable* if for all $x, y \in E$, determining whether $x \preccurlyeq y$ holds is computable. Given a subset X of E, we define its *upward closure* with respect to the quasi-order $\preccurlyeq$ by ${}_{\preccurlyeq}\uparrow X = \{e \in E \mid \exists x \in X, x \preccurlyeq e\}$. Given two



$\widehat{\Sigma}_{cr}$	c	r	ε
c	$\lessdot$	$\doteq$	$\gtrdot$
r	$\gtrdot$	$\gtrdot$	$\gtrdot$
ε	$\lessdot$	$\lessdot$	$\doteq$

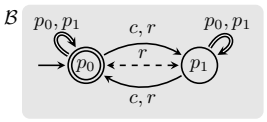


Figure 5 (left) OPA $\mathcal{A}$ over $\widehat{\Sigma}_{cr}$ recognizing the VPL of well-matched *call*/*return* words.

Figure 6 (right) OPA $\mathcal{B}$ over $\widehat{\Sigma}_{cr}$ recognizing the regular language of words of even length.

subsets $X, Y \subseteq E$ the set X is a *basis* for Y with respect to $\preccurlyeq$, denoted $\mathfrak{B}(X \preccurlyeq Y)$, whenever $X \subseteq Y$ and $\preccurlyeq\restriction X = \preccurlyeq\restriction Y$. The quasi-order $\preccurlyeq$ is a *well quasi-order* if and only if for each set $Y \subseteq E$ there exists a finite set $X \subseteq E$ such that $\mathfrak{B}(X \preccurlyeq Y)$. This property on bases is also known as the *finite basis property*. Other equivalent definitions of well quasi-orders can be found in the literature [23], we will use the following two:
(†) For every sequence $\{e_i\}_{i\in\mathbb{N}}$ in E, there exists $i, j \in \mathbb{N}$ with $i < j$ such that $e_i \preccurlyeq e_j$.
(‡) There is no sequence $\{X_i\}_{i\in\mathbb{N}}$ in 2^E such that $\preccurlyeq\restriction X_1 \subsetneq \preccurlyeq\restriction X_2 \subsetneq \dots$ holds.

Let L_1, L_2 be two languages. The main idea behind our inclusion algorithm is to compute a finite subset T of L_1, called a *query-basis*, such that $T \subseteq L_2 \Leftrightarrow L_1 \subseteq L_2$. Then, $L_1 \subseteq L_2$ holds if and only if each word of T belongs to L_2, which is checked via finitely many membership queries. The computation of a query-basis consists of collecting enough words of L_1 to obtain a finite basis T for L_1 with respect to a quasi-order $\preccurlyeq$ that abstracts L_2. When $\preccurlyeq$ is a well quasi-order, some basis is guaranteed to exist thanks to the finite basis property. To ensure the equivalence $L_1 \subseteq L_2 \Leftrightarrow T \subseteq L_2$ for any T such that $\mathfrak{B}(T \preccurlyeq L_1)$, a counterexample $w \in L_1 \setminus L_2$ can be discarded (not included in T), only if it there exists $w_0 \in T$ such that $w_0 \preccurlyeq w$ and w_0 is also a counterexample. Thus, we introduce the *language saturation* property asking a quasi-order $\preccurlyeq$ to satisfy the following: for all $w_0, w \in \widehat{\Sigma}^*$ if $w_0 \preccurlyeq w$ and $w_0 \in L_2$ then $w \in L_2$, or equivalently, $\preccurlyeq\restriction L_2 = L_2$. Intuitively, language saturation ensures the completeness of the language abstraction with respect to the inclusion. Finally, to guarantee that the query-basis T is iteratively constructible with an effective fixpoint computation, the quasi-order $\preccurlyeq$ must be both chain-monotonic and decidable. We now define the notion of *language abstraction* to identify the properties for a quasi-order over structured words that allow an effectively computable query-basis, as was done in [25, 36] in the context of Büchi automata for quasi-orders over unstructured infinite words.

▶ **Definition 30** (language abstraction). *Let $L \subseteq \widehat{\Sigma}^*$. A quasi-order $\preccurlyeq$ over $\widehat{\Sigma}^*$ is a* language abstraction *of L iff (1) it is decidable, (2) it is chain-monotonic, (3) it is a well quasi-order, and (4) it saturates L.*

In the next section, we provide an effective computation of a query-basis for an OPA, thanks to a quasi-order that abstracts its language.

▶ **Example 31.** The operator precedence alphabet $\widehat{\Sigma}_{cr}$ of $\mathcal{A}$ and $\mathcal{B}$ from Figures 5 and 6 induces four families of words: (1) the words of $\widehat{\Sigma}^*_{\doteq}$ where every c matches an r, (2) the words of $\widehat{\Sigma}^*_{\lessdot} = \widehat{\Sigma}^*_{\leqslant} \setminus \widehat{\Sigma}^*_{\doteq}$ where some c is pending for an r on its right, (3) the words of $\widehat{\Sigma}^*_{\gtrdot} = \widehat{\Sigma}^*_{\geqslant} \setminus \widehat{\Sigma}^*_{\doteq}$ where some r is pending for a c on its left, and (4) all other words of $\widehat{\Sigma}^*_{\not\doteq} = \Sigma^* \setminus \left(\widehat{\Sigma}^*_{\leqslant} \cup \widehat{\Sigma}^*_{\geqslant}\right)$.

We focus on deciding whether $L(\mathcal{B})$ is a subset of $L(\mathcal{A})$ and suppose that we are given the quasi-order $\ll$ that is a language abstraction of $L(\mathcal{A})$. Additionally, we have that two words compare with $\ll$ only if they belong to the same family, and we have the following bases: $\mathfrak{B}(\{cr\} \ll \widehat{\Sigma}^*_{=})$, $\mathfrak{B}(\{c\} \ll \widehat{\Sigma}^*_{\lessdot})$, $\mathfrak{B}(\{r\} \ll \widehat{\Sigma}^*_{\gtrdot})$, and $\mathfrak{B}(\{rc\} \ll \widehat{\Sigma}^*_{\neq})$. We observe that $\ll$ saturates $L(\mathcal{A})$ since $\widehat{\Sigma}^*_{=} \subseteq L(A)$ and $\widehat{\Sigma}^*_{\lessdot}, \widehat{\Sigma}^*_{\gtrdot}, \widehat{\Sigma}^*_{\neq} \not\subseteq L(\mathcal{A})$.

Among the representatives cr, c, r, and rc, we can construct the set $T = \{cr, rc\}$ since $c, r \notin L(\mathcal{B})$. The set T is a query-basis for deciding whether $L(\mathcal{B})$ is a subset of $L(\mathcal{A})$. In particular, $rc \in T$ witnesses that $L(\mathcal{B}) \not\subseteq L(\mathcal{A})$.

Note that the syntactic congruence is a natural language abstraction of OPLs.

▶ **Proposition 32.** *For every OPL L, $\equiv_L$ is a language abstraction of L.*

When the language to be abstracted is given by an OPA we are able to define a quasi-order, called *structural quasi-order*, that is based on the underlying structure of the automaton.

▶ **Definition 33** (structural quasi-order). *Given an OPA $\mathcal{A} = (Q, I, F, \Delta)$, we define the relation $\leqslant_{\mathcal{A}}$ over $\widehat{\Sigma}^*$ as follows:*

$$x \leqslant_{\mathcal{A}} y \iff x \approx y \wedge \forall q, q' \in Q, \forall \gamma \in \Gamma \cup \{\bot\} \bigwedge \begin{cases} f_x(q,\gamma) \subseteq f_y(q,\gamma) \\ g_x(q,q',\gamma) \subseteq g_y(q,q',\gamma) \\ (\Phi_x(q,\gamma))^\top \subseteq (\Phi_y(q,\gamma))^\top \end{cases}$$

▶ Remark 34. For every OPA $\mathcal{A}$, the quasi-order $\leqslant_{\mathcal{A}}$ relaxes the congruence $\equiv_A$ from Section 3. For every OPA $\mathcal{A}$, the quasi-order $\leqslant_{\mathcal{A}}$ relaxes the congruence $\equiv_A$ from Section 3.

Note that, for every OPA $\mathcal{A}$, the set $Q \times (\Gamma \cup \{\bot\})$ is finite. Consequently, $\leqslant_{\mathcal{A}}$ is computable, and it is a well quasi-order since there cannot exist an infinite sequence of incomparable elements, i.e., (†) holds.

▶ **Proposition 35.** *For every OPA $\mathcal{A}$, $\leqslant_{\mathcal{A}}$ is a computable chain-monotonic well quasi-order.*

Next, we establish that structural quasi-orders saturate their languages.

▶ **Lemma 36.** *For every OPA $\mathcal{A}$ and $w_1, w_2 \in \widehat{\Sigma}^*$, if $w_1 \leqslant_{\mathcal{A}} w_2$ and $w_1 \in L(\mathcal{A})$ then $w_2 \in L(\mathcal{A})$.*

The following comes as a direct consequence of Proposition 35 and Lemma 36.

▶ **Corollary 37.** *For every OPA $\mathcal{A}$, $\leqslant_{\mathcal{A}}$ is a language abstraction of $L(\mathcal{A})$.*

We continue Example 31, showing that the structural quasi-order agrees with the considered bases above.

▶ **Example 38.** The quasi-order $\ll$ described in Example 31 agrees with the structural quasi-order $\leqslant_{\mathcal{A}}$ of the OPA $\mathcal{A}$ in Figure 5. Indeed, due to the constraint that two comparable words $x, y \in \widehat{\Sigma}^*$ should be chain equivalent, i.e., $x \approx y$, the quasi-order $\leqslant_{\mathcal{A}}$ compares only the words from the same families among $\widehat{\Sigma}^*_{=}, \widehat{\Sigma}^*_{\lessdot}, \widehat{\Sigma}^*_{\gtrdot}$, and $\widehat{\Sigma}^*_{\neq}$. We also note that, for all words, adding a factor in $\widehat{\Sigma}^*_{=}$ cannot change the accessibility in $\mathcal{A}$ since reading such a factor has no effect on the stack or the current state. Additionally, reading several c in a row triggers a self loop and reading several r is not possible in $\mathcal{A}$. As a consequence, the base predicates mentioned in Example 31 hold, that is, $\mathfrak{B}(\{cr\} \leqslant_{\mathcal{A}} \widehat{\Sigma}^*_{=})$, $\mathfrak{B}(\{c\} \leqslant_{\mathcal{A}} \widehat{\Sigma}^*_{\lessdot})$, $\mathfrak{B}(\{r\} \leqslant_{\mathcal{A}} \widehat{\Sigma}^*_{\gtrdot})$, and $\mathfrak{B}(\{rc\} \leqslant_{\mathcal{A}} \widehat{\Sigma}^*_{\neq})$. Yet, we have that $cr \leqslant_{\mathcal{A}} \varepsilon$ because $(q_0, cr, \bot) \rightsquigarrow^* (q_2, \varepsilon, \langle c, q_0 \rangle)$ but $(q_0, \varepsilon, \bot) \not\rightsquigarrow^* (q_2, \varepsilon, \langle c, q_0 \rangle)$.

4.2 Fixpoint Characterization of Languages and Inclusion

In order to formulate our inclusion algorithm, it remains to give an effective computation of a query-basis. We do so through a fixpoint characterization of the languages recognized by OPAs. We introduce the function $\texttt{Cat}$ to construct words that follow the runs of the given OPA. Iterating the $\texttt{Cat}$ function $n \in \mathbb{N}$ times captures all words of length up to n, and the fixpoint of the iteration captures the entire language of a given OPA.

Let $\mathcal{A} = (Q, I, F, \Delta)$ be an OPA. Consider a vector of set of words $\vec{X}$ that accesses its fields with two states $s, t \in Q$, and three letters $a, b, c \in \widehat{\Sigma} \cup \{\varepsilon\}$. Intuitively, we aim at constructing $\vec{X}$ iteratively such that, reading any $w \in \vec{X}_{s,t}^{a,b,c}$ from the configuration (s, wc, α) where $\alpha^\top = a$ allows reaching (t, c, β) where $\beta^\top = b$ in $\mathcal{A}$. We recall that $\bot^\top = \varepsilon$. As the base case, we take $\vec{X}_{s,t}^{a,b,c} = \varepsilon$ when $a = b$ and $s = t$, otherwise $\vec{X}_{s,t}^{a,b,c} = \varnothing$. Then, we introduce operations (more explicitly, functions from sets of words to sets of words) that use the transitivity of $\rightsquigarrow^*$ in $\mathcal{A}$ to extend the sets of $\vec{X}$. We first introduce:

$$\texttt{CatShift}(\vec{X}_{s,t}^{a,b,c}) = \left\{ ub'v \;\middle|\; \begin{array}{l} a', b' \in \Sigma,\ q, s', t' \in Q,\ u \in \vec{X}_{s,s'}^{a,a',b'},\ v \in \vec{X}_{t',t}^{b',b,c}, \\ (s', \langle a', q\rangle \bot) \xrightarrow{b'} (t', \langle b', q\rangle \bot) \end{array} \right\}$$

Essentially, $\texttt{CatShift}$ adds $ub'v$ to $\vec{X}_{s,t}^{a,b,c}$ when some run over u can be appended with b' thanks to a shift-transition, and some run of v requires starting with b' at the top of the stack. Next, we introduce:

$$\texttt{CatChain}(\vec{X}_{s,t}^{a,b,c}) = \left\{ ub'v \;\middle|\; \begin{array}{l} a', b', c' \in \Sigma,\ q, s', t' \in Q,\ u \in \vec{X}_{s,q}^{a,b,b'},\ v \in \vec{X}_{s',t'}^{b',c',c}, \\ b \lessdot b' \wedge (q, \bot) \xrightarrow{b'} (s', \langle b', q\rangle \bot) \wedge (t', \langle c', q\rangle \bot) \Longrightarrow (t, \bot) \end{array} \right\}$$

Intuitively, $\texttt{CatChain}$ adds $ub'v$ to $\vec{X}_{s,t}^{a,b,c}$ when some run over u can be appended with b' thanks to a push-transition, and some run of v requires starting with b' at the top of the stack. Additionally, b' is guaranteed to be removed from the stack thanks to a pop-transition on the incoming letter c. Finally, we define:

$$\texttt{Cat}(\vec{X}_{s,t}^{a,b,c}) = \vec{X}_{s,t}^{a,b,c} \cup \texttt{CatShift}(\vec{X}_{s,t}^{a,b,c}) \cup \texttt{CatChain}(\vec{X}_{s,t}^{a,b,c})$$

Note that the function $\texttt{Cat}$ never removes words from the sets of $\vec{X}$, i.e., $\vec{X}_{s,t}^{a,b,c} \subseteq \texttt{Cat}(\vec{X}_{s,t}^{a,b,c})$. Iterating the $\texttt{Cat}$ function $n \in \mathbb{N}$ times allows us to extend the sets of $\vec{X}$ to words of length at most n that follow some run of $\mathcal{A}$. In particular, $\texttt{Cat}$ characterizes the language of $\mathcal{A}$ by $w \in L(\mathcal{A})$ if and only if $w \in \texttt{Cat}^*(\vec{X}_{q_I,q_F}^{\varepsilon,\varepsilon,\varepsilon})$ for some $q_I \in I$ and $q_F \in F$. This is formalized by the following lemma.

▶ **Lemma 39.** *Let $\mathcal{A} = (Q, I, F, \Delta)$ be an OPA, and let $\Gamma = \Sigma \times Q$. Considering $\vec{U}_{s,t}^{a,b,c} = \varepsilon$ when $a = b$ and $s = t$, otherwise $\vec{U}_{s,t}^{a,b,c} = \varnothing$. The following holds for all $n > 0$:*

$$\texttt{Cat}^n(\vec{U}_{s,t}^{a,b,c}) = \left\{ u \mid (s, uc, \alpha) \rightsquigarrow^* (t, c, \beta), |u| = n, \alpha \in \Theta_a, \beta \in \Theta_b, au \in \widehat{\Sigma}^*_{\lessdot}, uc \in \widehat{\Sigma}^*_{\gtrdot}, u^\triangleright = b \right\}$$

where, for all $a \in \widehat{\Sigma}$, the set of stack symbols $\Theta_a \subseteq \Gamma \cup \{\bot\}$ is defined by $\Theta_a = \{\bot\}$ if $a = \varepsilon$, and $\Theta_a = \{\langle a, q\rangle \mid q \in Q\}$ otherwise.

We continue Example 31, showing that $\texttt{Cat}$ agrees with the considered query-basis.

▶ **Example 40.** Let $\vec{U}_{s,t}^{a,b,c} = \varepsilon$ when $a = b$ and $s = t$, otherwise $\vec{U}_{s,t}^{a,b,c} = \varnothing$. Thanks to Lemma 39, we have that $L(\mathcal{B}) = \texttt{Cat}^*(\vec{U}_{p_0,p_0}^{\varepsilon,\varepsilon,\varepsilon})$. First observe that $c, r \notin \texttt{Cat}^*(\vec{U}_{p_0,p_0}^{\varepsilon,\varepsilon,\varepsilon})$. This comes from Lemma 39 and the fact that there is no run of $\mathcal{B}$ from p_0 to p_0 that reads a single letter. Next, we prove that $cr, rc \in \texttt{Cat}^2(\vec{U}_{p_0,p_0}^{\varepsilon,\varepsilon,\varepsilon})$.

We show that $r \in \texttt{Cat}(\vec{U}^{\varepsilon,\varepsilon,c}_{p_0,p_1})$ by `CatChain`. Indeed, we have $\varepsilon \in \vec{U}^{\varepsilon,\varepsilon,r}_{p_0,p_0}$, $\varepsilon \in \vec{U}^{r,r,c}_{p_1,p_1}$, $\varepsilon \preccurlyeq r$, and $(p_0,\bot) \xrightarrow{r} (p_1,\langle r,p_1\rangle\bot) \xRightarrow{c} (p_1,\bot)$. Then, $rc \in \texttt{Cat}^2(\vec{U}^{\varepsilon,\varepsilon,\varepsilon}_{p_0,p_0})$ by `CatChain` since $r \in \texttt{Cat}(\vec{U}^{\varepsilon,\varepsilon,c}_{p_0,p_1})$, $\varepsilon \in \vec{U}^{c,c,\varepsilon}_{p_0,p_0}$, $\varepsilon \preccurlyeq c$, and $(p_1,\bot) \xrightarrow{c} (p_0,\langle c,p_1\rangle\bot) \xRightarrow{\varepsilon} (p_1,\bot)$.

We show that $r \in \texttt{Cat}(\vec{U}^{c,r,\varepsilon}_{p_1,p_0})$ by `CatShift`. Indeed, we have $\varepsilon \in \vec{U}^{c,c,r}_{p_1,p_1}$, $\varepsilon \in \vec{U}^{r,r,\varepsilon}_{p_0,p_0}$, and $(p_1,\langle c,p\rangle\bot) \xrightarrow{r} (p_0,\langle r,p\rangle\bot)$, for all $p \in \{p_0,p_1\}$. Then, $cr \in \texttt{Cat}^2(\vec{U}^{\varepsilon,\varepsilon,\varepsilon}_{p_0,p_0})$ by `CatChain` since $\varepsilon \in \vec{U}^{\varepsilon,\varepsilon,c}_{p_0,p_0}$, $r \in \texttt{Cat}(\vec{U}^{c,r,\varepsilon}_{p_1,p_0})$, $\varepsilon \preccurlyeq c$, $(p_0,\bot) \xrightarrow{c} (p_1,\langle c,p_0\rangle\bot)$, and $(p_0,\langle r,p_0\rangle) \xRightarrow{\varepsilon} (p_0,\bot)$.

The computation of a query-basis for deciding whether L_1 is a subset of L_2 consists of iterating `Cat` to collect enough words to obtain a vector of finite bases with respect to the quasi-order $\preccurlyeq$ that is a language abstraction of L_2. In other words, we search for $n \in \mathbb{N}$ such that $\texttt{Cat}^n(\vec{X}^{a,b,c}_{s,t})$ is a basis for $\lim_{k\mapsto\infty} \texttt{Cat}^k(\vec{U}^{a,b,c}_{s,t})$ with respect to $\preccurlyeq$. The following lemma shows that when $\mathfrak{B}(\texttt{Cat}^n(\vec{X}^{a,b,c}_{s,t}) \preccurlyeq \texttt{Cat}^{n+1}(\vec{X}^{s,b,c}_{s,t}))$ holds for some $n \in \mathbb{N}$, then $\mathfrak{B}(\texttt{Cat}^n(\vec{X}^{a,b,c}_{s,t}) \preccurlyeq \lim_{k\mapsto\infty} \texttt{Cat}^k(\vec{X}^{a,b,c}_{s,t}))$ holds also, as long as the used quasi-order is chain-monotonic.

▶ **Lemma 41.** *Let $\preccurlyeq$ be a chain-monotonic quasi-order over $\widehat{\Sigma}^*$. For every $A = (Q,I,F,\Delta)$ and $\vec{X},\vec{Y}$ such that $\mathfrak{B}(\vec{X}^{a,b,c}_{s,t} \preccurlyeq \vec{Y}^{a,b,c}_{s,t})$ holds for all $s,t \in Q$ and all $a,b,c \in \Sigma \cup \{\varepsilon\}$, we have $\mathfrak{B}(\texttt{Cat}(\vec{X}^{a,b,c}_{s,t}) \preccurlyeq \texttt{Cat}(\vec{Y}^{a,b,c}_{s,t}))$ holds also for all $s,t \in Q$ and all $a,b,c \in \Sigma \cup \{\varepsilon\}$.*

Input: an OPL L_1 given by the OPA (Q,I,F,Δ)
Input: a language L_2 with a procedure deciding if $w \in L_2$
Input: a quasi-order $\preccurlyeq$ that is a language abstraction of L_2
Output: Returns `ok` if $L_1 \subseteq L_2$ and `ko` otherwise

1 **Function:**
2 **let** $\vec{U}$ **as** $\vec{U}^{a,b,c}_{s,t} \coloneqq \varepsilon$ **if** $a = b \wedge s = t$ **else** $\vec{U}^{a,b,c}_{s,t} \coloneqq \varnothing$
3 $\vec{X} \coloneqq \vec{U}$
4 **repeat**
5 **let** $\vec{X}$ **as** $\vec{X}^{a,b,c}_{s,t} \coloneqq \texttt{Cat}(\vec{X}^{a,b,c}_{s,t})$
6 **until** $\mathfrak{B}(\vec{X}^{a,b,c}_{s,t} \preccurlyeq \texttt{Cat}(\vec{X}^{a,b,c}_{s,t}))$ *for all* $s,t \in Q$ *and all* $a,b,c \in \Sigma \cup \{\varepsilon\}$
7 **for each** $(q_I,q_F) \in I \times F$ **do**
8 **for each** $w \in \vec{X}^{\varepsilon,\varepsilon,\varepsilon}_{q_I,q_F}$ **do**
9 **if** $w \notin L_2$ **then return** ko
10 **return** ok

Figure 7 Antichain inclusion algorithm.

Our inclusion algorithm is given in Figure 7. We can prove that it always terminates thanks to the finite base property of language abstractions. Additionally, its correctness is based on the following: Lemmas 39 and 41 ensure that the repeat-until loop computes a basis of the language L_1 given by an OPA while the language saturation ensures the completeness of this basis with respect to the inclusion problem.

▶ **Theorem 42.** *The algorithm from Figure 7 terminates and decides language inclusion.*

We establish that our inclusion algorithm for OPAs is in ExpTime as a consequence of Lemma 26, Remark 34, the facts that the vector $\vec{X}$ maintains polynomially many sets of words and the membership problem for OPAs is in PTime (Remark 17). We recall that inclusion and universality are ExpTime-C for both OPLs and VPLs [3, 43].

▶ **Theorem 43.** *For all OPAs $\mathcal{A}, \mathcal{B}$ with respectively $n_{\mathcal{A}}, n_{\mathcal{B}}$ states and m input letters, the inclusion algorithm from Figure 7 with $\leqslant_{\mathcal{B}}$ as the language abstraction quasi-order decides if $L(\mathcal{A}) \subseteq L(\mathcal{B})$ in time $\mathcal{O}(m \times n_{\mathcal{A}})^{\mathcal{O}(m \times n_{\mathcal{B}})^{\mathcal{O}(1)}}$.*

5 Conclusion

We provided, for the first time, a syntactic congruence that characterizes operator precedence languages (OPLs) in the following exact sense: for any language L, the syntactic congruence has finitely many equivalence classes if and only if L is an OPL. Second, we gave sufficient conditions for a quasi-order to yield an antichain algorithm for solving the universality and language inclusion problems for nondeterministic automata. These conditions are satisfied by our syntactic congruence, which, like any finite congruence, is monotonic for structured words (i.e., chain-monotonic) and saturates its language. This results in an exponential-time antichain algorithm for the inclusion of operator precedence automata (OPAs), which is the optimal worst-case complexity for the ExpTime-hard problem. This will allow efficient symbolic implementations of antichain algorithms to be extended to OPLs.

The possibility of future research directions regarding OPLs is still vast. One promising direction is to study OPAs from a runtime verification [6] perspective. For example, extending the runtime approaches for visibly pushdown automata [10, 49], one can study the monitor synthesis and right-universality problems for OPAs to establish them as an expressively powerful class of monitors. Also other methods developed for visibly pushdown automata may be generalizable to OPAs based on our syntactic congruence, such as learning algorithms [41].

While OPLs characterize the weakest known restrictions on stack operations which enable decidability of the inclusion problem, one may try to push the frontier of decidability by relaxing the restrictions on stack operations further. Investigating similar restrictions in the context of observability for counter automata can also provide new decidability results. For example, [7] shows that hardcoding the counter operations (increments and decrements) in the input letters yields decidable inclusion for one-counter automata. Another natural direction is to investigate quantitative versions of OPAs, for instance, through the addition of Presburger acceptance constraints, and to identify decidable fragments thereof [27].

References

1 Rajeev Alur and Dana Fisman. Colored nested words. *Formal Methods Syst. Des.*, 58(3):347–374, 2021. doi:10.1007/s10703-021-00384-2.

2 Rajeev Alur, Viraj Kumar, P. Madhusudan, and Mahesh Viswanathan. Congruences for visibly pushdown languages. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 1102–1114. Springer, 2005. doi:10.1007/11523468_89.

3 Rajeev Alur and P. Madhusudan. Visibly pushdown languages. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 202–211. ACM, 2004. doi:10.1145/1007352.1007390.

4 André Arnold. A syntactic congruence for rational omega-language. *Theor. Comput. Sci.*, 39:333–335, 1985. doi:10.1016/0304-3975(85)90148-3.

5 Alessandro Barenghi, Stefano Crespi-Reghizzi, Dino Mandrioli, and Matteo Pradella. Parallel parsing of operator precedence grammars. *Inf. Process. Lett.*, 113(7):245–249, 2013. doi:10.1016/j.ipl.2013.01.008.

6 Ezio Bartocci and Yliès Falcone, editors. *Lectures on Runtime Verification – Introductory and Advanced Topics*, volume 10457 of *Lecture Notes in Computer Science*. Springer, 2018. doi:10.1007/978-3-319-75632-5.

7 Benedikt Bollig. One-counter automata with counter observability. In Akash Lal, S. Akshay, Saket Saurabh, and Sandeep Sen, editors, *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2016, December 13-15, 2016, Chennai, India*, volume 65 of *LIPIcs*, pages 20:1–20:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.FSTTCS.2016.20.

8 Ahmed Bouajjani, Peter Habermehl, Lukás Holík, Tayssir Touili, and Tomás Vojnar. Antichain-based universality and inclusion testing over nondeterministic finite tree automata. In Oscar H. Ibarra and Bala Ravikumar, editors, *Implementation and Applications of Automata, 13th International Conference, CIAA 2008, San Francisco, California, USA, July 21-24, 2008. Proceedings*, volume 5148 of *Lecture Notes in Computer Science*, pages 57–67. Springer, 2008. doi:10.1007/978-3-540-70844-5_7.

9 Thomas Brihaye, Véronique Bruyère, Laurent Doyen, Marc Ducobu, and Jean-François Raskin. Antichain-based QBF solving. In Tevfik Bultan and Pao-Ann Hsiung, editors, *Automated Technology for Verification and Analysis, 9th International Symposium, ATVA 2011, Taipei, Taiwan, October 11-14, 2011. Proceedings*, volume 6996 of *Lecture Notes in Computer Science*, pages 183–197. Springer, 2011. doi:10.1007/978-3-642-24372-1_14.

10 Véronique Bruyère, Marc Ducobu, and Olivier Gauwin. Right-universality of visibly pushdown automata. In Axel Legay and Saddek Bensalem, editors, *Runtime Verification – 4th International Conference, RV 2013, Rennes, France, September 24-27, 2013. Proceedings*, volume 8174 of *Lecture Notes in Computer Science*, pages 76–93. Springer, 2013. doi:10.1007/978-3-642-40787-1_5.

11 Véronique Bruyère, Marc Ducobu, and Olivier Gauwin. Visibly pushdown automata: Universality and inclusion via antichains. In Adrian-Horia Dediu, Carlos Martín-Vide, and Bianca Truthe, editors, *Language and Automata Theory and Applications – 7th International Conference, LATA 2013, Bilbao, Spain, April 2-5, 2013. Proceedings*, volume 7810 of *Lecture Notes in Computer Science*, pages 190–201. Springer, 2013. doi:10.1007/978-3-642-37064-9_18.

12 Véronique Bruyère, Guillermo A. Pérez, and Gaëtan Staquet. Learning realtime one-counter automata. In Dana Fisman and Grigore Rosu, editors, *Tools and Algorithms for the Construction and Analysis of Systems – 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I*, volume 13243 of *Lecture Notes in Computer Science*, pages 244–262. Springer, 2022. doi:10.1007/978-3-030-99524-9_13.

13 Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Algorithms for omega-regular games with imperfect information' . In Zoltán Ésik, editor, *Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006, Proceedings*, volume 4207 of *Lecture Notes in Computer Science*, pages 287–302. Springer, 2006. doi:10.1007/11874683_19.

14 Michele Chiari, Dino Mandrioli, and Matteo Pradella. Operator precedence temporal logic and model checking. *Theor. Comput. Sci.*, 848:47–81, 2020. doi:10.1016/j.tcs.2020.08.034.

15 Christian Choffrut. Minimizing subsequential transducers: a survey. *Theor. Comput. Sci.*, 292(1):131–143, 2003. doi:10.1016/S0304-3975(01)00219-5.

16 Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In Robert M. Graham, Michael A. Harrison, and Ravi Sethi, editors, *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977*, pages 238–252. ACM, 1977. doi:10.1145/512950.512973.

17 Stefano Crespi-Reghizzi, Giovanni Guida, and Dino Mandrioli. Operator precedence grammars and the noncounting property. *SIAM J. Comput.*, 10(1):174–191, 1981. doi:10.1137/0210013.

18 Stefano Crespi-Reghizzi, Violetta Lonati, Dino Mandrioli, and Matteo Pradella. Toward a theory of input-driven locally parsable languages. *Theor. Comput. Sci.*, 658:105–121, 2017. doi:10.1016/j.tcs.2016.05.003.

19 Stefano Crespi-Reghizzi and Dino Mandrioli. Algebraic properties of structured context-free languages: old approaches and novel developments. *CoRR*, abs/0907.2130, 2009. arXiv:0907.2130.

20 Stefano Crespi-Reghizzi and Dino Mandrioli. Operator precedence and the visibly pushdown property. *J. Comput. Syst. Sci.*, 78(6):1837–1867, 2012. doi:10.1016/j.jcss.2011.12.006.

21 Stefano Crespi-Reghizzi, Dino Mandrioli, and David F. Martin. Algebraic properties of operator precedence languages. *Inf. Control.*, 37(2):115–133, 1978. doi:10.1016/S0019-9958(78)90474-6.

22 Stefano Crespi-Reghizzi and Matteo Pradella. Beyond operator-precedence grammars and languages. *J. Comput. Syst. Sci.*, 113:18–41, 2020. doi:10.1016/j.jcss.2020.04.006.

23 Aldo de Luca and Stefano Varricchio. Well quasi-orders and regular languages. *Acta Informatica*, 31(6):539–557, 1994. doi:10.1007/BF01213206.

24 Kyveli Doveri, Pierre Ganty, and Luka Hadži-Đokić. Antichains Algorithms for the Inclusion Problem Between ω-VPL. In *TACAS'23: Proc. 29th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 13993 of *LNCS*. Springer, 2023.

25 Kyveli Doveri, Pierre Ganty, and Nicolas Mazzocchi. Forq-based language inclusion formal testing. In Sharon Shoham and Yakir Vizel, editors, *Computer Aided Verification – 34th International Conference, CAV 2022, Haifa, Israel, August 7-10, 2022, Proceedings, Part II*, volume 13372 of *Lecture Notes in Computer Science*, pages 109–129. Springer, 2022. doi:10.1007/978-3-031-13188-2_6.

26 Kyveli Doveri, Pierre Ganty, Francesco Parolini, and Francesco Ranzato. Inclusion testing of büchi automata based on well-quasiorders. In Serge Haddad and Daniele Varacca, editors, *32nd International Conference on Concurrency Theory, CONCUR 2021, August 24-27, 2021, Virtual Conference*, volume 203 of *LIPIcs*, pages 3:1–3:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CONCUR.2021.3.

27 Manfred Droste, Stefan Dück, Dino Mandrioli, and Matteo Pradella. Weighted operator precedence languages. *Inf. Comput.*, 282:104658, 2022. doi:10.1016/j.ic.2020.104658.

28 Tomás Fiedor, Lukás Holík, Ondrej Lengál, and Tomás Vojnar. Nested antichains for WS1S. In Christel Baier and Cesare Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems – 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, volume 9035 of *Lecture Notes in Computer Science*, pages 658–674. Springer, 2015. doi:10.1007/978-3-662-46681-0_59.

29 Tomás Fiedor, Lukás Holík, Ondrej Lengál, and Tomás Vojnar. Nested antichains for WS1S. *Acta Informatica*, 56(3):205–228, 2019. doi:10.1007/s00236-018-0331-z.

30 Emmanuel Filiot, Olivier Gauwin, and Nathan Lhote. Logical and algebraic characterizations of rational transductions. *Log. Methods Comput. Sci.*, 15(4), 2019. doi:10.23638/LMCS-15(4:16)2019.

31 Emmanuel Filiot, Naiyong Jin, and Jean-François Raskin. An antichain algorithm for LTL realizability. In Ahmed Bouajjani and Oded Maler, editors, *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 – July 2, 2009. Proceedings*, volume 5643 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 2009. doi:10.1007/978-3-642-02658-4_22.

32 Michael J. Fischer. Some properties of precedence languages. In Patrick C. Fischer, Seymour Ginsburg, and Michael A. Harrison, editors, *Proceedings of the 1st Annual ACM Symposium on Theory of Computing, May 5-7, 1969, Marina del Rey, CA, USA*, pages 181–190. ACM, 1969. doi:10.1145/800169.805432.

33 Robert W. Floyd. Syntactic analysis and operator precedence. *J. ACM*, 10(3):316–333, 1963. doi:10.1145/321172.321179.

34 Pierre Ganty, Elena Gutiérrez, and Pedro Valero. A congruence-based perspective on automata minimization algorithms. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, volume 138 of *LIPIcs*, pages 77:1–77:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.MFCS.2019.77.

35 Pierre Ganty, Francesco Ranzato, and Pedro Valero. Language inclusion algorithms as complete abstract interpretations. In Bor-Yuh Evan Chang, editor, *Static Analysis – 26th International Symposium, SAS 2019, Porto, Portugal, October 8-11, 2019, Proceedings*, volume 11822 of *Lecture Notes in Computer Science*, pages 140–161. Springer, 2019. doi:10.1007/978-3-030-32304-2_8.

36 Pierre Ganty, Francesco Ranzato, and Pedro Valero. Complete abstractions for checking language inclusion. *ACM Trans. Comput. Log.*, 22(4):22:1–22:40, 2021. doi:10.1145/3462673.

37 Ferenc Gécseg. Classes of tree languages determined by classes of monoids. *Int. J. Found. Comput. Sci.*, 18(6):1237–1246, 2007. doi:10.1142/S0129054107005285.

38 Jelena Ignjatovic, Miroslav Ciric, and Stojan Bogdanovic. Determinization of fuzzy automata with membership values in complete residuated lattices. *Inf. Sci.*, 178(1):164–180, 2008. doi:10.1016/j.ins.2007.08.003.

39 Barbara Jobstmann and Roderick Bloem. Optimizations for LTL synthesis. In *Formal Methods in Computer-Aided Design, 6th International Conference, FMCAD 2006, San Jose, California, USA, November 12-16, 2006, Proceedings*, pages 117–124. IEEE Computer Society, 2006. doi:10.1109/FMCAD.2006.22.

40 Nils Klarlund, Anders Møller, and Michael I. Schwartzbach. MONA implementation secrets. *Int. J. Found. Comput. Sci.*, 13(4):571–586, 2002. doi:10.1142/S012905410200128X.

41 Viraj Kumar, P. Madhusudan, and Mahesh Viswanathan. Minimization, learning, and conformance testing of boolean programs. In Christel Baier and Holger Hermanns, editors, *CONCUR 2006 – Concurrency Theory, 17th International Conference, CONCUR 2006, Bonn, Germany, August 27-30, 2006, Proceedings*, volume 4137 of *Lecture Notes in Computer Science*, pages 203–217. Springer, 2006. doi:10.1007/11817949_14.

42 Martin Lange. P-hardness of the emptiness problem for visibly pushdown languages. *Inf. Process. Lett.*, 111(7):338–341, 2011. doi:10.1016/j.ipl.2010.12.013.

43 Violetta Lonati, Dino Mandrioli, Federica Panella, and Matteo Pradella. Operator precedence languages: Their automata-theoretic and logic characterization. *SIAM J. Comput.*, 44(4):1026–1088, 2015. doi:10.1137/140978818.

44 Oded Maler and Ludwig Staiger. On syntactic congruences for omega-languages. *Theor. Comput. Sci.*, 183(1):93–112, 1997. doi:10.1016/S0304-3975(96)00312-X.

45 Dino Mandrioli and Matteo Pradella. Generalizing input-driven languages: Theoretical and practical benefits. *Comput. Sci. Rev.*, 27:61–87, 2018. doi:10.1016/j.cosrev.2017.12.001.

46 Jakub Michaliszyn and Jan Otop. Learning deterministic visibly pushdown automata under accessible stack. In Stefan Szeider, Robert Ganian, and Alexandra Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022, August 22-26, 2022, Vienna, Austria*, volume 241 of *LIPIcs*, pages 74:1–74:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.MFCS.2022.74.

47 Jean-Eric Pin. Profinite methods in automata theory. In Susanne Albers and Jean-Yves Marion, editors, *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*, volume 3 of *LIPIcs*, pages 31–50. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany, 2009. doi:10.4230/LIPIcs.STACS.2009.1856.

48 Francesco Pontiggia, Michele Chiari, and Matteo Pradella. Verification of programs with exceptions through operator precedence automata. In Radu Calinescu and Corina S. Pasareanu, editors, *Software Engineering and Formal Methods – 19th International Conference, SEFM 2021, Virtual Event, December 6-10, 2021, Proceedings*, volume 13085 of *Lecture Notes in Computer Science*, pages 293–311. Springer, 2021. doi:10.1007/978-3-030-92124-8_17.

49 Grigore Rosu, Feng Chen, and Thomas Ball. Synthesizing monitors for safety properties: This time with calls and returns. In Martin Leucker, editor, *Runtime Verification, 8th International Workshop, RV 2008, Budapest, Hungary, March 30, 2008. Selected Papers*, volume 5289 of *Lecture Notes in Computer Science*, pages 51–68. Springer, 2008. doi:10.1007/978-3-540-89247-2_4.

50 Nguyen Van Tang and Hitoshi Ohsaki. On model checking for visibly pushdown automata. In Adrian-Horia Dediu and Carlos Martín-Vide, editors, *Language and Automata Theory and Applications – 6th International Conference, LATA 2012, A Coruña, Spain, March 5-9, 2012. Proceedings*, volume 7183 of *Lecture Notes in Computer Science*, pages 408–419. Springer, 2012. doi:10.1007/978-3-642-28332-1_35.

51 Niklaus Wirth and Helmut Weber. EULER: a generalization of ALGOL and it formal definition: Part 1. *Commun. ACM*, 9(1):13–25, 1966. doi:10.1145/365153.365162.

52 Martin De Wulf, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Antichains: A new algorithm for checking universality of finite automata. In Thomas Ball and Robert B. Jones, editors, *Computer Aided Verification, 18th International Conference, CAV 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*, volume 4144 of *Lecture Notes in Computer Science*, pages 17–30. Springer, 2006. doi:10.1007/11817963_5.

53 Martin De Wulf, Laurent Doyen, Nicolas Maquet, and Jean-François Raskin. Antichains: Alternative algorithms for LTL satisfiability and model-checking. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, pages 63–77. Springer, 2008. doi:10.1007/978-3-540-78800-3_6.

Positivity Problems for Reversible Linear Recurrence Sequences

George Kenison
Institute of Logic and Computation, TU Wien, Austria

Joris Nieuwveld
Max Planck Institute for Software Systems, Saarland Informatics Campus, Saarbrücken, Germany

Joël Ouaknine
Max Planck Institute for Software Systems, Saarland Informatics Campus, Saarbrücken, Germany

James Worrell
Department of Computer Science, University of Oxford, UK

Abstract

It is a longstanding open problem whether there is an algorithm to decide the Positivity Problem for linear recurrence sequences (LRS) over the integers, namely whether given such a sequence, all its terms are non-negative. Decidability is known for LRS of order 5 or less, i.e., for those sequences in which every new term depends linearly on the previous five (or fewer) terms. For *simple* LRS (i.e., those sequences whose characteristic polynomials have no repeated roots), decidability of Positivity is known up to order 9.

In this paper, we focus on the important subclass of *reversible* LRS, i.e., those integer LRS $\langle u_n \rangle_{n=0}^{\infty}$ whose bi-infinite completion $\langle u_n \rangle_{n=-\infty}^{\infty}$ also takes exclusively integer values; a typical example is the classical Fibonacci (bi-)sequence $\langle \ldots, 5, -3, 2, -1, 1, 0, 1, 1, 2, 3, 5, \ldots \rangle$. Our main results are that Positivity is decidable for reversible LRS of order 11 or less, and for simple reversible LRS of order 17 or less.

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics; Computing methodologies → Algebraic algorithms

Keywords and phrases The Positivity Problem, Linear Recurrence Sequences, Verification

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.130

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Funding *George Kenison*: Work supported by WWTF ICT19-018 grant ProbInG and ERC Consolidator Grant ARTIST 101002685.
Joël Ouaknine: Also affiliated with Keble College, Oxford as emmy.network Fellow, and supported by DFG grant 389792660 as part of TRR 248 (see `https://perspicuous-computing.science`).
James Worrell: Work supported by UKRI Frontier Research Grant EP/X033813/1.

1 Introduction

The Positivity Problem

The class of *threshold problems* considers whether a given loop program's variables remain above a fixed threshold before and after each iteration of the loop. In automated verification, this class of decision problems is relevant to program correctness, and particularly questions regarding termination, persistence, and reachability. The moniker *Positivity* is used when the chosen threshold is zero. In this paper, we shall consider the Positivity Problem (and its variants) for a particular class of integer-valued linear recurrence sequences.

An integer-valued *linear recurrence sequence (LRS)* $\langle u_n \rangle_n$ satisfies a relation of the form

$$u_{n+d} = a_{d-1}u_{n+d-1} + \cdots + a_1 u_{n+1} + a_0 u_n \tag{1}$$

where the coefficients $a_{d-1}, \ldots, a_1, a_0 \in \mathbb{Z}$ and, without loss of generality, we can assume that $a_0 \neq 0$. The sequence $\langle u_n \rangle_n$ is then wholly determined by the recurrence relation and the initial values $u_0, u_1, \ldots, u_{d-1}$. The relation in (1) is said to have *length* d and the *order* of an LRS $\langle u_n \rangle_n$ is equal to the length of the shortest relation that $\langle u_n \rangle_n$ satisfies. The polynomial $f(X) = X^d - a_{d-1}X^{d-1} - \cdots - a_1 X - a_0$ is the *characteristic polynomial* associated with relation (1).

Given an LRS $\langle u_n \rangle_n$, the *Positivity Problem* asks to determine whether $u_n \geq 0$ for each $n \in \mathbb{N}_0$. Positivity is a longstanding open problem and is intimately related to the well-known *Skolem Problem*, which asks to determine whether an LRS vanishes at some term [6, 8]. Indeed, if one could establish decidability of Positivity, then decidability of Skolem would necessarily follow (cf. [13]). One of the motivations to study Positivity lies in its connections to program verification [15]. Take, for example, the following linear loop P with inputs $\underline{w}, \underline{b} \in \mathbb{Z}^d$ and $A \in \mathbb{Z}^{d \times d}$ where

$$P\colon \underline{v} \leftarrow \underline{w};\ \textbf{while}\ \underline{b}^\top \cdot \underline{v} \geq 0\ \textbf{do}\ \underline{v} \leftarrow A\underline{v}. \tag{2}$$

Let $\langle u_n \rangle_n$ be the LRS with terms given by $u_n = \underline{b}^\top A^n \underline{w}$. It is clear that loop P terminates if and only if there exists an $n \in \mathbb{N}_0$ such that $u_n < 0$. Conversely, to each LRS $\langle u_n \rangle_n$ we can associate a linear loop of the form (2): one need only take A to be the transpose of the companion matrix associated with $\langle u_n \rangle_n$ so that

$$A = \begin{pmatrix} a_{d-1} & 1 & \cdots & 0 & 0 \\ \vdots & 0 & \ddots & 0 & 0 \\ a_2 & 0 & \cdots & 1 & 0 \\ a_1 & 0 & \cdots & 0 & 1 \\ a_0 & 0 & \cdots & 0 & 0 \end{pmatrix}, \underline{b}^\top = (u_{d-1}, \ldots, u_1, u_0), \text{ and } \underline{w} = (1, 0, \ldots, 0)^\top$$

in order to recover the terms $u_{n+d-1} = \underline{b}^\top A^n \underline{w}$.

Variants of the Positivity Problem have also garnered attention in the literature. For example, the *Ultimate Positivity Problem* weakens the guard clause: given an LRS $\langle u_n \rangle_n$, determine whether there exists an $N \in \mathbb{N}$ such that $u_n \geq 0$ if $n > N$. By contrast, the *Simple Positivity Problem* restricts the class of sequences under consideration to those that are simple. Here an LRS $\langle u_n \rangle_n$ is *simple* if each of the roots of the associated characteristic polynomial has algebraic multiplicity one. In this paper, we focus on the *Reversible Positivity Problem*, i.e., the restriction of the Positivity Problem to the class of LRS that are reversible (as defined below).

Background and Motivation

Lipton *et al.* [11] coined the term *reversible* to describe the class of LRS that assume exclusively integer values, whether the sequences are expanded forwards or backwards. Equivalently, such LRS can be shown to satisfy relations of the form (1) with the condition $a_0 = \pm 1$ (or, alternatively, the associated characteristic polynomial satisfies $f(0) = \pm 1$). The subclass of while loops (as in (2)) naturally associated with reversible sequences have *unimodular* update matrices. The inverse A^{-1} of a unimodular matrix A is likewise unimodular. Thus the uniquely defined bi-infinite extension $\langle u_n \rangle_{n=-\infty}^{\infty}$ with each $u_n = \underline{b}^\top A^n \underline{w}$ (as above) is integer-valued.

The decidability of Reversible Skolem – where one restricts the Skolem Problem to reversible LRS – was established up to order 7 in a recent paper by Lipton *et al.* [11]. Kenison [9] gave an alternative proof of this decidability result, leveraging a powerful result for algebraic units due to Dubickas and Smyth [5]. In general, the Skolem Problem is only known to be decidable for arbitrary LRS up to order four [12, 22]. The Positivity Problem is decidable for arbitrary LRS of order five or less [16], and for simple LRS of order 9 or less [13]. The Ultimate Positivity Problem is also decidable for arbitrary LRS of order 5 of less, as well as for simple LRS of arbitrary order [14, 16]. Previous work showed that the Positivity Problem is decidable for simple reversible LRS of order 10 or less [9].

Contributions

In this paper, we shall consider Positivity problems for reversible LRS. We will exploit spectral properties of reversible LRS and employ techniques from both Galois theory and Diophantine approximation to establish decidability at higher orders than is currently known for general positivity. Our main contributions are as follows:

▶ **Theorem 1.** *For reversible LRS, the Positivity and Ultimate Positivity Problems are both decidable up to order* 11*.*

▶ **Theorem 2.** *For simple reversible LRS, the Positivity Problem is decidable up to order* 17*.*

Structure

The remainder of this paper is structured as follows. In the next section we review necessary preliminary material. In Section 3, we prove results on the root structures of characteristic polynomials associated with reversible LRS. In Section 4, we prove Theorems 1 and 2. We also consider barriers impeding further progress to the state of the art (i.e., decidability results at higher orders) by exhibiting sequences that are not amenable to standard Diophantine approximation techniques due to certain spectral properties (see Section 5). The calculations involved in preparing these hard instances were performed in SageMath [4].

2 Preliminaries

2.1 Linear Recurrence Sequences

We expand upon the standard terminology for LRS given in the introduction. It is straightforward to see that an LRS $\langle u_n \rangle_n$ is wholly determined by a recurrence relation (as in (1)) and the initial values $u_0, u_1, \ldots, u_{d-1}$.

Let $\langle u_n \rangle_n$ be an LRS with characteristic polynomial f. It is well-known that an LRS admits a closed-form representation as an exponential polynomial; specifically, for each $n \in \mathbb{N}_0$, we have $u_n = A_1(n)\lambda_1^n + \cdots + A_\ell(n)\lambda_\ell^n$. Here the *characteristic roots* $\lambda_1, \ldots, \lambda_\ell$ are the distinct roots of f. Further, the polynomial coefficients $A_j \in \overline{\mathbb{Q}}[X]$ are completely determined by the initial values of $\langle u_n \rangle_n$. The polynomial coefficients for a simple LRS are all constants; that is, if $\langle u_n \rangle_n$ is simple, then $u_n = A_1\lambda_1^n + \cdots + A_\ell\lambda_\ell^n$.

An LRS is *degenerate* when there are two distinct characteristic roots whose quotient is a root of unity. Otherwise, the sequence is said to be *non-degenerate*.

Let $\lambda_1, \ldots, \lambda_\ell$ be the characteristic roots of an LRS $\langle u_n \rangle_n$. A characteristic root λ of $\langle u_n \rangle_n$ is *dominant* if $|\lambda| \geq |\lambda_j|$ for each $j \in \{1, \ldots, \ell\}$. By convention, when we talk about the number of dominant roots *we do not count multiplicity*; e.g., a recurrence sequence that satisfies the relation $u_{n+2} = 2u_{n+1} - u_n$ with characteristic polynomial $(X-1)^2$ has only one dominant root.

In the sequel, we will state and prove technical results for polynomials in $\mathbb{Z}[X]$. Here we say that a polynomial in $\mathbb{Z}[X]$ is *non-degenerate* if no quotient of distinct roots is a root of unity and we say that it is *reversible* if it is monic and has constant term ± 1. Note that our use of these terms for both recurrence sequences and their characteristic polynomials is consistent.

2.2 The Positivity Problem

In this subsection we briefly recall decidability results for the Positivity Problem for LRS. We first recall the standard assumptions that permit us to reduce the problem of deciding positivity to that of deciding positivity for LRS that are both non-degenerate and possess a positive dominant characteristic root.

First, it is well known (cf. [6, 8]) that we can effectively reduce the computational study of LRS to that of non-degenerate LRS. This observation follows because each degenerate LRS can be realised as an interleaving of finitely many non-degenerate LRS of the same order. Thus we need only consider non-degenerate LRS when studying positivity. We note also that this reduction preserves the quality of having simple characteristic roots.

Second, let us recall the following classical consequence of the Vivanti–Pringsheim Theorem from complex analysis [17, 23] (see also [21, Section 7.21]).

▶ **Lemma 3.** *Suppose that a non-zero LRS $\langle u_n \rangle_n$ has no positive dominant characteristic root. Then the sets $\{n \in \mathbb{N} : u_n > 0\}$ and $\{n \in \mathbb{N} : u_n < 0\}$ are both infinite.*

As a consequence of Lemma 3, we can reduce the problem of deciding positivity to LRS that possess a positive dominant characteristic root.

Together, the two preceding assumptions show that the sequences we consider have an odd number of dominant roots: the set of dominant roots comprises complex-conjugate pairs of roots and a single positive dominant root. Note that a second real dominant root would violate non-degeneracy.

Ouaknine and Worrell showed that the Simple Positivity Problem for LRS is decidable up to order nine [13]. The main technical contribution of that paper was the following result, which, in combination with the various observations above, covers all sequences up to order nine.

▶ **Theorem 4.** *Let $\langle u_n \rangle_n$ be a non-degenerate simple LRS with characteristic polynomial $f \in \mathbb{Z}[X]$ and a positive dominant root. If $f \in \mathbb{Z}[X]$ has either at most eight dominant roots or precisely nine roots, then we can determine whether $u_n \geq 0$ for each $n \in \mathbb{N}_0$.*

2.3 Number Theory

An algebraic integer is a *unit* if its multiplicative inverse is also an algebraic integer. It is a basic fact that an algebraic number is a unit if and only if its minimum polynomial in $\mathbb{Z}[X]$ is monic and has constant term ± 1. Thus the characteristic roots of a reversible LRS are all units.

A *number field* K is a field determined by a finite extension of $\mathbb{Q}$. The *splitting field* K for the polynomial $f \in \mathbb{Q}[X]$ is the field extension of $\mathbb{Q}$ with the following two properties. First, the polynomial f can be written as a product of linear factors in $K[X]$ (i.e., f splits completely in K) and second, f does not split completely over any proper subfield of K containing $\mathbb{Q}$.

2.4 Group Theory

A finite group G is said to act *transitively* on a finite set X if for each pair $x, y \in X$ there is a $g \in G$ such that $gx = y$. The *stabilizer* G_x of an element x in X is defined as the set $\{g \in G : gx = x\}$. The Orbit-Stabilizer Theorem (see, for example, [18, Theorem 3.19]) implies that if G acts transitively on X, the cardinality of G_x is the same for each $x \in X$. Further, $\#\{g \in G : gx = y\}$ is the same for all $x, y \in G$ in this scenario.

2.5 Galois Theory

We assume familiarity with basic notions in Galois theory and the theory of number fields. For reference, we recommend standard textbooks such as [3, 20]. The following includes some of the definitions and theory we use in the sequel.

Given a field extension K of $\mathbb{Q}$, we use $\mathrm{Gal}_{\mathbb{Q}}(K)$ to denote the *Galois group of K over $\mathbb{Q}$*; that is, the group of automorphisms of K that fix $\mathbb{Q}$. We shall refer to elements of a Galois group as *Galois automorphisms*. In the sequel, we shall frequently use the following property of irreducible polynomials. Let $f \in \mathbb{Q}[X]$ be an irreducible polynomial and K the splitting field of f. Then the Galois group $\mathrm{Gal}_{\mathbb{Q}}(K)$ acts transitively on the roots of f. Indeed, the *orbit* of a root of f (i.e., the set of images of the root under the group action) is the set of roots of f. In light of the above, for a given algebraic number α we use the term *Galois conjugates* to refer to set of roots of the minimal polynomial of α.

Recall the following theorem due to Kronecker [10].

▶ **Theorem 5.** *Let $f \in \mathbb{Z}[X]$ be a monic polynomial such that $f(0) \neq 0$. Suppose that all the roots of f have absolute value at most 1. Then all the roots of f are roots of unity.*

We deduce the following. If $f \in \mathbb{Z}[X]$ is the characteristic polynomial of a reversible LRS $\langle u_n \rangle_n$ such that the roots of f all lie in the unit disk $\{z \in \mathbb{C} : |z| \leq 1\}$, then the roots of f are all roots of unity and so $\langle u_n \rangle_n$ is of order one or degenerate. In the former case, positivity of $\langle u_n \rangle_n$ is easily determined and in the latter case, determining whether $\langle u_n \rangle_n$ is positive reduces to studying positivity for associated non-degenerate LRS. Thus in the sequel we shall always assume, without loss of generality, that the dominant roots of f lie on a circle with radius strictly larger than 1.

The roots of an irreducible polynomial are necessarily Galois conjugates. We call the quotient of two distinct roots of an irreducible polynomial a *conjugate ratio*.

Key to the technical lemmas we prove in the sequel are results concerning identities between the roots of irreducible polynomials. We employ a powerful result due to Dubickas and Smyth [5], Theorem 6 below, concerning necessary conditions for an algebraic unit and all its Galois conjugates to lie on two concentric circles centred at the origin. (Theorem 6 is a specialisation of the general result [5, Theorem 2.1].)

▶ **Theorem 6.** *Suppose that $f \in \mathbb{Z}[X]$ is an irreducible, reversible polynomial of degree d such that all the roots of f lie on two circles centred at the origin. Let r and R be the radii of the respective circles and, without loss of generality, suppose that at most half of the roots of f lie on the circle of radius r. Then we have the following: either d is even, in which case half of the roots lie on the circle of radius r; or d is a multiple of three and a third of the roots lie on the circle of radius r. In the latter case, we additionally have that for every root β on the circle of radius r there exists $n > 0$ such that $\beta^n \in \mathbb{R}$.*

We shall frequently employ the following lemma, versions of which were proved by Smyth [19] and Ferguson [7].

▶ **Lemma 7.** *Suppose that λ is an algebraic number with Galois conjugates β and γ satisfying $\lambda^2 = \beta\gamma$. Then the conjugate ratio λ/β is a root of unity.*

3 Root Analysis of Reversible Polynomials

The main result of this section is Theorem 14, concerning the number of dominant roots of a reversible polynomial. Essentially the theorem says that, excepting a number of special cases, no more than half of the roots of such a polynomial can be dominant. This is the key technical tool behind our main decidability results for the Positivity Problem for reversible LRS.

Let us begin with two lemmas concerning the dominant roots of reversible polynomials. These can be considered as weak forms of the main result of the section (and are used in the proof thereof).

▶ **Lemma 8.** *Let $f \in \mathbb{Z}[X]$ be an irreducible non-degenerate polynomial with a real dominant root λ. Then f has exactly one dominant root.*

Proof. Let λ be a real dominant root. Suppose β is also a dominant root. Then $\lambda^2 = \beta\overline{\beta}$ and hence λ/β is a root of unity by Lemma 7. Since f is non-degenerate we have $\lambda = \beta$; that is, f has exactly one dominant root. ◀

▶ **Lemma 9.** *Suppose that $f \in \mathbb{Z}[X]$ is irreducible, non-degenerate, and reversible, with $2m$ non-real dominant roots and no real dominant roots. Then $\deg(f) > 3m$ if $m \geq 2$. Further, $\deg(f) \leq 3m$ only if $(\deg(f), m) = (3, 1)$ or f is constant.*

Proof. Since f has at least $2m$ roots, it is clear that $\deg(f) \geq 2m$. The case where $m = 0$ pertains to constant polynomials, thus we need only consider the case when $m \geq 1$.

We will first show that $\deg(f) > 2m$. Assume, for a contradiction, that $\deg(f) = 2m$. Then the roots of f all lie on the circumference of some circle centred at the origin. We make the following two observations. First, f is reversible, and hence monic. Second, by Vieta's formulas, $|f(0)| = 1$ is equal to the absolute value of the product of the roots of f. From these observations, we conclude that the roots of f all lie on the unit circle and, by Theorem 5, are therefore roots of unity. As $m \geq 1$, f has at least two roots, and their conjugate ratio is thus a root of unity. We have reached a contradiction: f is assumed to be non-degenerate. Thus $\deg(f) > 2m$.

Consider the subcase where $m = 1$. Assume that $2m < \deg(f) \leq 3m$, then clearly we have $\deg(f) = 3$. The assertion in the lemma trivially holds. Hereafter we assume that $m \geq 2$.

We now show that under the assumption that $m \geq 2$, we necessarily have $\deg(f) \geq 3m$. Let $\lambda_1, \overline{\lambda_1}, \ldots, \lambda_m, \overline{\lambda_m}$ be the $2m$ dominant roots of f. Thus $\lambda_1\overline{\lambda_1} = \lambda_i\overline{\lambda_i}$ for each $i \in \{1, \ldots, m\}$. Since $2m < \deg(f)$, f has a non-dominant root γ. Further, since f is irreducible, there is a Galois automorphism σ such that $\sigma(\lambda_1) = \gamma$. We claim that for each $i \in \{2, \ldots, m\}$ at least one of $\sigma(\lambda_i)$ and $\sigma(\overline{\lambda_i})$ is a non-dominant root of f. Assume, for a contradiction, that the claim does not hold. Then there is an $i \in \{2, \ldots, m\}$ such that both $\sigma(\lambda_i)$ and $\sigma(\overline{\lambda_i})$ are dominant roots. The map σ necessarily preserves polynomial symmetries between the roots of f and so $\gamma\sigma(\overline{\lambda_1}) = \sigma(\lambda_i)\sigma(\overline{\lambda_i})$. However, since $\sigma(\lambda_1) = \gamma$ is strictly smaller in absolute value than both $\sigma(\lambda_i)$ and $\sigma(\overline{\lambda_i})$, we have $|\gamma\sigma(\overline{\lambda_1})| < |\sigma(\lambda_i)\sigma(\overline{\lambda_i})|$. This last inequality contradicts the aforementioned symmetry between dominant roots. We conclude that the list of non-dominant roots of f includes γ and at least one of $\sigma(\lambda_i)$ and $\sigma(\overline{\lambda_i})$ for each $i \in \{2, \ldots, m\}$. Thus f has at least m non-dominant roots and so $\deg(f) \geq m + 2m = 3m$.

Finally, we eliminate the case that $\deg(f) = 3m$ when $m \geq 2$. Assume, for a contradiction, that $\deg(f) = 3m$. We can apply the preceding argument to the reciprocal polynomial of f to deduce that the m non-dominant roots of f are equal in modulus and so all lie on a circle $\{z \in \mathbb{C} : |z| = r\}$ for some $r > 0$. Thus all roots of the irreducible and reversible polynomial

f lie on two circles centred at the origin: the dominant roots all lie on one circle, and all the non-dominant roots lie on another circle. Thus, by Theorem 6, each non-dominant root of f takes the form $re^{i\theta}$ where $e^{i\theta}$ is a root of unity. Since $m \geq 2$, there are at least two distinct roots of f, say, $re^{i\theta}$ and $re^{i\theta'}$ of the prescribed form. It follows that the conjugate ratio $re^{i\theta}/re^{i\theta'}$ is a root of unity, which contradicts our assumption that f is non-degenerate. Hence $\deg(f) > 3m$ if $m \geq 2$, from which the desired result follows. ◀

In order to improve the bound from from $\deg(f) > 3m$ to $\deg(f) \geq 4m$, we shall introduce new and novel techniques for counting symmetries in the roots of f. Let $\lambda_1, \ldots, \lambda_\ell$ be the roots of f. The interesting case occurs when all the dominant roots of f are non-real. Let us denote the dominant roots of f by $\lambda_1, \overline{\lambda_1}, \ldots, \lambda_m, \overline{\lambda_m}$. Let $\mu_1 := \lambda_1\overline{\lambda_1}$ and g be the minimal polynomial of μ_1 (hereafter we shall refer to g as the *dominating polynomial of* f). Let $\mu_2, \ldots, \mu_n$ be the Galois conjugates of μ_1 (and thus the other roots of g) and $\sigma_1, \ldots, \sigma_n$ the Galois automorphisms associated with g such that $\sigma_j(\mu_1) = \mu_j$.

Set $K = \mathbb{Q}(\mu_1, \ldots, \mu_n)$ and $L = \mathbb{Q}(\lambda_1, \ldots, \lambda_\ell)$. Clearly, $K \subset L$, and so each σ_j can be lifted to an automorphism $\widetilde{\sigma_j}$ in $\mathrm{Gal}_{\mathbb{Q}}(L)$ such that $\widetilde{\sigma_j}|_K = \sigma_j$. Applying these $\widetilde{\sigma_j}$ on $\lambda_1\overline{\lambda_1} = \cdots = \lambda_m\overline{\lambda_m} = \mu_1$ gives rise to the following n equations:

$$\begin{aligned} \alpha_{1,1,1}\alpha_{1,1,2} &= \ldots = \alpha_{m,1,1}\alpha_{m,1,2} &&= \mu_1 \\ \vdots \quad &\quad \vdots \quad &&\quad \vdots \\ \alpha_{1,n,1}\alpha_{1,n,2} &= \ldots = \alpha_{m,n,1}\alpha_{m,n,2} &&= \mu_n \end{aligned} \tag{3}$$

where $\alpha_{i,j,1} = \widetilde{\sigma_j}(\lambda_i)$ and $\alpha_{i,j,2} = \widetilde{\sigma_j}(\overline{\lambda_i})$. Since each $\alpha_{i,j,k}$ is a Galois conjugate of a dominant root of f, we determine that each $\alpha_{i,j,k}$ is also a root f. Given a root λ of f, we define the *equation number*

$$E = \#\{(i,j,k) : \alpha_{i,j,k} = \lambda \text{ for } 1 \leq i \leq m, 1 \leq j \leq n, k = 1,2\}.$$

In Lemma 11, we will show that E is independent of the choice of root λ. It is useful to see the two roots of f in one position in one equation in (3) as a *pair*. In other words, $\alpha_{i,j,1}$ and $\alpha_{i,j,2}$ are *paired* for all $1 \leq i \leq m$ and $1 \leq j \leq n$. Further, for $j = 1, \ldots, n$, let $\mathscr{A}_j := \{\alpha_{1,j,1}, \alpha_{1,j,2}, \ldots, \alpha_{m,j,1}, \alpha_{m,j,2}\}$. Note that $\#\mathscr{A}_j = 2m$, as $\widetilde{\sigma_j}$ is a bijection between the set of dominant roots of f and $\mathscr{A}_j$.

We claim that $\mathscr{A}_j$ is independent of the choice of lift $\widetilde{\sigma_j}$ of σ_j. If λ and λ' are roots of f such that $\lambda\lambda' = \mu_j$, then $\widetilde{\sigma_j}^{-1}(\lambda)\widetilde{\sigma_j}^{-1}(\lambda') = \mu_1$. Because $\widetilde{\sigma_j}^{-1}(\lambda)$ and $\widetilde{\sigma_j}^{-1}(\lambda')$ are roots of f whose product is equal to $\mu_1 = \lambda_1\overline{\lambda_1}$, we easily deduce that both $\widetilde{\sigma_j}^{-1}(\lambda)$ and $\widetilde{\sigma_j}^{-1}(\lambda')$ are dominant roots of f. Further, since $\widetilde{\sigma_j}$ is a bijection, $\lambda = \widetilde{\sigma_j}\big(\widetilde{\sigma_j}^{-1}(\lambda)\big) \in \mathscr{A}_j$ (and similarly $\lambda' \in \mathscr{A}_j$). We make two deductions. First, if μ_j is the product of two distinct roots of f then those roots are two elements of $\mathscr{A}_j$. Second, we infer our claim that $\mathscr{A}_j$ is independent of the choice of $\widetilde{\sigma_j}$.

In the case of one dominant root, the same construction applies: g is defined as the minimum polynomial of λ_1^2, where λ_1 is the sole dominant root of f. By non-degeneracy, the squares of all roots of f are distinct, and so $\deg(f) = \deg(g)$, $\mu_j = \lambda_j^2$ for $j = 1, \ldots, \deg(f)$ and $E = 2$ for all roots of f (it appears once as a square). Only, $\mathscr{A}_j = \{\lambda_j\}$ consists of exactly one root of f.

▶ **Lemma 10.** *Suppose that $f \in \mathbb{Z}[x]$ is reversible, non-degenerate, and irreducible with $2m$ non-real dominant roots and has degree less than $4m$. Write g for the dominating polynomial of f. Then g is also non-degenerate.*

Proof. Assume, for a contradiction, that the conjugate ratio $\mu_j/\mu_{j'}$ of g is a root of unity. Both root sets $\mathscr{A}_j$ and $\mathscr{A}_{j'}$ have cardinality $2m$. Since $\deg(f) < 4m = \#\mathscr{A}_j + \#\mathscr{A}_{j'}$, we deduce that $\mathscr{A}_j \cap \mathscr{A}_{j'}$ is non-empty. Let $\lambda \in \mathscr{A}_j \cap \mathscr{A}_{j'}$ and κ, κ' be roots of f such that $\lambda\kappa = \mu_j$ and $\lambda\kappa' = \mu_{j'}$. Since $\mu_j \neq \mu_{j'}$, we have $\kappa \neq \kappa'$. It follows that f is degenerate because $\kappa/\kappa' = \mu_j/\mu_{j'}$ is a root of unity. From this contradiction, we deduce that g is non-degenerate. ◀

▶ **Lemma 11.** *Suppose that $f \in \mathbb{Z}[X]$ is reversible, non-degenerate, and irreducible with $2m$ non-real dominant roots. Write g for the dominant polynomial of f. Then all the roots of f have the same equation number E and*

$$2m \deg(g) = E \deg(f). \tag{4}$$

Proof. We use the notation of $\lambda_i, \mu_j, \sigma_j, \widetilde{\sigma_j}$, $\alpha_{i,j,k}$, K, L, etc. as above.

Set $H = \mathrm{Gal}_{\mathbb{Q}}(K)$ and $G = \mathrm{Gal}_{\mathbb{Q}}(L)$. By the Orbit-Stabilizer Theorem, the number of $\sigma \in H$ such that $\sigma(\mu_1) = \mu_j$ is independent of the choice of $j \in \{1, \ldots, n\}$. Now each $\sigma \in H$ has the same number of lifts to G, and so the number of elements of G that map μ_1 to each μ_j is independent of $j \in \{1, \ldots, n\}$. Thus the number of elements of G such that the image of $\mathscr{A}_1$ is $\mathscr{A}_j$ is also independent of the choice of j.

We make the following claim whose proof is given immediately below.

▷ Claim 12. In the setting defined above, there is no pair of distinct j_1 and j_2 for which $\mathscr{A}_{j_1} = \mathscr{A}_{j_2}$.

We also make the following observation. By the Orbit-Stabilizer Theorem, for every choice of two roots λ and λ' of f, the number of $\sigma \in G$ such that $\sigma(\lambda) = \sigma(\lambda')$ is equal. Thus for each root λ of f, the number of $\sigma \in G$ such that one of $\widetilde{\sigma}(\lambda_1), \widetilde{\sigma}(\overline{\lambda_1}), \ldots, \widetilde{\sigma}(\lambda_m), \widetilde{\sigma}(\overline{\lambda_m})$ equals λ is independent of the choice of λ. This shows that the equation number E is independent of the choice of the root λ.

The equation $2m \deg(g) = E \deg(f)$ follows from counting the number of $\alpha_{i,j,k}$. On the one hand, there are $\deg(g)$ equations with $2m$ entries (Claim 12). On the other hand, there are $\deg(f)$ roots each appearing E times. ◀

Proof of Claim 12. Let us assume, for a contradiction, that $\mathscr{A}_{j_1} = \mathscr{A}_{j_2}$ for $j_1 \neq j_2$. Then $\mu_{j_1} \neq \mu_{j_2}$ and

$$\mu_{j_1}^m = \prod_{i=1}^{m} \alpha_{i,j_1,1}\alpha_{i,j_1,2} = \prod_{i=1}^{m} \alpha_{i,j_2,1}\alpha_{i,j_2,2} = \mu_{j_2}^m.$$

Thus μ_{j_1}/μ_{j_2} is a root of unity. Since $\alpha_{1,j_1,1} \in \mathscr{A}_{j_2}$, there are $1 \leq i \leq m$ and $k \in \{1, 2\}$ such that $\alpha_{1,j_1,1} = \alpha_{i,j_2,k}$. Then we have that the conjugate ratio $\alpha_{1,j_1,2}/\alpha_{i,j_2,3-k}$ given by

$$1 \neq \frac{\mu_{j_1}}{\mu_{j_2}} = \frac{\alpha_{1,j_1,1}\alpha_{1,j_1,2}}{\alpha_{i,j_2,1}\alpha_{i,j_2,2}} = \frac{\alpha_{1,j_1,2}}{\alpha_{i,j_2,3-k}}$$

is also a root of unity. Since $\alpha_{1,j_1,2}$ and $\alpha_{i,j_2,3-k}$ are distinct roots of f whose quotient is a root of unity, it follows that f is degenerate. We have reached a contradiction to our assumption that f is non-degenerate. Thus we have the claimed result. ◁

The next result increases the bound on the degree of f to $\deg(f) \geq 4m$.

▶ **Theorem 13.** *Let $f \in \mathbb{Z}[X]$ be an irreducible, non-degenerate, and reversible polynomial with $2m$ dominant non-real roots and no real dominant roots, then $(\deg(f), m) = (3, 1)$ or $\deg(f) \geq 4m$.*

Proof. Assume, for a contradiction, that f is a minimal counterexample in the sense that all polynomials of strictly smaller degree satisfy the statement in Theorem 13.

From Lemma 9, we deduce that $\deg(f) > 3m$ if we are not in the exceptional case $(\deg(f), m) = (3, 1)$. As we assume that f is a counterexample to Theorem 13, $\deg(f) < 4m$ as well. We shall employ the preceding notation for the dominating polynomial g, the sets of roots $\mathscr{A}_j$ of f, and the equation number E.

Consider that there are $2m$ distinct roots of f in each equation in (3). Since $\deg(f) < 4m$ and f has $2m$ dominant roots, there is at least one dominant root of f in each such equation. Let γ be a root of f with minimal absolute value, then $|\gamma\lambda_1|$ is the minimal absolute value attained by any root of g. We now show that at least half of the roots of g lie on the circle $\{z \in \mathbb{C} : |z| = |\gamma\lambda_1|\}$. Observe that γ is witnessed in E (and so more than half) of the pairings in (3) and, further, is necessarily paired with a dominant root (for otherwise, a pairing between γ and a non-dominant root breaks the equality in (3)). From (4) and our assumption that $\deg(f) < 4m$, we deduce that $2E > \deg(g)$, and so γ appears in more than half of the equations in (3). Each such equation is in correspondence with a root of g of minimal absolute value.

Consider the polynomial $h(X) := g(0)X^n g(X^{-1})$. The polynomial $X^n g(X^{-1})$ is the reciprocal polynomial of g and so immediately, the roots of h are precisely $\mu_1^{-1}, \ldots, \mu_n^{-1}$ and $n = \deg(g) = \deg(h)$. From the preceding discussion, more than half of the roots of h are dominant. Moreover, we can easily deduce that h is reversible, irreducible, and non-degenerate as g has these properties. Thus h is another counterexample to the statement in Theorem 13. All that remains is to derive a contradiction from our assumption that f has minimal degree. We derive this contradiction by proving that $\deg(h) < \deg(f)$ and h does not belong to either one of the exceptional cases.

Observe that h cannot belong to one of the exceptional cases since (4) has no integer solutions when $n = 1, 2, 3$ and $3m < \deg(f) < 4m$. Thus it remains to show that $n \geq \deg(f)$ is absurd. Let us assume, for a contradiction, that λ_1 appears in a product pair with a dominant root other than $\overline{\lambda_1}$ in the jth equation of (3). Then μ_1 and μ_j have equal absolute value. If μ_j is real, $\mu_1/\mu_j = \pm 1$ contradicting our non-degeneracy assumption (Lemma 10). Similarly, we derive a contradiction to our non-degeneracy assumption if μ_j is non-real by Lemma 7. Thus, we can pair λ_1 with the $\deg(f) - 2m < 2m$ non-dominant roots of f and $\overline{\lambda_1}$. This gives the upper bound $E \leq \deg(f) - 2m + 1 \leq 2m$ on E. We substitute our assumption that $n = \deg(g) \geq \deg(f)$ into (4) to obtain a lower bound $2m \leq E$. Thus, $E = 2m$.

We use the equality $E = 2m$ to deduce that $\deg(f) = 4m - 1$ and make the following observations. Each of the $2m$ dominant roots of f pair with their respective complex conjugate and all of the $2m - 1$ non-dominant roots of f. Thus we can pair each non-dominant root of f with $2m$ dominant roots. Further, every pair of non-dominant roots of f appears in at least one equation in (3). Thus the roots of g and h lie on two concentric circles centred at the origin. The roots of g are distributed so that g has exactly one dominant root and $2m + (2m - 1) - 1 = 4m - 2$ non-dominant roots. By construction, h has exactly one non-dominant root and $4m - 2$ dominant roots. This distribution of roots is not possible by Theorem 6, hence a contradiction.

In summary, f cannot be a counterexample to Theorem 13 of smallest possible degree. We thus deduce that all polynomials that satisfy the hypothesis in the theorem obey the bound $\deg(f) \geq 4m$, as required. ◀

The only superfluous assumption in Theorem 13 is that f is irreducible. We circumvent the irreducibility assumption with a careful case analysis.

▶ **Theorem 14.** *Let f be a non-degenerate reversible polynomial. Suppose that more than half of the roots of f are dominant. Then either f is linear or f is cubic with two dominant roots.*

Proof. Let f be a counterexample of minimal degree, and factor f into irreducible polynomials $f_1, \ldots, f_k$. For $1 \leq i \leq k$, let m_i' be the number of dominant roots of f_i. Call an irreducible factor *sharp* if $2m_i' = \deg(f_i)$ and *special* if $2m_i' > \deg(f_i)$. From Lemma 8 and Theorem 13, it follows that if an irreducible factor is special, then $(\deg(f_i), m_i') = (1,1)$ or $(3,2)$. If $k = 1$, then f is irreducible and the result follows automatically. Thus we can freely assume that $k \geq 2$. Since f is a counterexample of minimal degree, a straightforward proof by contradiction permits us to assume $k = 2$. Thus our argument reduces to the following cases: we need only show that the product of either two special polynomials or a special and a sharp polynomial breaks the hypothesis. By renumbering, we can assume f_1 is special and f_2 is either sharp or special. We observe that under our assumptions the dominant roots of f_1 and f_2 are necessarily equal in absolute value and, as we do not count multiplicity, $f_1 \neq f_2$.

We begin our case analysis. First, consider the case where $(\deg(f_1), m_1') = (1,1)$. Then $f_1(X) = X \pm 1$ as f_1 is reversible. Since the root ∓ 1 of f_1 is a dominant root of f, we deduce that all roots of f lie on the unit circle as the roots of f are algebraic units. When we combine Lemma 8, Theorem 13, and our assumption that at least half of the roots of f are dominant, we deduce that $(\deg(f_2), m_2') = (1,1)$ and so $f_2(X) = X \mp 1$. Thus -1 and 1 are both roots of f, which contradicts our assumption that f is non-degenerate.

Second, let us suppose that $(\deg(f_1), m_1') = (3,2)$. Following the argument in the preceding case, either $(\deg(f_2), m_2') = (3,2)$ or $\deg(f_2) = 2m_2'$. In the former, the non-dominant roots γ_1 and γ_2 of f_1 and f_2 (respectively) are both real and equal in modulus. This is straightforward to see since each f_j is of the form $f_j = (x - \gamma_j)(x - R\mathrm{e}^{\mathrm{i}\theta_j})(x - R\mathrm{e}^{-\mathrm{i}\theta_j})$ for some $R > 1$ and $\gamma_j := \pm R^{-2}$. We cannot have two such irreducible factors since then the ratio $\gamma_1/\gamma_2 = \pm 1$, which breaks either the non-degeneracy assumption on f or the assumption that $f_1 \neq f_2$.

We continue with the latter subcase $(\deg(f_1), m_1') = (3,2)$ and $\deg(f_2) = 2m_2'$. Since the dominant roots of f_1 and f_2 are dominant roots of f, the dominating polynomials of f_1 and f_2 are one and the same, say g. Let E_1 and E_2 be the respective equation numbers of f_1 and f_2. From (4), $2\deg(g) = E_1 \deg(f_1) = 3E_1$. We thus deduce that E_1 is even. Since $1 \leq E_1 \leq \deg(f_1) = 3$ (each pairing is distinct), we have that $E_1 = 2$ and, it follows immediately, $\deg(g) = 3$. We substitute this result and our assumption that $\deg(f_2) = 2m_2'$ into (4) in order to obtain $m_2' \deg(g) = 3m_2' = 2E_2 m_2'$. We have reached a contradiction: $E_2 = 3/2$ is not an integer. We have exhausted the possibilities for constructing a minimal counterexample f and find that no such counterexample exists. We have thus proved Theorem 14. ◀

4 Decidability at Low Orders

In this section we complete the proofs of our two main theorems concerning the Positivity Problem for reversible LRS. We start with Theorem 2, which states that positivity of reversible sequences that are moreover simple is decidable up to order 17.

Proof of Theorem 2. As previously noted, we can reduce the Simple Reversible Positivity Problem to deciding positivity for the subclass of simple reversible LRS that are additionally both non-degenerate and in possession of a positive dominant root.

In light of the preceding paragraph, consider the subclass of non-degenerate, simple, and reversible LRS with a positive dominant root. Let f be the characteristic polynomial associated with a sequence in this class. Without loss of generality, we can additionally

assume that fewer than half of the roots of f are dominant. If $f \in \mathbb{Z}[X]$ has at least nine dominant roots, then, by Theorem 14, we have the bound $\deg(f) \geq 18$. Taking the contrapositive, if f is again the characteristic polynomial of a sequence in this subclass with $\deg(f) \leq 17$, then f has at most eight dominant roots.

Now we invoke Theorem 4 to deduce that, in the aforementioned subclass, positivity is decidable for LRS up to order 17. As noted at the beginning of this proof, this deduction is sufficient to obtain the desired result: simple reversible positivity is decidable up to order 17. ◀

We now turn our attention to general reversible sequences; i.e., we no longer assume that the characteristic roots are simple. Here, as stated in Theorem 1, we have decidability up to order 11.

Proof of Theorem 1. Assume, for a contradiction, that $\langle u_n \rangle_n$ is a reversible LRS and counterexample to the statement; that is to say, $\langle u_n \rangle_n$ is a reversible LRS with order at most 11 for which we cannot determine positivity or ultimate positivity.

From our earlier discussion on the Positivity Problem and Ultimate Positivity Problem in Subsection 2.2, it follows that we can reduce both problems for reversible LRS to deciding (ultimate) positivity for the subclass of reversible LRS that are additionally both non-degenerate and in possession of a positive dominant root.

For the class of reversible LRS with one dominant root, decidability of (ultimate) positivity is considered folklore. Thus we freely assume that $\langle u_n \rangle_n$ has at least three dominant roots (the positive root and a pair of complex conjugate roots). By Theorem 2 for positivity and the earlier mentioned results in [14] for ultimate positivity, we can also assume that $\langle u_n \rangle_n$ has a non-simple characteristic root. Now consider the exponential polynomial representation of $\langle u_n \rangle_n$: deciding (ultimate) positivity for LRS whose dominant characteristic roots are all simple reduces to deciding (ultimate) positivity for simple LRS. So, in addition, we shall assume that $\langle u_n \rangle_n$ has a non-simple dominant characteristic root. We will use the following claims, whose proofs are given immediately below.

▷ Claim 15. Suppose that the real positive dominant root ρ of sequence $\langle u_n \rangle_n$ (as above) is the only non-simple dominant root of $\langle u_n \rangle_n$. Then we can determine whether $\langle u_n \rangle_n$ is (ultimately) positive.

▷ Claim 16. Suppose that sequence $\langle u_n \rangle_n$ (as above) possesses non-real dominant roots that are not simple and, further, that the real dominant root ρ is simple. Then $\langle u_n \rangle_n$ is neither ultimately positive nor positive.

In light of the preceding claims, we freely assume that the counterexample $\langle u_n \rangle_n$ has at least three non-simple dominant characteristic roots and this collection must include the real dominant root ρ as well as a complex conjugate pair λ and $\overline{\lambda}$.

Let f be the monic integer-valued polynomial of the smallest degree with ρ and λ as roots. Then, f is non-degenerate and reversible. By Theorem 14, it follows that at most half of the roots of f are dominant if f is neither linear nor cubic with two dominant roots. As such, f has degree at least 6 and, additionally, as each of these roots is non-simple being a Galois conjugate of either ρ or λ, $\langle u_n \rangle_n$ has order at least 12.

We thus deduce the desired result: positivity and ultimate positivity are decidable for sequences up to order 11. ◀

Proof of Claim 15. Suppose that the real positive dominant root ρ of $\langle u_n \rangle_n$ is the only non-simple dominant root of the sequence. If such a phenomenon were to take place then $u_n = A_\rho(n)\rho^n + O(\rho^n)$ where A_ρ is a non-constant polynomial. It is straightforward to deduce that $\langle u_n \rangle_n$ is (ultimately) positive if and only if $A_\rho(n)$ is (ultimately) positive in this instance. ◁

Proof of Claim 16. We will show that the claim follows from Lemma 17 (cf. [1]).

▶ **Lemma 17.** *Let $\gamma_1, \ldots, \gamma_k \in \{z \in \mathbb{C} : |z| = 1, z \neq 1\}$ be distinct complex numbers, $\alpha_1, \ldots, \alpha_k \in \mathbb{C} \setminus \{0\}$, and $w_n = \sum_{\ell=1}^{k} \alpha_\ell \gamma_\ell^n$. Then there is a $c < 0$ such that $\mathrm{Re}(w_n) < c$ for infinitely many n.*

To prove the claim, let d be the maximum of the degrees of the roots of $\langle u_n \rangle_n$. Note $d \geq 1$ since, by assumption, $\langle u_n \rangle_n$ has a non-real dominant root that is not simple. We consider the normalised sequence $\langle v_n \rangle_n$ with terms given by $v_n = u_n/(\rho^n n^d)$ where ρ is the dominant root of $\langle u_n \rangle_n$. Note it is sufficient to establish that $\langle v_n \rangle_n$ is neither positive nor ultimately positive to obtain the desired result.

An analysis of the exponential polynomial of $\langle u_n \rangle_n$ leads to

$$v_n = \sum_{\ell=1}^{2k} \frac{A_\ell(n)}{n^d} \frac{\lambda_\ell^n}{\rho^n} + O(n^{-d})$$

where $\lambda_1, \ldots, \lambda_{2k}$ are the non-real dominant roots of $\langle u_n \rangle_n$ and the implied constant associated with $O(n^{-d})$ is real-valued. Let α_ℓ be the leading coefficient of $A_\ell(n)$. Then $A_\ell(n)/n^d \to \alpha_\ell$ as $n \to \infty$. Now

$$v_n < r(n) + \sum_{\ell=1}^{2k} \alpha_\ell \frac{\lambda_\ell^n}{\rho^n} =: r(n) + w_n$$

where $r(n) \in O(n^{-1})$ is real-valued and the LRS $\langle w_n \rangle_n$ is both real-valued and simple. In addition, the characteristic roots of $\langle w_n \rangle_n$ are all non-real and lie on the unit circle. For the avoidance of doubt, the exponential polynomial defining $\langle w_n \rangle_n$ is real-valued since the summands $\alpha_\ell \lambda_\ell^n/\rho^n$ for non-real λ_ℓ occur in complex-conjugate pairs. Thus, w_n satisfies the hypothesis in Lemma 17, and so the inequalities $v_n < r(n) + w_n < r(n) + c$ hold for some $c < 0$ and infinitely many n. Since $r(n) \in O(n^{-1})$, we find that for infinitely many n, $v_n < 0$. Hence $\langle u_n \rangle_n$ is neither positive nor ultimately positive. ◁

5 Hard Instances

In this section we discuss obstacles to extending our results for deciding positivity of reversible LRS of higher orders. Specifically, we construct a simple reversible LRS of order 18 and sketch the construction of a reversible LRS of order 12 that, to the best of our knowledge, lie outside the known classes for which the Positivity Problem can be decided. In particular, these examples lie beyond the scope of Theorem 4.

We start with simple reversible LRS of order 18. In order to illustrate the technical arguments and guide our construction of a hard instance, it is useful to recall the techniques employed by Ouaknine and Worrell in their proof of Theorem 4 [13]. For the sake of brevity, we shall give only a brief outline here; we direct the interested reader to the full argument given in [13].

5.1 Sketch proof of Theorem 4

Let $\langle u_n \rangle_n$ be a simple LRS satisfying the assumptions of Theorem 4. We first normalise $\langle u_n \rangle_n$ and so assume that the dominant roots $\lambda_1, \ldots, \lambda_k$ of $\langle u_n \rangle_n$ lie on the unit circle in the complex plane. Then, for each $n \in \mathbb{N}$,

$$u_n = \alpha_1 \lambda_1^n + \cdots + \alpha_k \lambda_k^n + \beta_1 \xi_1^n + \cdots + \beta_{k'} \xi_{k'}^n$$

where $\xi_1, \ldots, \xi_{k'}$ are the non-dominant roots of $\langle u_n \rangle_n$ and $\alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_{k'}$ are algebraic numbers.

We then compute a basis for the multiplicative relations between the dominant roots and consider a maximal subset $\lambda_1, \ldots, \lambda_\ell$ whose elements are multiplicatively independent. By Kronecker's Theorem on simultaneous Diophantine approximation (cf. [2, page 53]), $\{(\lambda_1^n, \ldots, \lambda_\ell^n) : n \in \mathbb{N}\}$ is a dense subset of the torus $T := \{z \in \mathbb{C} : |z| = 1\}^\ell$, which is compact.

Ouaknine and Worrell then construct a continuous function $\tau \colon T \to \mathbb{R}$ given by

$$\tau(\lambda_1^n, \ldots, \lambda_\ell^n) = \alpha_1 \lambda_1^n + \cdots + \alpha_k \lambda_k^n$$

with the following properties. If $\min_T \tau = 0$, the given sequence $\langle u_n \rangle_n$ is ultimately positive. That is to say, there is a number N such that $u_n \geq 0$ for all $n \geq N$. If $\min_T \tau < 0$, the sequence is not ultimately positive (and thus also not positive). Finally, if $\min_T \tau > 0$, then the sequence grows quickly, and deciding positivity is relatively straightforward. Hence the critical case occurs when $\min_T \tau = 0$. Moreover, we can determine which of the three cases occur (that is, compute $\min_T \tau$).

In the critical case where $\min_T \tau = 0$, we can sometimes exploit the set of points $Z = \{(z_1, \ldots, z_\ell) \in T : \tau(z_1, \ldots, z_\ell) = 0\}$ where the minimum is attained. If $(z_1, \ldots, z_\ell) \in Z$, then Baker's Theorem on linear forms shows that λ_1^n cannot get too "close" to z_1 for n greater than a computable bound. As such, if Z is finite, then we can decide whether $\langle u_n \rangle_n$ is positive.

Theorem 4 is now proven as follows. If $\langle u_n \rangle_n$ has at most eight dominant characteristic roots and falls into the critical case, then Z is finite. Likewise, if $\langle u_n \rangle_n$ has exactly nine characteristic roots all of which are dominant, then $\langle u_n \rangle_n$ is positive in the critical case as $u_n \geq \min_T \tau = 0$.

The approach described breaks down when there are nine dominant roots since then Z is possibly infinite. Briefly, in this setting the state of the art cannot show that $(\lambda_1^n, \ldots, \lambda_\ell^n)$ does not approach this infinite set too "closely". Thus we encounter examples of LRS where we cannot currently determine positivity.

5.2 Constructing a hard example of a simple sequence of order 18

Our hard example is constructed from a function τ that assumes its minimum infinitely often on the torus $T = \{(z_1, z_2) \in \mathbb{C} : |z_1| = |z_2| = 1\}$. To this end, we define $\tau : T \to \mathbb{R}$ by

$$\tau(z_1, z_2) = (a z_1 + \overline{a} z_1^{-1} + b z_2 + \overline{b} z_2^{-1})^2 \tag{5}$$

for some non-zero $a, b \in \mathbb{C}$ with $|a| \neq |b|$. Then $\min_T \tau$ is equal to 0 and τ attains its minimum on an infinite subset of T. This property prevents the application of Theorem 4.

▶ **Example 18.** We shall construct a simple reversible LRS sequence of order 18. An analysis of the spectral properties of this sequence shows that it lies beyond the current state-of-the-art techniques for deciding positivity. This hard instance is derived from the irreducible polynomial

$$f(X) = X^8 - 3X^7 + 4X^6 - 4X^5 + 11X^4 - 21X^3 + 19X^2 - 7X + 1,$$

which has eight non-real roots $\lambda_1, \ldots, \overline{\lambda_4}$ such that λ_1 and λ_2 are dominant, λ_3 and λ_4 are both non-dominant, and $1.143 \approx |\lambda_3| > 1 > |\lambda_4|$.

Let $\phi := (1+\sqrt{5})/2$ denote the golden ratio. Then, with a certain labelling of complex conjugates,

$$\lambda_1\overline{\lambda_1} = \lambda_2\overline{\lambda_2} = \phi^2 \quad \text{and} \quad \lambda_3\lambda_4 = \overline{\lambda_3}\,\overline{\lambda_4} = \phi^{-2},$$

which, due to the number of relations, severely limits the possible Galois automorphisms. In particular, the Galois group has the form of a *wreath product* $D_4 \wr C_2$. Thus a dihedral group D_4 acts on $\lambda_1, \overline{\lambda_1}, \lambda_2$, and $\overline{\lambda_2}$ and is generated by the elements (written in cycle notation) $(\lambda_1\ \lambda_2\ \overline{\lambda_1}\ \overline{\lambda_2})$ and $(\lambda_1\ \overline{\lambda_1})$. A second dihedral group D_4 acts on $\lambda_3, \overline{\lambda_3}, \lambda_4, \overline{\lambda_4}$ and is generated by $(\lambda_3\ \overline{\lambda_3}\ \lambda_4\ \overline{\lambda_4})$ and $(\lambda_3\ \lambda_4)$. Lastly, there is a cyclic C_2 group acting on these two sets of four roots generated by the permutation $(\lambda_1\ \lambda_3)(\overline{\lambda_1}\ \lambda_4)(\lambda_2\ \overline{\lambda_3})(\overline{\lambda_2}\ \overline{\lambda_4})$.

The terms in the sequence $\langle u_n \rangle_n$ are given as follows:

$$u_n = \frac{1}{\sqrt{5}}\Big((1+\lambda_1)\lambda_1^n + (1+\overline{\lambda_1})\overline{\lambda_1}^n + (1+\lambda_2)\lambda_2^n + (1+\overline{\lambda_2})\overline{\lambda_2}^n\Big)^2 \\ - \frac{1}{\sqrt{5}}\Big((1+\lambda_3)\lambda_3^n + (1+\overline{\lambda_3})\overline{\lambda_3}^n + (1+\lambda_4)\lambda_4^n + (1+\overline{\lambda_4})\overline{\lambda_4}^n\Big)^2.$$

By the action of the Galois group, it can be seen that each term u_n is rational and further that $\langle u_n \rangle_n$ is simple, reversible, and has exactly order 18. The initial values $u_0, \ldots, u_{17}$ of $\langle u_n \rangle_n$ are

$$-11, -8, 0, 240, 704, -20, 192, 5508, 46305, 2625, 13425, 73117, \\ 2469800, 536000, 554151, 77287, 108792361, 66461616.$$

The simple LRS $\langle u_n \rangle_n$ satisfies the relation

$$u_{n+18} = u_{n+17} - 10u_{n+16} + 6u_{n+15} + 43u_{n+14} - 93u_{n+13} + 672u_{n+12} - 596u_{n+11} \\ + 120u_{n+10} + 3972u_{n+9} - 15345u_{n+8} + 29654u_{n+7} - 36108u_{n+6} + 23847u_{n+5} \\ - 9572u_{n+4} + 2361u_{n+3} - 325u_{n+2} + 26u_{n+1} - u_n.$$

Observe that u_0, u_1, and u_5 are negative, but up to $n = 10^5$ these are the only negative terms. Thus, the question is to prove that $u_n \geq 0$ for all $n \geq 6$. We reiterate that, as far as the authors are aware, there are no known techniques in the state of the art that can tackle this question.

It remains to show that the torus T associated with $\langle u_n \rangle_n$ has the prescribed "squaring form" (as in (5)) and that $\langle u_n \rangle_n$ is non-degenerate. To start, u_n is positive if and only if $\frac{u_n}{\phi^{2n}}$. Moreover, we observe that $|1+\lambda_1| \neq |1+\lambda_2|$ and that both λ_1/ϕ and λ_2/ϕ lie on the unit circle. For $a = 1+\lambda_1, b = \lambda_2$ and some $0 < r < 1$, we have that

$$\frac{u_n}{\phi^{2n}} = \frac{1}{\phi^{2n}}\Big((1+\lambda_1)\lambda_1^n + (1+\overline{\lambda_1})\overline{\lambda_1}^n + (1+\lambda_2)\lambda_2^n + (1+\overline{\lambda_2})\overline{\lambda_2}^n\Big)^2 + O(r^n) \\ = \left(a\left(\frac{\lambda_1}{\phi}\right)^n + \overline{a}\left(\frac{\lambda_1}{\phi}\right)^{-n} + b\left(\frac{\lambda_2}{\phi}\right)^n + \overline{b}\left(\frac{\lambda_2}{\phi}\right)^{-n}\right)^2 + O(r^n)$$

is close to the "squaring form" discussed at (5). In fact, we have that

$$u_n/\phi^{2n} = \tau((\lambda_1/\phi)^n, (\lambda_2/\phi)^n) + O(r^n).$$

Here, the term $O(r^n)$ decreases exponentially fast and determines how closely the square should approach zero to contradict positivity.

We now show that we cannot apply Theorem 4 in this instance. To this end, we need to show that the points to which we restrict τ are dense on the torus T. That is, we need to show that λ_1/ϕ and λ_2/ϕ are multiplicatively independent. This lack of multiplicative relations also immediately implies that $\langle u_n \rangle_n$ is non-degenerate. We complete the spectral analysis of sequence $\langle u_n \rangle_n$ with the following proposition.

▶ **Proposition 19.** *We have that $\lambda_1/|\lambda_1|$ and $\lambda_2/|\lambda_2|$ are multiplicatively independent.*

Proof. Note that $|\lambda_1| = |\lambda_2| = \phi$ as $\lambda_1\overline{\lambda_1} = \lambda_2\overline{\lambda_2} = \phi^2$. By the earlier described Galois action, we see that there are Galois automorphisms σ and τ such that $\sigma(\lambda_1) = \tau(\lambda_1) = \lambda_3$, $\sigma(\lambda_2) = \lambda_4$ and $\tau(\lambda_2) = \overline{\lambda_3}$. Further, by this choice, $\sigma(\phi) = \tau(\phi) = -\phi^{-1}$.

Assume, for a contradiction, that $\lambda_1/|\lambda_1|$ and $\lambda_2/|\lambda_2|$ are multiplicatively dependent; that is to say, there are $a, b \in \mathbb{Z}$, not both 0, such that $(\lambda_1/|\lambda_1|)^a (\lambda_2/|\lambda_2|)^b = 1$. By applying σ to this identity we obtain

$$1 = \left(\frac{\lambda_3}{-\phi^{-1}}\right)^a \left(\frac{\lambda_4}{-\phi^{-1}}\right)^b = \zeta \left(\frac{|\lambda_3\lambda_4|}{\phi^{-2}}\right)^a \left(\frac{\lambda_4}{-\phi^{-1}}\right)^{b-a} = \zeta \left(\frac{\lambda_4}{-\phi^{-1}}\right)^{b-a}$$

for some ζ on the unit circle. Since $|\lambda_4/-\phi^{-1}| \neq 1$, we conclude that $a = b$. Then when we apply τ to the identity $(\lambda_1/|\lambda_1|)^a (\lambda_2/|\lambda_2|)^b = 1$ we obtain

$$1 = \left(\frac{\lambda_3}{-\phi^{-1}}\right)^a \left(\frac{\overline{\lambda_3}}{-\phi^{-1}}\right)^b = \zeta' \left(\frac{|\lambda_3|}{|\overline{\lambda_3}|}\right)^a \left(\frac{\overline{\lambda_3}}{-\phi^{-1}}\right)^{b+a} = \zeta' \left(\frac{\overline{\lambda_3}}{-\phi^{-1}}\right)^{b+a}$$

for some ζ' on the unit circle. Since $|\overline{\lambda_3}/-\phi^{-1}| \neq 1$, this implies that $a = -b$. Together with $a = b$, we deduce that $a = b = 0$. Thus $\lambda_1/|\lambda_1|$ and $\lambda_2/|\lambda_2|$ are multiplicatively independent. ◀

5.3 Constructing a hard example of a non-simple sequence of order 12

In this subsection, we briefly consider a reversible LRS of order 12 where we cannot decide positivity nor ultimate positivity. Explicit examples are easier to construct than in the simple case and are closely related to the extensive discussion in [16]. Let us recall the following point from Theorem 1: a non-simple LRS that is a hard example of (ultimate) positivity possesses three simple dominant roots of which one is real and positive. One choice, closely resembling Example 4.5 in [11], is to take

$$\rho = \sqrt{2} + 1 \quad \text{and} \quad \lambda = \frac{1 + \sqrt{1 - 4\rho^2}}{2}.$$

Then we have that ρ and λ are units of equal modulus, ρ has one Galois conjugate $\tilde{\rho}$ of smaller modulus, and λ has three Galois conjugates. The three Galois conjugates of λ are its complex conjugate and two real numbers, say, λ_3 and λ_4 of smaller modulus. Lastly, let $q \in \mathbb{Q}_{>0}$. Then define the non-simple reversible rational-valued LRS $\langle u_n^q \rangle_n$ as follows:

$$u_n^q = (n+\rho)\rho^n + (n+\tilde{\rho})\tilde{\rho}^n + q(n+\lambda)\lambda^n + q(n+\overline{\lambda})\overline{\lambda}^n + q(n+\lambda_3)\lambda_3^n + q(n+\lambda_4)\lambda_4^n.$$

For small q, $\langle u_n^q \rangle_n$ is positive and so ultimately positive. For sufficiently large q, $\langle u_n^q \rangle_n$ is neither positive nor ultimately positive. However, given the current state of the art, it is not known how to determine where an arbitrary q falls in this partition. Thus, at the time of writing, we cannot tell whether LRS of the form $\langle u_n^q \rangle_n$ are (ultimately) positive.

Following [11, Section 4.2]), we can construct further LRS (akin to $\langle u_n^q \rangle_n$) where the state of the art is unable to settle positivity and ultimate positivity. In this direction, we may take a real quadratic unit $\rho > 1$ and find a non-real algebraic unit λ of equal modulus such that λ has a minimum polynomial of degree 4.

References

1 Mark Braverman. Termination of integer linear programs. In *International conference on computer aided verification*, pages 372–385. Springer, 2006.

2 J. W. S. Cassels. *An introduction to Diophantine approximation.* Cambridge Tracts in Mathematics and Mathematical Physics, No. 45. Cambridge University Press, New York, 1957.

3 Henri Cohen. *A course in computational algebraic number theory*, volume 138 of *Graduate Texts in Mathematics.* Springer-Verlag, Berlin, 1993.

4 The SageMath Developers. *SageMath, the Sage Mathematics Software System (Version 9.7)*, 2022. https://www.sagemath.org.

5 A. Dubickas and C. J. Smyth. On the Remak height, the Mahler measure and conjugate sets of algebraic numbers lying on two circles. *Proc. Edinb. Math. Soc. (2)*, 44(1):1–17, 2001. doi:10.1017/S001309159900098X.

6 Graham Everest, Alf van der Poorten, Igor Shparlinski, and Thomas Ward. *Recurrence sequences*, volume 104 of *Mathematical Surveys and Monographs.* Amer. Math. Soc., Providence, RI, 2003.

7 Ronald Ferguson. Irreducible polynomials with many roots of equal modulus. *Acta Arith.*, 78(3):221–225, 1997. doi:10.4064/aa-78-3-221-225.

8 Vesa Halava, Tero Harju, Mika Hirvensalo, and Juhani Karhumäki. Skolem's problem–on the border between decidability and undecidability. Technical report, Turku Centre for Computer Science, 2005.

9 George Kenison. On the Skolem Problem for Reversible Sequences. In Stefan Szeider, Robert Ganian, and Alexandra Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, volume 241 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 61:1–61:15, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2022.61.

10 L. Kronecker. Zwei Sätze über Gleichungen mit ganzzahligen Coefficienten. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1857(53):173–175, January 1857. doi:10.1515/crll.1857.53.173.

11 Richard Lipton, Florian Luca, Joris Nieuwveld, Joël Ouaknine, David Purser, and James Worrell. On the Skolem Problem and the Skolem Conjecture. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '22, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3531130.3533328.

12 Maurice Mignotte, Tarlok Shorey, and Robert Tijdeman. The distance between terms of an algebraic recurrence sequence. *Journal für die Reine und Angewandte Mathematik*, pages 63–76, 1984.

13 Joël Ouaknine and James Worrell. On the Positivity Problem for Simple Linear Recurrence sequences,. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming*, pages 318–329, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

14 Joël Ouaknine and James Worrell. Ultimate positivity is decidable for simple linear recurrence sequences. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 330–341. Springer, 2014. doi:10.1007/978-3-662-43951-7_28.

15 Joël Ouaknine and James Worrell. On linear recurrence sequences and loop termination. *ACM SIGLOG News*, 2(2):4–13, April 2015.

16 Joël Ouaknine and James Worrell. *Positivity Problems for Low-Order Linear Recurrence Sequences*, pages 366–379. ACM, New York, 2014. doi:10.1137/1.9781611973402.27.

17 Alfred Pringsheim. Ueber Functionen, welche in gewissen Punkten endliche Differentialquotienten jeder endlichen Ordnung, aber keine Taylor'sche Reihenentwickelung besitzen. *Mathematische Annalen*, 44(1):41–56, 1894.

18 J.J. Rotman. *An Introduction to the Theory of Groups*. Graduate Texts in Mathematics. Springer New York, 2012.

19 C. Smyth. Conjugate algebraic numbers on conics. *Acta Arithmetica*, 40(4):333–346, 1982.

20 I. Stewart and D. Tall. *Algebraic number theory and Fermat's last theorem*. CRC Press, Boca Raton, FL, fourth edition, 2016.

21 Edward Charles Titchmarsh. *The theory of functions*. Oxford University Press, 2nd edition, 1939.

22 Nikolai Vereshchagin. Occurrence of zero in a linear recursive sequence. *Mathematical notes of the Academy of Sciences of the USSR*, 38(2):609–615, August 1985.

23 Giulio Vivanti. Sulle serie di potenze. *Annali di Matematica Pura ed Applicata (1867-1897)*, 21(1):193–194, 1893.

Coverability in VASS Revisited: Improving Rackoff's Bound to Obtain Conditional Optimality

Marvin Künnemann
RPTU Kaiserslautern-Landau, Germany

Filip Mazowiecki
University of Warsaw, Poland

Lia Schütze
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Henry Sinclair-Banks
Centre for Discrete Mathematics and its Applications (DIMAP) & Department of Computer Science, University of Warwick, Coventry, UK

Karol Węgrzycki
Saarland University and Max Planck Institute for Informatics, Saarbrücken, Germany

Abstract

Seminal results establish that the coverability problem for Vector Addition Systems with States (VASS) is in EXPSPACE (Rackoff, '78) and is EXPSPACE-hard already under unary encodings (Lipton, '76). More precisely, Rosier and Yen later utilise Rackoff's bounding technique to show that if coverability holds then there is a run of length at most $n^{2^{\mathcal{O}(d \log d)}}$, where d is the dimension and n is the size of the given unary VASS. Earlier, Lipton showed that there exist instances of coverability in d-dimensional unary VASS that are only witnessed by runs of length at least $n^{2^{\Omega(d)}}$. Our first result closes this gap. We improve the upper bound by removing the twice-exponentiated $\log(d)$ factor, thus matching Lipton's lower bound. This closes the corresponding gap for the exact space required to decide coverability. This also yields a deterministic $n^{2^{\mathcal{O}(d)}}$-time algorithm for coverability. Our second result is a matching lower bound, that there does not exist a deterministic $n^{2^{o(d)}}$-time algorithm, conditioned upon the Exponential Time Hypothesis.

When analysing coverability, a standard proof technique is to consider VASS with bounded counters. Bounded VASS make for an interesting and popular model due to strong connections with timed automata. Withal, we study a natural setting where the counter bound is linear in the size of the VASS. Here the trivial exhaustive search algorithm runs in $\mathcal{O}(n^{d+1})$-time. We give evidence to this being near-optimal. We prove that in dimension one this trivial algorithm is conditionally optimal, by showing that $n^{2-o(1)}$-time is required under the k-cycle hypothesis. In general fixed dimension d, we show that $n^{d-2-o(1)}$-time is required under the 3-uniform hyperclique hypothesis.

2012 ACM Subject Classification Theory of computation → Models of computation

Keywords and phrases Vector Addition System, Coverability, Reachability, Fine-Grained Complexity, Exponential Time Hypothesis, k-Cycle Hypothesis, Hyperclique Hypothesis

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.131

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: https://arxiv.org/abs/2305.01581 [31]

Funding *Marvin Künnemann*: Research partially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 462679611.
Filip Mazowiecki: Supported by the ERC grant INFSYS, agreement no. 950398.
Henry Sinclair-Banks: Supported by EPSRC Standard Research Studentship (DTP), grant number EP/T5179X/1.
Karol Węgrzycki: Supported by the ERC grant TIPEA, agreement no. 850979.

Acknowledgements We would like to thank our anonymous reviewers for their comments, for their time, and especially for highlighting a piece of related work [30].

1 Introduction

Vector Addition Systems with States (VASS) are a popular model of concurrency with a number of applications in database theory [9], business processes [49], and more (see the survey [47]). A d-dimensional VASS (d-VASS) consists of a finite automaton equipped with d non-negative valued counters that can be updated by transitions. A configuration in a d-VASS consists of a state and a d-dimensional vector over the naturals. One of the central decision problems for VASS is the *coverability problem*, that asks whether there is a run from a given initial configuration to some configuration with at least the counter values of a given target configuration. Coverability finds application in the verification of safety conditions, which often equate to whether or not a particular state can be reached without any precise counter values [13, 24]. Roughly speaking, one can use VASS as a modest model for concurrent systems where the dimension corresponds with the number of locations a process can be in and each counter value corresponds with the number of processes in a particular location [21, 25].

In 1978, Rackoff [45] showed that coverability is in EXPSPACE, by proving that if coverability holds then there exists a run of double-exponential length. Following, Rosier and Yen [46] analysed and discussed Rackoff's ideas in more detail and argued that if a coverability holds then it is witnessed by a run of length at most $n^{2^{\mathcal{O}(d \log d)}}$, where n is the size of the given unary encoded d-VASS. Furthermore, this yields a $2^{\mathcal{O}(d \log d)} \cdot \log(n)$-space algorithm for coverability. Prior to this in 1976, Lipton [37] proved that coverability is EXPSPACE-hard even when VASS is encoded in unary, by constructing an instance of coverability witnessed only by a run of double-exponential length $n^{2^{\Omega(d)}}$. Rosier and Yen [46] also presented a proof that generalises Lipton's constructions to show that $2^{\Omega(d)} \cdot \log(n)$-space is required for coverability. Although this problem is EXPSPACE-complete in terms of classical complexity, a gap was left open for the exact space needed for coverability [46, Section 1]. By using an approach akin to Rackoff's argument, we close this thirty-eight-year-old gap by improving the upper bound to match Lipton's lower bound.

Result 1: If coverability holds then there exists a run of length at most $n^{2^{\mathcal{O}(d)}}$ (Theorem 3.3). Accordingly, we obtain an optimal $2^{\mathcal{O}(d)} \cdot \log(n)$-space algorithm that decides coverability (Corollary 3.4).

Our bound also implies the existence of a deterministic $n^{2^{\mathcal{O}(d)}}$-time algorithm for coverability. We complement this with a matching lower bound on the deterministic running time that is conditioned upon the Exponential Time Hypothesis (ETH).

Result 2: Under ETH, there is no deterministic $n^{2^{o(d)}}$-time algorithm deciding coverability in unary d-VASS (Theorem 4.2).

While our results establish a fast-increasing, conditionally optimal exponent of $2^{\Theta(d)}$ in the time complexity of the coverability problem, they rely on careful constructions that enforce the observation of large counter values. In certain settings, however, it is natural to instead consider a restricted version of coverability, where all counter values remain *bounded*. This yields one of the simplest models, fixed-dimension bounded unary VASS, for which we obtain even tighter results. Decision problems for B-bounded VASS, where B forms part of the input, have been studied due to their strong connections to timed automata [27, 22, 41].

We consider linearly-bounded unary VASS, that is when the maximum counter value is bounded above by a constant multiple of the size of the VASS. Interestingly, coverability and reachability are equivalent in linearly-bounded unary VASS. The trivial algorithm that employs depth-first search on the space of configurations runs in $\mathcal{O}(n^{d+1})$-time for both coverability and reachability. We provide evidence that the trivial algorithm is optimal.

Result 3: Reachability in linearly-bounded unary 1-VASS requires $n^{2-o(1)}$-time, subject to the k-cycle hypothesis (Theorem 5.4).

This effectively demonstrates that the trivial algorithm is optimal in the one-dimensional case. For the case of large dimensions, we show that the trivial algorithm only differs from an optimal deterministic-time algorithm by at most an $n^{3+o(1)}$-time factor.

Result 4: Reachability in linearly-bounded unary d-VASS requires $n^{d-2-o(1)}$-time, subject to the 3-uniform k-hyperclique hypothesis (Theorem 5.8).

Broadly speaking, these results add a time complexity perspective to the already known space complexity, that is for any fixed dimension d, coverability in unary d-VASS is NL-complete [45].

Organisation and Overview. Section 3 contains our first main result, the improved upper bound on the space required for coverability. Most notably, in Theorem 3.3 we show that if coverability holds then there exists a run of length at most $n^{2^{\mathcal{O}(d)}}$. Then, in Corollary 3.4 we are able to obtain a non-deterministic $2^{\mathcal{O}(d)} \cdot \log(n)$-space algorithm and a deterministic $n^{2^{\mathcal{O}(d)}}$-time algorithm for coverability. In much of the same way as Rackoff, we proceed by induction on the dimension. The difference is in the inductive step; Rackoff's inductive hypothesis dealt with a case where all counters are bounded by the same well-chosen value. Intuitively speaking, the configurations are bounded within a d-hypercube. This turns out to be suboptimal. This is due to the fact that the volume of a d-hypercube with sides of length ℓ is ℓ^d; unrolling the induction steps gives a bound of roughly $n^{d\cdot(d-1)\cdot\ldots\cdot 1} = n^{d!} = n^{2^{\mathcal{O}(d \log d)}}$, hence the twice-exponentiated $\log(d)$ factor. The key ingredient in our proof is to replace the d-hypercubes with a collection of objects with greatly reduced volume, thus reducing the number of configurations in a run witnessing coverability.

Section 4 contains our second main result, the matching lower bound on the time required for coverability that is conditioned upon ETH. In Lemma 4.3, we first reduce from finding a k-clique in a graph to an instance of coverability in bounded unary 2-VASS with zero-tests. Then, via Lemma 4.4, we implement the aforementioned technique of Rosier and Yen to, when there is a counter bound, remove the zero-tests at the cost of increasing to a d-dimensional unary VASS. Then, in Theorem 4.2 we are able to conclude, by setting $k = 2^d$, that if ETH holds, then there is no deterministic $n^{2^{o(d)}}$-time algorithms for coverability in unary d-VASS. This is because ETH implies that there is no $f(k) \cdot n^{o(k)}$-time algorithm for finding a k-clique in a graph with n vertices (Theorem 4.1).

Section 5 contains our other results where we study bounded fixed dimension unary VASS. Firstly, Theorem 5.4 states that under the k-cycle hypothesis (Hypothesis 5.2), there does not exist a deterministic $n^{2-o(1)}$-time algorithm deciding reachability in linearly-bounded unary 1-VASS. Further, we conclude in Corollary 5.5, if the k-cycle hypothesis is assumed then there does not exist a deterministic $n^{2-o(1)}$-time algorithm for coverability in (not bounded) unary 2-VASS. Following, we prove Theorem 5.8, that claims there does not exist a deterministic $n^{d-o(1)}$-time algorithm reachability in linearly-bounded unary $(d+2)$-VASS under the 3-uniform k-hyperclique hypothesis (Hypothesis 5.7). We achieve this with two

components. First, in Lemma 5.9, we first reduce from finding a $4d$-hyperclique to an instance of reachability in a bounded unary $(d+1)$-VASS with a fixed number of zero-tests. Second, via Lemma 5.10, we implement the newly developed controlling counter technique of Czerwiński and Orlikowski [16] to remove the fixed number of zero-tests at the cost of increasing the dimension by one.

Related Work. The coverability problem for VASS has plenty of structure that still receives active attention. The set of configurations from which the target can be covered is upwards-closed, meaning that coverability still holds if the initial counter values are increased. An alternative approach, the *backwards algorithm* for coverability, relies on this phenomenon. Starting from the target configuration, one computes the set of configurations from which it can be covered [1]. Thanks to the upwards-closed property, it suffices to maintain the collection of minimal configurations. The backwards algorithm terminates due to Dickson's lemma, however, using Rackoff's bound one can show it runs in double-exponential time [10]. This technique has been deeply analysed for coverability in VASS and some extensions [23, 32]. Despite high complexity, there are many implementations of coverability relying on the backwards algorithm that work well in practice. Intuitively, the idea is to prune the set of configurations, using relaxations that can be efficiently implemented in SMT solvers [21, 7, 8].

Another central decision problem for VASS is the *reachability problem*, asking whether there is a run from a given initial configuration to a given target configuration. Reachability is a provably harder problem. In essence, reachability differs from coverability by allowing one zero-test to each counter. Counter machines, well-known to be equivalent to Turing machines [43], can be seen as VASS with the ability to arbitrarily zero-test counters; coverability and reachability are equivalent here and are undecidable. In 1981, Mayr proved that reachability in VASS is decidable [39], making VASS one of the richest decidable variants of counter machines. Only recently, after decades of work, has the complexity of reachability in VASS been determined to be Ackermann-complete [35, 16, 34]. A widespread technique for obtaining lower bounds for coverability and reachability problems in VASS is to simulate counter machines with some restrictions. Our overall approach to obtaining lower bounds follows suit; we first reduce finding cliques in graphs, finding cycles in graphs, and finding hypercliques in hypergraphs to various intermediate instances of coverability in VASS with extra properties such as bounded counters or a fixed number of zero-tests. These VASS, that are counter machines restricted in some way, are then simulated by standard higher-dimensional VASS. Such simulations are brought about by the two previously developed techniques. Rosier and Yen leverage Lipton's construction to obtain VASS that can simulate counter machines with bounded counters [46]. Czerwiński and Orlikowski have shown that the presence of an additional counter in a VASS, with carefully chosen transition effects and reachability condition, can be used to implicitly perform a limited number of zero-tests [16].

Recently, some work has been dedicated to the coverability problem for low-dimensional VASS [3, 42]. Furthermore, reachability in low-dimensional VASS has been given plenty of attention, in particular for 1-VASS [48, 26] and for 2-VASS [28, 6]. In the restricted class of flat VASS, other fixed dimensions have also been studied [15, 17].

Another studied variant, *bidirected* VASS, has the property that for every transition $(p, \mathbf{x}, q)$, the reverse transition $(q, -\mathbf{x}, p)$ is also present. The reachability problem in bidirected VASS is equivalent to the uniform word problem in commutative semigroups, both of which are EXPSPACE-complete [40]; not to be confused with the reversible reachability problem in general VASS which is also EXPSPACE-complete [33]. In 1982, Meyer and Mayr listed an open problem that stated, in terms of commutative semigroups, the best known upper

bound for coverability in general VASS [45], the best known lower bound for coverability in bidirected VASS [37], and asked for improvements to these bounds [40, Section 8, Problem 3]. Subsequently, Rosier and Yen refined the upper bound for coverability in general VASS to $2^{\mathcal{O}(d \log d)} \cdot \log(n)$-space [46]. Finally, Koppenhagen and Mayr showed that the coverability problem in bidirected VASS can be decided in $2^{\mathcal{O}(n)}$-space [30], matching the lower bound.

2 Preliminaries

We use bold font for vectors. We index the i-th component of a vector $\mathbf{v}$ by writing $\mathbf{v}[i]$. Given two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{Z}^d$ we write $\mathbf{u} \leq \mathbf{v}$ if $\mathbf{u}[i] \leq \mathbf{v}[i]$ for each $1 \leq i \leq d$. For every $1 \leq i \leq d$, we write $\mathbf{e}_i \in \mathbb{Z}^d$ to represent the i-th standard basis vector that has $\mathbf{e}_i[i] = 1$ and $\mathbf{e}_i[j] = 0$ for all $j \neq i$. Given a vector $\mathbf{v} \in \mathbb{Z}^d$ we define $\|\mathbf{v}\| = \max\{1, |\mathbf{v}[1]|, \ldots, |\mathbf{v}[d]|\}$. Throughout, we assume that log has base 2. We use $\mathsf{poly}(n)$ to denote $n^{\mathcal{O}(1)}$.

A *d-dimensional Vector Addition System with States* (d-VASS) $\mathcal{V} = (Q, T)$ consists of a non-empty finite set of states Q and a non-empty set of transitions $T \subseteq Q \times \mathbb{Z}^d \times Q$. A *configuration* of a d-VASS is a pair $(q, \mathbf{v}) \in Q \times \mathbb{N}^d$ consisting of the current state q and current counter values $\mathbf{v}$, denoted $q(\mathbf{v})$. Given two configurations $p(\mathbf{u})$, $q(\mathbf{v})$, we write $p(\mathbf{u}) \to q(\mathbf{v})$ if there exists $t = (p, \mathbf{x}, q) \in T$ where $\mathbf{x} = \mathbf{v} - \mathbf{u}$. We may refer to $\mathbf{x}$ as the *update* of a transition and may also write $p(\mathbf{v}) \xrightarrow{t} q(\mathbf{w})$ to emphasise the transition t taken.

A *path* in a VASS is a (possibly empty) sequence of transitions $((p_1, \mathbf{x}_1, q_1), \ldots, (p_\ell, \mathbf{x}_\ell, q_\ell))$, where $(p_i, \mathbf{x}_i, q_i) \in T$ for all $1 \leq i \leq \ell$ and such that the start and end states of consecutive transitions match $q_i = p_{i+1}$ for all $1 \leq i \leq \ell - 1$. A *run* π in a VASS is a sequence of configurations $\pi = (q_0(\mathbf{v}_0), \ldots, q_\ell(\mathbf{v}_\ell))$ such that $q_i(\mathbf{v}_i) \to q_{i+1}(\mathbf{v}_{i+1})$ for all $1 \leq i \leq \ell - 1$. We denote the length of the run by $\mathrm{len}(\pi) = \ell + 1$. If there is such a run π, we can write $q_0(\mathbf{v}_0) \xrightarrow{\pi} q_\ell(\mathbf{v}_\ell)$. We may also write $p(\mathbf{u}) \xrightarrow{*} q(\mathbf{v})$ if there exists a run from $p(\mathbf{u})$ to $q(\mathbf{v})$. The *underlying path* of a run π is sequence of transitions $(t_1, \ldots, t_\ell)$ taken between each of the configurations in π, so $q_i(\mathbf{v}_i) \xrightarrow{t_{i+1}} q_{i+1}(\mathbf{v}_{i+1})$ for all $0 \leq i \leq \ell - 1$.

A *B-bounded d-VASS*, in short (B, d)-VASS, is given as an integer upper bound on the counter values $B \in \mathbb{N}$ and d-VASS $\mathcal{V}$. A configuration in a (B, d)-VASS is a pair $q(\mathbf{v}) \in Q \times \{0, \ldots, B\}^d$. The notions of paths and runs in bounded VASS remain the same as for VASS, but are accordingly adapted for the appropriate bounded configurations. We note that one should think that B forms part of the problem statement, not the input, as it will be given implicitly by a function depending on the size of the VASS. For example, we later consider *linearly-bounded* d-VASS, that represent occasions where $B = \mathcal{O}(\|\mathcal{V}\|)$.

We do allow for zero-dimensional VASS, that is VASS with no counters, which can be seen as just directed graphs. A *hypergraph* is a generalisation of the graph. Formally, a hypergraph is a tuple $H = (V, E)$ where V is a set of vertices and E is a collection of non-empty subsets of V called *hyperedges*. For an integer μ, a hypergraph is *μ-uniform* if each hyperedge has cardinality μ. Note that a 2-uniform hypergraph is a standard graph.

We study the complexity of the *coverability problem*. An instance $(\mathcal{V}, p(\mathbf{u}), q(\mathbf{v}))$ of coverability asks whether there is a run in the given VASS $\mathcal{V}$ from the given initial configuration $p(\mathbf{u})$ to a configuration $q(\mathbf{v}')$ with at least the counter values $\mathbf{v}' \geq \mathbf{v}$ of the given target configuration $q(\mathbf{v})$. At times, we also consider the *reachability problem* that additionally requires $\mathbf{v}' = \mathbf{v}$ so that the target configuration is reached exactly.

To measure the complexity of these problems we need to discuss the encoding used. In *unary encoding*, a d-VASS $\mathcal{V} = (Q, T)$ has *size* $\|\mathcal{V}\| = |Q| + \sum_{(p,\mathbf{x},q) \in T} \|\mathbf{x}\|$. We define a *unary* d-VASS $\mathcal{U} = (Q', T')$ to have restricted transitions $T' \subseteq Q' \times \{-1, 0, 1\}^d \times Q'$, the size is therefore $\|\mathcal{U}\| = |Q'| + |T'|$. For any unary encoded d-VASS $\mathcal{V}$ there exists an equivalent

unary d-VASS $\mathcal{U}$ such that $\|\mathcal{U}\| = \|\mathcal{V}\|$. An instance $(\mathcal{V}, p(\mathbf{u}), q(\mathbf{v}))$ of coverability has size $n = \|\mathcal{V}\| + \|\mathbf{u}\| + \|\mathbf{v}\|$. An equal in size, equivalent instance $(\mathcal{V}', p'(\mathbf{0}), q'(\mathbf{0}))$ of coverability exists; consider adding an initial transition $(p', \mathbf{s}, p)$ and a final transition $(q, -\mathbf{t}, q')$.

It is well known that for d-VASS, the coverability problem can be reduced to the reachability problem. Indeed, for an instance $(\mathcal{V}, p(\mathbf{u}), q(\mathbf{v}))$ of coverability, define $\mathcal{V}' = (Q, T')$ that has additional decremental transitions at the target states $T' = T \cup \{(q, \mathbf{e}_i, q) : 1 \leq i \leq d\}$. It is clear that $p(\mathbf{u}) \xrightarrow{*} q(\mathbf{v}')$, for some $\mathbf{v}' \geq \mathbf{v}$, in $\mathcal{V}$ if and only if $p(\mathbf{u}) \xrightarrow{*} q(\mathbf{v})$ in $\mathcal{V}'$.

▶ **Lemma 2.1** (folklore). *Let $(\mathcal{V}, p(\mathbf{u}), q(\mathbf{v}))$ be an instance of coverability. It can be reduced to an instance of reachability $(\mathcal{V}', p(\mathbf{u}), q(\mathbf{v}))$ such that $\|\mathcal{V}'\| = \mathcal{O}(\|\mathcal{V}\|)$.*

A *d-dimensional Vector Addition System* (d-VAS) $\mathcal{V}$ is a system without states, consisting only of a non-empty collection of transitions $\mathcal{V} \subseteq \mathbb{Z}^d$. All definitions, notations, and problems carry over for VAS except that, for simplicity, we drop the states across the board. For example, a configuration in a VAS is just a vector $\mathbf{v} \in \mathbb{N}^d$. Another well-known result from the seventies by Hopcroft and Pansiot, one can simulate the states of a VASS at the cost of three extra dimensions in a VAS [28]. For clarity, the VAS obtained has an equivalent reachability relation between configurations; a configuration $q(\mathbf{x})$ in the original VASS corresponds with a configuration $(\mathbf{x}, a, b, c)$ in the VAS, where a, b, and c represent the state q.

▶ **Lemma 2.2** ([28, Lemma 2.1]). *A d-VASS $\mathcal{V}$ can be simulated by $(d+3)$-VAS $\mathcal{V}'$ such that $\|\mathcal{V}'\| = \mathit{poly}(\|\mathcal{V}\|)$.*

3 Improved Bounds on the Maximum Counter Value

This section is devoted to our improvement of the seminal result of Rackoff. Throughout, we fix our attention to the arbitrary instance $(\mathcal{V}, p(\mathbf{s}), q(\mathbf{t}))$ of the coverability problem in a d-VASS $\mathcal{V} = (Q, T)$ from the initial configuration $p(\mathbf{s})$ to a configuration $q(\mathbf{t}')$ with at least the counter values of the target configuration $q(\mathbf{t})$. We denote $n = \|\mathcal{V}\| + \|\mathbf{s}\| + \|\mathbf{t}\|$. Informally, n may as well be the number of states plus the absolute value of the greatest update on any transition, for these differences can be subsumed by the second exponent in our following upper bounds. The following two theorems follow from Rackoff's technique and subsequent work by Rosier and Yen, in particular see [45, Lemma 3.4 and Theorem 3.5] and [46, Theorem 2.1 and Lemma 2.2].

▶ **Theorem 3.1** (Corollary of [45, Lemma 3.4] and [46, Theorem 2.1]). *Suppose $p(\mathbf{s}) \xrightarrow{*} q(\mathbf{t}')$ for some $\mathbf{t}' \geq \mathbf{t}$. Then there exists a run π such that $p(\mathbf{s}) \xrightarrow{\pi} q(\mathbf{t}'')$ for some $\mathbf{t}'' \geq \mathbf{t}$ and $\mathrm{len}(\pi) \leq n^{2^{\mathcal{O}(d \log d)}}$.*

▶ **Theorem 3.2** (cf. [45, Theorem 3.5]). *For a given d-VASS $\mathcal{V}$, integer ℓ, and two configurations $p(\mathbf{s})$ and $q(\mathbf{t})$, there is an algorithm that determines the existence of a run π of length $\mathrm{len}(\pi) \leq \ell$ that witnesses coverability, so $p(\mathbf{s}) \xrightarrow{\pi} q(\mathbf{t}')$ for some $\mathbf{t}' \geq \mathbf{t}$. The algorithm can be implemented to run in non-deterministic $\mathcal{O}(d \log(n \cdot \ell))$-space or deterministic $2^{\mathcal{O}(d \log(n \cdot \ell))}$-time.*

Note that Theorem 3.1 combined with Theorem 3.2, that is proved in the full version [31], yield non-deterministic $2^{\mathcal{O}(d \log d)}$-space and deterministic $n^{2^{\mathcal{O}(d \log(d))}}$-time algorithms for coverability. Our result improves this by a $\mathcal{O}(\log(d))$ factor in the second exponent.

▶ **Theorem 3.3.** *Suppose $p(\mathbf{s}) \xrightarrow{*} q(\mathbf{t}')$ for some $\mathbf{t}' \geq \mathbf{t}$. Then there exists a run π such that $p(\mathbf{s}) \xrightarrow{\pi} q(\mathbf{t}'')$ for some $\mathbf{t}'' \geq \mathbf{t}$ and $\mathrm{len}(\pi) \leq n^{2^{\mathcal{O}(d)}}$.*

This combined with Theorem 3.2 yields the following corollary.

▶ **Corollary 3.4.** *Coverability in d-VASS can be decided by both a non-deterministic $2^{\mathcal{O}(d)} \cdot \log(n)$-space algorithm and a deterministic $n^{2^{\mathcal{O}(d)}}$-time algorithm.*

Note that by Lemma 2.2, we may handle VAS instead of VASS. Recall that, as there are no states, a d-VAS consists only of a set of vectors in $\mathbb{Z}^d$ that we still refer to as transitions. A configuration is just a vector in $\mathbb{N}^d$. Accordingly, we may fix our attention on the instance $(\mathcal{V}, \mathbf{s}, \mathbf{t})$ of the coverability problem in a d-VAS $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ from the initial configuration $\mathbf{s}$ to a configuration $\mathbf{t}'$ that is at least as great as the target configuration $\mathbf{t}$. The rest of this section is dedicated to the proof of Theorem 3.3. Imitating Rackoff's proof, we proceed by induction on the dimension d. Formally, we prove a stronger statement; Theorem 3.3 is a direct corollary of the following lemma.

▶ **Lemma 3.5.** *Define $L_i \coloneqq n^{4^i}$, and let $\mathbf{t} \in \mathbb{N}^d$ such that $\|\mathbf{t}\| \leq n$. For any $\mathbf{s} \in \mathbb{N}^d$, if $\mathbf{s} \xrightarrow{*} \mathbf{t}'$ for some $\mathbf{t}' \geq \mathbf{t}$ then there exists a run π such that $\mathbf{s} \xrightarrow{\pi} \mathbf{t}''$ for some $\mathbf{t}'' \geq \mathbf{t}$ and $\mathrm{len}(\pi) \leq L_d$.*

The base case is $d = 0$. In a 0-dimensional VAS, the only possible configuration is the empty vector $\boldsymbol{\varepsilon}$ and therefore there is only the trivial run $\boldsymbol{\varepsilon} \xrightarrow{*} \boldsymbol{\varepsilon}$. This trivially satisfies the lemma.

For the inductive step, when $d \geq 1$, we assume that Lemma 3.5 holds for all lower dimensions $0, \ldots, d-1$. Let $\pi = (\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_\ell)$ be a run with minimal length such that $\mathbf{s} \xrightarrow{\pi} \mathbf{t}'$ for some $\mathbf{t}' \geq \mathbf{t}$, so in particular, $\mathbf{c}_0 = \mathbf{s}$ and $\mathbf{c}_n = \mathbf{t}'$. Our objective is to prove that $\mathrm{len}(\pi) = \ell + 1 \leq L_d$. Observe that configurations $\mathbf{c}_i$ need to be distinct, else π could be shortened trivially. We introduce the notion of a *thin configuration*.

▶ **Definition 3.6** (Thin Configuration). *In a d-VAS, we say that a configuration $\mathbf{c} \in \mathbb{N}^d$ is* thin *if there exists a permutation σ of $\{1, \ldots, d\}$ such that $\mathbf{c}[\sigma(i)] < M_i$ for every $i \in \{1, \ldots, d\}$, where $M_0 \coloneqq n$ and for $i \geq 1$, $M_i \coloneqq L_{i-1} \cdot n$.*

Recall, from above, the run $\pi = (\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_\ell)$. Let $t \in \{0, \ldots, \ell\}$ be the first index where $\mathbf{c}_t$ is not thin, otherwise let $t = \ell + 1$ if every configuration in π is thin. We decompose the run about the t-th configuration $\pi = \pi_{\text{thin}} \cdot \pi_{\text{tail}}$, where $\pi_{\text{thin}} \coloneqq (\mathbf{c}_0, \ldots, \mathbf{c}_{t-1})$ and $\pi_{\text{tail}} \coloneqq (\mathbf{c}_t, \ldots, \mathbf{c}_\ell)$. Note that π_{thin} or π_{tail} can be empty. Subsequently, we individually analyse the lengths of π_{thin} and π_{tail} (see Figure 1). We will also denote $\mathbf{m} = \mathbf{c}_t$ to be the first configuration that is not thin.

▷ Claim 3.7. $\mathrm{len}(\pi_{\text{thin}}) \leq d! \cdot n^d \cdot L_{d-1} \cdot \ldots \cdot L_0$.

Proof. By definition, every configuration in π_{thin} is thin. Moreover, since π has a minimal length, no configurations in π repeat, let alone in π_{thin}. We now count the number of possible thin configurations. There are $d!$ many permutations of $\{1, \ldots, d\}$. For a given permutation σ and an index $i \in \{1, \ldots, d\}$, we know that for a thin configuration $\mathbf{c}$, $0 \leq \mathbf{c}[\sigma(i)] < M_i$, so there are at most $M_i = L_{i-1} \cdot n$ many possible values on the $\sigma(i)$-th counter. Hence the total number of thin configurations is at most $d! \cdot \prod_{i=1}^{d} (L_{i-1} \cdot n) = d! \cdot n^d \cdot L_{d-1} \cdot \ldots \cdot L_0$. ◁

▷ Claim 3.8. $\mathrm{len}(\pi_{\text{tail}}) \leq L_{d-1}$.

Proof. Consider $\mathbf{m} \in \mathbb{N}^d$, the first configuration of π_{tail}. Let σ be a permutation such that $\mathbf{m}[\sigma(1)] \leq \mathbf{m}[\sigma(2)] \leq \ldots \leq \mathbf{m}[\sigma(d)]$. Given that $\mathbf{m}$ is not thin, for every permutation σ' there exists an $i \in \{1, \ldots, d\}$ such that $\mathbf{m}[\sigma'(i)] \geq M_i$; in particular, this holds for σ. Note that this also implies $M_i \leq \mathbf{m}[\sigma(i+1)] \leq \ldots \leq \mathbf{m}[\sigma(d)]$.

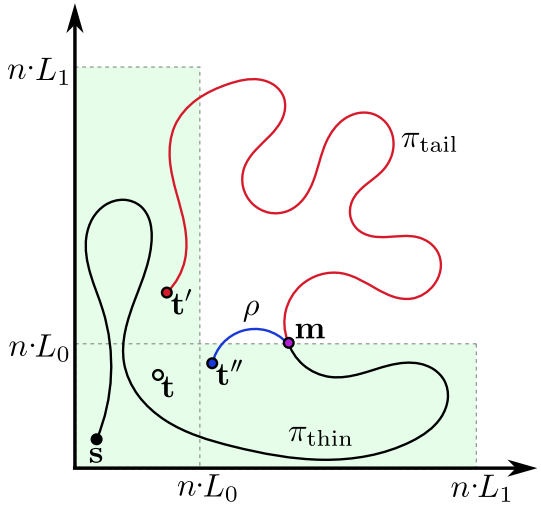


Figure 1 The schematic view of proofs of Claim 3.7 and Claim 3.8, restricted to the two-dimensional case. Note that $\mathbf{s}$ is the initial configuration and $\mathbf{t}$ is the target configuration. Every configuration inside the green shaded polygon is thin, where each rectangular component of the green shaded polygon corresponds to a permutation of the indices. Observe that $\mathbf{m}$ is the first configuration, just outside the green shaded polygon, that is not thin. Claim 3.7 bounds π_{thin}, and therefore its maximum length, by the volume of the green polygon. Claim 3.8 argues that there is an executable run ρ (drawn in blue) from $\mathbf{m}$ to $\mathbf{t}'' \geq \mathbf{t}$ of length at most L_{d-1} that can be used in place of the run π_{tail} (drawn in red) from $\mathbf{m}$ to $\mathbf{t}' \geq \mathbf{t}$.

We construct an $(i-1)$-VAS $\mathcal{U}$ from $\mathcal{V}$ by ignoring the counters $\sigma(i), \ldots, \sigma(d)$. Formally, $\mathbf{u} \in \mathcal{U}$ if there is $\mathbf{v} \in \mathcal{V}$ such that $\mathbf{u}[j] = \mathbf{v}[\sigma(j)]$ for each $1 \leq j \leq i-1$. In such a case we say $\mathbf{u}$ is the *projection* of $\mathbf{v}$ via σ. We will use the inductive hypothesis to show that there is a short path ρ' in $\mathcal{U}$ from (the projection of) $\mathbf{m}$ covering (the projection of) $\mathbf{t}$. We will then show that the remaining components of $\mathbf{m}$ are large enough that the embedding of ρ' into $\mathcal{V}$ maintains its covering status.

Recall that $\mathbf{t}'$ is the final configuration of the run π. Note that the run π_{tail} induces a run π'_{tail} in $\mathcal{U}$ by permuting and projecting every configuration. More precisely, $(\mathbf{m}[\sigma(1)], \ldots, \mathbf{m}[\sigma(i-1)]) \xrightarrow{\pi'_{\text{tail}}} (\mathbf{t}'[\sigma(1)], \ldots, \mathbf{t}'[\sigma(i-1)])$. By the inductive hypothesis there exists a run ρ' in $\mathcal{U}$ such that $(\mathbf{m}[\sigma(1)], \ldots, \mathbf{m}[\sigma(i-1)]) \xrightarrow{\rho'} (\mathbf{t}''[\sigma(1)], \ldots, \mathbf{t}''[\sigma(i-1)])$, such that $(\mathbf{t}''[\sigma(1)], \ldots, \mathbf{t}''[\sigma(i-1)]) \geq (\mathbf{t}[\sigma(1)], \ldots, \mathbf{t}[\sigma(i-1)])$ and $\text{len}(\rho') \leq L_{i-1}$.

Let $(\mathbf{u}_1, \ldots, \mathbf{u}_{\text{len}(\rho')})$ be the underlying path of the run ρ', that is, the sequence of transitions in $\mathcal{U}$ that are sequentially added to form the run ρ'. By construction, each transition vector $\mathbf{u}_i \in \mathcal{U}$ has a corresponding transition vector $\mathbf{v}_i \in \mathcal{V}$ where $\mathbf{u}_i$ is the projection of $\mathbf{v}_i$ via σ. We will now show that the following run witnesses coverability of $\mathbf{t}$.

$$\rho = \left(\mathbf{m},\ \mathbf{m} + \mathbf{v}_1,\ \mathbf{m} + \mathbf{v}_1 + \mathbf{v}_2,\ \ldots,\ \mathbf{m} + \sum_{j=1}^{\text{len}(\rho')} \mathbf{v}_j \right)$$

To this end, we verify that (i) ρ is a run, that is, all configurations lie in $\mathbb{N}^d$, and (ii) the final configuration indeed covers $\mathbf{t}$. For components $\sigma(1), \ldots, \sigma(i-1)$, this follows directly from the inductive hypothesis. For all other components we will show that *all* configurations of ρ are covering $\mathbf{t}$ in these components. This satisfies both (i) and (ii).

Let j be any of the remaining components. Recall that by the choice of $\mathbf{m}$, $\mathbf{m}[j] \geq M_i = n \cdot L_{i-1}$. Since $n > \|\mathcal{V}\| \geq \|\mathbf{v}_j\|$ for every $1 \leq j \leq \mathrm{len}(\rho')$, this means that in a single step, the value of a counter can change by at most n. Given that $\mathrm{len}(\rho) = \mathrm{len}(\rho') \leq L_{i-1}$, the value on each of the remaining components must be at least n for every configuration in ρ. In particular, observing that $\|\mathbf{t}\| \leq n$, the final configuration of ρ satisfies

$$\mathbf{m} + \sum_{j=1}^{\mathrm{len}(\rho')} \mathbf{v}_j \geq \mathbf{t}.$$

Finally, observe that $\mathrm{len}(\rho) = \mathrm{len}(\rho') \leq L_{i-1} \leq L_{d-1}$. ◁

To conclude this section, we show that Lemma 3.5 follows from Claim 3.7 and Claim 3.8.

Proof of Lemma 3.5. From Claim 3.7 and Claim 3.8,

$$\begin{aligned}\mathrm{len}(\pi) \leq \mathrm{len}(\pi_{\mathrm{thin}}) + \mathrm{len}(\pi_{\mathrm{tail}}) &\leq d! \cdot n^d \cdot L_{d-1} \cdot \ldots \cdot L_0 + L_{d-1} \\ &\leq 2 \cdot d! \cdot n^d \cdot L_{d-1} \cdot \ldots \cdot L_0.\end{aligned}$$

Recall that $n \geq 2$ and observe that $2 \cdot d! \cdot n^d \leq n^{2^d}$. Hence,

$$\mathrm{len}(\pi) \leq n^{2^d} \cdot L_{d-1} \cdot \ldots \cdot L_0.$$

Next, we use the definition of $L_i \coloneqq n^{4^i}$ to show

$$\mathrm{len}(\pi) \leq n^{2^d} \cdot \prod_{i=0}^{d-1} n^{4^i} \leq n^{\left(2^d + \sum_{i=0}^{d-1} 4^i\right)}.$$

Finally, when $d \geq 1$, $2^d + \sum_{i=0}^{d-1} 4^i \leq 4^d$ holds, therefore

$$\mathrm{len}(\pi) \leq n^{4^d} = L_d.$$ ◀

4 Conditional Time Lower Bound for Coverability

In this section, we present a conditional lower bound based on the *Exponential Time Hypothesis* (ETH) [29]. Roughly speaking, ETH is a conjecture that an n-variable instance of 3-SAT cannot be solved by a deterministic $2^{o(n)}$-time algorithm (for a modern survey, see [38]). In our reductions, it will be convenient for us to work with the k-clique problem instead. In the k-clique problem we are given a graph $G = (V, E)$ as an input and the task is to decide whether there is a set of k pairwise adjacent vertices in V. The naive algorithm for k-clique runs in $\mathcal{O}(n^k)$ time. Even though the exact constant in the dependence on k can be improved [44], ETH implies that the exponent must have a linear dependence on k.

▶ **Theorem 4.1** ([11, Theorem 4.2], [12, Theorem 4.5], and [14, Theorem 14.21]). *Assuming the Exponential Time Hypothesis, there is no algorithm running in $f(k) \cdot n^{o(k)}$-time for the k-clique problem for any computable function f. Moreover one can assume that G is k-partite, i.e. $G = (V_1 \cup \ldots \cup V_k, E)$ and edges belong to $V_i \times V_j$ for $i \neq j \in \{1, \ldots, k\}$.*

We will use Theorem 4.1 to show the following conditional lower bound for coverability in unary d-VASS, which is proved at the end of this section.

▶ **Theorem 4.2.** *Assuming the Exponential Time Hypothesis, there does not exist an $n^{2^{o(d)}}$-time algorithm deciding coverability in a unary d-VASS with n states.*

We first reduce the k-clique problem to coverability in bounded 2-VASS with the ability to perform a fixed number of zero-tests. We will then leverage a result by Rosier and Yen to construct an equivalent, with respect to coverability, $(\mathcal{O}(\log k))$-VASS without zero-tests.

▶ **Lemma 4.3.** *Given a k-partite graph $G = (V_1 \cup \cdots \cup V_k, E)$ with n vertices, there exists a unary $(\mathcal{O}(n^{2k}), 2)$-VASS $\mathcal{T}$ such that there is a k-clique in G if and only if there exists a run from $q_I(\mathbf{0})$ to $q_F(\mathbf{v})$ in $\mathcal{T}$, for some $\mathbf{v} \geq \mathbf{0}$. Moreover, $\|\mathcal{T}\| \leq \mathsf{poly}(n+k)$ and $\mathcal{T}$ can be constructed in $\mathsf{poly}(n+k)$-time.*

Proof. Without loss of generality, we may assume that each of the k vertex subsets in the graph has the same size $|V_1| = \ldots = |V_k| = \ell$. Thus $n = k \cdot \ell$. For convenience, we denote $V = \{1, \ldots, k\} \times \{1, \ldots, \ell\}$.

We begin by sketching the main ideas behind the reduction before they are implemented. We start by finding the first $n = k \cdot \ell$ primes and associating a distinct prime $p_{i,j}$ to each vertex $(i, j) \in V$. Note that a product of k different primes uniquely corresponds to selecting k vertices. Thus the idea is to guess such a product, and test whether the corresponding verticies form a k-clique. To simplify the presentation we present VASS also as counter programs, inspired by Esparza's presentation of Lipton's lower bound [20, Section 7].

We present an overview of our construction in Algorithm 1. Note that the counter y is used only by subprocedures. Initially both counter values are 0, as in the initial configuration of the coverability instance. The program is non-deterministic and we are interested in the existence of a certain run. One should think that coverability holds if and only if there is a run through the code without getting stuck so to say. In this example a run can be stuck only in the `Edge`$[e]$ subprocedure, that will be explained later. The precise final counter values are not important, as we are simply aiming to cover the target counter values $\mathbf{0}$. The variable i (in the first loop) and variables i and j (in the second loop) are just syntactic sugar for copying similar code multiple times. The variables j (in the first loop) and e (in the second loop) allow us to neatly represent non-determinism in a VASS.

Algorithm 1 A counter program for a VASS with zero tests with two counters x and y.

input : $\mathsf{x} = 0$, $\mathsf{y} = 0$

$\mathsf{x} \mathrel{+}= 1$
for $i \leftarrow 1$ **to** k **do**
  guess $j \in \{1, \ldots, \ell\}$
  `Multiply`$[\mathsf{x}, p_{i,j}]$
end
for $(i, j) \in \{1, \ldots, k\}^2$, $i \neq j$ **do**
  guess $e \in E \cap (V_i \times V_j)$
  `Edge`$[e]$
end

Algorithm 1 uses the `Multiply`$[\mathsf{x}, p]$ and `Edge`$[e]$ subprocedures. These two subprocedures will be implemented later. Note that `Multiply`$[\mathsf{x}, p]$ takes a counter x as input as we later reuse this subprocedure when there is more than one counter subject to multiplication. The intended behaviour of `Multiply`$[\mathsf{x}, p]$ is that it can be performed if and only if as a result we get $\mathsf{x} = \mathsf{x} \cdot p$, despite the fact that VASS can only additively increase and decrease counters. The subprocedure `Edge`$[e]$ can be performed if and only if both vertices of the edge e are encoded in the value of the counter x. Overall, Algorithm 1 is designed so that in the first part the variable x is multiplied by $p_{i,j}$, where for every i one j is guessed. This equates to selecting one vertex from each V_i. Then the second part the algorithm checks whether between every pair of selected vertices from V_i and V_j there is an edge. Clearly there is a run through the program that does not get stuck if and only if there is k-clique in G.

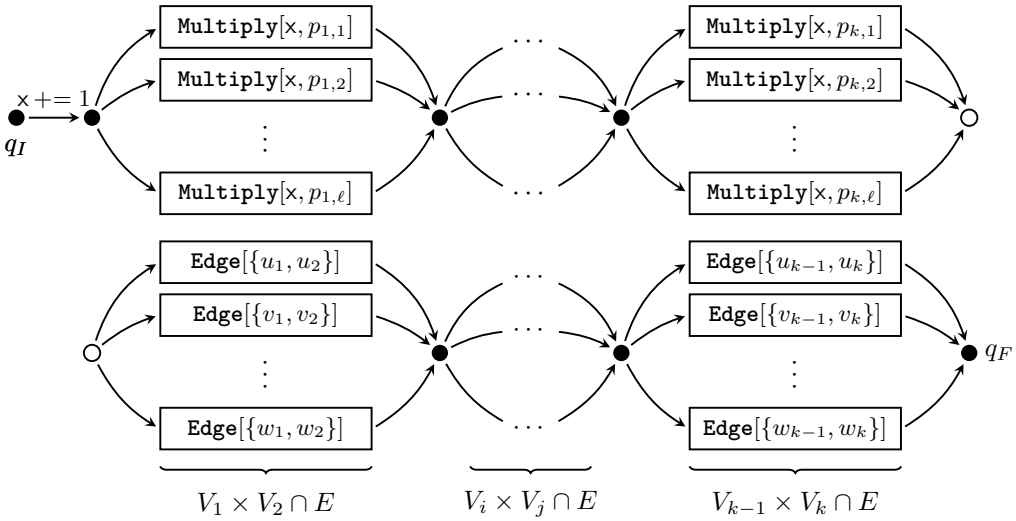


Figure 2 The top part of the VASS implements the first line and the first loop in Algorithm 1. The variable x is multiplied by k non-deterministically chosen primes $p_{i,j}$, each corresponding to a vertex in V_i. The bottom part of the VASS implements the second loop in Algorithm 1. For every pair $i \neq j$ the VASS non-deterministically chooses $e \in V_i \cap V_j$ and invokes the subprocedure **Edge**$[e]$.

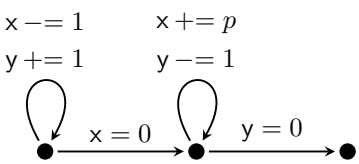
In Figure 2 we present a VASS with zero-tests implementing Algorithm 1. The construction will guarantee that $q_F(\mathbf{0})$ can be covered from $q_I(\mathbf{0})$ if and only if there is a k-clique in G.

It remains to define the subprocedures. One should think that every call of a subprocedure corresponds to a unique part of the VASS, like a gadget of sorts. To enter and leave the subprocedure one needs to add trivial transitions that to do not change the counter values. All subprocedures rely on the invariant $\mathsf{y} = 0$ at the beginning and admit the invariant at the end.

Algorithm 2 The counter program of **Multiply**$[\mathsf{x}, p]$ above its VASS implementation (left) and the counter program of **Divide**$[\mathsf{x}, p]$ above its VASS implementation (right).

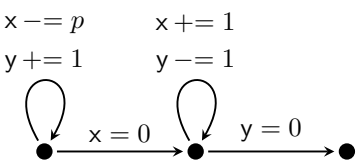
input : $\mathsf{x} = v$, $\mathsf{y} = 0$
output : $\mathsf{x} = v \cdot p$, $\mathsf{y} = 0$

repeat
  $\mathsf{x} -= 1$; $\mathsf{y} += 1$
until $\mathsf{x} = 0$
repeat
  $\mathsf{x} += p$; $\mathsf{y} -= 1$
until $\mathsf{y} = 0$



input : $\mathsf{x} = v \cdot p$, $\mathsf{y} = 0$
output : $\mathsf{x} = v$, $\mathsf{y} = 0$

repeat
  $\mathsf{x} -= p$; $\mathsf{y} += 1$
until $\mathsf{x} = 0$
repeat
  $\mathsf{x} += 1$; $\mathsf{y} -= 1$
until $\mathsf{y} = 0$

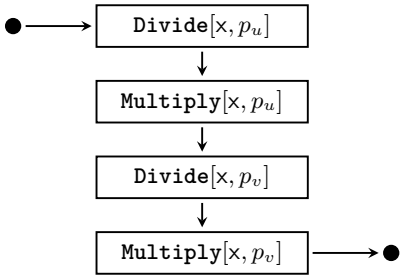


We start with **Multiply**[x, p] and **Divide**[x, p] that indeed multiply and divide x by p, respectively. See Algorithm 2 for the counter program and VASS implementations. Notice that the repeat loops correspond to the self-loops in the VASS. In the **Multiply**[x, p] gadget, it is easy to see that a run passes through the procedure if and only if the counter x is multiplied by p. Similarly, in the **Divide**[x, p] gadget, it is easy to see that a run pass through the procedure if and only if the counter x is divided by p wholly. Indeed, the division procedure would get stuck if $p \nmid$ x because it will be impossible to exit the first loop.

Algorithm 3 The counter program for **Edge**[$\{u, v\}$] and its VASS implementation.

input : x $= v$, y $= 0$
output : x $= v$, y $= 0$

Divide[x, p_u]
Multiply[x, p_u]
Divide[x, p_v]
Multiply[x, p_v]



The procedure **Edge**[$\{u, v\}$] is very simple, it is a sequence of four subprocedures, see Algorithm 3. Indeed, to check if the vertices from edge e are encoded in x we simply check whether x is divisible by the corresponding primes. Afterwards we multiply x with the same primes so that the value does not change and it is ready for future edge checks.

It remains to analyse the size of the VASS and its construction time in this reduction time. In every run from $q_I(\mathbf{0})$ to $q_F(\mathbf{v})$, for some $\mathbf{v} \geq \mathbf{0}$, the greatest counter value observable can be bounded above by p^k where p is the n-th prime. By the Prime Number Theorem (for example, see [51]), we know that $p^k \leq \mathcal{O}((n \log(n))^k) \leq \mathcal{O}(n^{2k})$ is an upper bound on the counter values observed. Hence $\mathcal{T}$ is an $\mathcal{O}(n^{2k})$-bounded unary 2-VASS.

Finally, the **Multiply** and **Divide** subprocedures contain three states and five transitions. Since the n-th prime is bounded above by $\mathcal{O}(n \log(n))$, we also get $\|\mathcal{T}\| = \mathcal{O}(n \log(n))$, hence our VASS can be represented using unary encoding. Analysing Algorithm 1, it is easy to see that overall the number of states is polynomial in n. Finally, the first n primes can be found in $\mathcal{O}(n^{1+o(1)})$-time [2]. Therefore, in total $\mathcal{T}$ has size $\|\mathcal{T}\| = \mathsf{poly}(n + k)$ and can be constructed in $\mathsf{poly}(n + k)$-time. ◀

To attain conditional lower bounds for coverability we must replace the zero-tests. We make use of a technique of Rosier and Yen [46] that relies on the construction of Lipton [37]. They show that a $(2n)^{2^k}$-bounded counter machine with finite state control can be simulated by a unary $(\mathcal{O}(k))$-VASS with n states. As Rosier and Yen detail after their proof, it is possible to apply this technique to multiple counters with zero-tests at once [46]. This accordingly results in the number of VASS counters increasing, but we instantiate this with just two counters. We remark that the VASS constructed in Lemma 4.3 is structurally bounded, so for any initial configuration there is a limit on the largest observable counter, as is the case in the VASS Lipton constructed [37].

▶ **Lemma 4.4** (Corollary of [46, Lemma 4.3]). *Let $\mathcal{T}$ be an n-state unary $(n^{\mathcal{O}(k)}, 2)$-VASS with zero-tests, for some parameter k. Then there exists an $\mathcal{O}(n)$-state $(\mathcal{O}(\log k))$-VASS $\mathcal{V}$, such that there is a run from $q_I(\mathbf{0})$ to $q_F(\mathbf{v})$, for some $\mathbf{v} \geq \mathbf{0}$, in $\mathcal{T}$ if and only if there is a run from $q'_I(\mathbf{0})$ to $q'_F(\mathbf{w})$, for some $\mathbf{w} \geq \mathbf{0}$, in $\mathcal{V}$. Moreover, $\mathcal{V}$ has size $\mathcal{O}(|\mathcal{T}|)$ and can be constructed in the same time.*

Proof of Theorem 4.2. Let $k = 2^d$. We instantiate Lemma 4.3 on k-partite graphs G with n vertices. We therefore obtain a unary $(n^{2^{\mathcal{O}(d)}}, 2)$-VASS with zero tests $\mathcal{T}$ such that G contains a k-clique if and only if there is a run from $q_I(\mathbf{0})$ to $q_F(\mathbf{v})$, for some $\mathbf{v} \geq \mathbf{0}$, in $\mathcal{T}$.

Given the bound on the value of the counters, we can apply Lemma 4.4 to $\mathcal{T}$. This gives us an $\mathcal{O}(n)$-state $(\mathcal{O}(d))$-VASS $\mathcal{V}$ such that G contains a k-clique if and only if there is a run from $q'_I(\mathbf{0})$ to $q'_F(\mathbf{w})$, for some $\mathbf{w} \geq \mathbf{0}$, in $\mathcal{V}$.

By Theorem 4.1 we conclude that under the Exponential Time Hypothesis there does not exist an $n^{2^{o(d)}}$-time algorithm deciding coverability in unary d-VASS. ◀

5 Coverability and Reachability in Bounded Unary VASS

In this section, we give even tighter bounds for coverability in *bounded* fixed dimension unary VASS. Specifically, for a time constructible function $B(n)$, the coverability problem in $(B(n), d)$-VASS asks, for a given $(B(n), d)$-VASS $\mathcal{V} = (Q, T)$ of size n as well as configurations $p(\mathbf{u})$, $q(\mathbf{v})$, whether there is a run in $\mathcal{V}$ from $p(\mathbf{u})$ to $q(\mathbf{v}')$ for some $\mathbf{v}' \geq \mathbf{v}$ such that each counter value remains in $\{0, \ldots, B(n)\}$ throughout. We would like to clarify the fact that the bound is not an input parameter. We focus on the natural setting of linearly-bounded fixed dimension VASS, that is $(\mathcal{O}(n), d)$-VASS. There is a simple algorithm, given in the proof of Observation 5.1 which can be found in the full version [31], that yields an immediate $\mathcal{O}(n^{d+1})$ upper bound for the time needed to decide the coverability problem. We accompany this observation with closely matching lower bounds, see Table 1 for an overview.

Table 1 Conditional lower bounds and upper bounds of the time complexity of coverability and reachability in unary $(\mathcal{O}(n), d)$-VASS. For clarity, we remark that Theorem 5.4 is subject to Hypothesis 5.2 and that Theorem 5.8 is subject to Hypothesis 5.7. Note that the lower bounds for dimensions $d = 2$ and $d = 3$ follow from Theorem 5.4 by just adding components consisting of only zeros. All upper bounds follow from Observation 5.1.

d	Lower Bound	Upper Bound
0	$\Omega(n)$ (trivial)	$\mathcal{O}(n)$
1	$n^{2-o(1)}$ (Theorem 5.4)	$\mathcal{O}(n^2)$
2	$n^{2-o(1)}$ (from above)	$\mathcal{O}(n^3)$
3	$n^{2-o(1)}$ (from above)	$\mathcal{O}(n^4)$
$d \geq 4$	$n^{d-2-o(1)}$ (Theorem 5.8)	$\mathcal{O}(n^{d+1})$

▶ **Observation 5.1.** *Coverability in an n-sized unary $(B(n), d)$-VASS can be solved in $\mathcal{O}(n(B(n)+1)^d)$-time.*

Lower Bounds for Coverability in Linearly-Bounded VASS

Now, we consider lower bounds for the coverability problem in linearly-bounded fixed dimension unary VASS. Firstly, in dimension one, we show that quadratic running time is conditionally optimal under the k-cycle hypothesis. Secondly, in dimensions four and higher, we require a running time at least $n^{d-2-o(1)}$ under the 3-uniform hyperclique hypothesis. Together, this provides evidence that the simple $\mathcal{O}(n^{d+1})$ algorithm for coverability in $(\mathcal{O}(n), d)$-VASS is close to optimal, as summarised in Table 1.

▶ **Hypothesis 5.2** (k-Cycle Hypothesis). *For every $\varepsilon > 0$, there exists a k such that there does not exist a $\mathcal{O}(m^{2-\varepsilon})$-time algorithm for finding a k-cycle in directed graphs with m edges.*

The k-cycle hypothesis arises from the state-of-the-art $\mathcal{O}(m^{2-\frac{c}{k}+o(1)})$-time algorithms, where c is some constant [4, 50, 19]. It has been previously used as an assumption for hardness results, for example, see [36, 5, 18]. It is a standard observation, due to colour-coding arguments, that we may without loss of generality assume that the graph given is a k-circle-layered graph [36, Lemma 2.2]. Specifically, we can assume that the input graph $G = (V, E)$ has vertex partition $V = V_0 \cup \cdots \cup V_{k-1}$ such that each edge $\{u, v\} \in E$ is in $V_i \times V_{i+1 \pmod k}$ for some $0 \leq i < k$. Furthermore, we may assume $|V| \leq |E|$.

The core of the upcoming lower bounds is captured in the following lemma; see Figure 3 for an overview and the full version [31] for the proof.

▶ **Lemma 5.3.** *Given a k-circle-layered graph $G = (V_0 \cup \cdots \cup V_{k-1}, E)$ with m edges, there exists a unary $(\mathcal{O}(n), 1)$-VASS $\mathcal{V}$ such that there is a k-cycle in G if and only if there exists a run from $p(0)$ to $q(0)$ in $\mathcal{V}$. Moreover, $\mathcal{V}$ has size $n \leq \mathcal{O}(m)$ and can be constructed in $\mathcal{O}(m)$-time.*

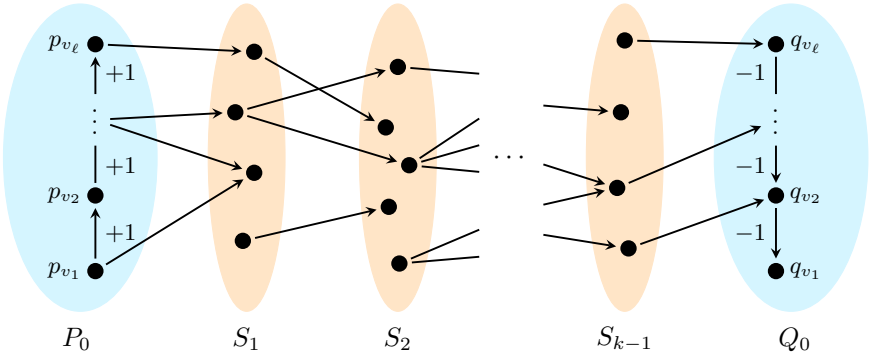


Figure 3 The $(\mathcal{O}(n), 1)$-VASS $\mathcal{V}$ of size $n \leq \mathcal{O}(m)$ for finding k-cycle in a k-circle-layered graphs with m edges. Note that unlabelled transitions have zero effect. Observe that the graph is mostly copied into the states and transitions of the linearly-bounded 1-VASS. Importantly, two copies of V_0 are created. By starting at $p_{v_1}(0)$ in the first copy, a vertex from V_0 belonging to the k-cycle can be selected by loading the sole counter with a value corresponding to that vertex. Then, in the second copy, $q_{v_1}(0)$ can only be reached if the state first arrived at corresponds to the vertex selected in the beginning. Accordingly, there is a run from $p_{v_1}(0)$ to $q_{v_1}(0)$ if and only if there exists a k-cycle, since the states visited in the underlying path of the run correspond to the vertices of the k-cycle.

▶ **Theorem 5.4.** *Assuming the k-cycle hypothesis, coverability and reachability in unary $(\mathcal{O}(n), 1)$-VASS of size n require $n^{2-o(1)}$-time.*

Proof. Assume for contradiction that reachability in a unary $(\mathcal{O}(n), 1)$-VASS of size n can be solved in $\mathcal{O}(n^{2-\varepsilon})$-time for some $\varepsilon > 0$. By the k-cycle hypothesis (Hypothesis 5.2), there exists a k such that the problem of finding a k-cycle in a k-circle layered graph with m vertices cannot be solved in $\mathcal{O}(m^{2-\varepsilon})$-time. Via the reduction presented above in Lemma 5.3, we create a $(\mathcal{O}(n), 1)$-VASS $\mathcal{V}$ of size $n \leq \mathcal{O}(m)$ together with an initial configuration $p(0)$ and a target configuration $q(0)$, such that deciding reachability from $p(0)$ to $q(0)$ in $\mathcal{V}$ determines the existence of a k-cycle in G. Thus the $\mathcal{O}(n^{2-\varepsilon})$ algorithm for reachability would give a $\mathcal{O}(m^{2-\varepsilon})$ algorithm for finding k-cycles, contradicting the k-cycle hypothesis.

By the equivalence of coverability and reachability in unary $(\mathcal{O}(n), 1)$ VASS in Lemma 5.6, the same lower bound holds for coverability. ◀

▶ **Corollary 5.5.** *Assuming the k-cycle hypothesis, coverability in unary 2-VASS of size n requires $n^{2-o(1)}$-time.*

Reachability in $(\mathcal{O}(n), d)$-VASS can be decided in $\mathcal{O}(n(B(n)+1)^d)$-time using the simple algorithm for Observation 5.1 with a trivially modified acceptance condition. It turns out that coverability and reachability are equivalent in unary $(\mathcal{O}(n), d)$-VASS. The following lemma is proved in the full version [31].

▶ **Lemma 5.6.** *For a $(B(n), d)$-VASS, let $C^{B(n)}(n)$ and $R^{B(n)}(n)$ denote the optimal running times for coverability and reachability, respectively. For any $\gamma > 0$, there exists some $\delta > 0$ such that $C^{\gamma \cdot n}(n) \leq \mathcal{O}(R^{\delta \cdot n}(n))$. Conversely, for any $\gamma > 0$, there exists some $\delta > 0$ such that $R^{\gamma \cdot n}(n) \leq \mathcal{O}(C^{\delta \cdot n}(n))$.*

Lower Bounds for Reachability in Linearly-Bounded VASS

To obtain further lower bounds for the coverability problem in $(\mathcal{O}(n), d)$-VASS, by Lemma 5.6, we can equivalently find lower bounds for the reachability problem in $(\mathcal{O}(n), d)$-VASS. In Theorem 5.8, we will assume a well-established hypothesis concerning the time required to find hypercliques in 3-uniform hypergraphs. In fact, Lincoln, Vassilevska Williams, and Williams state and justify an even stronger hypothesis about μ-uniform hypergraphs for every $\mu \geq 3$ [36, Hypothesis 1.4]. We will use this computational complexity hypothesis to expose precise lower bounds on the time complexity of reachability in linearly-bounded fixed dimension unary VASS.

▶ **Hypothesis 5.7** (k-Hyperclique Hypothesis [36, Hypothesis 1.4])**.** *Let $k \geq 3$ be an integer. On Word-RAM with $\mathcal{O}(\log(n))$ bit words, finding an k-hyperclique in a 3-uniform hypergraph on n vertices requires $n^{k-o(1)}$-time.*

▶ **Theorem 5.8.** *Assuming Hypothesis 5.7, reachability in unary $(\mathcal{O}(n), d+2)$-VASS of size n requires $n^{d-o(1)}$-time.*

For the remainder of this section, we focus on the proof of Theorem 5.8. The lower bound is obtained via reduction from finding hyperclique in 3-uniform hypergraphs, hence it is subject to the k-Hyperclique Hypothesis. We present our reduction in two steps. The first step is an intermediate step, in Lemma 5.9 we offer a reduction to an instance of reachability in unary VASS with a limited number of zero-tests (proved in the full version [31]). The second step extends the first, in Lemma 5.10 we modify the reduction by adding a counter so zero-tests are absented. This extension leverages the recently developed *controlling counter technique* of Czerwiński and Orlikowski [16]. This technique allows for implicit zero-tests to be performed in the presence of a dedicated counter whose transition effects and reachability condition ensure the implicit zero-tests were indeed performed correctly.

It has been shown that we may assume that the hypergraph is ℓ-partite for the k-Hyperclique Hypothesis [36, Theorem 3.1]. Thus, we may assume that the vertices can be partitioned into ℓ disjoint subsets $V = V_1 \cup \cdots \cup V_\ell$ and all hyperedges contain three vertices from distinct subsets $\{u, v, w\} \in V_i \times V_j \times V_k$ for some $1 \leq i < j < k \leq \ell$.

▶ **Lemma 5.9.** *Let $d \geq 1$ be a fixed integer. Given a $4d$-partite 3-uniform hypergraph $H = (V_1 \cup \ldots \cup V_{4d}, E)$ with n vertices, there exists a unary $(\mathcal{O}(n^{4+o(1)}), d+1)$-VASS with $\mathcal{O}(d^3)$ zero-tests $\mathcal{T}$ such that there is a $4d$-hyperclique in H if and only if there is a run from $q_I(\mathbf{0})$ to $q_F(\mathbf{v})$, for some $\mathbf{v} \geq \mathbf{0}$, in $\mathcal{T}$. Moreover, $\mathcal{T}$ can be constructed in $\mathsf{poly}(d) \cdot n^{4+o(1)}$-time.*

▶ **Lemma 5.10** ([16, Lemma 10])**.** *Let ρ be a run in a $(d+2)$-VASS such that $q_I(\mathbf{0}) \xrightarrow{\rho} q_F(\mathbf{0})$. Further, let $q_0(\mathbf{v}_0), q_1(\mathbf{v}_1) \ldots, q_r(\mathbf{v}_r)$ be some distinguished configurations observed along the run ρ with $q_0(\mathbf{v}_0) = q_I(\mathbf{0})$ and $q_r(\mathbf{v}_r) = q_F(\mathbf{0})$ and let ρ_j be the segment of ρ that is between $q_{j-1}(\mathbf{v}_{j-1})$ and $q_j(\mathbf{v}_j)$, so ρ can be described as*

$$q_I(\mathbf{0}) = q_0(\mathbf{v}_1) \xrightarrow{\rho_1} q_1(\mathbf{v}_1) \to \cdots \to q_{r-1}(\mathbf{v}_{r-1}) \xrightarrow{\rho_r} q_r(\mathbf{v}_r) = q_F(\mathbf{0}).$$

Let $S_1, \ldots, S_d, S_{d+1} \subseteq \{0, 1, \ldots, r\}$ be the sets of indices of the distinguished configurations where zero-tests could be performed on counters $\mathsf{x}_1, \ldots, \mathsf{x}_d, \mathsf{x}_{d+1}$, respectively. Let $t_{j,i} = |\{s \geq j : s \in S_i\}|$ be the number of zero-test for the counter x_i in the remainder of the run $\rho_{j+1} \cdots \rho_r$. Given that $\mathbf{v}_0 = \mathbf{0}$ and $\mathbf{v}_r = \mathbf{0}$, if

$$\mathrm{eff}(\rho_j)[d+2] = \sum_{i=1}^{d+1} t_{j,i} \cdot \mathrm{eff}(\rho_j)[i], \tag{1}$$

then for every $i \in \{1, \ldots, d, d+1\}$ and $j \in S_i$, we know that $\mathbf{v}_j[i] = 0$.

With Lemma 5.10 in hand, we can ensure that the $\mathcal{O}(d^3)$ zero-tests performed by $\mathcal{T}$, from Lemma 5.9, are executed correctly. We conclude this section with a proof of Theorem 5.8.

Proof of Theorem 5.8. Consider the reduction, presented in Lemma 5.9, from finding a $4d$-hyperclique in a $4d$-partite 3-uniform hypergraph H to reachability in $(\mathcal{O}(n^{4+o(1)}), d+1)$-VASS with $\mathcal{O}(d^3)$ zero-tests. Now, given Lemma 5.10, we will add a controlling counter to $\mathcal{T}$ so that the zero-tests on the $d+1$ counters $\mathsf{x}_1, \ldots, \mathsf{x}_d, \mathsf{y}$ are instead performed implicitly. So we introduce another counter z that receives updates on transitions, consistent with Equation 1, whenever any of the other counters are updated. Note that counters y and z, for the sake of a succinct and consistent description, are respectively referred to as counters x_{d+1} and x_{d+2} in the statement of Lemma 5.10. Moreover, notice that the maximum value of z is bounded by $\mathsf{poly}(d) \cdot \left(\sum_{i=1}^{d+1} \mathsf{x}_i\right) \in \mathsf{poly}(d) \cdot n^{4+o(1)}$.

Therefore, we have constructed a unary $(\mathsf{poly}(d) \cdot n^{4+o(1)}, d+2)$-VASS $\mathcal{V}$ with the property that there H contains a $4d$-hyperclique if and only if there is a run from $q'_I(\mathbf{0})$ to $q'_F(\mathbf{0})$ in $\mathcal{V}$. Such a $(\mathsf{poly}(d) \cdot n^{4+o(1)}, d+2)$-VASS $\mathcal{V}$ has size $\mathcal{O}(t \cdot |\mathcal{T}|)$ where $t \in \mathsf{poly}(d)$ is the number of zero-tests performed on the run from $q_I(\mathbf{0})$ to $q_F(\mathbf{0})$ in $\mathcal{T}$. Moreover, $\mathcal{V}$ can be constructed in $\mathsf{poly}(d) \cdot n^{4+o(1)}$ time. Hence, if reachability in $(\mathcal{O}(n), d+2)$-VASS of size n can be solved faster than $n^{d-o(1)}$, then one can find a $4d$-hyperclique in a 3-uniform hypergraph faster than $n^{4d-o(1)}$, contradicting Hypothesis 5.7. ◀

6 Conclusion

Summary. In this paper, we have revisited a classical problem of coverability in d-VASS. We have closed the gap left by Rosier and Yen [46] on the length of runs witnessing instances of coverability in d-VASS. We have lowered the upper bound of $n^{2^{\mathcal{O}(d \log d)}}$, from Rackoff's technique [45], to $n^{2^{\mathcal{O}(d)}}$ (Theorem 3.3), matching the $n^{2^{\Omega(d)}}$ lower bound from Lipton's construction [37]. This accordingly closes the gap on the exact space required for the coverability problem and yields a deterministic $n^{2^{\mathcal{O}(d)}}$-time algorithm for coverability in d-VASS (Corollary 3.4). We complement this with a matching lower bound conditional on ETH; there does not exist a deterministic $n^{2^{o(d)}}$-time algorithm for coverability (Theorem 4.2). By and large, this settles the exact space and time complexity of coverability in VASS.

In addition, we study linearly-bounded unary d-VASS. Here, coverability and reachability are equivalent and the trivial exhaustive search $\mathcal{O}(n^{d+1})$ algorithm is near-optimal. We prove that reachability in linearly-bounded 1-VASS requires $n^{2-o(1)}$-time under the k-cycle hypothesis (Theorem 5.4), matching the trivial upper bound. We further prove that reachability in linearly-bounded $(d+2)$-VASS requires $n^{d-o(1)}$-time under the 3-uniform hyperclique hypothesis (Theorem 5.8).

Open Problems. The *boundedness problem*, a problem closely related to coverability, asks whether, from a given initial configuration, the set of all reachable configurations is finite. This problem was also studied by Lipton then Rackoff and is EXPSPACE-complete [37, 45].

Boundedness was further analysed by Rosier and Yen [46, Theorem 2.1] and the same gap also exists for the exact space required. We leave the same improvement, to eliminate the same twice-exponentiated $\log(d)$ factor, as an open problem.

Our lower bounds for the time complexity of coverability and reachability in linearly-bounded unary d-VASS, for $d \geq 2$, leave a gap of up to $n^{3+o(1)}$, see Table 1. We leave it as an open problem to either improve upon the upper bound $\mathcal{O}(n^{d+1})$ given by the trivial algorithm, or to raise our conditional lower bounds.

References

1 Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Yih-Kuen Tsay. Algorithmic analysis of programs with well quasi-ordered domains. *Inf. Comput.*, 160(1-2):109–127, 2000. doi:10.1006/inco.1999.2843.

2 Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in P. *Annals of Mathematics*, pages 781–793, 2004.

3 Shaull Almagor, Nathann Cohen, Guillermo A. Pérez, Mahsa Shirmohammadi, and James Worrell. Coverability in 1-VASS with Disequality Tests. In Igor Konnov and Laura Kovács, editors, *31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference)*, volume 171 of *LIPIcs*, pages 38:1–38:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CONCUR.2020.38.

4 Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997. doi:10.1007/BF02523189.

5 Bertie Ancona, Monika Henzinger, Liam Roditty, Virginia Vassilevska Williams, and Nicole Wein. Algorithms and hardness for diameter in dynamic graphs. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 13:1–13:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.13.

6 Michael Blondin, Matthias Englert, Alain Finkel, Stefan Göller, Christoph Haase, Ranko Lazić, Pierre McKenzie, and Patrick Totzke. The Reachability Problem for Two-Dimensional Vector Addition Systems with States. *J. ACM*, 68(5):34:1–34:43, 2021. doi:10.1145/3464794.

7 Michael Blondin, Alain Finkel, Christoph Haase, and Serge Haddad. Approaching the Coverability Problem Continuously. In Marsha Chechik and Jean-François Raskin, editors, *Tools and Algorithms for the Construction and Analysis of Systems – 22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, volume 9636 of *Lecture Notes in Computer Science*, pages 480–496. Springer, 2016. doi:10.1007/978-3-662-49674-9_28.

8 Michael Blondin, Christoph Haase, and Philip Offtermatt. Directed reachability for infinite-state systems. In Jan Friso Groote and Kim Guldstrand Larsen, editors, *Tools and Algorithms for the Construction and Analysis of Systems – 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 – April 1, 2021, Proceedings, Part II*, volume 12652 of *Lecture Notes in Computer Science*, pages 3–23. Springer, 2021. doi:10.1007/978-3-030-72013-1_1.

9 Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data words. *ACM Trans. Comput. Log.*, 12(4):27:1–27:26, 2011. doi:10.1145/1970398.1970403.

10 Laura Bozzelli and Pierre Ganty. Complexity analysis of the backward coverability algorithm for VASS. In Giorgio Delzanno and Igor Potapov, editors, *Reachability Problems – 5th International Workshop, RP 2011, Genoa, Italy, September 28-30, 2011. Proceedings*, volume 6945 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2011. doi:10.1007/978-3-642-24288-5_10.

11 Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David W. Juedes, Iyad A. Kanj, and Ge Xia. Tight lower bounds for certain parameterized NP-hard problems. *Inf. Comput.*, 201(2):216–231, 2005. doi:10.1016/j.ic.2005.05.001.

12 Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.*, 72(8):1346–1367, 2006. doi:10.1016/j.jcss.2006.04.007.

13 Hubert Comon and Yan Jurski. Multiple counters automata, safety analysis and presburger arithmetic. In Alan J. Hu and Moshe Y. Vardi, editors, *Computer Aided Verification, 10th International Conference, CAV '98, Vancouver, BC, Canada, June 28 – July 2, 1998, Proceedings*, volume 1427 of *Lecture Notes in Computer Science*, pages 268–279. Springer, 1998. doi:10.1007/BFb0028751.

14 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

15 Wojciech Czerwiński, Slawomir Lasota, Ranko Lazic, Jérôme Leroux, and Filip Mazowiecki. Reachability in Fixed Dimension Vector Addition Systems with States. In Igor Konnov and Laura Kovács, editors, *31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference)*, volume 171 of *LIPIcs*, pages 48:1–48:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CONCUR.2020.48.

16 Wojciech Czerwiński and Łukasz Orlikowski. Reachability in Vector Addition Systems is Ackermann-complete. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1229–1240. IEEE, 2021. doi:10.1109/FOCS52979.2021.00120.

17 Wojciech Czerwiński and Łukasz Orlikowski. Lower Bounds for the Reachability Problem in Fixed Dimensional VASSes. In Christel Baier and Dana Fisman, editors, *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2–5, 2022*, pages 40:1–40:12. ACM, 2022. doi:10.1145/3531130.3533357.

18 Mina Dalirrooyfard, Ce Jin, Virginia Vassilevska Williams, and Nicole Wein. Approximation Algorithms and Hardness for n-Pairs Shortest Paths and All-Nodes Shortest Cycles. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 – November 3, 2022*, pages 290–300. IEEE, 2022. doi:10.1109/FOCS54457.2022.00034.

19 Mina Dalirrooyfard, Thuy Duong Vuong, and Virginia Vassilevska Williams. Graph pattern detection: Hardness for all induced patterns and faster noninduced cycles. *SIAM J. Comput.*, 50(5):1627–1662, 2021. doi:10.1137/20M1335054.

20 Javier Esparza. Decidability and Complexity of Petri Net Problems – An Introduction. In Wolfgang Reisig and Grzegorz Rozenberg, editors, *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets, held in Dagstuhl, September 1996*, volume 1491 of *Lecture Notes in Computer Science*, pages 374–428. Springer, 1996. doi:10.1007/3-540-65306-6_20.

21 Javier Esparza, Ruslán Ledesma-Garza, Rupak Majumdar, Philipp J. Meyer, and Filip Niksic. An SMT-Based Approach to Coverability Analysis. In Armin Biere and Roderick Bloem, editors, *Computer Aided Verification – 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, volume 8559 of *Lecture Notes in Computer Science*, pages 603–619. Springer, 2014. doi:10.1007/978-3-319-08867-9_40.

22 John Fearnley and Marcin Jurdziński. Reachability in Two-Clock Timed Automata Is PSPACE-Complete. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming – 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, volume 7966 of *Lecture Notes in Computer Science*, pages 212–223. Springer, 2013. doi:10.1007/978-3-642-39212-2_21.

23 Diego Figueira, Santiago Figueira, Sylvain Schmitz, and Philippe Schnoebelen. Ackermannian and Primitive-Recursive Bounds with Dickson's Lemma. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada*, pages 269–278. IEEE Computer Society, 2011. doi:10.1109/LICS.2011.39.

24 Pierre Ganty and Rupak Majumdar. Algorithmic verification of asynchronous programs. *ACM Trans. Program. Lang. Syst.*, 34(1):6:1–6:48, 2012. doi:10.1145/2160910.2160915.

25 Steven M. German and A. Prasad Sistla. Reasoning about systems with many processes. *J. ACM*, 39(3):675–735, 1992. doi:10.1145/146637.146681.

26 Christoph Haase, Stephan Kreutzer, Joël Ouaknine, and James Worrell. Reachability in Succinct and Parametric One-Counter Automata. In Mario Bravetti and Gianluigi Zavattaro, editors, *CONCUR 2009 – Concurrency Theory, 20th International Conference, CONCUR 2009, Bologna, Italy, September 1-4, 2009. Proceedings*, volume 5710 of *Lecture Notes in Computer Science*, pages 369–383. Springer, 2009. doi:10.1007/978-3-642-04081-8_25.

27 Christoph Haase, Joël Ouaknine, and James Worrell. On the relationship between reachability problems in timed and counter automata. In Alain Finkel, Jérôme Leroux, and Igor Potapov, editors, *Reachability Problems – 6th International Workshop, RP 2012, Bordeaux, France, September 17-19, 2012. Proceedings*, volume 7550 of *Lecture Notes in Computer Science*, pages 54–65. Springer, 2012. doi:10.1007/978-3-642-33512-9_6.

28 John E. Hopcroft and Jean-Jacques Pansiot. On the Reachability Problem for 5-Dimensional Vector Addition Systems. *Theor. Comput. Sci.*, 8:135–159, 1979. doi:10.1016/0304-3975(79)90041-0.

29 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

30 Ulla Koppenhagen and Ernst W. Mayr. Optimal algorithms for the coverability, the subword, the containment, and the equivalence problems for commutative semigroups. *Inf. Comput.*, 158(2):98–124, 2000. doi:10.1006/inco.1999.2812.

31 Marvin Künnemann, Filip Mazowiecki, Lia Schütze, Henry Sinclair-Banks, and Karol Węgrzycki. Coverability in VASS Revisited: Improving Rackoff's Bound to Obtain Conditional Optimality, 2023. arXiv:2305.01581.

32 Ranko Lazic and Sylvain Schmitz. The ideal view on Rackoff's coverability technique. *Inf. Comput.*, 277:104582, 2021. doi:10.1016/j.ic.2020.104582.

33 Jérôme Leroux. Vector addition system reversible reachability problem. *Log. Methods Comput. Sci.*, 9(1), 2013. doi:10.2168/LMCS-9(1:5)2013.

34 Jérôme Leroux. The Reachability Problem for Petri Nets is Not Primitive Recursive. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1241–1252. IEEE, 2021. doi:10.1109/FOCS52979.2021.00121.

35 Jérôme Leroux and Sylvain Schmitz. Reachability in Vector Addition Systems is Primitive-Recursive in Fixed Dimension. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–13. IEEE, 2019. doi:10.1109/LICS.2019.8785796.

36 Andrea Lincoln, Virginia Vassilevska Williams, and R. Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1236–1252. SIAM, 2018. doi:10.1137/1.9781611975031.80.

37 Richard Lipton. The Reachability Problem Requires Exponential Space. *Department of Computer Science. Yale University*, 62, 1976.

38 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the Exponential Time Hypothesis. *Bulletin of EATCS*, 3(105), 2013.

39 Ernst W. Mayr. An Algorithm for the General Petri Net Reachability Problem. *SIAM J. Comput.*, 13(3):441–460, 1984. doi:10.1137/0213029.

40 Ernst W. Mayr and Albert R. Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in Mathematics*, 46(3):305–329, 1982. doi:10.1016/0001-8708(82)90048-2.

41 Filip Mazowiecki and Michał Pilipczuk. Reachability for Bounded Branching VASS. In Wan J. Fokkink and Rob van Glabbeek, editors, *30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands*, volume 140 of *LIPIcs*, pages 28:1–28:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.CONCUR.2019.28.

42 Filip Mazowiecki, Henry Sinclair-Banks, and Karol Węgrzycki. Coverability in 2-VASS with One Unary Counter is in NP. In Orna Kupferman and Paweł Sobociński, editors, *Foundations of Software Science and Computation Structures*, pages 196–217. Springer Nature Switzerland, 2023. doi:10.1007/978-3-031-30829-1_10.

43 Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Inc., 1967.

44 Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

45 Charles Rackoff. The Covering and Boundedness Problems for Vector Addition Systems. *Theor. Comput. Sci.*, 6:223–231, 1978. doi:10.1016/0304-3975(78)90036-1.

46 Louis E. Rosier and Hsu-Chun Yen. A Multiparameter Analysis of the Boundedness Problem for Vector Addition Systems. *J. Comput. Syst. Sci.*, 32(1):105–135, 1986. doi:10.1016/0022-0000(86)90006-1.

47 Sylvain Schmitz. The Complexity of Reachability in Vector Addition Systems. *ACM SIGLOG News*, 3(1):4–21, 2016. URL: https://dl.acm.org/citation.cfm?id=2893585, doi:10.1145/2893582.2893585.

48 Leslie G. Valiant and Mike Paterson. Deterministic One-Counter Automata. *J. Comput. Syst. Sci.*, 10(3):340–350, 1975. doi:10.1016/S0022-0000(75)80005-5.

49 Wil M. P. van der Aalst. Verification of Workflow Nets. In Pierre Azéma and Gianfranco Balbo, editors, *Application and Theory of Petri Nets 1997, 18th International Conference, ICATPN '97, Toulouse, France, June 23-27, 1997, Proceedings*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426. Springer, 1997. doi:10.1007/3-540-63139-9_48.

50 Raphael Yuster and Uri Zwick. Detecting short directed cycles using rectangular matrix multiplication and dynamic programming. In J. Ian Munro, editor, *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 254–260. SIAM, 2004. URL: http://dl.acm.org/citation.cfm?id=982792.982828.

51 Don Zagier. Newman's short proof of the prime number theorem. *The American mathematical monthly*, 104(8):705–708, 1997.

First Order Logic on Pathwidth Revisited Again

Michael Lampis
Université Paris-Dauphine, PSL University, CNRS, LAMSADE, 75016, Paris, France

Abstract

Courcelle's celebrated theorem states that all MSO-expressible properties can be decided in linear time on graphs of bounded treewidth. Unfortunately, the hidden constant implied by this theorem is a tower of exponentials whose height increases with each quantifier alternation in the formula. More devastatingly, this cannot be improved, under standard assumptions, even if we consider the much more restricted problem of deciding FO-expressible properties on trees.

In this paper we revisit this well-studied topic and identify a natural special case where the dependence of Courcelle's theorem can, in fact, be improved. Specifically, we show that all FO-expressible properties can be decided with an elementary dependence on the input formula, if the input graph has bounded pathwidth (rather than treewidth). This is a rare example of treewidth and pathwidth having different complexity behaviors. Our result is also in sharp contrast with MSO logic on graphs of bounded pathwidth, where it is known that the dependence has to be non-elementary, under standard assumptions. Our work builds upon, and generalizes, a corresponding meta-theorem by Gajarský and Hliněný for the more restricted class of graphs of bounded tree-depth.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Algorithmic Meta-Theorems, FO logic, Pathwidth

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.132

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: https://arxiv.org/abs/2210.09899

Funding Supported by ANR projects ANR-21-CE48-0022 (S-EX-AP-PE-AL) and ANR-18-CE40-0025 (ASSK).

1 Introduction

Algorithmic meta-theorems are general statements of the form "all problems in a certain class are tractable on a particular class of inputs". Probably the most famous and celebrated result of this type is Courcelle's theorem [5], which states that all graph properties expressible in Monadic Second Order (MSO) logic are solvable in linear time on graphs of bounded treewidth. This result has proved to be of immense importance to parameterized complexity theory, because a vast collection of natural NP-hard problems can be expressed in MSO logic (and its variations that allow optimization objectives [1]) and because treewidth is the most well-studied structural graph parameter. Thanks to Courcelle's theorem, we immediately obtain that all such problems are fixed-parameter tractable (FPT) parameterized by treewidth.

Despite its great success, Courcelle's theorem suffers from a significant weakness: the algorithm it guarantees has a running time that is astronomical for most problems. Indeed, a careful reading of the theorem shows that the running time increases as a tower of exponentials whose height is equal to the number of quantifier alternations of the input MSO formula. Hence, even though Courcelle's theorem shows that any MSO formula ϕ can be decided on n-vertex graphs of treewidth tw in time $f(\phi, \mathrm{tw})n$, the function f is *non-elementary*, that is, it cannot be bounded from above by any tower of exponentials of fixed height.

Table 1 Summary of the state of the art for FO and MSO model checking on graphs of bounded treewidth, pathwidth, and tree-depth. Elementary (green cells) indicates that there is an algorithm which, when the corresponding width is bounded by an absolute constant, decides any formula ϕ in time $f(\phi)n^{O(1)}$, where f is a function that can be bounded above by a finite tower of exponentials. For the remaining cases, this is known to be impossible, under standard assumptions, hence it is inevitable to have an $f(\phi)$ that is a tower of exponentials whose height increases with ϕ.

Parameter	FO	MSO
Treewidth	Non-elementary on Trees [13]	Non-elementary on Trees [13]
Pathwidth	Elementary (Theorem 24)	Non-elementary on Caterpillars [13]
Tree-depth	Elementary [14]	Elementary [14]

One could hope that this terrible dependence on ϕ is an artifact of Courcelle's proof technique. Unfortunately, it was shown in a very influential work by Frick and Grohe [13] that this non-elementary dependence on the number of quantifiers of ϕ is best possible (under standard assumptions), even if one considers the severely restricted special case of model-checking First Order (FO) logic on trees. Recall that FO logic is a basic logic formalism that allows us to express graph properties using quantification over the vertices of the graph, while MSO logic also allows quantification over sets of vertices. Since FO logic is trivially a subset of MSO logic and trees have treewidth 1, this result established that Courcelle's theorem is essentially best possible.

Frick and Grohe's lower bound thus provided the motivation for the search for subclasses of bounded-treewidth graphs where avoiding the non-elementary dependence on ϕ may be possible. The obvious next place to look was naturally, pathwidth, which is the most well-known restriction (and close cousin) of treewidth. Unfortunately, Frick and Grohe's paper provided a negative result for MSO model checking also for this parameter. More precisely, they showed that MSO model checking on strings with a total order relation has a non-elementary dependence on the formula (unless P=NP), but such structures can easily be embedded into caterpillars (which are graphs of pathwidth 1) if one allows quantification over sets. Notice, however, that this does not settle the complexity of FO logic for graphs of counstant pathwidth, as it is not clear how one could implement the total ordering relation of a string without access to set quantifiers (we expand on this question further below).

On the positive side, Frick and Grohe's lower bounds motivated the discovery of several meta-theorems with elementary dependence on the formula for other, more restricted variations of treewidth (we review some such results below). Of all these results, the one that is "closest" to treewidth, is the theorem of Gajarský and Hliněný [14], which states that on graphs of constant tree-depth, MSO (and hence FO) model checking has elementary dependence on the input formula. It is known that for all n-vertex graphs G we have $\mathrm{tw}(G) \leq \mathrm{pw}(G) \leq \mathrm{td}(G) \leq \mathrm{tw}(G) \log n$, where $\mathrm{tw}, \mathrm{pw}, \mathrm{td}$ denote the treewidth, pathwidth and tree-depth. In a sense, this positive result seemed to go as far as one could possibly go towards emulating treewidth, while retaining the elementary dependence on the formula and avoiding the lower bound of Frick and Grohe. This state of the art is summarized in Table 1.

Our result. In this paper we revisit this well-studied topic and address the one remaining case of Table 1 where it is still unknown whether it is possible to obtain an elementary dependence on the formula for model checking. We answer this question positively, showing that if we restrict ourselves to graphs of pathwidth p, where p is an absolute constant, then FO formulas with q quantifiers can be decided in time $f(q)n^{O(1)}$, where f is an elementary function of q. More precisely, the function f is at most a tower of exponentials of height $O(p)$. In other words, our result trades the non-elementary dependence on q which is inherent in Courcelle's theorem, with a non-elementary dependence on p. Though this may seem

disappointing at first, it is known that this is the best one could have hoped for. In fact, the meta-theorem of [14] also has this behavior (its parameter dependence is a tower of exponentials whose height increases with the tree-depth), and it was shown in [24] that this is best possible (under standard assumptions). Since pathwidth is a more general parameter, we cannot evade this lower bound and our algorithm needs to have a non-elementary dependence on pathwidth, if its dependence on the formula is elementary.

The result we obtain is, therefore, in a sense best possible and fills a natural gap in our knowledge regarding FO model checking for a well-studied graph width. Beyond filling this gap, the fact that we are able to give a positive answer to this question and obtain an algorithm with "good" dependence on the formula is interesting, and perhaps even rather striking, for several reasons. First, in many cases in this domain, it is impossible to obtain an elementary dependence on q, no matter how much we are willing to sacrifice on our dependence on the graph width, as demonstrated by the fact that the lower bounds of Table 1 apply for classes with the smallest possible width (trees and caterpillars). Second, even though FO seems much weaker than MSO in general, the complexities of model checking the two logics seem to be similar (that is, at most one level of exponentiation apart) for most parameters (we review some further examples below). Indeed, a main contribution of [14] was to prove that for graphs of bounded tree-depth, the two logics are actually equivalent. It is therefore somewhat unusual (for this context) that for pathwidth FO has quite different complexity from MSO logic. Third, even though treewidth and pathwidth are arguably the two most well-studied graph widths in parameterized complexity, by and large the complexities of the vast majority of problems are the same for both parameters (for more information on this, see [2] which only recently discovered the first example of a natural problem separating the two parameters). It is therefore remarkable that the complexity of FO model checking is so different for pathwidth and treewidth.

Finally, one aspect of our result that makes it more surprising is that it does not seem to generalize to dense graphs. Meta-theorems that give a non-elementary dependence on the formula by using a restriction of treewidth, generally have a dense graph analogue, using a restriction of clique-width (the dense graph analogue of treewidth). Indeed, this is the case for vertex cover [23] (neighborhood diversity [23], twin cover [16]) but also for tree-depth (shrub-depth [17]). One may have expected something similar to hold in our case. However, the natural dense analogue of pathwidth is linear clique-width and it is already known that FO logic has a non-elementary dependence on threshold graphs [24]. Since threshold graphs have linear clique-width 2, we cannot hope to extend our result to this parameter and it appears that the positive result of this paper is an isolated island of "tractability".

High-level proof overview. Our technique extends and builds upon the meta-theorem of [14] which handles the more restricted case of graphs of bounded tree-depth. We recall that the heart of this meta-theorem is the basic observation that FO logic has bounded counting power: if our graph contains $q+1$ identical parts (for some appropriate definition of "identical"), then deleting one cannot affect the validity of any FO formula with q quantifiers. The approach of [14] is to partition the vertices of the graph depending on their height in the tree-depth decomposition, then identify (and delete) identical vertices in the bottom level. This bounds the degree of vertices one level up, which allows us to partition them into a bounded number of types, delete components of the same type if we have too many, hence bound the degree of vertices one level up, and so on until the size of the whole graph is bounded.

Our approach borrows much of this general strategy: we will appropriately rank the vertices of the graph and then move from lower to higher ranks, at each step bounding the maximum degree of any vertex of the current rank. Besides the fact that ranking vertices into levels is less obvious when given a path decomposition, rather than a tree of fixed height,

the main difficulty we encounter is that no matter where we start, we cannot in general easily find identical parts where something can be safely deleted. Intuitively, this is demonstrated by the contrast between the simplest bounded tree-depth graph (a star, where leaves are twins, hence one can easily delete one if we have at least $q+1$) and the simplest bounded pathwidth graph (a path, which contains no twins). In order to handle this more general case, we need to combine the previous approach with arguments that rely on the locality of FO logic.

To understand informally what we mean by this, recall the classical argument which proves that Reachability is not expressible in FO logic. One way this is proved, is to show that a graph G_1 which is a long path (of say, 4^q vertices) and a graph G_2 which is a union of a path and a cycle (of say, $2 \cdot 4^{q-1}$ vertices each) are indistinguishable for FO formulas with q quantifiers. Our strategy is to flip this argument: if we are asked to model check a formula on a long path, we might as well model check the same formula on a simpler (less connected) graph which contains a shorter path and a cycle. Of course, our input graphs will be more complicated than long paths; we will, however, be dealing with long path-like structures, as our graph has small pathwidth. Our strategy is to perform a surgical rewiring operation on the path decomposition, producing the union of a shorter decomposition and a ring-like structure, while still satisfying the same formulas (the reader may skip ahead to Figure 1 to get a feeling for this operation). In other words, the main technical ingredient of our algorithm is inspired by (and exploits) a classical impossibility result on the expressiveness of FO logic. The abstract idea is (in a rough sense) to apply this argument repeatedly, so that if we started with a long path decomposition, we end up with a short path decomposition and many "disconnected rings". Eventually, we will be able to produce some such rings which are identical, delete them, and simplify the graph.

There are, of course, now various technical difficulties we need to overcome in order to turn this intuition into a precise argument. First, when we cut at two points in the path decomposition to extract the part that will form the "ring", we need to make sure that at an appropriate radius around the cut points the decompositions are isomorphic. It is not hard to calculate the appropriate radius we need in the graph (it is known that q-quantifier FO formulas depend on balls of radius roughly 2^q), but a priori two vertices which are close in the graph could be far in the path decomposition. To handle this, we take care when we rank the vertices, so that vertices of lower rank are guaranteed to only appear in a bounded number of bags, hence distances in the path decomposition approximate distances in the graph. Second, we need to calculate how long our decomposition needs to be before we can guarantee that we will be able to find some appropriate cut points. Here we use some counting arguments and pigeonhole principle to show that a path decomposition with length double-exponential in the desired radius is sufficient. Finally, once we find sufficiently many points to rewire and produce sufficiently many "rings", we need to prove that this did not affect the validity of the formula. Then, we are free to delete one, using the same argument as [14] and obtain a smaller equivalent graph. In the end, once we can no longer repeat this process, we obtain a bounded-degree graph, where it is known that FO model checking has an elementary dependence on the formula.

Overall, even though the algorithm we present seems somewhat complicated, the basic ingredients are simple and well-known: the fact that deleting one of many identical parts does not affect the validity of the formula (which is also used in [14]); the fact that FO formulas are not affected if we edit the graph in a way that preserves balls of a small radius around each vertex; and simple counting arguments and the pigeonhole principle.

Paper Organization. We conclude this section below with a short overview of other related work on algorithmic meta-theorems and continue in Section 2 with definitions and notation. The rest of the paper is organized as follows:

1. In Section 3 we present two lemmas, which are standard facts on FO logic, with minor adjustments to our setting. In particular, in Section 3.1 we present the lemma that states that if we have $q+1$ identical parts, it is safe to delete one; and in Section 3.2 we present the lemma that states that if two graphs agree on the local extended neighborhoods around each vertex (for some appropriate radius), then they satisfy the same formulas (that is, FO logic is local). Since these facts are standard, the reader may wish to skip the proofs of Section 3, which are given for the sake of completeness, during a first reading.
2. Then, in Section 4 we present the specific tools we will use to simplify our graph. In Section 4.1 we explain how we rank the vertices of a path decomposition so that each vertex has few neighbors of higher rank (but possibly many neighbors of lower rank). This allows us to process the ranks from lower to higher, simplifying the graph step by step. Then, in Section 4.2 we use some counting arguments to calculate the length of a path decomposition that guarantees the existence of long isomorphic blocks, on which we will apply the rewiring operation. We also show how distances in the graph can be approximated by distances in the path decomposition, if we have bounded the number of occurrences of each vertex in the decomposition. Finally, in Section 4.3 we formally define the rewiring operation and show that if the points where we apply it are in the middle of sufficiently long isomorphic blocks of the decomposition, this operation is safe. We also show that the "rings" it produces can be considered identical, in a sense that will allow us to invoke the lemma of Section 3.1 and delete one.
3. We put everything together in Section 5, where we explain how the lemmas we have presented form parts of an algorithm that ranks the vertices of a graph supplied with a path decomposition and then processes ranks one by one, decreasing the number of occurences of each vertex in the decomposition without affecting the validity of any formula (with q quantifiers). In the end, the processed graph has bounded degree and we invoke known results to decide the formula.

Other related work. Algorithmic meta-theorems are a very well-studied topic in parameterized complexity ([20]) and much work has been devoted in improving and extending Courcelle's theorem. Among such results, we mention the generalization of this theorem to MSO for clique-width, which covers dense graphs [6]. For FO logic, fixed-parameter tractability can be extended to much wider classes of graphs, with the recently introduced notion of twin-width nicely capturing many results in the area [4, 9, 11, 12]. Of course, since all these classes include the class of all trees, the non-elementary dependence on the formula implied by the lower bound of [13] still applies. Meta-theorems have also been given for logics other than FO and MSO, with the goal of either targeting a wider class of problems [18, 21, 22, 28], or achieving better complexity [26]. Kernelization [3, 10, 19] and approximation [8] are also topics where meta-theorems have been studied.

The meta-theorems which are more relevant to the current work are those which explicitly try to improve upon the parameter dependence given by Courcelle, by considering more restricted parameters. We mention here the meta-theorems for vertex cover, max-leaf, and neighborhood diversity [23], twin-cover [16], shrub-depth [17], and vertex integrity [25]. As mentioned, one common aspect of these meta-theorems is that they handle both FO and MSO logic, without a huge difference in complexity (at most one extra level of exponentiation in the parameter dependence), which makes the behavior of FO logic on treewidth somewhat

unusual. The only exception, is the meta-theorem on graphs of bounded max-leaf number of [23] which does not generalize to MSO logic. It was later shown that this is with good reason, as MSO logic has a non-elementary dependence even for unlabeled paths [24], which have the smallest possible max-leaf number. This is therefore the only previous result in the literature which mirrors the situation for pathwidth.

A classical result, incomparable to the parameters mentioned above, is the fact that FO model checking is FPT (with an elementary, triple-exponential dependence on the formula) on graphs of bounded degree [27]. We will use this fact as the last step of our algorithm.

The complexity of model checking FO and MSO formulas on structures other than graphs, such as posets [15] and strings has also been investigated. As mentioned, the case of strings is of particular interest to us, because the standard structure that represents a string over a fixed alphabet (a universe that contains the letters of the string, unary predicates that indicate for each letter which character of the alphabet it corresponds to, and a total ordering relation $\prec$ which indicates the ordering of the letters in the string) allows us to easily translate MSO properties of strings into MSO properties of an appropriate caterpillar. Indeed, to embed a string into a caterpillar, we can start with a path with endpoints s, t, and use one vertex of the path to represent each letter in the string. We can attach an appropriate (constant) number of leaves on each vertex to signify which character it represents. The precedence relation $x \prec y$ of the string now becomes the relation "every connected set that contains s and y, also contains x", which is MSO-expressible. Thanks to this simple transformation, the lower bound result of [13] on model checking MSO (and even FO) logic on strings, immediately carries over to graphs of pathwidth 1. Note, however, that the existence of the ordering relation is crucial, as FO model checking on other models of strings (e.g. with a successor relation) has elementary dependence on the formula, as such structures have bounded degree [13]. Hence, it seems that if we focus on FO (rather than MSO) logic, the similarity between model checking on bounded pathwidth graphs and strings becomes much weaker: FO model checking is easier on graphs of bounded pathwidth than on strings with an ordering relation, but harder than on strings with only a successor relation (as the lower bound of [24] for tree-depth applies to pathwidth, and rules out an algorithm with "only" triple-exponential dependence).

2 Definitions and Preliminaries

We use standard graph-theoretic notation and assume the reader is familiar with the basics of parameterized complexity (see e.g. [7]). For a graph $G = (V, E)$, and $S \subseteq V$, we use $G[S]$ to denote the subgraph of G induced by S. When r is a positive integer, we use $[r]$ to denote the set $\{1, \ldots, r\}$, while for two integers s, t, we use $[s, t]$ to denote the set $\{i \in \mathbb{Z} \mid s \leq i \leq t\}$. Note that if $t < s$ then $[s, t] = \emptyset$. We define $\mathrm{tow}(i, n)$ as follows: $\mathrm{tow}(0, n) = n$ and $\mathrm{tow}(i + 1, n) = 2^{\mathrm{tow}(i,n)}$. A function $f : \mathbb{N} \to \mathbb{N}$ is elementary if there exists a fixed i such that for all n we have $f(n) \leq \mathrm{tow}(i, n)$.

We recall the standard notion of path decomposition: a path decomposition of a graph $G = (V, E)$ is an ordered sequence of bags $B_1, B_2, \ldots, B_\ell$, where each B_i is a subset of V, that satisfies the following: (i) $\bigcup_{j \in [\ell]} B_j = V$ and for all $uv \in E$ there exists $i \in [\ell]$ such that $\{u, v\} \subseteq B_i$ (ii) for all $i_1 < i_2 < i_3$, with $i_1, i_2, i_3 \in [r]$ we have $B_{i_1} \cap B_{i_3} \subseteq B_{i_2}$. The width of a path decomposition is the number of vertices in the largest bag (minus one). The pathwidth of a graph G is the smallest width of any path decomposition of G.

First Order Logic. We use a standard form of First Order (FO) logic on graphs, where quantified variables are allowed to range over vertices. To simplify the presentation of some results, we will allow our formulas to also refer to vertex constants, corresponding to some specific vertices of the graph. More formally, the structures on which we will perform model checking are k-terminal graphs as defined below.

▶ **Definition 1.** *For a positive integer k, a k-terminal graph $G = (V, E)$ is a graph supplied with a function $\mathcal{T} : [k] \to V$, called the terminal labeling function. For $i \in [k]$, we say that $\mathcal{T}(i)$ is the i-th terminal of G. The set of terminals is the set T of images of $\mathcal{T}$ in V. Vertices of $V \setminus T$ are called non-terminals.*

Intuitively, terminals will play two roles: on the one hand, we define FO logic on graphs (below) in a way that allows formulas to refer to the terminal vertices; on the other, in some parts of our algorithm we will use a set of terminals that form a separator of the graph and hence allow us to break down the graph into smaller components. Note, however, that Definition 1 does not require the k terminals to be a separator, or have any other particular property.

A formula of FO logic is made up of the following vocabulary: (i) vertex variables, denoted $x_1, x_2, \ldots$ (ii) vertex constants denoted $\ell_1, \ell_2, \ldots$ (iii) existential quantification $\exists$ (iv) the boolean operations $\neg, \vee$ (v) the binary predicates $\sim$ (for adjacency) and $=$ (for equality). More formally, a First Order formula is a formula produced by the following grammar, where x represents a vertex variable and y represents a vertex variable or constant:

$$\phi \to \exists x.\phi \mid \neg\phi \mid \phi \vee \phi \mid y \sim y \mid y = y$$

A FO formula ϕ is called a sentence if every vertex variable x appearing in ϕ is quantified, that is, x appears within the scope of $\exists x$. A variable that is not quantified is called a free variable. For a formula ϕ that contains a free variable x, we will write $\phi[x/\ell_i]$ to denote the formula obtained by replacing every occurrence of x in ϕ by the constant ℓ_i.

The main problem we are concerned with is model checking: given a k-terminal graph G and a sentence ϕ, decide if G satisfies ϕ. We define the semantics of what this means inductively in a standard way, as follows. We say that a k-terminal graph $G = (V, E)$ with labeling function $\mathcal{T}$ models (or satisfies) a formula ϕ, and write $G, \mathcal{T} \models \phi$ (or simply $G \models \phi$ if $\mathcal{T}$ is clear from the context) if and only if we have one of the following:

1. $\phi := (\ell_i = \ell_j)$, where $i, j \in [k]$ and $\mathcal{T}(i)$ is the same vertex as $\mathcal{T}(j)$.
2. $\phi := (\ell_i \sim \ell_j)$, where $i, j \in [k]$ and $\mathcal{T}(i)\mathcal{T}(j) \in E$.
3. $\phi := (\neg\psi)$ and it is not the case that $G, \mathcal{T} \models \psi$.
4. $\phi := (\psi_1 \vee \psi_2)$ and at least one of $G \models \psi_1$, $G \models \psi_2$ holds.
5. $\phi := (\exists x.\psi)$ and there exists $v \in V$ such that $G, \mathcal{T}' \models \psi[x/\ell_{(k+1)}]$, where $\mathcal{T}'$ is the labeling function that sets $\mathcal{T}'(k+1) = v$ and $\mathcal{T}'(i) = \mathcal{T}(i)$ for $i \in [k]$.

Note that we have not included in our definition universal quantification or other boolean connectives such as $\wedge$. However, this is without loss of generality as $\forall x.\phi$ can be thought of as shorthand for $\neg\exists x.\neg\phi$ and all missing boolean connectives can be simulated using $\neg$ and $\vee$.

Let us also define a kind of isomorphism between labeled graphs that is guaranteed to leave terminal vertices untouched.

▶ **Definition 2.** *A terminal-respecting isomorphism between two k-terminal graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with terminal labeling functions $\mathcal{T}_1, \mathcal{T}_2$ is a bijective function $f : V_1 \to V_2$ such that (i) for all $u, u' \in V_1$ we have $uu' \in E_1$ if and only if $f(u)f(u') \in E_2$ (ii) for each $i \in [k]$, $f(\mathcal{T}_1(i)) = \mathcal{T}_2(i)$.*

We recall the following basic fact about FO logic which states that isomorphic structures satisfy the same sentences (see e.g. Lemma 9 of [25] for a proof).

▶ **Lemma 3.** *If G_1, G_2 are two k-terminal graphs such there exists a terminal-respecting isomorphism from G_1 to G_2, then, for all FO sentences ϕ we have $G_1 \models \phi$ if and only if $G_2 \models \phi$.*

3 Two Basic Lemmas

The purpose of this section is to establish two basic ingredients that will allow us to simplify the input graph without affecting whether it satisfies any FO formula with at most a given number q of quantified variables. The first lemma (Lemma 5) is rather simple and states that if a graph contains many "identical" components, we can safely remove one. Despite its simplicity, this idea has been sufficient to obtain many of the best currently known meta-theorems with non-elementary dependence in the formula, such as the meta-theorem of [14] for graphs of bounded tree-depth.

The second lemma (Lemma 9) is a variation of standard arguments regarding the locality of FO logic. It states that if we have two graphs which look locally the same, in the sense that for each vertex of one graph there exists a vertex of the other whose r-neighborhood is the same, for some appropriately chosen r, then actually the two graphs are indistinguishable by FO formulas with q quantifiers (even though they are not necessarily isomorphic). As we explained, we intend to use this to allow us to take parts of the graph that resemble "long", low-pathwidth components and cut them up into smaller, disconnected components. The strategy is to eventually produce a large enough number of such components that we can apply Lemma 5 and simplify the graph.

3.1 Identical Parts

We would now like to show that if the given graph contains many (say, at least $q+1$) "identical" parts, then it is safe to delete one without affecting whether the graph satisfies any FO formula with at most q quantifiers. We first define what we mean that two sets of vertices are identical in a k-terminal graph and then prove that if we can find $q+1$ such sets in a graph, we can safely delete one without affecting whether any FO formula with at most q quantifiers is satisfied.

▶ **Definition 4.** *Let $G = (V, E)$ be a k-terminal graph with labeling function $\mathcal{T}$ and terminal set T. We say that two disjoint sets of vertices C_1, C_2 are* identical *if there exists a terminal-respecting isomorphism from G to G that maps all vertices of C_1 to C_2 and all vertices of C_2 to C_1, and maps every vertex of $V \setminus (C_1 \cup C_2)$ to itself.*

Before we proceed, let us make two easy observations. First, if C_1, C_2 are identical, it must be the case that $(C_1 \cup C_2) \cap T = \emptyset$, because C_1, C_2 are disjoint and terminal-respecting isomorphisms must map vertices of T to themselves. Second, the relation of being identical is an equivalence relation on a collection of pairwise disjoint sets of vertices, that is, if C_1, C_2, C_3 are disjoint, C_1 is identical to C_2, and C_2 is identical to C_3, then C_1 is identical to C_3 (the fact that the relation is reflexive and symmetric is easy to see).

▶ **Lemma 5.** *Fix a positive integer q. Let $G = (V, E)$ be a k-terminal graph with labeling function $\mathcal{T}$ and terminal set T and suppose that $C_1, C_2, \ldots, C_{q+1}$ are $q+1$ sets of vertices of G which are pairwise identical. Then, for all FO sentences with at most q quantifiers we have that $G, \mathcal{T} \models \phi$ if and only if $G[V \setminus C_1], \mathcal{T} \models \phi$.*

3.2 Similar Neighborhoods

We now move on to present a lemma that will allow us to claim that two graphs are indistinguishable for FO formulas with q quantifiers if they are locally the same. This is a standard argument in FO logic, going back to Gaifman, though we need to adjust the proof to our purposes to handle terminal vertices appropriately. In particular, we will on the one hand be stricter on the isomorphisms we allow before we consider that the neighborhoods of two vertices are the same (because we only allow terminal-respecting isomorphisms), but on the other, we will only consider the extended neighborhood around a vertex by considering paths that go through non-terminals. This is important, because it allows us to work around the case where, for example, a terminal vertex is connected to everything and hence the diameter of the graph is 2. In such a case, the extended neighborhood of a non-terminal vertex will not trivially contain the whole graph, because we exclude paths that go through the supposed universal terminal.

According to this discussion, we define the notion of a ball of radius r around a vertex v, denoted $B_r(v)$, in a way that only takes into account paths whose internal vertices are non-terminals, as follows.

▶ **Definition 6.** *Let $G = (V, E)$ be a k-terminal graph with terminal labeling function $\mathcal{T}$ and terminal set T, r be a positive integer, and $v \in V$. We define $B_r^G(v)$ (and simply write $B_r(v)$ if G is clear from the context) to be the k-terminal subgraph of G that has labeling function $\mathcal{T}$ and is induced by $T \cup V'$, where V' is the set of all vertices reachable by v via a path of length at most r whose internal vertices are all in $V \setminus T$.*

▶ **Definition 7.** *Let $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$ be two k-terminal graphs, with terminal labeling functions $\mathcal{T}_1, \mathcal{T}_2$ and terminal sets T_1, T_2. For a non-negative integer r, we will say that $v_1 \in V_1$ is* r-similar *to $v_2 \in V_2$, if there exists a terminal-respecting isomorphism from $B_r^{G_1}(v_1)$ to $B_r^{G_2}(v_2)$ that maps v_1 to v_2.*

Note that in the above definition, G_1, G_2 may be the same graph. It is not hard to see that r-similarity is an equivalence relation on the vertices of $V_1 \cup V_2$. Definition 7 allows us to set $r = 0$, in which case we are testing if the graphs induced by $T \cup \{v_1\}$ and $T \cup \{v_2\}$ are isomorphic. Let us also make the following easy observation that decreasing r cannot make two similar vertices dissimilar.

▶ **Observation 8.** *Let G_1, G_2 be two graphs as in Definition 7 and $v_1 \in V(G_1), v_2 \in V(G_2)$ be two vertices which are r-similar. Then, for all non-negative integers $r' \leq r$, v_1 is r'-similar to v_2.*

The main lemma of this section is then the following.

▶ **Lemma 9.** *Let q, k be positive integers and set $r = 2^q - 1$. Let G_1, G_2 be two k-terminal graphs that contain some non-terminal vertices, with labeling functions $\mathcal{T}_1, \mathcal{T}_2$ and terminal sets T_1, T_2. Suppose that there exists a bijective mapping $f : V(G_1) \to V(G_2)$ such that (i) for all $i \in [k]$ we have $f(\mathcal{T}_1(i)) = \mathcal{T}_2(i)$ (ii) for all non-terminal vertices $v \in V(G_1) \setminus T_1$ we have that v is r-similar to $f(v) \in V(G_2) \setminus T_2$. Then, for all FO sentences ϕ with at most q quantifiers we have $G_1 \models \phi$ if and only if $G_2 \models \phi$.*

4 Simplification Operations on Path Decompositions

In this section we present the main technical ingredients of our algorithm. In Section 4.1 we show how we can rank the vertices to bound the number of higher-rank neighbors of any vertex; in Section 4.2 we use the pigeonhole principle to show that for sufficiently long path

decompositions we can always find long isomorphic blocks; and in Section 4.3 we describe the rewiring operation we will use in these blocks and show that it does not affect the validity of any formula and that it produces identical parts, in the sense of Lemma 5.

4.1 Normalized Path Decompositions

▶ **Definition 10.** *A* ranked *path decomposition of a graph $G = (V, E)$ where all bags have size at most p is a path decomposition together with a ranking function $\rho : V \to \mathbb{N}$ that has the property that no bag B_i of the decomposition contains two vertices $u, v \in B_i$ for which $\rho(u) = \rho(v)$.*

▶ **Lemma 11.** *Given a graph $G = (V, E)$ and a path decomposition of G where each bag contains at most p vertices, it is possible in polynomial time to convert it into a ranked path decomposition with a ranking function $\rho : V \to [8p]$ and the property that for each $i < j$ with $i, j \in [8p]$ we have that for every vertex v with $\rho(v) = i$, there exist at most two vertices u_1, u_2 with $\rho(u_1) = \rho(u_2) = j$ that appear in a bag together with v. Furthermore, the produced decomposition has the property that each bag contains at least one vertex that does not appear in the previous bag.*

We will call the ranked path decompositions that satisfy the properties of the decompositions produced by Lemma 11 *normalized* path decompositions. Since such a decomposition can always be obtained without using too many ranks in the ranking function, we will from now on focus on the case where we are given a normalized decomposition. Furthermore, we will usually use p to denote the maximum rank, rather than the pathwidth; this will not have a significant impact as, according to Lemma 11 we can make sure that the two are at most a constant factor apart.

4.2 Finding Isomorphic Bag Intervals

As mentioned, our high-level strategy will be to identify parts of the graph which are locally isomorphic, so that we can apply Lemma 9 to obtain a simpler (less well-connected) graph, and eventually Lemma 5 in order to decrease the size of the graph. In order to identify such parts, we first define what it means for two blocks of bags of a given decomposition to be isomorphic.

▶ **Definition 12.** *Let $G = (V, E)$ be a k-terminal graph with terminal set T, and $B_1, \ldots, B_\ell$ a ranked path decomposition of G with ranking function $\rho : V \to [p]$. Let s_1, t_1, s_2, t_2 be positive integers with $s_1 \leq t_1$ and $t_1 - s_1 = t_2 - s_2$. We define the* block *corresponding to $[s_1, t_1]$ and write $\mathcal{B}(s_1, t_1)$ to be $\{B_j \mid j \in [s_1, t_1]\}$. We say that $\mathcal{B}(s_1, t_1)$ is* block-isomorphic *to $\mathcal{B}(s_2, t_2)$ if*

1. *For each $j \in [s_1, t_1]$ and rank i we have $|\rho^{-1}(i) \cap B_j| = |\rho^{-1}(i) \cap B_{s_2+(j-s_1)}|$.*
2. *For each $j \in [s_1 + 1, t_1]$ and rank i we have that B_j contains a vertex v with $\rho(v) = i$ such that $v \notin B_{j-1}$ if and only if $B_{s_2+(j-s_1)}$ contains a vertex v' with $\rho(v') = i$ such that $v' \notin B_{s_2+(j-s_1)-1}$.*
3. *The following mapping f is a terminal-respecting isomorphism from $G[T \cup (\bigcup_{j\in[s_1,t_1]} B_j)]$ to $G[T \cup (\bigcup_{j\in[s_2,t_2]} B_j)]$. For each $v \in \bigcup_{j\in[s_1,t_1]} B_j$ we let j_v be the minimum index in $[s_1, t_1]$ such that $v \in B_{j_v}$ and define $f(v)$ to be the (unique) vertex of $B_{s_2+(j_v-s_1)}$ such that $\rho(v) = \rho(f(v))$.*

▶ **Definition 13.** *Let $L \geq 0$ and G, k, T, ℓ, ρ as in Definition 12. For positive integers $s_1, s_2 \in [\ell - L]$ we will write $s_1 \approx_L s_2$ to indicate that $\mathcal{B}(s_1, s_1 + L)$ is block-isomorphic to $\mathcal{B}(s_2, s_2 + L)$.*

Note that the isomorphism of Definition 12 is well-defined, because according to the first condition, if B_j contains a vertex of rank i, then so does $B_{s_2+(j_v-s_1)}$, and such a vertex is unique by the definition of ranked path decomposition. According to Definition 12, two blocks of bags are isomorphic only if the subgraphs induced by the bags they contain (and the terminals of G) are isomorphic under the trivial mapping function which maps each vertex of a bag from one block to the vertex of the corresponding bag of the other block that has the same rank. Despite the fact that this restricts the class of isomorphisms we may consider quite a bit, the block-isomorphism relation is an equivalence relation that does not have too many equivalence classes. In particular, we have the following.

▶ **Lemma 14.** *Let $L \geq 0$, $G = (V, E)$ be a k-terminal graph with terminal set T, and $B_1, \ldots, B_\ell$ a ranked path decomposition of G with ranking function $\rho : V \to [p]$. Let t_1, t_2 be integers such that for all $j, j' \in [t_1, t_2]$ we have $B_j \cap T = B_{j'} \cap T$. Then, the relation $\approx_L$ is an equivalence relation on the set $[t_1, t_2 - L]$ with at most $2^{(L+1)(p^2+2p+kp)}$ equivalence classes.*

Proof. The fact that $\approx_L$ is an equivalence relation is easy to see, as terminal-respecting isomorphisms can be composed to show transitivity. The interesting part of the lemma is then the bound on the number of equivalence classes. We prove this by induction on L.

For $L = 0$, we claim there are at most 2^{p+p^2+kp} equivalence classes of bags (in this case, each block consists of a single bag). Indeed, in order to decide if $\mathcal{B}(s_1, s_1) = \{B_{s_1}\}$ and $\mathcal{B}(s_2, s_2) = \{B_{s_2}\}$ are block-isomorphic, we first need to check if B_{s_1}, B_{s_2} contain vertices of the same ranks, and for this there are 2^p equivalence classes. If they do, then we must check, for each $i_1, i_2 \in [p]$ if the vertices of ranks i_1, i_2 in each of B_{s_1}, B_{s_2} are adjacent, and for this we have $2^{\binom{p}{2}} < 2^{p^2}$ equivalence classes. Finally, since the isomorphism has to be terminal-respecting, we have to check for each rank $i \in [p]$ if the vertex of rank i in each of B_{s_1}, B_{s_2} is connected to each of the k terminals, which gives at most kp edges which may or may not exist. (Note that we have to check these edges, even though the two bags contain the same terminals, because terminal-respecting isomorphisms must also preserve the edges towards terminals outside the bag). Overall we have at most $2^{p+p^2+kp} < 2^{p^2+2p+kp}$ choices. If we make the same choices for two bags, the two bags are block-isomorphic, hence we have bounded the number of equivalence classes for $L = 0$.

Suppose now that $L > 0$ and we have shown that the number of equivalence classes of $\approx_{L-1}$ is at most $2^{L(p^2+2p+kp)}$. Consider two indices s_1, s_2 for which we want to check if $s_1 \approx_L s_2$. We claim that for this it is sufficient to have $s_1 \approx_{L-1} s_2$ and to satisfy certain conditions for the bags B_{s_1+L}, B_{s_2+L} for which we have at most $2^{p^2+2p+kp}$ choices. More precisely, for each rank i, we have three possibilities for the bag B_{s_1+L}: either the bag contains no vertex of rank i; or it contains a vertex of rank i that also appears in B_{s_1+L-1}; or it contains a vertex of rank i that appears for the first time in B_{s_1+L} (and hence this vertex is a non-terminal). Suppose now that for each rank i, the bags B_{s_1+L}, B_{s_2+L} agree on the choice of which of these three possibilities holds (there are $3^p < 2^{2p}$ possibilities in total), and furthermore, that the graphs induced by $B_{s_1+L} \cup T$ and $B_{s_2+L} \cup T$ are isomorphic for the natural terminal-respecting isomorphism that matches vertices of the same rank (at most 2^{p^2+kp} possibilities). Then, if $s_1 \approx_{L-1} s_2$, we now have $s_1 \approx_L s_2$. Therefore, each of the $2^{L(p^2+2p+kp)}$ equivalence classes of $\approx_{L-1}$ has been refined into at most 2^{2p+p^2+kp} equivalence classes, giving that the number of equivalence classes of $\approx_L$ is at most $2^{(L+1)(p^2+2p+kp)}$, as desired. ◀

Now that we know that block-isomorphism has a bounded number of equivalence classes (if k, p, L are bounded), we can try to look for "copies" of the same block in our path decomposition. We observe the following lemma.

▶ **Lemma 15.** *Let L be a non-negative integer, $G = (V, E)$ be a k-terminal graph with terminal set T, and $B_1, \ldots, B_\ell$ a ranked path decomposition of G with ranking function $\rho : V \to [p]$. We define $R = (L+1)(2^{(L+1)(p^2+2p+kp)} + 1)$. Let t_1, t_2 be integers such that for all $j, j' \in [t_1, t_2]$ we have $B_j \cap T = B_{j'} \cap T$. Then, for every $s \in [t_1, t_2 - R]$, there exist $s_1, s_2 \in [s, s + R - (L+1)]$ such that $s_1 + L < s_2$ and $s_1 \approx_L s_2$.*

What we have shown so far is that if we take sufficiently many (at least R) consecutive bags in our decomposition, we will find two blocks of length (roughly) L which are block-isomorphic. Let us now move a step further.

▶ **Lemma 16.** *Let L be a non-negative integer, $G = (V, E)$ be a k-terminal graph with terminal set T, $B_1, \ldots, B_\ell$ a ranked path decomposition of G with ranking function $\rho : V \to [p]$, and t_1, t_2 as defined in Lemma 15. Let q, R be positive integers. We define $R^* = (R+1)(q2^{(R+1)(p^2+2p+kp)} + 1)$. Then, for every $s \in [t_1, t_2 - R^*]$ there exist $q+1$ distinct $s_1, s_2, \ldots, s_{q+1} \in [s, s + R^* - (R+1)]$, such that for any two distinct s_i, s_j with $i, j \in [q+1]$ we have $|s_i - s_j| > R$ and $s_i \approx_R s_j$.*

Note that Lemma 15 and Lemma 16 are non-vacuous only if we find a long enough interval where all bags contain the same terminals, that is if $t_2 - t_1 \geq R$ or $t_2 - t_1 \geq R^*$ respectively. We will take this into account when we use these lemmas in the next section.

At this point we are almost done in our search for appropriate isomorphic parts of the graph. What we have proved is that, if we fix some appropriate radius L, there is some larger radius R^* (double-exponential in L), such that if we look at any interval of the path decomposition of length R^*, we will be able to find $q+1$ isomorphic R-blocks, which are long enough to guarantee the existence of two isomorphic L-blocks inside them. What remains is to ask what value of L will be appropriate for our purposes. Ideally, we would like to calculate a value L that will allow us to preserve the balls around vertices for a suitable radius and apply Lemma 9. However, we can only give such a bound if we know that vertices of our path decomposition do not appear in too many bags.

▶ **Lemma 17.** *Let $G = (V, E)$ be a k-terminal graph with terminal set T and $B_1, \ldots, B_\ell$ a ranked path decomposition of G with the additional property that any non-terminal vertex appears in at most Δ bags of the decomposition. Then, for each $r \geq 0$ and for each non-terminal vertex v, if $v \in B_j$, then each non-terminal vertex of $B_r(v)$ is contained in a bag of $\mathcal{B}(j - r\Delta, j + r\Delta)$.*

4.3 Rewiring Operation

The goal of Section 4.2 was to present the basic tools which will allow us to find isomorphic parts of the input graph. Ideally, we would then like to use Lemma 5 and delete one such part. However, this is in general not possible, as the isomorphism guaranteed by the lemmas of Section 4.2 is not sufficient to obtain identical sets, in the sense of Definition 4. What we need to do, then, is to edit the graph in a way that does not affect the validity of any FO formula with q quantifiers but leverages the isomorphic parts we have found to construct $q+1$ identical parts on which Lemma 5 can be applied. We now present the basic edit operation which will allow us to achieve this for appropriate parameters.

▶ **Definition 18** (Rewiring). *Let $G = (V, E)$ be a k-terminal graph for which we are given a ranked path decomposition with ranking function $\rho : V \to [p]$. Let B_{s_1}, B_{s_2} be two bags of this decomposition, for $s_1 < s_2$. We define the* rewiring *operation on (s_1, s_2) as follows: (i) for every non-terminal vertex $v \in B_{s_1}$ which is adjacent to a non-terminal vertex $u \in$*

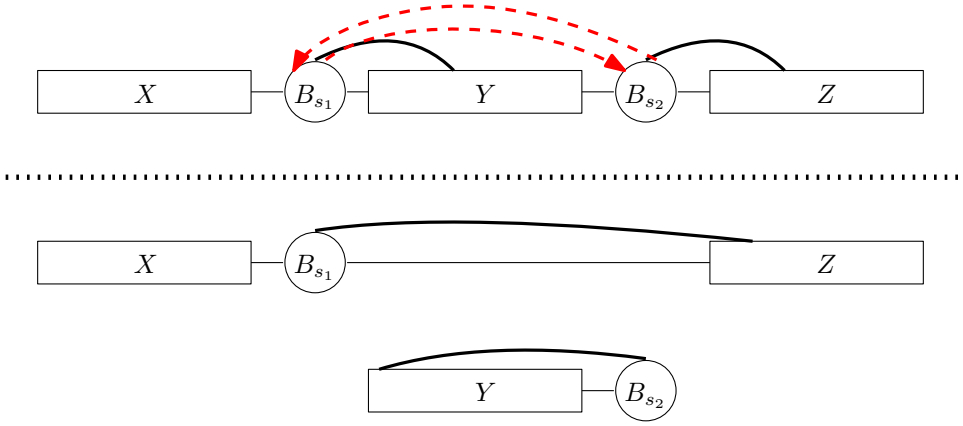


Figure 1 The rewiring operation of Definition 18. Edges from Y to B_{s_1} are rerouted towards B_{s_2}, while edges from Z to B_{s_2} are rerouted towards B_{s_1}. Edges incident on terminals are not modified.

$B_j \setminus (B_{s_1} \cup B_{s_2})$ for some $j \in [s_1 + 1, s_2 - 1]$ we delete the edge uv and add to the graph the edge uv', where $v' \in B_{s_2}$ and $\rho(v) = \rho(v')$, if such a v' exists (ii) for every non-terminal vertex $v \in B_{s_2}$ which is adjacent to a non-terminal vertex $u \in B_j \setminus B_{s_2}$ for some $j > s_2$, we delete the edge uv and add to the graph the edge uv', where $v' \in B_{s_1}$ and $\rho(v) = \rho(v')$, if such a v' exists.

Some explanations are in order regarding the motivation of the rewiring operation. We refer the reader to Figure 1. From standard properties of path decompositions, B_{s_1}, B_{s_2} are separators which break down the graph into three parts, call them X, Y, Z, which are respectively vertices which appear in a bag before B_{s_1}, between B_{s_1} and B_{s_2}, and after B_{s_2}. The rewiring operation leaves all edges incident on terminals and all edges incident on X unchanged. What it does is replace edges from Y to B_{s_1} with edges from Y to B_{s_2} and edges from Z to B_{s_2} with edges from Z to B_{s_1}. Intuitively, what this is meant to achieve is to break down the long path-like structure $X - B_{s_1} - Y - B_{s_2} - Z$ into the shorter path-like structure $X - B_{s_1} - Z$ and the ring-like structure $Y - B_{s_2}$. The idea here is that the $Y - B_{s_2}$ part is "disconnected" from the rest of the graph (more precisely, the k terminals separate this part from the rest of the graph, since terminals are not modified by this operation). Hence if we find many isomorphic such parts, they will also be identical in the sense of Definition 4, allowing us to delete one using Lemma 5. This argument is made precise in Lemma 20.

Before we do all these things, however, we need to be sure that the rewiring operation did not affect the validity of any FO formula of at most q quantifiers. The main claim now is that if s_1, s_2 are sufficiently far apart, we have a bound on the number of occurrences of non-terminal vertices in bags, and a sufficiently large block around B_{s_1} is block-isomorphic to a sufficiently large block around B_{s_2}, then the ball of radius $r = 2^q - 1$ around any vertex has remained unchanged. Hence, we can invoke Lemma 9 to conclude that the rewired graph is indistinguishable from the original graph for FO formulas with q quantifiers. Our main tool in proving this will be the following lemma.

▶ **Lemma 19.** *Let $G = (V, E)$ be a k-terminal graph with terminal set T, $B_1, \ldots, B_\ell$ a ranked path decomposition of G with ranking function $\rho : V \to [p]$ with the additional property that any non-terminal vertex appears in at most Δ bags of the decomposition. Fix an integer $q \geq 0$ and let $L = \Delta(2^q - 1)$. Let s_1, s_2 be such that (i) we have $s_1 > 4L$, $s_2 < \ell - 4L$, $s_2 - s_1 > 6L$ (ii) $\mathcal{B}(s_1 - L, s_1 + L)$ is block-isomorphic to $\mathcal{B}(s_2 - L, s_2 + L)$. Let G' be the graph obtained by applying the rewiring operation on (s_1, s_2). Then, for all FO formulas ϕ with at most q quantifiers we have $G \models \phi$ if and only if $G' \models \phi$.*

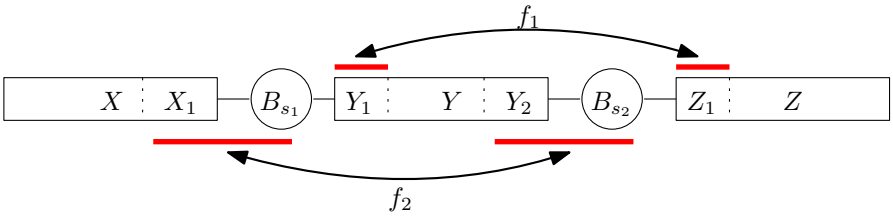


Figure 2 Schematic view of the two mappings of the proof of Lemma 19.

Finally, we argue that if we apply the rewiring operation on two block-isomorphic parts, then we obtain two parts of the graph which are identical in the sense of Definition 4. This will allow us to delete a part of the graph, once we gather sufficiently many identical parts.

▶ **Lemma 20.** *Let R be a positive integer, $G = (V, E)$ be a k-terminal graph with terminal set T, $B_1, \ldots, B_\ell$ a ranked path decomposition of G with ranking function $\rho : V \to [p]$ with the property that no non-terminal vertex appears in more than R bags. Let s_1, s_2 be positive integers such that $s_2 - s_1 > 4R$ and $\mathcal{B}(s_1, s_1 + R)$ is block-isomorphic to $\mathcal{B}(s_2, s_2 + R)$. Let $j_1, j_2 \in [0, R-1]$ with $j_1 < j_2$ and let G' be the graph obtained after applying the rewiring operation on $(s_1 + j_1, s_1 + j_2)$ and also on $(s_2 + j_1, s_2 + j_2)$. Let Y_1 be the set of vertices that appear in a bag with index in $[s_1 + j_1 + 1, s_1 + j_2 - 1]$, but not in $B_{s_1+j_1} \cup B_{s_1+j_2}$. Similarly, let Y_2 be the set of vertices that appear in a bag with index in $[s_2 + j_1 + 1, s_2 + j_2 - 1]$, but not in $B_{s_2+j_1} \cup B_{s_2+j_2}$. Then $(Y_1 \cup B_{s_1+j_2}) \setminus T$ is identical to $(Y_2 \cup B_{s_2+j_2}) \setminus T$.*

5 Putting Everything Together

We are now ready to put everything together to obtain our model checking algorithm for FO logic. We formulate a procedure which can either simplify the graph in a way that does not affect the validity of the given formula (or any formula with the same number of quantifiers), or certify that the graph has bounded degree, and hence we can use known algorithms with an elementary dependence on the formula. On a high level, we take as input a graph G, a path decomposition of G, and a formula ϕ with q quantifiers and we will do the following:

1. Use Lemma 11 to normalize the decomposition and obtain a ranking of the vertices. In this ranking, vertices of rank 1 appear in a constant number of bags. We would like to extend this so that every vertex appears in a bounded number of bags. In the remainder we will use the number of bags a vertex appears in as a proxy bound for its degree.
2. Define a function $\Delta(i)$ which defines an acceptable bound for the number of occurrences in distinct bags for a vertex of rank i. This function will be a tower of exponentials of height roughly $2i$, but this is acceptable, since the maximum rank is upper-bounded by a function of the pathwidth, which we consider to be an absolute constant.
3. Examine the graph and check if any vertex of rank i appears in more than $\Delta(i)$ bags. If this is not the case, we can bound the maximum degree of the graph, and we are done.
4. Otherwise, find a vertex v of minimum rank i that appears more than $\Delta(i)$ times. Find a section of the decomposition where v appears, and where all bags contain the same vertices of rank higher than i (if $\Delta(i)$ is large, we can find such a section that is quite long). We label as terminals the vertices of the first and last bag of the section, and the vertices of rank at least i appearing in the section.
5. Now, the remaining vertices of the section appear a bounded (by $\Delta(i-1)$) number of times, and are separated from the rest of the graph by $k = O(p)$ terminals. However, they are quite numerous, as we assumed that v appears too many times. Therefore,

we can invoke the machinery of Section 4.2 to find some isomorphic parts. Note that it is important that vertices of rank at least i (which are now terminals) are common throughout the section, which allows us to invoke Lemmas 15 and 16.

6. Having found many isomorphic parts, we use the tools of Section 4.3 to perform the rewiring operation that will produce $q+1$ identical parts, of which we can remove one. We then "undo" the operation on the remaining parts, and obtain a smaller graph, where v appears in fewer bags, without changing whether ϕ is satisfied.

▶ **Definition 21.** *Let p, q be positive integers. We define the function $\Delta_{p,q}(i)$ as follows: $\Delta_{p,q}(1) = 3p$ and $\Delta_{p,q}(i+1) = 2^{2^{\Delta_{p,q}(i)\cdot 2^{20qp^2}}}$. When p, q are clear from the context, we will write $\Delta(i)$ to denote $\Delta_{p,q}(i)$.*

▶ **Observation 22.** *For each fixed p, i, the function $\Delta_{p,q}(i)$ is an elementary function of q. Furthermore, $\Delta_{p,q}(i)$ is a strictly increasing function of i.*

▶ **Lemma 23.** *There is an algorithm that takes as input an FO formula ϕ with q quantifiers, an n-vertex graph $G = (V, E)$, a normalized ranked path decomposition of G with ranking function $\rho : V \to [p]$, such that G contains a vertex that appears in at least $3p \cdot \Delta(p)$ bags of the decomposition. Then, the algorithm runs in polynomial time and outputs a smaller graph G' and a normalized ranked path decomposition of G' with the same ranking function ρ, such that $G \models \phi$ if and only if $G' \models \phi$.*

▶ **Theorem 24.** *For every fixed p, model checking a formula ϕ on a graph G with pathwidth p can be performed in time $f(\phi)|G|^{O(1)}$, where f is an elementary function.*

6 Conclusions

We have shown that FO model checking for graphs of bounded pathwidth has a complexity behavior that is in sharp contrast with both MSO logic for the same class of graphs and the complexity of FO logic on graphs of bounded treewidth. It may be interesting to improve upon our result by noting that our algorithm's dependence on the pathwidth is a tower of exponentials whose height is $O(\mathrm{pw})$, where the hidden constant is roughly 16. This is in contrast with the meta-theorem of [14], where the height of the tower is roughly equal to the tree-depth. Can the height of the tower in our case can be made $\mathrm{pw} + O(1)$, or is FO model checking on bounded pathwidth truly harder than for bounded tree-depth?

Another interesting research direction would be to explore parameters which lie between pathwidth and treewidth to attempt to trace the frontier of where the arguments of this paper break down. One idea would be to consider tree decompositions with a bounded number of leaf bags, which would generalize pathwidth, and see if, as long as the number of leaf bag and the width of the decomposition is an absolute constant, we can hope for an elementary dependence on the formula for FO model checking.

References

1 Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *J. Algorithms*, 12(2):308–340, 1991. doi:10.1016/0196-6774(91)90006-K.

2 Rémy Belmonte, Eun Jung Kim, Michael Lampis, Valia Mitsou, and Yota Otachi. Grundy distinguishes treewidth from pathwidth. In *ESA*, volume 173 of *LIPIcs*, pages 14:1–14:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

3 Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (meta) kernelization. *J. ACM*, 63(5):44:1–44:69, 2016.

4 Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width I: tractable FO model checking. In *FOCS*, pages 601–612. IEEE, 2020.

5 Bruno Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

6 Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000.

7 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

8 Anuj Dawar, Martin Grohe, Stephan Kreutzer, and Nicole Schweikardt. Approximation schemes for first-order definable optimisation problems. In *LICS*, pages 411–420. IEEE Computer Society, 2006.

9 Zdenek Dvořák, Daniel Král, and Robin Thomas. Testing first-order properties for subclasses of sparse graphs. *J. ACM*, 60(5):36:1–36:24, 2013.

10 Eduard Eiben, Robert Ganian, and Stefan Szeider. Meta-kernelization using well-structured modulators. *Discret. Appl. Math.*, 248:153–167, 2018.

11 Markus Frick. Generalized model-checking over locally tree-decomposable classes. *Theory Comput. Syst.*, 37(1):157–191, 2004.

12 Markus Frick and Martin Grohe. Deciding first-order properties of locally tree-decomposable structures. *J. ACM*, 48(6):1184–1206, 2001.

13 Markus Frick and Martin Grohe. The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Log.*, 130(1-3):3–31, 2004.

14 Jakub Gajarský and Petr Hliněný. Kernelizing MSO properties of trees of fixed height, and some consequences. *Log. Methods Comput. Sci.*, 11(1), 2015.

15 Jakub Gajarský, Petr Hlinený, Daniel Lokshtanov, Jan Obdrzálek, Sebastian Ordyniak, M. S. Ramanujan, and Saket Saurabh. FO model checking on posets of bounded width. In *FOCS*, pages 963–974. IEEE Computer Society, 2015.

16 Robert Ganian. Improving vertex cover as a graph parameter. *Discret. Math. Theor. Comput. Sci.*, 17(2):77–100, 2015.

17 Robert Ganian, Petr Hliněný, Jaroslav Nešetřil, Jan Obdržálek, and Patrice Ossona de Mendez. Shrub-depth: Capturing height of dense graphs. *Log. Methods Comput. Sci.*, 15(1), 2019.

18 Robert Ganian and Jan Obdržálek. Expanding the expressive power of monadic second-order logic on restricted graph classes. In *IWOCA*, volume 8288 of *Lecture Notes in Computer Science*, pages 164–177. Springer, 2013.

19 Robert Ganian, Friedrich Slivovsky, and Stefan Szeider. Meta-kernelization with structural parameters. *J. Comput. Syst. Sci.*, 82(2):333–346, 2016.

20 Martin Grohe and Stephan Kreutzer. Methods for algorithmic meta theorems. *Model Theoretic Methods in Finite Combinatorics*, 558:181–206, 2011.

21 Dusan Knop, Martin Koutecký, Tomás Masarík, and Tomás Toufar. Simplified algorithmic metatheorems beyond MSO: treewidth and neighborhood diversity. *Log. Methods Comput. Sci.*, 15(4), 2019.

22 Dusan Knop, Tomás Masarík, and Tomás Toufar. Parameterized complexity of fair vertex evaluation problems. In *MFCS*, volume 138 of *LIPIcs*, pages 33:1–33:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

23 Michael Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2012. doi:10.1007/s00453-011-9554-x.

24 Michael Lampis. Model checking lower bounds for simple graphs. *Log. Methods Comput. Sci.*, 10(1), 2014.

25 Michael Lampis and Valia Mitsou. Fine-grained meta-theorems for vertex integrity. In *ISAAC*, volume 212 of *LIPIcs*, pages 34:1–34:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

26 Michal Pilipczuk. Problems parameterized by treewidth tractable in single exponential time: A logical approach. In *MFCS*, volume 6907 of *Lecture Notes in Computer Science*, pages 520–531. Springer, 2011.

27 Detlef Seese. Linear time computable problems and first-order descriptions. *Math. Struct. Comput. Sci.*, 6(6):505–526, 1996.

28 Stefan Szeider. Monadic second order logic on graphs with local cardinality constraints. *ACM Trans. Comput. Log.*, 12(2):12:1–12:21, 2011.

Witnessed Symmetric Choice and Interpretations in Fixed-Point Logic with Counting

Moritz Lichter
TU Darmstadt, Germany

Abstract

At the core of the quest for a logic for PTIME is a mismatch between algorithms making arbitrary choices and isomorphism-invariant logics. One approach to tackle this problem is witnessed symmetric choice. It allows for choices from definable orbits certified by definable witnessing automorphisms.

We consider the extension of fixed-point logic with counting (IFPC) with witnessed symmetric choice (IFPC+WSC) and a further extension with an interpretation operator (IFPC+WSC+I). The latter operator evaluates a subformula in the structure defined by an interpretation. When similarly extending pure fixed-point logic (IFP), IFP+WSC+I simulates counting which IFP+WSC fails to do. For IFPC+WSC, it is unknown whether the interpretation operator increases expressiveness and thus allows studying the relation between WSC and interpretations beyond counting.

In this paper, we separate IFPC+WSC from IFPC+WSC+I by showing that IFPC+WSC is not closed under FO-interpretations. By the same argument, we answer an open question of Dawar and Richerby regarding non-witnessed symmetric choice in IFP. Additionally, we prove that nesting WSC-operators increases the expressiveness using the so-called CFI graphs. We show that if IFPC+WSC+I canonizes a particular class of base graphs, then it also canonizes the corresponding CFI graphs. This differs from various other logics, where CFI graphs provide difficult instances.

2012 ACM Subject Classification Theory of computation → Finite Model Theory; Theory of computation → Complexity theory and logic

Keywords and phrases witnessed symmetric choice, interpretation, fixed-point logic, counting, CFI graphs, logic for PTime

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.133

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: https://arxiv.org/abs/2210.07869 [30]

Funding This work received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (EngageS: agreement No. 820148).

1 Introduction

The quest for a logic for PTIME is one of the prominent open questions in finite model theory [6, 19]. It asks whether there is a logic defining exactly all polynomial-time decidable properties of finite structures. While Fagin's theorem [14] initiated descriptive complexity theory by showing that there is a logic capturing NPTIME, the question for PTIME is still open. One problem at the core of the question is a mismatch between logics and algorithms. For algorithms, it is common to make arbitrary choices as long as the output is still isomorphism-invariant. In general, it is undecidable whether an algorithm is isomorphism-invariant. Showing this is usually part of the proof that the algorithm is correct. On the other hand, every reasonable logic is required to be isomorphism-invariant by design [22, 13], so in contrast to algorithms we cannot define something non-isomorphism-invariant. That is, a logic has to enforce isomorphism-invariance syntactically and it is generally not clear how algorithms making choices can be implemented in a logic.

For totally ordered structures, inflationary fixed-point logic (IFP) captures Ptime due to the Immerman-Vardi Theorem [27]. On ordered structures, no arbitrary choices are needed and the total order is used to "choose" the unique minimal element. Thus, the lack of making choices is crucial on unordered structures. We therefore would like to support choices in a logic while still guaranteeing isomorphism-invariance. There are logics in which arbitrary choices can be made [2, 10], but for these it is undecidable whether a formula is isomorphism-invariant [2]. In particular, such logics fail to be reasonable in the sense of Gurevich [22]. Similarly, when extending structures by an arbitrary order it is undecidable whether a formula is order-invariant, i.e., it evaluates equally for all such orders (see [16]).

One approach to overcome the lack of choices in logics is to support a restricted form of choice. If only choices from definable orbits (of the automorphism group of the input structure) are allowed, that is, from sets of definable objects related by an automorphism of the input structure, the output is guaranteed to be still isomorphism-invariant [9, 15]. This form of choice is called *symmetric choice* (SC). However, it is unknown whether orbits can be computed in Ptime. So it is unknown whether a logic with symmetric choice can be evaluated in Ptime because during the evaluation we have to verify that the choice-sets are indeed orbits. This is solved by handing over the obligation to check whether the choice-sets are orbits from the evaluation to the formulas themselves. To make a choice, not only a choice-set but also a set of witnessing automorphisms has to be defined. These automorphisms certify that the choice-set is indeed an orbit in the following way: For every pair of elements a and b in the choice-set, an automorphism mapping a to b has to be provided. This condition guarantees evaluation in Ptime. We call this restricted form of choice *witnessed symmetric choice* (WSC).

Besides witnessed symmetric choice, other operators were proposed to extend the expressiveness of logics not capturing Ptime including a counting operator (see [35]) and an operator based on logical interpretations [15]. It was shown that witnessed symmetric choice increases the expressiveness of IFP [15] and that counting operators increase the expressiveness of IFP and the logic of Choiceless Polynomial Time (CPT) [4] (one usually refers with CPT to its extension with counting, which we also do from now). However, for the combination of counting and choices not much is known. In this paper, we investigate the relation of counting, witnessed symmetric choice, and interpretations to better understand their expressive power.

Extending IFP with symmetric and witnessed symmetric choice was first studied by Gire and Hoang [15]. They extend IFP with symmetric choice (which we denote IFP+SC) and which witnessed symmetric choice (which we denote IFP+WSC). The authors show that IFP+WSC distinguishes CFI graphs over ordered base graphs, which IFP (and even fixed-point logic with counting IFPC) fails to do [5]. Afterwards IFP+SC was studied by Dawar and Richerby [9]. They allowed for nested symmetric choice operators, proved that parameters of choice operators increase the expressiveness, showed that nested symmetric choice operators are more expressive than a single one, and conjectured that with additional nesting depth the expressiveness increases.

Recently, extending CPT with witnessed symmetric choice (CPT+WSC) was studied by Lichter and Schweitzer [32]. CPT+WSC has the interesting property that a CPT+WSC-definable isomorphism test on a class of structures implies a CPT+WSC-definable canonization for this class. Canonization is the task of defining an isomorphic but totally ordered copy. The only requirement is that the class is closed under individualization, so under assigning unique colors to vertices. This is often unproblematic [28, 33]. Individualization is natural in the context of choices because a choice is, in some sense, an individualization. The concept of canonization is essential in the quest for a logic for Ptime. It provides the routinely employed

approach to capture Ptime on a class of structures: Define canonization, obtain isomorphic and ordered structures, and apply the Immerman-Vardi Theorem (e.g. [42, 20, 21, 31]). While in CPT+WSC defining isomorphism implies canonization, we do not know whether the same holds for CPT or whether CPT+WSC is more expressive than CPT. Proving this requires separating CPT from Ptime, which has been open for a long time.

(Witnessed) symmetric choice has the drawback that it can only choose from orbits of the input structure. This structure might have complicated orbits that we cannot define or witness in the logic. However, there could be a reduction to a different structure with easier orbits exploitable by witnessed symmetric choice. For logics, the natural concept of a reduction is an interpretation, i.e., defining a structure in terms of another one. Interpretations are in some sense incompatible with (witnessed) symmetric choice because we always have to choose from orbits of the input structure. Orbits of the interpreted structure are always unions of orbits of the input structure, i.e., an interpretation may add more automorphisms but never removes one. To exploit a combination of choices and interpretations, Gire and Hoang proposed an interpretation operator [15]. It evaluates a formula in the image of an interpretation. For logics closed under interpretations (e.g. IFP, IFPC, and CPT) such an interpretation operator does not increase expressiveness. However, for the extension with witnessed symmetric choice this is different: IFP+WSC is less expressive than the extension of IFP+WSC with the interpretation operator. The interpretation operator in combination with WSC simulates counting, which for WSC alone is indicated not to be the case [15].

We are interested in the relation between witnessed symmetric choice and the interpretation operator not specifically for IFP but more generally. Most of the existing results in [15, 9] showing that (witnessed) symmetric choice or the interpretation operator increases in some way the expressiveness of IFP are based on counting. However, counting is not the actual reason for using witnessed symmetric choice. Counting can be achieved more naturally in IFPC. Thus, it is unknown whether the interpretation operator increases expressiveness of IFPC+WSC. In CPT, it is not possible to show that witnessed symmetric choice or the interpretation operator increases expressiveness without separating CPT from Ptime [32].

Overall, a natural base logic for studying the interplay of witnessed symmetric choice and the interpretation operator is IFPC. In IFPC, separation results based on counting are not applicable. But there are known IFPC-undefinable Ptime properties, namely the already mentioned CFI query, which can be used to separate extensions of IFPC. The CFI construction assigns to a connected graph, the so-called base graph, two non-isomorphic CFI graphs: One is called even and the other one is called odd. The CFI query is to define whether a given CFI graph is even.

Results. We define the logics IFPC+WSC and IFPC+WSC+I, which extend IFPC by a fixed-point operator featuring witnessed symmetric choice and the latter additionally by an interpretation operator. We show that the interpretation operator increases expressiveness:

▶ **Theorem 1.** IFPC+WSC $<$ IFPC+WSC+I $\leq$ *Ptime*.

In particular, this separates IFPC+WSC from Ptime. Such a result does not follow from existing techniques because separating IFP+WSC from Ptime is based on counting in [15]. Moreover, we show that both IFPC+WSC and IFP+SC are not even closed under FO-interpretations. This answers an open question of Dawar and Richerby [9]. Proving Theorem 1 relies on the CFI construction and on defining the CFI query for certain classes of base graphs. To show this, we show that IFPC+WSC+I-distinguishable orbits imply an IFPC+WSC+I-definable canonization (similarly to [32]). We apply this to CFI graphs:

▶ **Theorem 2.** *If* IFPC+WSC+I *canonizes a class of colored base graphs $\mathcal{K}$ (closed under individualization), then* IFPC+WSC+I *canonizes the class of CFI graphs* $\mathsf{CFI}(\mathcal{K})$ *over* $\mathcal{K}$.

The conclusion is that for IFPC+WSC+I canonization of a class of CFI graphs is not more difficult than canonization of the corresponding class of base graphs, which is different in many other logics [5, 17, 29, 7]. However, to canonize the CFI graphs in our proof, the nesting depth of WSC-fixed-point operators and interpretation operators increases. We show that this increase is unavoidable: For $L \subseteq$ IFPC+WSC+I, we denote by WSCI(L) the IFPC+WSC+I-fragment which uses IFPC-formula-formation-rules to compose L-formulas and an additional interpretation operator nested inside a WSC-fixed-point operator.

▶ **Theorem 3.** *There is a class of base graphs $\mathcal{K}$, for which* WSCI(IFPC) *defines a canonization but does not define the CFI query for* $\mathsf{CFI}(\mathcal{K})$ *and* WSCI(WSCI(IFPC)) *canonizes* $\mathsf{CFI}(\mathcal{K})$.

Theorem 3 provides a first step towards an operator nesting hierarchy for IFPC+WSC+I.

Our Techniques. We adapt the techniques of [32] from CPT to IFPC to define a WSC-fixed-point operator. It has some small but essential differences to [15, 9]. Similar to [32] for CPT, Gurevich's canonization algorithm [23] is expressible in IFPC+WSC: It suffices to distinguish orbits of a class of individualization-closed structures to define a canonization.

To prove Theorem 2, we use the interpretation operator to show that if IFPC+WSC+I distinguishes orbits of the base graphs, then IFPC+WSC+I distinguishes also orbits of the CFI graphs and thus canonizes the CFI graphs. The CFI-graph-canonizing formula nests one WSC-fixed-point operator (for Gurevich's algorithm) and one interpretation operator (to distinguish orbits) more than the orbit-distinguishing formula of the base graphs. To show that this increase in nesting depth is necessary, we construct double CFI graphs. We start with a class of CFI graphs $\mathsf{CFI}(\mathcal{K}')$ canonized in WSCI(IFPC). We create new base graphs $\mathcal{K}$ from the $\mathsf{CFI}(\mathcal{K}')$-graphs. Applying the CFI construction once more, $\mathsf{CFI}(\mathcal{K})$ is canonized in WSCI(WSCI(IFPC)) but not in WSCI(IFPC): To define orbits of $\mathsf{CFI}(\mathcal{K})$, we have to define orbits of the base graph, for which we need to define the CFI query for $\mathsf{CFI}(\mathcal{K}')$.

To prove IFPC+WSC $<$ IFPC+WSC+I, we construct a class of asymmetric structures, i.e., structures without non-trivial automorphisms, for which isomorphism is not IFPC-definable. Because asymmetric structures have only singleton orbits, witnessed symmetric choice is not beneficial, thus IFPC+WSC = IFPC, and isomorphism is not IFPC+WSC-definable. These structures combine CFI graphs and the so-called multipedes [24], which are asymmetric and for which IFPC fails to distinguish orbits. An interpretation removes the multipedes and reduces the isomorphism problem to the ones of CFI graphs. Thus, isomorphism of this class of structures is IFPC+WSC+I-definable.

Related Work. The logic IFPC was separated from Ptime using the CFI graphs [5]. CFI graphs not only turned out to be difficult for IFPC but variants of them were also used to separate rank logic [29] and the more general linear-algebraic logic [7] from Ptime. CPT was shown to define the so-called CFI query for ordered base graphs [12] and base graphs of maximal logarithmic color class size [37]. Defining the CFI query for these graphs in CPT turned out to be comparatively more complicated than in IFP+WSC for ordered base graphs in [15]. In general, it is still open whether CPT defines the CFI query for all base graphs.

The definitions of the (witnessed) symmetric choice operator in [15, 9] differ at crucial points form the one in [32] and in this paper: The formula defining the witnessing automorphism has access to the obtained fixed-point. This is essential to implement Gurevich's

canonization algorithm but has the drawback to impose (possibly) stronger orbit conditions. Although it is unknown whether these two variants of witnessed symmetric choice have the same expressiveness, existing results [15, 9] transfer to the variant used in this paper. We expect that the results of this paper also hold for the other variant. However, proving Theorem 2 will be more effort because Gurevich's canonization algorithm cannot be used.

CPT+WSC in [32] is a three-valued logic using, beside true and false, an error marker for non-witnessed choices. This is needed for CPT because fixed-point computations in CPT do not necessarily terminate in a polynomial number of steps. Instead, computation is aborted (and orbits cannot be witnessed). For other approaches to integrate choice in first-order logic, but which are no PTIME-logic candidates, see e.g. [3, 36, 10]. We refer to [38] for an overview.

Logical interpretations can also be used to characterize CPT. The logic CPT as the same expressive power as polynomial-time interpretation logic [18]. This logic essentially applies a first-order interpretation (with an appropriate counting extension) iteratively. The iterative application of an interpretation and thereby the ability to iteratively take quotients strictly increases the expressive power. The interpretation operator that we consider evaluates a formula in the interpreted structure, that is, it evaluates an interpretation only once. By nesting this operator, constantly many interpretations can be nested. Hence, for logics closed under interpretations, the interpretation operator does not increase the expressive power.

Multipedes [24] are a class of asymmetric structures not characterized up to isomorphism in k-variable counting logic for every fixed k. Asymmetry turns multipedes to hard examples for graph isomorphism algorithms in the individualization-refinement framework [34, 1]. The size of a multipede not identifiable in k-variable counting logic is large with respect to k. There also exists a class of asymmetric graphs [8] for which the number of variables needed for identification is linear. Both classes are based on the CFI construction.

There is another remarkable but not directly-connected coincidence to lengths of resolution proofs. Resolution proofs for non-isomorphism of CFI-graphs have exponential size [40]. When adding a global symmetry rule (SRC-I), which exploits automorphisms of the formula (so akin to symmetric choice), the length becomes polynomial [39]. For asymmetric multipedes the length in the SRC-I system is still exponential [41]. But when considering the local symmetry rule (SRC-II) exploiting local automorphisms (so somewhat akin to symmetric choice after restricting to a substructure with an interpretation) the length becomes polynomial again [39].

2 Preliminaries

We set $[k] := \{1, \dots, k\}$. The i-th entry of a k-tuple $\bar{t} \in N^k$ is denoted by t_i and its length by $|\bar{t}| = k$. The set of all tuples of length at most k is $N^{\leq k}$ and the set of all finite tuples is N^*.

A *relational signature* τ is a set of relation symbols $\{R_1, \dots, R_\ell\}$ of *arities* $\mathrm{ar}(R_i)$. We use letters τ and σ for signatures. A *τ-structure* is a tuple $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \dots, R_\ell^{\mathfrak{A}})$ where $R_i^{\mathfrak{A}} \subseteq A^{\mathrm{ar}(R_i)}$. The set A is called *universe* and its elements *vertices*. We use letters $\mathfrak{A}$ and $\mathfrak{B}$ for structures, A and B for their universes, and u, v, and w for vertices. The *reduct* $\mathfrak{A} \restriction \sigma$ is the restriction of $\mathfrak{A}$ to the relations in $\sigma \subseteq \tau$. This paper considers finite structures.

A *colored graph* is an $\{E, \preceq\}$-structure $G = (V, E^G, \preceq^G)$. The relation E is the edge relation and the relation $\preceq$ is a total preorder. Its equivalence classes are the *color classes* or *colors*. We often write $G = (V, E, \preceq)$ for a colored graph. The *neighborhood* of a vertex $u \in V$ is $N_G(u)$. The *induced subgraph* of G by $W \subseteq V$ is $G[W]$. The graph G is *k-connected* if $|V| > k$ and, for every $V' \subseteq V$ of size at most $k-1$, the graph $G \setminus V'$ is connected. The *treewidth* $\mathrm{tw}(G)$ of G measures how close G is to being a tree. We omit a definition (see [11]) and only use the fact that if G is a *minor* of H, i.e., G is obtained from H by deleting vertices or edges and contracting edges, then $\mathrm{tw}(G) \leq \mathrm{tw}(H)$.

Let $\mathfrak{A}$ and $\mathfrak{B}$ be two τ-structures and $\bar{u} \in A^k$ and $\bar{v} \in B^k$. We write $(\mathfrak{A}, \bar{u}) \cong (\mathfrak{B}, \bar{v})$ if there is a *isomorphism* $\varphi\colon \mathfrak{A} \to \mathfrak{B}$ satisfying $\varphi(\bar{u}) = \bar{v}$. An *automorphism* φ of $(\mathfrak{A}, \bar{u})$ is an isomorphism $(\mathfrak{A}, \bar{u}) \to (\mathfrak{A}, \bar{u})$. We say that φ *fixes* $\bar{u}$ and write $\mathsf{Aut}((\mathfrak{A}, \bar{u}))$ for the set of all automorphisms fixing $\bar{u}$. We will use the same notation also for other objects, e.g., for automorphisms fixing relations. A *k-orbit* of $(\mathfrak{A}, \bar{u})$ is a maximal set of k-tuples $O \subseteq A^k$ such that for every $\bar{v}, \bar{w} \in O$, there is an automorphism $\varphi \in \mathsf{Aut}((\mathfrak{A}, \bar{u}))$ satisfying $\varphi(\bar{v}) = \bar{w}$.

Fixed-Point Logic with Counting. We recall fixed-point logic with counting IFPC (proposed in [26], see [35]). Let τ be a signature and $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \dots, R_\ell^{\mathfrak{A}})$ be a τ-structure. We extend τ and $\mathfrak{A}$ with counting. Set $\tau^{\#} := \tau \uplus \{\cdot, +, 0, 1\}$ and $\mathfrak{A}^{\#} := (A, R_1^{\mathfrak{A}}, \dots, R_\ell^{\mathfrak{A}}, \mathbb{N}, \cdot, +, 0, 1)$ to be the two-sorted $\tau^{\#}$-structure that is the disjoint union of $\mathfrak{A}$ and $\mathbb{N}$. IFPC$[\tau]$ is a two-sorted logic using the signature $\tau^{\#}$. *Element variables* range over vertices and *numeric variables* range over $\mathbb{N}$. The letters x, y, and z are used for element variables, ν and μ for numeric variables, and s and t for numeric terms. IFPC-formulas are built from first-order formulas, a fixed-point operator, and counting terms. The range of numeric variables needs to be bounded to ensure PTIME-evaluation: For an IFPC-formula Φ, a closed numeric IFPC-term s, a numeric variable ν, and a quantifier $Q \in \{\forall, \exists\}$, the formula $Q\nu \leq s.\, \Phi$ is an IFPC-formula. An *inflationary fixed-point operator* defines a relation R. For an IFPC$[\tau, R]$-formula Φ and variables $\bar{x}\bar{\mu}$, the fixed-point operator $[\mathsf{ifp}R\bar{x}\bar{\mu} \leq \bar{s}.\, \Phi]\,(\bar{\nu}\bar{\mu})$ is an IFPC$[\tau]$-formula. The tuple $\bar{s}$ of $|\bar{\mu}|$ closed numeric terms bounds the values of $\bar{\mu}$. The crucial element of IFPC are *counting terms*. For an IFPC-formula Φ, variables $\bar{x}\bar{\nu}$, and $|\bar{\nu}|$ closed numeric IFPC-terms $\bar{s}$, $\#\bar{x}\bar{\nu} \leq \bar{s}.\, \Phi$ is a numeric IFPC-term.

IFPC-formulas (or terms) are evaluated over $\mathfrak{A}^{\#}$. For a numeric term $s(\bar{x}\bar{\nu})$, the function $s^{\mathfrak{A}}\colon A^{|\bar{x}|} \times \mathbb{N}^{|\bar{\nu}|} \to \mathbb{N}$ maps the values for the free variables of s to the value that s takes in $\mathfrak{A}^{\#}$. Likewise, for a formula $\Phi(\bar{x}\bar{\nu})$, we write $\Phi^{\mathfrak{A}} \subseteq A^{|\bar{x}|} \times \mathbb{N}^{|\bar{\nu}|}$ for the set of values for the free variables satisfying Φ. Evaluating a counting term for a formula $\Phi(\bar{y}\bar{x}\bar{\mu}\bar{\nu})$ is defined as follows: $(\#\bar{x}\bar{\nu} \leq s.\, \Phi)^{\mathfrak{A}}(\bar{u}\bar{m}) := |\{\bar{w}\bar{n} \in A^{|\bar{x}|} \times \mathbb{N}^{|\bar{\nu}|} \mid n_i \leq s_i^{\mathfrak{A}} \text{ for all } i \in [|\bar{\nu}|], \bar{u}\bar{w}\bar{m}\bar{n} \in \Phi^{\mathfrak{A}}\}|$.

Finite Variable Counting Logic. The k-variable logic with counting $\mathcal{C}_k$ extends the k-variable fragment of first-order logic (FO) with counting quantifiers $\exists^{\geq j} x.\, \Phi$ ("at least j distinct vertices satisfy Φ"). For every fixed $n \in \mathbb{N}$, every k-variable IFPC-formula is equivalent to a $\mathcal{C}_{\mathcal{O}(k)}$-formula [35] on structures of order up to n. Let $\mathfrak{A}$ and $\mathfrak{B}$ be two τ-structures and $\bar{u} \in A^\ell$ and $\bar{v} \in B^\ell$. A logic L *distinguishes* $(\mathfrak{A}, \bar{u})$ from $(\mathfrak{B}, \bar{v})$ if there is an L-formula Φ with ℓ free variables such that $\bar{u} \in \Phi^{\mathfrak{A}}$ and $\bar{v} \notin \Phi^{\mathfrak{B}}$. Otherwise, the structures are *L-equivalent*. We write $(\mathfrak{A}, \bar{u}) \simeq_{\mathcal{C}}^k (\mathfrak{B}, \bar{v})$ if $(\mathfrak{A}, \bar{u})$ and $(\mathfrak{B}, \bar{v})$ are $\mathcal{C}_k$-equivalent. The logics $\mathcal{C}_k$ are used to prove IFPC-undefinability: Let $(\mathfrak{A}_k, \mathfrak{B}_k)$ be a sequence of finite structures for every $k \in \mathbb{N}$ such that $\mathfrak{A}_k$ has a property P but $\mathfrak{B}_k$ does not. If $\mathfrak{A}_k \simeq_{\mathcal{C}}^k \mathfrak{B}_k$ for every k, then IFPC does not define P. The logic $\mathcal{C}_k$ is characterized by the *bijective k-pebble game* [25]. The game is played on two structures $\mathfrak{A}$ and $\mathfrak{B}$ by two players called Spoiler and Duplicator. There are pebble pairs (p_i, q_i) for every $i \in [k]$. Positions in the game are tuples $(\mathfrak{A}, \bar{u}; \mathfrak{B}, \bar{v})$ for tuples $\bar{u} \in A^{\leq k}$ and $\bar{v} \in B^{\leq k}$ of the same length. A pebble p_j is placed on u_i and q_j is placed on v_i for some $j \in [k]$. No pebbles are placed initially. If $|A| \neq |B|$, then Spoiler wins. If not, Spoiler picks up a pair of pebbles (p_i, q_i). Duplicator answers with a bijection $\lambda\colon A \to B$. Spoiler places p_i on $w \in A$ and q_i on $\lambda(w) \in B$. If in the resulting position $(\mathfrak{A}, \bar{u}; \mathfrak{B}, \bar{v})$ the map $u_i \mapsto v_i$ is not a *pebble-respecting* local isomorphism $(\mathfrak{A}[\bar{u}], \bar{u}) \to (\mathfrak{B}[\bar{v}], \bar{v})$, then Spoiler wins. Otherwise, the game continues with the next round. Duplicator wins if Spoiler never wins. Spoiler/Duplicator has a *winning strategy* in position $(\mathfrak{A}, \bar{u}; \mathfrak{B}, \bar{v})$ if they can always win the game. Spoiler has a winning strategy in position $(\mathfrak{A}, \bar{u}; \mathfrak{B}, \bar{v})$ if and only if $(\mathfrak{A}, \bar{u}) \not\simeq_{\mathcal{C}}^k (\mathfrak{B}, \bar{v})$ [25].

Logical Interpretations. Interpretations define maps between structures via formulas evaluated on tuples containing vertices and numbers. We use $\bar{x}$, $\bar{y}$, and $\bar{z}$ for a tuple of element and numeric variables and $\bar{u}$ and $\bar{v}$ for a tuple of vertices and numbers in the following.

Let $\sigma := \{R_1, \dots, R_\ell\}$. A *$d$-dimensional* IFPC$[\tau,\sigma]$*-interpretation* $\Theta(\bar{z})$ with parameters $\bar{z}$ is a tuple

$$\Theta(\bar{z}) = \big(\Phi_{\mathrm{dom}}(\bar{z}\bar{x}), \Phi_{\cong}(\bar{z}\bar{x}\bar{y}), \Phi_{R_1}(\bar{z}\bar{x}_1 \dots \bar{x}_{\mathrm{ar}(R_1)}), \dots, \Phi_{R_\ell}(\bar{z}\bar{x}_1 \dots \bar{x}_{\mathrm{ar}(R_\ell)}), \bar{s}\big)$$

of IFPC$[\tau]$-formulas and a j-tuple $\bar{s}$ of closed numeric IFPC$[\tau]$-terms, where j is the number of numeric variables in $\bar{x}$. The tuples of variables $\bar{x}$, $\bar{y}$, and the $\bar{x}_i$ are all of length d and agree on whether the k-th variable is an element variable or not. Let $\mathfrak{A}$ be a τ-structure and $\bar{u} \in (A \cup \mathbb{N})^{|\bar{z}|}$ match the parameter variables (element or numeric). Assume that the first j variables in $\bar{x}$ are numeric variables and set $D := \{0, ..., s_1^{\mathfrak{A}}\} \times \cdots \times \{0, ..., s_j^{\mathfrak{A}}\} \times A^{d-j}$. We define the σ-structure $\mathfrak{B} = (B, R_1^{\mathfrak{B}}, \dots, R_\ell^{\mathfrak{B}})$ via

$$B := \big\{\, \bar{v} \in D \;\big|\; \bar{u}\bar{v} \in \Phi_{\mathrm{dom}}^{\mathfrak{A}} \,\big\} \text{ and}$$
$$R_i^{\mathfrak{B}} := \big\{\, (\bar{v}_1, \dots, \bar{v}_{\mathrm{ar}(R_i)}) \in B^{\mathrm{ar}(R_i)} \;\big|\; \bar{u}\bar{v}_1 \dots \bar{v}_{\mathrm{ar}(R_i)} \in \Phi_{R_i}^{\mathfrak{A}} \,\big\}$$

and the relation $E := \{(\bar{v}_1, \bar{v}_2) \in B^2 \mid \bar{u}\bar{v}_1\bar{v}_2 \in \Phi_{\cong}^{\mathfrak{A}}\}$. Set $\Theta(\mathfrak{A}, \bar{u}) := \mathfrak{B}/E$ if E is a congruence relation on $\mathfrak{B}$ and otherwise leave $\Theta(\mathfrak{A}, \bar{u})$ undefined. An interpretation is called *equivalence-free* if $\Phi_{\cong}(\bar{z}\bar{x}\bar{y})$ is the formula $\bar{x} = \bar{y}$. For extensions L of IFPC, the notion of an $L[\tau,\sigma]$-interpretation is defined similarly. For logics without numeric variables like FO or IFP, interpretations are defined analogously omitting the numeric parts.

A property P of τ-structures is *L-reducible* to a property Q of σ-structures if there is a parameter-free $L[\tau,\sigma]$-interpretation Θ such that $\mathfrak{A} \in P$ if and only if $\Theta(\mathfrak{A}) \in Q$ for every τ-structure $\mathfrak{A}$. A logic L' is *closed under L-interpretations* if for every property P that is L-reducible to an L'-definable property Q, the property P itself is L'-definable (cf. [13, 35]).

3 Witnessed Symmetric Choice

We extend IFPC with an inflationary fixed-point operator with witnessed symmetric choice. Let τ be a relational signature and R, R^*, and S be relation symbols not in τ satisfying $k = \mathrm{ar}(R) = \mathrm{ar}(R^*)$. The relation R will be used for stages in the fixed-point computation, R^* for a fixed-point, and S will be a singleton relation containing a chosen tuple.

We define the WSC-fixed-point operator with parameters $\bar{p}\bar{\nu}$. If $\Phi_{\mathrm{step}}(\bar{p}\bar{x}\bar{\nu})$ is an IFPC + WSC$[\tau, R, S]$-formula such that $|\bar{x}| = \mathrm{ar}(R)$, $\Phi_{\mathrm{choice}}(\bar{p}\bar{y}\bar{\nu})$ is a IFPC + WSC$[\tau, R]$-formula such that $|\bar{y}| = \mathrm{ar}(S)$, $\Phi_{\mathrm{wit}}(\bar{p}\bar{y}\bar{y}'z_1z_2\bar{\nu})$ is an IFPC + WSC$[\tau, R, R^*]$-formula where $|\bar{y}| = |\bar{y}'| = \mathrm{ar}(S)$, and $\Phi_{\mathrm{out}}(\bar{p}\bar{\nu})$ is an IFPC + WSC$[\tau, R^*]$-formula, then

$$\Phi(\bar{p}\bar{\nu}) = \mathsf{ifp\text{-}wsc}_{R,\bar{x};R^*;S,\bar{y},\bar{y}';z_1z_2}.\ \big(\Phi_{\mathrm{step}}(\bar{p}\bar{x}\bar{\nu}), \Phi_{\mathrm{choice}}(\bar{p}\bar{y}\bar{\nu}), \Phi_{\mathrm{wit}}(\bar{p}\bar{y}\bar{y}'z_1z_2\bar{\nu}), \Phi_{\mathrm{out}}(\bar{p}\bar{\nu})\big)$$

is an IFPC + WSC$[\tau]$-formula. The formulas Φ_{step}, Φ_{choice}, Φ_{wit}, and Φ_{out} are called *step formula*, *choice formula*, *witnessing formula*, and *output formula* respectively. Note that only element variables are used for defining the fixed-point in the WSC-fixed-point operator. This suffices for our purpose in this paper. We expect that our arguments also work with numeric variables. We omit the free numeric variables $\bar{\nu}$ when defining the semantics of the WSC-fixed-point operator because numeric parameters do not change automorphisms.

Intuitively, Φ is evaluated as follows. The formula Φ defines the set of vertex-tuples $\bar{u}$ that, when interpreting $\bar{p}$ with $\bar{u}$, satisfy the following process: Initialize R as the empty relation. Define a choice-set using the choice formula. Pick an arbitrary tuple of this set and let S

only contain this tuple. Define the next stage from R and S using the step formula. Then set R to be this next stage and repeat until a fixed-point R^* is reached. If the witnessing formula certifies that all choice-set were indeed orbits and the output formula is satisfied by R^*, then $\bar{u}$ satisfies Φ.

We now consider the evaluation in more detail. Let $\mathfrak{A}$ be a τ-structure and $\bar{u} \in A^{|\bar{p}|}$. We define a sequence of relations called *stages* $\emptyset =: R_1^{\mathfrak{A}}, \dots, R_\ell^{\mathfrak{A}} = R_{\ell+1}^{\mathfrak{A}} =: (R^*)^{\mathfrak{A}}$. Given the relation $R_i^{\mathfrak{A}}$, the choice formula defines the choice-set

$$T_{i+1}^{\mathfrak{A}} := \left\{ \bar{v} \;\middle|\; \bar{u}\bar{v} \in \Phi_{\text{choice}}^{\mathfrak{A}, R_i^{\mathfrak{A}}} \right\}.$$

We pick an arbitrary tuple $\bar{v} \in T_{i+1}^{\mathfrak{A}}$ and set $S_{i+1}^{\mathfrak{A}} := \{\bar{v}\}$ or $S_{i+1}^{\mathfrak{A}} := \emptyset$ if $T_{i+1}^{\mathfrak{A}} = \emptyset$. The step formula defines the next stage on the structure $(\mathfrak{A}, R_i^{\mathfrak{A}}, S_{i+1}^{\mathfrak{A}})$:

$$R_{i+1}^{\mathfrak{A}} := R_i^{\mathfrak{A}} \cup \left\{ \bar{w} \;\middle|\; \bar{u}\bar{w} \in \Phi_{\text{step}}^{(\mathfrak{A}, R_i^{\mathfrak{A}}, S_{i+1}^{\mathfrak{A}})} \right\}.$$

We proceed until a fixed-point $(R^*)^{\mathfrak{A}}$ is reached. This fixed-point is in general not isomorphism-invariant, i.e., not invariant under automorphisms of $(\mathfrak{A}, \bar{u})$. Isomorphism-invariance of Φ is ensured as follows: Choices are only allowed from orbits, which is certified by the witnessing formula. A set $N \subseteq \mathsf{Aut}((\mathfrak{A}, \bar{u}))$ *witnesses a relation* $R \subseteq A^k$ *as* $(\mathfrak{A}, \bar{u})$*-orbit*, if for very $\bar{v}, \bar{v}' \in R$ there is a $\varphi \in N$ with $\bar{v} = \varphi(\bar{v}')$. Because we consider isomorphism-invariant sets, the relation R is never a proper subset of an orbit. We require that Φ_{wit} defines a set of automorphisms. For $\bar{v}, \bar{v}' \in T_{i+1}^{\mathfrak{A}}$, a map $\varphi_{\bar{v},\bar{v}'}$ is defined by

$$w \mapsto w' \text{ whenever } \bar{u}\bar{v}\bar{v}'ww' \in \Phi_{\text{wit}}^{(\mathfrak{A}, R_i^{\mathfrak{A}}, (R^*)^{\mathfrak{A}})}.$$

The set $\{\varphi_{\bar{v},\bar{v}'} \mid \bar{v}, \bar{v}' \in T_{i+1}^{\mathfrak{A}}\}$ has to witness $T_{i+1}^{\mathfrak{A}}$ as $(\mathfrak{A}, \bar{u}, R_1^{\mathfrak{A}}, \dots, R_i^{\mathfrak{A}})$-orbit. The witnessing formula always has access to the fixed-point (note that orbits are witnessed after the fixed-point is computed). This is possible because $T_{i+1}^{\mathfrak{A}}$ is an $(\mathfrak{A}, \bar{u}, R_1^{\mathfrak{A}}, \dots, R_i^{\mathfrak{A}})$-orbit and not just an $(\mathfrak{A}, \bar{u}, R_i^{\mathfrak{A}})$-orbit. If some choice is not witnessed, then $\bar{u} \notin \Phi^{\mathfrak{A}}$. Otherwise, the output formula is evaluated on the fixed-point:

$$\Phi^{\mathfrak{A}} := \Phi_{\text{out}}^{(\mathfrak{A}, (R^*)^{\mathfrak{A}})}.$$

Because all choices are witnessed, all possible fixed-points (for different choices) are related by an automorphism of $(\mathfrak{A}, \bar{u})$ and thus either all or none satisfy the output formula.

An Example. We give an illustrating example (an adaption of [32]). We show that the class of threshold graphs (i.e., graphs which can be reduced to the empty graph by iteratively deleting universal or isolated vertices) is IFPC+WSC-definable (it is actually IFP-definable). The set of all isolated or universal vertices of a graph forms a 1-orbit (there cannot be an isolated and a universal vertex in a nontrivial graph). We chose one such vertex, collect it in a unary relation R, and repeat as follows: The choice formula

$$\Phi_{\text{choice}}(y) := \neg R(y) \wedge \big((\forall z.\, \neg R(y) \Rightarrow E(y,z)) \vee (\forall z.\, \neg R(y) \Rightarrow \neg E(y,z))\big)$$

defines the set of all isolated or universal vertices after deleting the vertices in R. The step formula $\Phi_{\text{step}}(x) := R(x) \vee S(x)$ adds the chosen vertex contained in the relation S to R. The output formula $\Phi_{\text{out}} := \forall x.\, R^*(x)$ defines whether it was possible to delete all vertices. Witnessing orbits is easy: To show that two isolated (or universal) vertices y and y' are related by an automorphism, it suffices to define their transposition via

$$\Phi_{\text{wit}}(y, y', z_1, z_2) := (z_1 = y \wedge z_2 = y') \vee (z_2 = y \wedge z_1 = y') \vee (y \neq z_1 = z_2 \neq y').$$

The formula $\mathsf{ifp\text{-}wsc}_{R,x;R^*;S,y,y';z_1 z_2}.\, (\Phi_{\text{step}}, \Phi_{\text{choice}}, \Phi_{\text{wit}}, \Phi_{\text{out}})$ defines the class of threshold graphs.

Reduct Semantics. The semantics is defined formally using the WSC^*-operator from [32]. It formalizes the former evaluation strategy. The WSC-fixed-point operator Φ is evaluated in the structure $\mathfrak{A} \restriction \mathrm{sig}(\Phi)$, where $\mathrm{sig}(\Phi)$ are all relations symbols used in Φ. So adding a relation to $\mathfrak{A}$ that is not mentioned in Φ but possibly changes the orbits of $\mathfrak{A}$ does not change $\Phi^{\mathfrak{A}}$. This is a desirable property of a logic [13]. The *reduct semantics* of a choice operator can also be found in [9].

Extension with an Operator for Logical Interpretations. We extend IFPC+WSC with another operator using interpretations. Every IFPC+WSC-formula is an IFPC+WSC+I-formula. If $\Theta(\bar{p}\bar{\nu})$ is an IFPC+WSC+I$[\tau,\sigma]$-interpretation with parameters $\bar{p}\bar{\nu}$ and Φ is an IFPC+WSC+I$[\sigma]$-sentence, then the *interpretation operator*

$$\Psi(\bar{p}\bar{\nu}) := \mathsf{I}(\Theta(\bar{p}\bar{\nu}); \Phi)$$

is an IFPC+WSC+I$[\tau]$-formula with free variables $\bar{p}\bar{\nu}$. The semantics is defined by

$$\mathsf{I}(\Theta(\bar{p}\bar{\nu}); \Phi)^{\mathfrak{A}} := \left\{ \bar{u}\bar{n} \in A^{|\bar{p}|} \times \mathbb{N}^{|\bar{\nu}|} \;\middle|\; \Phi^{\Theta(\mathfrak{A},\bar{u}\bar{n})} \neq \emptyset \right\}.$$

Note that $\Phi^{\Theta(\mathfrak{A},\bar{u}\bar{n})} = \{()\}$ if Φ is satisfied and $\Phi^{\Theta(\mathfrak{A},\bar{u}\bar{n})} = \emptyset$ otherwise. The interpretation operator evaluates a subformula in the image of an interpretation. For IFPC, such an operator does not increase expressiveness because IFPC is closed under interpretations [35]. For IFPC+WSC this is not clear: Because $\Theta(\mathfrak{A}, \bar{u}\bar{n})$ may have a different automorphism structure, Φ possibly can exploit the WSC-fixed-point operator in a way impossible on $\mathfrak{A}$. Indeed, we will see that IFPC+WSC is not even closed under FO-interpretations. We now study the properties of IFPC+WSC+I and its relation to IFPC+WSC.

4 The CFI Construction

We give an overview of the CFI construction. For more details, we refer to [5]. The *degree-d CFI gadget* consists of d pairs of *edge vertices* $\{a_{i0}, a_{i1}\}$ for every $i \in [d]$ and the set of *gadget vertices* $\{\bar{b} \in \mathbb{F}_2^d \mid b_1 + \cdots + b_d = 0\}$. There is an edge $\{a_{ij}, \bar{b}\}$ if and only if $b_i = j$. If we use d colors to additionally color the vertices $\{a_{i0}, a_{i1}\}$ for every $i \in [d]$, then the CFI gadget realizes precisely the automorphisms exchanging the vertices $\{a_{i0}, a_{i1}\}$ for an even number of $i \in [d]$. We later need a relational variant of the CFI gadgets in which every gadget vertex $\bar{b}$ is replaced by the d-tuple $(a_{1b_1}, \ldots, a_{db_d})$. This gadget has the same automorphisms [24]. A *base graph* is a simple connected graph. Let $G = (V, E, \preceq)$ be a colored base graph. We call V *base vertices* and E *base edges* and use fraktur letters for base vertices or edges. For $f\colon E \to \mathbb{F}_2$, we construct the *CFI graph* $\mathsf{CFI}(G, f)$ as follows: Replace every base vertex by a CFI gadget of the same degree. For every base edge $\{\mathfrak{u}, \mathfrak{v}\} \in E$, we obtain two edge-vertex-pairs $\{a_{i0}, a_{i1}\}$ and $\{a'_{j0}, a'_{j1}\}$. The first one is given by the gadget of $\mathfrak{u}$ and the second one by the gadget of $\mathfrak{v}$. Now add the edges $\{a_{ik}, a'_{j\ell}\}$ satisfying $k + \ell = f(\{\mathfrak{u}, \mathfrak{v}\})$. The gadget vertices of the gadget for a base vertex $\mathfrak{u}$ *originate* from $\mathfrak{u}$, the edge vertices $\{a_{i0}, a_{i1}\}$ originate from $(\mathfrak{u}, \mathfrak{v})$, and the edge vertices $\{a'_{j0}, a'_{j1}\}$ originate from $(\mathfrak{v}, \mathfrak{u})$. The color of edge and gadget vertices is obtained from the color of its origin.

It is known [5] that $\mathsf{CFI}(G, g) \cong \mathsf{CFI}(G, f)$ if and only if $\sum g := \sum_{\mathfrak{e} \in E} g(\mathfrak{e}) = \sum f$. If we are interested in the graph up to isomorphism, we write $\mathsf{CFI}(G, 0)$ and $\mathsf{CFI}(G, 1)$. CFI graphs with $\sum g = 0$ are called *even* and the others *odd*. A base edge $\mathfrak{e} \in E$ is *twisted* by f and g if $g(\mathfrak{e}) \neq f(\mathfrak{e})$. Twisted edges can be moved by path isomorphisms [29]: If $\mathfrak{u}_1, \ldots, \mathfrak{u}_\ell$ is a path in G, then there is an isomorphism $\varphi\colon \mathsf{CFI}(G, g) \to \mathsf{CFI}(G, g')$, where $g'(\mathfrak{e}) = g(\mathfrak{e})$ apart from

$\mathfrak{e}_1 := \{\mathfrak{u}_1, \mathfrak{u}_2\}$ and $\mathfrak{e}_2 := \{\mathfrak{u}_{\ell-1}, \mathfrak{u}_\ell\}$ satisfying $g'(\mathfrak{e}_i) = g(\mathfrak{e}_i) + 1$. If G is totally ordered, then every isomorphism is composed of path-isomorphisms. When considering a cycle instead of a path, we obtain an automorphism of $\mathsf{CFI}(G, g)$.

For a class of base graphs $\mathcal{K}$, set $\mathsf{CFI}(\mathcal{K}) := \{\mathsf{CFI}(G, g) \mid G = (V, E) \in \mathcal{K}, g\colon E \to \mathbb{F}_2\}$. The *CFI query* for $\mathsf{CFI}(\mathcal{K})$ is to decide whether a given CFI graph in $\mathsf{CFI}(\mathcal{K})$ is even.

▶ **Lemma 4** ([11]). *If G is of minimum degree 2 and has treewidth at least k, in particular if G is k-connected, then $\mathsf{CFI}(G, 0) \simeq^k_{\mathcal{C}} \mathsf{CFI}(G, 1)$.*

▶ **Lemma 5.** *Let $G = (V, E, \preceq)$ be $(k+2)$-connected and $\mathfrak{A} = \mathsf{CFI}(G, f)$ for some $f\colon E \to \mathbb{F}_2$. Let $\bar{u} \in A^{\leq k}$ and $\{\mathfrak{u}, \mathfrak{v}\} \in E$ be a base edge such that no vertex in $\bar{u}$ has origin $\mathfrak{u}$, $\mathfrak{v}$, $(\mathfrak{u}, \mathfrak{v})$, or $(\mathfrak{v}, \mathfrak{u})$. Then the two edge vertices with origin $(\mathfrak{u}, \mathfrak{v})$ are contained in the same orbit of $(\mathfrak{A}, \bar{u})$.*

5 Canonization of CFI Graphs in IFPC+WSC+I

We show that canonizing CFI graphs in IFPC+WSC+I is not harder than canonizing the base graphs. We work with a class of base graphs closed under individualization, i.e., intuitively closed under assigning new unique colors to some vertices.

▶ **Definition 6** (Individualization of Vertices). *Let $\mathfrak{A}$ be a τ-structure. A binary relation $\trianglelefteq^{\mathfrak{A}} \subseteq A^2$ is an* individualization of $V \subseteq A$ *if $\trianglelefteq^{\mathfrak{A}}$ is a total order on V and $\trianglelefteq^{\mathfrak{A}} \subseteq V^2$. We say that $\trianglelefteq^{\mathfrak{A}}$ is an individualization if it is an individualization of some $V \subseteq A$ and that a vertex $u \in A$ is* individualized *by $\trianglelefteq^{\mathfrak{A}}$ if $u \in V$.*

The *closure under individualization* of a class of τ-structures $\mathcal{K}$ is the class $\mathcal{K}^{\trianglelefteq}$ of $(\tau \uplus \{\trianglelefteq\})$-structures such that $(\mathfrak{A}, \trianglelefteq^{\mathfrak{A}}) \in \mathcal{K}^{\trianglelefteq}$ for every $\mathfrak{A} \in \mathcal{K}$ and every individualization $\trianglelefteq^{\mathfrak{A}}$. Instead of $(\mathfrak{A}, \trianglelefteq^{\mathfrak{A}})$, one can think of $(\mathfrak{A}, \bar{u})$ where $u_1 \trianglelefteq^{\mathfrak{A}} \cdots \trianglelefteq^{\mathfrak{A}} u_{|\bar{u}|}$ are the $\trianglelefteq^{\mathfrak{A}}$-individualized vertices. In the following, let L be one of the logics IFPC, IFPC+WSC, or IFPC+WSC+I. We adapt some notions related to canonization from [32] to our first-order setting. Note that all definitions that follow implicitly include the closure under individualization.

▶ **Definition 7** (Canonization). *Let $\mathcal{K}$ be a class of τ-structures. An* L-canonization *for $\mathcal{K}$ is an $L[\tau \uplus \{\trianglelefteq\}, \tau \uplus \{\trianglelefteq, \leq\}]$-interpretation Θ such that $\leq^{\Theta(\mathfrak{A})}$ is a total order on $\Theta(\mathfrak{A})$ for every $\mathfrak{A} \in \mathcal{K}^{\trianglelefteq}$, $\mathfrak{A} \cong \Theta(\mathfrak{A}) \restriction (\tau \uplus \{\trianglelefteq\})$ for every $\mathfrak{A} \in \mathcal{K}^{\trianglelefteq}$, and $\Theta(\mathfrak{A}) \cong \Theta(\mathfrak{B})$ if and only if $\mathfrak{A} \cong \mathfrak{B}$ for all $\mathfrak{A}, \mathfrak{B} \in \mathcal{K}^{\trianglelefteq}$. L* canonizes *$\mathcal{K}$ if there is an L-canonization for $\mathcal{K}$.*

▶ **Definition 8** (Distinguishable Orbits). *The logic L* distinguishes k-orbits *for a class of τ-structures $\mathcal{K}$ if some $L[\tau \uplus \{\trianglelefteq\}]$-formula $\Phi(\bar{x}, \bar{y})$ with $|\bar{x}| = |\bar{y}| = k$ defines, for every $\mathfrak{A} \in \mathcal{K}^{\trianglelefteq}$, a total preorder on A^k whose equivalence classes form the k-orbit partition of $\mathfrak{A}$, i.e., Φ orders the k-orbits.*

▶ **Definition 9** (Ready for Individualization). *A class of τ-structures $\mathcal{K}$ is* ready for individualization *in L if there is an $L[\tau \uplus \{\trianglelefteq\}]$-formula $\Phi(x)$ defining for every $\mathfrak{A} \in \mathcal{K}^{\trianglelefteq}$ a set of vertices $O = \Phi^{\mathfrak{A}}$ such that O is a 1-orbit of $\mathfrak{A}$, $|O| > 1$ if $\mathfrak{A}$ has a non-trivial 1-orbit, and if $O = \{u\}$ is a singleton set, then u is not individualized by $\trianglelefteq^{\mathfrak{A}}$ unless $\trianglelefteq^{\mathfrak{A}}$ individualizes A.*

Let L be one logic of IFPC+WSC and IFPC+WSC+I. The following lemma is similar to [32] for CPT+WSC and the proof is analogous (we do not include definable isomorphism):

▶ **Lemma 10.** *Let $\mathcal{K}$ be a class of τ-structures. Then the following are equivalent:*

1. *L defines a canonization for $\mathcal{K}$.*
2. *L distinguishes the k-orbits of $\mathcal{K}$ for every $k \in \mathbb{N}$.*
3. *$\mathcal{K}$ is ready for individualization in L.*

Gurevich's canonization algorithm [23] is used to define the canonization. It requires the WSC-fixed-point operator. When canonizing using Lemma 10, defining witnessing automorphisms is hidden in Gurevich's algorithm and they do not have to be defined explicitly.

▶ **Lemma 11.** *Let $\mathcal{K}$ be a class of colored base graphs. If* IFPC+WSC+I *distinguishes* 2*-orbits of $\mathcal{K}$, then* $\mathsf{CFI}(\mathcal{K})$ *is ready for individualization in* IFPC+WSC+I.

Proof Sketch. For a CFI graph $\mathfrak{A}$ over $G \in \mathcal{K}^{\trianglelefteq}$, the base graph G is definable by an IFPC-interpretation. With an interpretation operator, we evaluate a 2-orbit-defining formula on G.

a) If there is a 2-orbit O of G such that for all edges in O there is a cycle not using the origins of individualized vertices, we define the non-trivial orbit of the CFI graph containing the edge-vertex-pairs with origin in O.

b) Otherwise, we can order each edge-vertex-pair. If there is a non-trivial 2-orbit O of G, we define the non-trivial orbit of the CFI graph containing the greater edge vertex of each edge-vertex-pair (with respect to the edge-vertex-pair-order) with origin in O.

c) Otherwise, the edge-vertex-pair-order extends to a total order. ◀

Proof of Theorem 2. The claim follows immediately from Lemmas 10 and 11. ◀

▶ **Corollary 12.** *If $\mathcal{K}$ is a class of base graphs of bounded degree, then* IFPC+WSC+I *defines canonization for $\mathcal{K}$ if and only if* IFPC+WSC+I *defines canonization for* $\mathsf{CFI}(\mathcal{K})$.

Proof. One direction is by Theorem 2. For the other direction, a base graph G of bounded degree is canonized by defining $\mathsf{CFI}(G, 0)$ (which is possible because G is of bounded degree), canonizing $\mathsf{CFI}(G, 0)$, and defining the base graph of the ordered copy of $\mathsf{CFI}(G, 0)$. ◀

Theorem 2 can be applied iteratively: If IFPC+WSC+I canonizes $\mathcal{K}$, then IFPC+WSC+I canonizes $\mathsf{CFI}(\mathcal{K})$, and so IFPC+WSC+I canonizes $\mathsf{CFI}(\mathsf{CFI}(\mathcal{K}))$. Every iteration adds one WSC-fixed-point operator (Gurevich's algorithm in Lemma 10) and one interpretation operator (define the base graph in Lemma 11), i.e., the nesting depth of these operators increases.

6 The CFI Query and Nesting of Operators

We show that the increased nesting depth of operators in Theorem 2 is unavoidable. If IFPC distinguishes orbits of the base graphs, then the nesting depth of WSC-fixed-point operators has to increase because IFPC does not define the CFI query. To show this for IFPC+WSC+I-distinguishable orbits, we combine non-isomorphic CFI graphs into a new base graph and apply the CFI construction again. To define orbits of these double CFI graphs, one has to define the CFI query for the base CFI graphs, which requires a WSC-fixed-point operator. However, parameters to WSC-fixed-point operators will complicate matters.

Nested WSC-Fixed-Point and Interpretation Operators. Let $\text{IFPC} \subseteq L \subseteq \text{IFPC+WSC+I}$. We write $\text{WSC}(L)$ for formulas composed by IFPC-formula-formation rules from L-formulas and WSC-fixed-point-operators, for which all subformulas are L-formulas. We define $\text{I}(L)$ similarly: One can use interpretation operators $I(\Theta, \Psi)$ where Θ is an L-interpretation and Ψ is an L-formula. We set $\text{WSCI}(L) := \text{WSC}(\text{I}(L))$ and $\text{WSCI}^{k+1}(L) := \text{WSCI}(\text{WSCI}^k(L))$. Note the construction in Lemmas 10 and 11:

▶ **Corollary 13.** *Let $\mathcal{K}$ be a class of base graphs.*

1. *If L distinguishes* 2*-orbits of $\mathcal{K}$, then* $\mathsf{CFI}(\mathcal{K})$ *is ready for individualization in* $\text{I}(L)$.
2. *If* $\mathsf{CFI}(\mathcal{K})$ *is ready for individualization in L, then* $\text{WSC}(L)$ *canonizes* $\mathsf{CFI}(\mathcal{K})$.
3. *If L distinguishes* 2*-orbits of $\mathcal{K}$, then* $\text{WSCI}(L)$ *canonizes* $\mathsf{CFI}(\mathcal{K})$.

Color Class Joins. Let $G_1, \dots, G_\ell$ be connected colored graphs such that all G_i have c colors. The *color class join* $J_{\mathrm{cc}}(G_1, \dots, G_\ell)$ is the following graph: Start with the disjoint union of the G_i and add c additional vertices $u_1, \dots, u_c$. Add, for every $i \in [c]$, edges between u_i and every vertex v in the i-th color class of all G_j. The resulting colored graph $J_{\mathrm{cc}}(G_1, \dots, G_\ell)$ has $2c$ color classes: For every $i \in [c]$, the vertex u_i forms a singleton color class and the union of the i-th color classes of every G_j forms a color class. The G_j are the *parts* of $J_{\mathrm{cc}}(G_1, \dots, G_\ell)$. The u_j are the *join vertices* and the others the *part vertices*. The *part* of a part vertex v is the G_j containing v. Defining orbits of $J_{\mathrm{cc}}(G_1, \dots, G_\ell)$ is at least as hard as defining isomorphism of the G_j.

▶ **Lemma 14.** *If two part vertices v and v' are in the same orbit of $J_{cc}(G_1, \dots, G_\ell)$, then the part of v is isomorphic to the one of v'.*

We set $J^k_{\mathrm{cc}}(G, H, K) := J_{\mathrm{cc}}(G, \dots, G, H, \dots, H, K, \dots, K)$, where G, H, and K are repeated k times. To consider color class joins of CFI graphs, let $\mathcal{K}$ be a class of colored base graphs. For $G \in \mathcal{K}$ and $g \in \mathbb{F}_2$, we set

$$\begin{aligned} \mathsf{CFI}^k(G, g) &:= J^k_{\mathrm{cc}}(\mathsf{CFI}(G, 0), \mathsf{CFI}(G, g), \mathsf{CFI}(G, 1)), \\ \mathsf{CFI}^k(\mathcal{K}) &:= \{\, \mathsf{CFI}^k(G, g) \mid G \in \mathcal{K}, g \in \mathbb{F}_2 \,\}, \\ \mathsf{CFI}^\omega(\mathcal{K}) &:= \bigcup_{k \in \mathbb{N}} \mathsf{CFI}^k(\mathcal{K}). \end{aligned}$$

▶ **Lemma 15.** *If L canonizes $\mathsf{CFI}(\mathcal{K})$, then L canonizes $\mathsf{CFI}^\omega(\mathcal{K})$.*

Let $G_1, \dots, G_\ell$ be colored base graphs and $h \in \mathbb{F}_2$. We transfer the notion of part and join vertices from $H := J_{\mathrm{cc}}(G_1, \dots, G_\ell)$ to $\mathfrak{A} := \mathsf{CFI}(H, h)$. The *$G_i$-part* of $\mathfrak{A}$ is the set of vertices originating from G_i in H. These vertices are called *part vertices of G_i*. A vertex is a part vertex, if it is a part vertex of some G_i. The remaining vertices are the *join vertices*.

We consider a special class of individualizations of $\mathfrak{A}$. Let $\bar{u} \in A^*$. A part of $\mathfrak{A}$ is *pebbled by $\bar{u}$* if the part contains u_i for some i. The set of *$\bar{u}$-pebbled-part vertices* $V_{\bar{u}}(\mathfrak{A})$ is the set of all join vertices and all part vertices of a part pebbled by $\bar{u}$. The set of *$\bar{u}$-pebbled-part individualizations* $P_{\bar{u}}(\mathfrak{A})$ is the set of all individualizations of $V_{\bar{u}}(\mathfrak{A})$.

▶ **Definition 16** (Unpebbled-Part-Distinguishing)**.** *For a tuple $\bar{u} \in A^\ell$, a relation $R \subseteq A^k$ is* $\bar{u}$-unpebbled-part-distinguishing *if there are $m \in [k]$ and $i \neq j \in [\ell]$ such that both the G_i-part and the G_j-part of $\mathfrak{A}$ are $\bar{u}$-unpebbled, there is a $\bar{v} \in R$ such that v_m is a part vertex of G_i, and for every $\bar{w} \in R$, the vertex w_m is not a part vertex of G_j.*

If $G_i \not\cong G_j$ are not $\bar{u}$-pebbled, then every k-orbit O of $(\mathfrak{A}, \bar{u})$ satisfies $O \subseteq V_{\bar{u}}(\mathfrak{A})^k$ or is $\bar{u}$-unpebbled-part-distinguishing because G_i- and G_j-part vertices are not in the same orbit.

Quantifying over Pebbled-Part Individualizations. We now define an extension of $\mathcal{C}_k$ which allows for quantifying over pebbled-part individualizations. This (unnatural) extension can only be evaluated on CFI graphs over color class joins. We use this logic for proving WSCI(IFPC)-undefinability. If $\Phi(\bar{x})$ is a $\mathcal{C}_k[\tau, \trianglelefteq_P]$-formula, then $(\exists^P \trianglelefteq_P . \Phi)(\bar{x})$ is a $\mathcal{P}_k[\tau]$-formula. $\mathcal{P}_k[\tau]$-formulas can be combined as usual in $\mathcal{C}_k$ with Boolean operators and counting quantifiers. Note that $\exists^P$-quantifiers *cannot* be nested. Let $G_1, \dots, G_\ell$ be base graphs, $g \in \mathbb{F}_2$, and $\mathfrak{A} = \mathsf{CFI}(J_{\mathrm{cc}}(G_1, \dots, G_n), g)$. The $\exists^P$-quantifier has the following semantics:

$$(\exists^P \trianglelefteq_P . \Phi)^{\mathfrak{A}} := \{\, \bar{u} \mid \bar{u} \in \Phi^{(\mathfrak{A}, \trianglelefteq_P^{\mathfrak{A}})} \text{ for some } \trianglelefteq_P^{\mathfrak{A}} \in P_{\bar{u}}(\mathfrak{A}) \,\}.$$

▶ **Lemma 17.** *Let $G_1, \dots, G_{k+1}$ be colored base graphs, each with $c > k \geq 3$ color classes, such that $\mathsf{CFI}(G_i, 0) \simeq^k_{\mathcal{C}} \mathsf{CFI}(G_i, 1)$ for every $i \in [k+1]$. Then $\mathsf{CFI}(J_{cc}(G_1, \dots, G_{k+1}), 0)$ and $\mathsf{CFI}(J_{cc}(G_1, \dots, G_{k+1}), 1)$ are $\mathcal{P}_k$-equivalent.*

Proof Sketch. The lemma is proven by a game characterization of $\mathcal{P}_k$. Essentially, because only k of the $k+1$ parts can be pebbled by k pebbles, the twist can always be moved in the pebble-free part, which is not affected by quantifying over pebbled-part individualizations. ◀

Nesting Operators to Define the CFI Query is Necessary. Let $\mathcal{K} := \{G_k \mid k \in \mathbb{N}\}$ be a set of ordered 3-regular base graphs such that G_k has treewidth at least k for every $k \in \mathbb{N}$.

▶ **Lemma 18.** $\mathsf{CFI}(\mathsf{CFI}(G_k, g), 0) \simeq^k_{\mathcal{C}} \mathsf{CFI}(\mathsf{CFI}(G_k, g), 1)$ *for every $k \in \mathbb{N}$ and $g \in \mathbb{F}_2$.*

Proof. The graph G_k is a minor of $\mathsf{CFI}(G_k, g)$ for every $g \in \mathbb{F}_2$ (cf. [11]). Hence, $\mathsf{CFI}(G_k, g)$ has treewidth at least k. The claim follows by Lemma 4. ◀

▶ **Lemma 19.** $\mathrm{WSCI}^2(\mathrm{IFPC})$ *defines the CFI query for* $\mathsf{CFI}(\mathsf{CFI}^\omega(\mathcal{K}))$.

Proof. IFPC distinguishes 2-orbits of $\mathcal{K}$ and so WSCI(IFPC) canonizes $\mathsf{CFI}(\mathcal{K})$ (Corollary 13) and $\mathsf{CFI}^\omega(\mathcal{K})$ (Lemma 15) and so also distinguishes 2-orbits of $\mathsf{CFI}^\omega(\mathcal{K})$. Thus, $\mathrm{WSCI}^2(\mathrm{IFPC})$ canonizes $\mathsf{CFI}(\mathsf{CFI}^\omega(\mathcal{K}))$ (Corollary 13) and hence defines the CFI query for $\mathsf{CFI}(\mathsf{CFI}^\omega(\mathcal{K}))$. ◀

To show WSCI(IFPC)-undefinability, there are two cases: If a choice is made from an orbit of parts not pebbled by parameters, then CFI graphs of $\mathsf{CFI}(\mathcal{K})$ are distinguished. Otherwise, CFI graphs of $\mathsf{CFI}(\mathsf{CFI}^\omega(\mathcal{K}))$ are distinguished only by choices from parameter-pebbled parts, so by (at most) individualizing all pebbled-part vertices, i.e., the graphs are distinguished by $\mathcal{P}_k$. By Lemmas 4 and 17, each case only applies to finitely many $\mathcal{K}$-graphs.

▶ **Lemma 20.** WSCI(IFPC) *does not define the CFI query for* $\mathsf{CFI}(\mathsf{CFI}^\omega(\mathcal{K}))$.

Proof Sketch. Suppose, towards a contradiction, that Φ is a WSCI(IFPC)-formula defining the CFI query for $\mathsf{CFI}(\mathsf{CFI}^\omega(\mathcal{K}))$. Because IFPC is closed under interpretations, we can assume that Φ is a WSC(IFPC)-formula.

Let $\Psi_1(\bar{x}_1), \dots, \Psi_p(\bar{x}_p)$ be all WSC-fixed-point operators in Φ and suppose all $\bar{x}_i$ are element variables (for numeric ones see the full version [30]). Let the number of distinct variables of Φ be k and let $\ell := \ell(k) \geq \max\{k, 3\}$. We consider the subclass $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(\mathcal{K})) \subseteq \mathsf{CFI}(\mathsf{CFI}^\omega(\mathcal{K}))$ and partition $\mathcal{K}$ as follows: Let $\mathcal{K}_{\mathrm{orb}}$ be the set of all $G \in \mathcal{K}$ such that, for every $g \in \mathbb{F}_2$, there are $h \in \mathbb{F}_2$, $j \in [p]$, and a $|\bar{x}_j|$-tuple $\bar{u}$ of $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G, g), h))$ such that
a) all choice-sets during the evaluation of $\Psi_j(\bar{u})$ on $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G, g), h)$ are orbits and
b) some choice-set is $\bar{u}$-unpebbled-part-distinguishing.
Set $\mathcal{K}_{\mathrm{cfi}} := \mathcal{K} \setminus \mathcal{K}_{\mathrm{orb}}$. At least one of $\mathcal{K}_{\mathrm{orb}}$ and $\mathcal{K}_{\mathrm{cfi}}$ is infinite. Assume that $\mathcal{K}_{\mathrm{orb}}$ is infinite. We claim that the CFI query for $\mathsf{CFI}(\mathcal{K}_{\mathrm{orb}})$ is IFPC-definable. There is an IFPC-interpretation that for $G \in \mathcal{K}$ maps $\mathsf{CFI}(G, g)$ and $h \in \mathbb{F}_2$ to $(\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G, g), h), \trianglelefteq)$ such that $\trianglelefteq$ individualizes the vertices of $\ell + 1$ many $\mathsf{CFI}(G, 0)$-parts, $\ell + 1$ many $\mathsf{CFI}(G, 1)$-parts, and all join vertices. For every $\Psi_j(\bar{x}_j)$, we try all $h \in \mathbb{F}_2$ and all tuples $\bar{u}$ of $\trianglelefteq$-individualized vertices for $\bar{x}_j$. We simulate Ψ_j as long as all choices are made from $\bar{u}$-pebbled parts (which are resolved using $\trianglelefteq$). If that is not the case, we check if the choice-set is $\bar{u}$-unpebbled-part distinguishing. If not, the choice-set is not an orbit and we evaluate to false. Otherwise, the choice-set contains vertices of $\mathsf{CFI}(G, g)$-parts and either of $\mathsf{CFI}(G, 0)$-parts or of $\mathsf{CFI}(G, 1)$-parts. At least one $\mathsf{CFI}(G, 0)$-part and one $\mathsf{CFI}(G, 1)$-part is not $\bar{u}$-pebbled because $|\bar{u}| \leq k < \ell + 1$, which is isomorphic to the $\mathsf{CFI}(G, g)$-parts. So we defined the parity of $\mathsf{CFI}(G, g)$ and IFPC defines the CFI query for $\mathsf{CFI}(\mathcal{K}_{\mathrm{orb}})$ contradicting Lemma 4.

So $\mathcal{K}_{\text{cfi}}$ must be infinite. Then there is an $\ell' > \ell$ such that $G := G_{\ell'} \in \mathcal{K}_{\text{cfi}}$. Hence there is a $g \in \mathbb{F}_2$ such that for all $h \in \mathbb{F}_2$, $j \in [p]$, and all $|\bar{x}_j|$-tuples $\bar{u}$ of $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G,g),h))$
a) some choice-set during the evaluation of $\Psi_j(\bar{u})$ on $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G,g),h)$ is not an orbit or
b) all choice-sets are not $\bar{u}$-unpebbled-part-distinguishing.
We construct a $\mathcal{P}_\ell$-formula equivalent to Φ on $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G,g),0)$ and $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G,g),1)$. The general idea is that we quantify over all $\bar{u}$-pebbled-part individualizations. Choices from choice-sets using only vertices of the $\bar{u}$-pebbled parts can be resolved deterministically using the individualization. If this is not always the case, one choice-set will not be an orbit and we evaluate to false. So a $\mathcal{P}_\ell$-formula distinguishes $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G,g),0)$ and $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G,g),1)$. This finally contradicts Lemma 17: The graphs G and $\mathsf{CFI}(G,g')$ have more than ℓ color classes for every $g' \in \mathbb{F}_2$ and we have $\mathsf{CFI}(\mathsf{CFI}(G,g'),0) \simeq^k_{\mathcal{C}} \mathsf{CFI}(\mathsf{CFI}(G,g'),1)$ by Lemma 18. ◀

Proof of Theorem 3. Consider $\mathsf{CFI}^\omega(\mathcal{K})$. The claim follows from Lemmas 15, 19, and 20. ◀

▶ **Corollary 21.** $\mathrm{IFPC} < \mathrm{WSCI}(\mathrm{IFPC}) < \mathrm{WSCI}^2(\mathrm{IFPC})$.

It seems natural that $\mathrm{WSCI}^n(\mathrm{IFPC}) < \mathrm{WSCI}^{n+1}(\mathrm{IFPC})$ for every $n \in \mathbb{N}$. Possibly, this hierarchy can be shown by iterating our construction (e.g., using $\mathsf{CFI}((\mathsf{CFI}^\omega)^n(\mathcal{K}))$).

7 Separating IFPC+WSC from IFPC+WSC+I

We define a class of *asymmetric* structures $\mathcal{K}$, i.e., structures without non-trivial automorphisms, for which isomorphism can be reduced to isomorphism of CFI graphs via an interpretation. To do so, we combine CFI graphs and the so-called multipedes.

7.1 Multipedes

We review the multipedes construction [24]. Let $G = (V, W, E, \leq)$ be an ordered bipartite graph, where every vertex in V has degree 3. We obtain the *multipede* $\mathsf{MP}(G)$ as follows. For every vertex $\mathfrak{u} \in W$, there is a vertex pair $F(\mathfrak{u}) = \{u_0, u_1\}$ called a *segment*. We also call $\mathfrak{u} \in W$ a segment. A single vertex u_i is a *foot*. Vertices $\mathfrak{v} \in V$ are *constraint vertices*. For every constraint vertex $\mathfrak{v} \in V$, a degree-3 CFI gadget with three edge-vertex-pairs $\{a^j_0, a^j_1\}$ ($j \in [3]$) is added. Let $N_G(\mathfrak{v}) = \{\mathfrak{u}^1, \mathfrak{u}^2, \mathfrak{u}^3\}$. Then a^j_i is identified with the foot u^j_i for all $j \in [3]$ and $i \in \mathbb{F}_2$. We use the relation-based CFI gadgets, i.e., we do not add further vertices. All base constraint vertices have degree 3 and we obtain a ternary $\{R, \preceq\}$-structure. The coloring $\preceq$ is obtained from $\leq$ such that feet in the same segment have the same color.

The *feet-induced* subgraph by $X \subseteq W$ is $G[[X]] := G[X \cup \{\mathfrak{v} \in V \mid N_G(\mathfrak{v}) \subseteq X\}]$. We extend the notation to the multipede: $\mathsf{MP}(G)[[X]]$ is the induced substructure of all feet whose segment is contained in X. For a tuple $\bar{u}$ of feet of $\mathsf{MP}(G)$, we define $S(\bar{u}) := \{\mathfrak{u} \in W \mid u_i \in F(\mathfrak{u}) \text{ for some } i \leq |\bar{u}|\}$ to be the set of the segments of the u_i.

A bipartite graph $G = (V, W, E, \leq)$ is *odd* if for every $\emptyset \neq X \subseteq W$, there exists a $\mathfrak{v} \in V$ such that $|X \cap N_G(\mathfrak{v})|$ is odd. The graph G is *k-meager*, if for every set $X \subseteq W$ of size $|X| \leq 2k$, it holds that $|\{\mathfrak{v} \in V \mid N_G(\mathfrak{v}) \subseteq X\}| \leq 2|X|$.

▶ **Lemma 22** ([24]). *If G is an odd and ordered bipartite graph, then $\mathsf{MP}(G)$ is asymmetric.*

▶ **Lemma 23** ([24]). *Let G be a k-meager bipartite graph, $\mathfrak{A} = \mathsf{MP}(G)$, and $\bar{u}, \bar{v} \in A^k$. If there is a local isomorphism $\varphi \in \mathsf{Aut}(\mathfrak{A}[[S(\bar{u}\bar{v})]])$ with $\varphi(\bar{u}) = \bar{v}$, then $(\mathfrak{A}, \bar{u}) \simeq^k_{\mathcal{C}} (\mathfrak{A}, \bar{v})$.*

Odd and k-meager graphs exist [24]. A closer inspection shows that for these graphs there are sets of vertices of pairwise large distance. For a bipartite graph $G = (V, W, E)$, a set $X \subseteq W$ is *k-scattered* if all distinct $\mathfrak{u}, \mathfrak{v} \in X$ have distance at least $2k$ in G.

▶ **Lemma 24.** *For every k, there is an odd and k-meager bipartite graph $G = (V, W, E)$ and a k-scattered set $X \subseteq W$ of size $|X| \geq k^2$.*

We will use a k-scattered set X to ensure that in the bijective k-pebble game placing a pebble on one foot of a segment in X creates no restrictions on the other segments in X. For now, fix a bipartite graph $G = (V, W, E, \leq)$. The *attractor* of a set $X \subseteq W$ is

$$\operatorname{attr}(X) := X \cup \bigcup_{u \in V \colon |N_G(u) \setminus X| \leq 1} N_G(u).$$

The set X is *closed* if $X = \operatorname{attr}(X)$. The *closure* $\operatorname{cl}(X)$ of X is the inclusion-wise minimal closed superset of X.

▶ **Lemma 25** ([24])**.** *Assume that $X \subseteq W$, $|X| \leq k$, G is k-meager, $\mathfrak{A} = \mathsf{MP}(G)$, and $\varphi \in \mathsf{Aut}(\mathfrak{A}[[X]])$. Then there is an extension of φ that is an automorphism of $\mathfrak{A}[[\operatorname{cl}(X)]]$.*

▶ **Lemma 26.** *Let G be $2k$-meager, $X = \{\mathfrak{u}_1, \dots, \mathfrak{u}_\ell\} \subseteq W$ be $6k$-scattered, $Y \subseteq W$ such that $X \cap \operatorname{cl}(Y) = \emptyset$ and $|X| + |Y| < k$, $\mathfrak{A} = \mathsf{MP}(G)$, and $\varphi \in \mathsf{Aut}(\mathfrak{A}[[Y]])$. Then for all $\bar{u}, \bar{u}' \in F(\mathfrak{u}_1) \times \cdots \times F(\mathfrak{u}_\ell)$, there is an extension ψ of φ to $\mathfrak{A}[[X \cup Y]]$ satisfying $\psi(\bar{u}) = \bar{u}'$.*

The previous lemma turns out to be useful in the bijective k-pebble game: If the pebbles are placed on the feet in Y, we can simultaneously for all feet in X place arbitrary pebbles and still maintain a local automorphism. Such sets X will allow us to glue another graph to the multipede at the feet in X: Whatever restrictions on placing pebbles are imposed by the other graph, we still can maintain partial automorphisms in the multipede.

7.2 Gluing Multipedes to CFI Graphs

We now use multipedes to make CFI graphs asymmetric. We alter the CFI graphs in this section. Instead of two edge-vertex-pairs for the same base edge $\{\mathfrak{u}, \mathfrak{v}\}$, we contract the edges between the two vertex pairs and obtain a single edge-vertex-pair with origin $\{\mathfrak{u}, \mathfrak{v}\}$. This preserves all relevant properties of CFI graphs. In this section we write $\mathsf{CFI}(H, f)$ for CFI graphs of this modified construction. A single edge-vertex-pair per base edge removes technical details from the following.

Let $G = (V^G, W^G, E^G, \leq^G)$ be an ordered bipartite graph, $H = (V^H, E^H, \leq^H)$ be an ordered base graph, $f \colon E^H \to \mathbb{F}_2$, and $X \subseteq W^G$ have size $|X| = |E^H|$. We define the *gluing* $\mathsf{MP}(G) \cup_X \mathsf{CFI}(H, f)$ of the multipede $\mathsf{MP}(G) = (A, R^{\mathsf{MP}(G)}, \preceq^{\mathsf{MP}(G)})$ and the CFI graph $\mathsf{CFI}(H, f) = (B, E^{\mathsf{CFI}(H,f)}, \preceq^{\mathsf{CFI}(H,f)})$ at X as follows: We start with the disjoint union of $\mathsf{MP}(G)$ and $\mathsf{CFI}(H, f)$ and identify the i-th edge-vertex-pair of $\mathsf{CFI}(H, f)$ (according to $\leq^H$) with the i-th segment in X (according to $\leq^G$). We turn the edges $E^{\mathsf{CFI}(H,f)}$ into a ternary relation by extending every edge (u, v) to (u, v, v). In that way, we obtain a $\{R, \preceq\}$-structure, where R is the union of $R^{\mathsf{MP}(G)}$ and the triples (u, v, v) defined before and $\preceq$ is the total preorder obtained from combining $\preceq^{\mathsf{MP}(G)}$ and $\preceq^{\mathsf{CFI}(H,f)}$.

▶ **Lemma 27.** *If $\mathsf{MP}(G)$ is asymmetric, then $\mathsf{MP}(G) \cup_X \mathsf{CFI}(H, f)$ is asymmetric.*

Let $\bar{u}$ be a tuple of at most k vertices of $\mathsf{MP}(G) \cup_X \mathsf{CFI}(H, f)$, i.e., $\bar{u}$ contains either gadget vertices of $\mathsf{CFI}(H, f)$ or feet of $\mathsf{MP}(G)$. We call the set of segments $S(\bar{u})$ of all feet in $\bar{u}$ *directly-fixed by* $\bar{u}$ and the segments $\operatorname{cl}(S(\bar{u})) \setminus S(\bar{u})$ *closure-fixed by* $\bar{u}$. A segment $\mathfrak{u} \in X$ is *gadget-fixed by* $\bar{u}$ if the feet of $\mathfrak{u}$ are identified with an edge-vertex-pair with origin $\{\mathfrak{v}, \mathfrak{w}\}$ in $\mathsf{CFI}(H, f)$ such that there is a gadget vertex with origin $\mathfrak{v}$ or $\mathfrak{w}$ in $\bar{u}$. A segment is *fixed by* $\bar{u}$ if it is directly fixed, closure-fixed, or gadget-fixed.

▶ **Lemma 28.** *Let $r \geq k \geq 2$. If H is r-regular, G is $2k$-meager, and X is $6k$-scattered, then at most $r \cdot |\bar{u}|$ segments are fixed. If $\bar{u}$ contains i gadget vertices and ℓ segments in X are directly-fixed, then at most $|\bar{u}| - i - \ell$ segments in X are closure-fixed.*

We now combine winning strategies of Duplicator on multipedes and CFI graphs:

▶ **Lemma 29.** *Let G be $2rk$-meager, H be r-regular and at least $(k+2)$-connected, and X be $6rk$-scattered. Then $\mathsf{MP}(G) \cup_X \mathsf{CFI}(H,0) \simeq^k_{\mathcal{C}} \mathsf{MP}(G) \cup_X \mathsf{CFI}(H,1)$.*

Proof Sketch. Assume $\mathfrak{A} = \mathsf{MP}(G)$, $\mathfrak{B} = \mathsf{CFI}(H,0)$, and $\mathfrak{B}' = \mathsf{CFI}(H,1)$. We show that Duplicator has a winning strategy in the bijective k-pebble game on $\mathfrak{A} \cup_X \mathfrak{B}$ and $\mathfrak{A} \cup_X \mathfrak{B}'$. For a set of segments Y and a tuple $\bar{u}$, we denote by $\bar{u}_Y$ the restriction of $\bar{u}$ to all feet whose segment is contained in Y, by $\bar{u}_{\mathrm{G}}$ the restriction to all gadget vertices, and by $\bar{u}_{\mathrm{F}}$ to all feet. Duplicator maintains the following invariant. At every position $(\mathfrak{A} \cup_X \mathfrak{B}, \bar{u}; \mathfrak{A} \cup_X \mathfrak{B}', \bar{u}')$ in the game, there exist tuples of feet $\bar{v}_{\mathrm{gf}}, \bar{v}_{\mathrm{cf}}$ of $\mathfrak{A} \cup_X \mathfrak{B}$ and $\bar{v}'_{\mathrm{gf}}, \bar{v}'_{\mathrm{cf}}$ of $\mathfrak{A} \cup_X \mathfrak{B}'$ satisfying the following:

1. $\bar{v}_{\mathrm{gf}}$ (respectively $\bar{v}'_{\mathrm{gf}}$) contains for every segment gadget-fixed by $\bar{u}$ exactly one foot and no others.
2. $\bar{v}_{\mathrm{cf}}$ (respectively $\bar{v}'_{\mathrm{cf}}$) contains for every segment in X closure-fixed by $\bar{u}$ exactly one foot and no others.
3. There is a local isomorphism $\varphi \in \mathsf{Aut}(\mathfrak{A}[[S(\bar{u}_{\mathrm{F}}\bar{v}_{\mathrm{gf}}\bar{v}_{\mathrm{cf}})]])$ satisfying $\varphi(\bar{u}_{\mathrm{F}}\bar{v}_{\mathrm{gf}}\bar{v}_{\mathrm{cf}}) = \bar{u}'_{\mathrm{F}}\bar{v}'_{\mathrm{gf}}\bar{v}'_{\mathrm{cf}}$.
4. $(\mathfrak{B}, \bar{u}_X\bar{u}_{\mathrm{G}}\bar{v}_{\mathrm{cf}}) \simeq^k_{\mathcal{C}} (\mathfrak{B}', \bar{u}'_X\bar{u}'_{\mathrm{G}}\bar{v}'_{\mathrm{cf}})$.
5. For every base vertex $\mathfrak{u}$, it holds that $(\mathfrak{B}', \bar{u}_{\mathrm{G}}\bar{v}_{\mathrm{gf}})[V_{\mathfrak{u}}] \cong (\mathfrak{B}, \bar{u}'_{\mathrm{G}}\bar{v}'_{\mathrm{gf}})[V_{\mathfrak{u}}]$, where $V_{\mathfrak{u}}$ is the set of all gadget vertices with origin $\mathfrak{u}$ and all edge vertices with origin $\{\mathfrak{u}, \mathfrak{v}\}$ for some $\mathfrak{v}$.

For Property 3, note that $S(\bar{u}_{\mathrm{F}}\bar{v}_{\mathrm{gf}}\bar{v}_{\mathrm{cf}}) = S(\bar{u}'_{\mathrm{F}}\bar{v}'_{\mathrm{gf}}\bar{v}'_{\mathrm{cf}})$ and $|\bar{u}_{\mathrm{F}}\bar{v}_{\mathrm{gf}}\bar{v}_{\mathrm{cf}}| = |\bar{u}'_{\mathrm{F}}\bar{v}'_{\mathrm{gf}}\bar{v}'_{\mathrm{cf}}| \leq rk$ by Lemma 28 because G is $2rk$-meager. For Property 4, note that $|\bar{u}_X\bar{u}_{\mathrm{G}}\bar{v}_{\mathrm{cf}}| = |\bar{u}'_X\bar{u}'_{\mathrm{G}}\bar{v}'_{\mathrm{cf}}| \leq k$: By Lemma 28, the number of closure-fixed segments in X is at most $|\bar{v}_{\mathrm{cf}}| \leq k - |\bar{u}_{\mathrm{G}}| - |\bar{u}_X|$. Property 5 guarantees that the vertices $\bar{v}_{\mathrm{gf}}$ and $\bar{v}'_{\mathrm{gf}}$ are picked consistently. This is needed because $|\bar{v}_{\mathrm{gf}}|$ exceeds k and thus cannot be included in Property 4.

We play two games. Game I is played with rk pebbles on $(\mathfrak{A}, \bar{u}_{\mathrm{F}}\bar{v}_{\mathrm{gf}}\bar{v}_{\mathrm{cf}}; \mathfrak{A}, \bar{u}'_{\mathrm{F}}\bar{v}'_{\mathrm{gf}}\bar{v}'_{\mathrm{cf}})$. Game II is played with k pebbles on $(\mathfrak{B}, \bar{u}_X\bar{u}_{\mathrm{G}}\bar{v}_{\mathrm{cf}}; \mathfrak{B}', \bar{u}'_X\bar{u}'_{\mathrm{G}}\bar{v}'_{\mathrm{cf}})$. From the winning strategies of Duplicator in both games (Lemmas 4 and 23) we construct a winning strategy on $(\mathfrak{A} \cup_X \mathfrak{B}, \bar{u}; \mathfrak{A} \cup_X \mathfrak{B}', \bar{u}')$. We can do so because in Game I we fixed all gadget-fixed segments and in Game II we fixed all closure-fixed segments in X. When placing a pebble on a gadget vertex, we extend the tuples $\bar{v}_{\mathrm{gf}}$ and $\bar{v}'_{\mathrm{gf}}$ using Lemma 26. When a pebble is placed on a segment not in X, at most one segment in X gets closure fixed and the edge vertices are in the same orbit of the CFI graphs (Lemma 5). ◀

▶ **Theorem 30.** *There is an* FO*-interpretation Θ and, for every $k \in \mathbb{N}$, a pair of ternary $\{R, \preceq\}$-structures $(\mathfrak{A}_k, \mathfrak{B}_k)$ such that $\preceq$ is a total preorder, $\mathfrak{A}_k$ and $\mathfrak{B}_k$ are asymmetric, $\mathfrak{A}_k \simeq^k_{\mathcal{C}} \mathfrak{B}_k$, $\mathfrak{A}_k \ncong \mathfrak{B}_k$, and $\Theta(\mathfrak{A}_k) \ncong \Theta(\mathfrak{B}_k)$ are CFI graphs of the same ordered base graph.*

Proof Sketch. We use the gluings constructed before for suitable base graphs and multipedes (Lemma 24). By Lemma 29, the odd and even gluing are $\mathcal{C}_k$-equivalent (but surely not isomorphic) and asymmetric by Lemmas 22 and 27. There is an FO-interpretation Θ removing the multipede: Shorten R-triples (u, v, v) back to edges (u, v) and remove the others. ◀

Proof Sketch of Theorem 1. Consider the $\{R, \preceq\}$-structures $\mathcal{K}$ from Theorem 30. To ensure that the reduct semantics does not add automorphisms, $\preceq$ is encoded into R by attaching paths of different lengths to the vertices. This preserves asymmetry and non-isomorphism. The paths are removed by an FO-interpretation $\Theta_{\mathcal{K}}$. All structures are asymmetric and have a single relation, so IFPC = IFPC+WSC and IFPC does not define isomorphism.

Let Φ_{CFI} be a WSCI(IFPC) = WSC(IFPC)-formula defining the CFI query for ordered base graphs (Corollary 13) and let Θ_{CFI} be the FO-interpretation extracting the CFI graphs from $\mathcal{K}$-structures given by Theorem 30. Then the I(WSC(IFPC))-formula $\mathsf{I}(\Theta_{\text{CFI}} \circ \Theta_{\mathcal{K}}; \Phi_{\text{CFI}})$ defines the isomorphism problem of $\mathcal{K}$-structures. ◀

▶ **Corollary 31.** IFPC+WSC $<$ *Ptime.*

▶ **Corollary 32.** WSC(IFPC) $<$ I(WSC(IFPC)).

Note that the prior corollary refines Corollary 21. We actually expect that

$$\text{WSC(IFPC)} < \text{I(WSC(IFPC))} < \text{WSC(I(WSC(IFPC)))}$$

because it seems unlikely that I(WSC(IFPC)) defines the CFI query of the base graphs of Theorem 3.

▶ **Corollary 33.** IFPC+WSC *is not closed under* IFPC*-interpretations and not even under* 1*-dimensional equivalence-free* FO*-interpretations.*

Using the same structures, we can answer an open question of Dawar and Richerby in [9]:

▶ **Corollary 34.** IFP+SC *is not closed under* 1*-dimensional equivalence-free* FO*-interpretations.*

8 Discussion

We defined the logics IFPC+WSC and IFPC+WSC+I to study the combination of witnessed symmetric choice and interpretations beyond simulating counting. Instead, we provided graph constructions to prove lower bounds. IFPC+WSC+I canonizes CFI graphs if it canonizes the base graphs, but operators have to be nested. We proved that this increase in nesting depth is unavoidable using double CFI graphs obtained by essentially applying the CFI construction twice. Does iterating our construction further show an operator nesting hierarchy in IFPC+WSC+I? We have seen that also in the presence of counting the interpretation operator strictly increases the expressiveness. So indeed both, witnessed symmetric choice and interpretations are needed to possibly capture Ptime. This answers the question to the relation between witnessed symmetric choice and interpretations for IFPC. But it remains open whether IFPC+WSC+I captures Ptime. Here, iterating our CFI construction is of interest again: If one shows an operator nesting hierarchy using this construction, then one in particular will separate IFPC+WSC+I from Ptime because our construction does not change the signature of the structures. Studying this remains for future work.

References

1 Markus Anders and Pascal Schweitzer. Search problems in trees with symmetries: Near optimal traversal strategies for individualization-refinement algorithms. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 16:1–16:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.16.

2 Vikraman Arvind and Somenath Biswas. Expressibility of first order logic with a nondeterministic inductive operator. In *STACS 87, 4th Annual Symposium on Theoretical Aspects of Computer Science, Passau, Germany, February 19-21, 1987, Proceedings*, volume 247 of *Lecture Notes in Computer Science*, pages 323–335. Springer, 1987. doi:10.1007/BFb0039616.

3 Andreas Blass and Yuri Gurevich. The logic of choice. *J. Symb. Log.*, 65(3):1264–1310, 2000. doi:10.2307/2586700.

4 Andreas Blass, Yuri Gurevich, and Saharon Shelah. On polynomial time computation over unordered structures. *J. Symb. Log.*, 67(3):1093–1125, 2002. doi:10.2178/jsl/1190150152.

5 Jin-yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992. doi:10.1007/BF01305232.

6 Ashok K. Chandra and David Harel. Structure and complexity of relational queries. *J. Comput. Syst. Sci.*, 25(1):99–128, 1982. doi:10.1016/0022-0000(82)90012-5.

7 Anuj Dawar, Erich Grädel, and Moritz Lichter. Limitations of the invertible-map equivalences. *J. Log. Comput.*, 2022. doi:10.1093/logcom/exac058.

8 Anuj Dawar and Kashif Khan. Constructing hard examples for graph isomorphism. *J. Graph Algorithms Appl.*, 23(2):293–316, 2019. doi:10.7155/jgaa.00492.

9 Anuj Dawar and David Richerby. A fixed-point logic with symmetric choice. In *Computer Science Logic, 17th International Workshop, CSL 2003, 12th Annual Conference of the EACSL, and 8th Kurt Gödel Colloquium, KGC 2003, Vienna, Austria, August 25-30, 2003, Proceedings*, volume 2803 of *Lecture Notes in Computer Science*, pages 169–182. Springer, 2003. doi:10.1007/978-3-540-45220-1_16.

10 Anuj Dawar and David Richerby. Fixed-point logics with nondeterministic choice. *J. Log. Comput.*, 13(4):503–530, 2003. doi:10.1093/logcom/13.4.503.

11 Anuj Dawar and David Richerby. The power of counting logics on restricted classes of finite structures. In *Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL, Lausanne, Switzerland, September 11-15, 2007, Proceedings*, volume 4646 of *Lecture Notes in Computer Science*, pages 84–98. Springer, 2007. doi:10.1007/978-3-540-74915-8_10.

12 Anuj Dawar, David Richerby, and Benjamin Rossman. Choiceless Polynomial Time, counting and the Cai-Fürer-Immerman graphs. *Ann. Pure Appl. Logic*, 152(1-3):31–50, 2008. doi:10.1016/j.apal.2007.11.011.

13 Heinz-Dieter Ebbinghaus. Extended logics: the general framework. In *Model-Theoretic Logics*, Perspectives in MathematicalLogic, pages 25–76. Association for Symbolic Logic, 1985.

14 Ronald Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In *Complexity of computation (Proc. SIAM-AMS Sympos. Appl. Math., New York, 1973)*, pages 43–73. SIAM–AMS Proc., Vol. VII, 1974.

15 Françoise Gire and H. Khanh Hoang. An extension of fixpoint logic with a symmetry-based choice construct. *Inf. Comput.*, 144(1):40–65, 1998. doi:10.1006/inco.1998.2712.

16 Erich Grädel, Phokion G. Kolaitis, Leonid Libkin, Maarten Marx, Joel Spencer, Moshe Y. Vardi, Yde Venema, and Scott Weinstein. *Finite Model Theory and Its Applications*. Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin, Heidelberg, 2007. doi:10.1007/3-540-68804-8.

17 Erich Grädel and Wied Pakusa. Rank logic is dead, long live rank logic! *J. Symb. Log.*, 84(1):54–87, 2019. doi:10.1017/jsl.2018.33.

18 Erich Grädel, Wied Pakusa, Svenja Schalthöfer, and Lukasz Kaiser. Characterising Choiceless Polynomial Time with first-order interpretations. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 677–688. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.68.

19 Martin Grohe. The quest for a logic capturing PTIME. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*, pages 267–271. IEEE Computer Society, 2008. doi:10.1109/LICS.2008.11.

20 Martin Grohe. *Descriptive Complexity, Canonization, and Definable Graph Structure Theory*. Cambridge University Press, 2017.

21 Martin Grohe and Daniel Neuen. Canonisation and definability for graphs of bounded rank width. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–13. IEEE Computer Society, 2019. doi:10.1109/LICS.2019.8785682.

22 Yuri Gurevich. Logic and the challenge of computer science. In *Current Trends in Theoretical Computer Science*, pages 1–57. Computer Science Press, 1988.

23 Yuri Gurevich. From invariants to canonization. *Bull. EATCS*, 63, 1997.

24 Yuri Gurevich and Saharon Shelah. On finite rigid structures. *J. Symb. Log.*, 61(2):549–562, 1996. doi:10.2307/2275675.

25 Lauri Hella. Logical hierarchies in PTIME. *Information and Computation*, 129(1):1–19, 1996. doi:10.1006/inco.1996.0070.

26 Neil Immerman. Expressibility as a complexity measure: results and directions. In *Proceedings of the Second Annual Conference on Structure in Complexity Theory, Cornell University, Ithaca, New York, USA, June 16-19, 1987*. IEEE Computer Society, 1987.

27 Neil Immerman. Languages that capture complexity classes. *SIAM J. Comput.*, 16(4):760–778, 1987. doi:10.1137/0216051.

28 Sandra Kiefer, Pascal Schweitzer, and Erkal Selman. Graphs identified by logics with counting. In *Mathematical Foundations of Computer Science 2015 – 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part I*, volume 9234 of *Lecture Notes in Computer Science*, pages 319–330. Springer, 2015. doi:10.1007/978-3-662-48057-1_25.

29 Moritz Lichter. Separating rank logic from polynomial time. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 – July 2, 2021*, pages 1–13. IEEE Computer Society, 2021. doi:10.1109/LICS52264.2021.9470598.

30 Moritz Lichter. Witnessed symmetric choice and interpretations in fixed-point logic with counting. *CoRR*, abs/2210.07869, 2022. arXiv preprint. doi:10.48550/arXiv.2210.07869.

31 Moritz Lichter and Pascal Schweitzer. Canonization for bounded and dihedral color classes in Choiceless Polynomial Time. In *29th EACSL Annual Conference on Computer Science Logic, CSL 2021, January 25-28, 2021, Ljubljana, Slovenia (Virtual Conference)*, volume 183 of *LIPIcs*, pages 31:1–31:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CSL.2021.31.

32 Moritz Lichter and Pascal Schweitzer. Choiceless Polynomial Time with witnessed symmetric choice. In *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2–5, 2022*, pages 30:1–30:13. ACM, 2022. doi:10.1145/3531130.3533348.

33 Rudolf Mathon. A note on the graph isomorphism counting problem. *Inf. Process. Lett.*, 8(3):131–132, 1979. doi:10.1016/0020-0190(79)90004-8.

34 Daniel Neuen and Pascal Schweitzer. An exponential lower bound for individualization-refinement algorithms for graph isomorphism. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 138–150. ACM, 2018. doi:10.1145/3188745.3188900.

35 Martin Otto. *Bounded variable logics and counting – A study in finite models*, volume 9 of *Lecture Notes in Logic*. Springer, 1997.

36 Martin Otto. Epsilon-logic is more expressive than first-order logic over finite structures. *J. Symb. Log.*, 65(4):1749–1757, 2000. doi:10.2307/2695073.

37 Wied Pakusa, Svenja Schalthöfer, and Erkal Selman. Definability of Cai-Fürer-Immerman problems in Choiceless Polynomial Time. In *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 – September 1, 2016, Marseille, France*, volume 62 of *LIPIcs*, pages 19:1–19:17. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.CSL.2016.19.

38 David Richerby. *Fixed-point logics with choice*. PhD thesis, University of Cambridge, 2004.

39 Pascal Schweitzer and Constantin Seebach. Resolution with symmetry rule applied to linear equations. In *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference)*, volume 187 of *LIPIcs*, pages 58:1–58:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.STACS.2021.58.

40 Jacobo Torán. On the resolution complexity of graph non-isomorphism. In *Theory and Applications of Satisfiability Testing – SAT 2013 – 16th International Conference, Helsinki, Finland, July 8-12, 2013. Proceedings*, volume 7962 of *Lecture Notes in Computer Science*, pages 52–66. Springer, 2013. doi:10.1007/978-3-642-39071-5_6.

41 Jacobo Torán and Florian Wörz. Number of variables for graph differentiation and the resolution of GI formulas. In *30th EACSL Annual Conference on Computer Science Logic, CSL 2022, February 14-19, 2022, Göttingen, Germany (Virtual Conference)*, volume 216 of *LIPIcs*, pages 36:1–36:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CSL.2022.36.

42 Faried Abu Zaid, Erich Grädel, Martin Grohe, and Wied Pakusa. Choiceless Polynomial Time on structures with small abelian colour classes. In *Mathematical Foundations of Computer Science 2014 – 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part I*, volume 8634 of *Lecture Notes in Computer Science*, pages 50–62. Springer, 2014. doi:10.1007/978-3-662-44522-8_5.

On the Complexity of Diameter and Related Problems in Permutation Groups

Markus Lohrey
Universität Siegen, Germany

Andreas Rosowski
Universität Siegen, Germany

Abstract

We prove that it is Π_2^{P}-complete to verify whether the diameter of a given permutation group $G = \langle A \rangle$ is bounded by a unary encoded number k. This solves an open problem from a paper of Even and Goldreich, where the problem was shown to be NP-hard. Verifying whether the diameter is exactly k is complete for the class consisting of all intersections of a Π_2^{P}-language and a Σ_2^{P}-language. A similar result is shown for the length of a given permutation π, which is the minimal k such that π can be written as a product of at most k generators from A. Even and Goldreich proved that it is NP-complete to verify, whether the length of a given π is at most k (with k given in unary encoding). We show that it is DP-complete to verify whether the length is exactly k. Finally, we deduce from our result on the diameter that it is Π_2^{P}-complete to check whether a given finite automaton with transitions labelled by permutations from S_n produces all permutations from S_n.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases algorithms for finite groups, diameter of permutation groups, rational subsets in groups

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.134

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Funding Both authors were supported by the DFG research project Lo748/12-2.

1 Introduction

Algorithmic problems for finite groups, in particular permutation groups, are an active research at the borderline between mathematics and theoretical computer science. Among the many applications of permutation group algorithms in computer science, let us just mention the work on the graph isomorphism problem that culminated with Babai's quasi-polynomial time algorithm [2]. For a comprehensive introduction into the area permutation group algorithms, see Serres' textbook [23]. In this paper we are concerned with algorithmic problems related to the diameter of finite permutation groups. We start with a few basic definitions.

Let G be a finite group. For a subset $A \subseteq G$ we denote with $\langle A \rangle$ the subgroup of G generated by the elements from A (i.e., the closure of A under the group multiplication). If $\langle A \rangle = G$ then A is called a generating set of G. For $k \geq 0$ we write $A^{\leq k}$ for the set of all products $a_1 a_2 \cdots a_l \in G$ with $l \leq k$ and $a_1, \ldots, a_l \in A$. For an element $g \in \langle A \rangle$ we denote with $|g|_A$ (the A-length of g) the smallest integer k such that $g \in A^{\leq k}$. The diameter $d(G, A)$ of G with respect to the generating A is the smallest number d such that $\langle A \rangle = A^{\leq d}$. Note that such a d exists since G is finite. There is a quite extensive literature on upper and lower bounds on the diameter in various finite groups; see e.g. [3, 4, 5, 6, 7, 8, 9, 13, 17, 19]. Let us mention in this context a famous (and still open) conjecture of Babai and Seress [8] stating that for every finite non-abelian simple group G and every generating set A, $d(G, A)$

is bounded by $\mathcal{O}((\log |G|)^c)$ for some universal constant c. This would imply in particular that the diameters of A_n and S_n (with respect to any generating sets) are bounded by a polynomial in n. The currently best known upper bound is $\exp(\mathcal{O}(\log^4 n \log\log n))$ [13].

Many problems about mechanical puzzles reduce to questions about the diameter of finite groups. As an example let us mention Rubik's cube. For a long time it was open how many moves in Rubik's cube are needed to transform an arbitrary initial configuration into the target configuration. This number is simply the diameter of the so-called Rubik's cube group. The precise value of this diameter was open for a long time. In 2013 Rokicki et al. proved that is 20 [22].

1.1 Computing diameter and length

In the first part of the paper (Sections 3 and 4) we investigate the complexity of certain decision variants of the following computational problems:

(i) computing the length of a given element from a permutation group and

(ii) computing the diameter of a permutation group.

Let us define the problems that we will investigate more precisely. With S_n we denote the group of all permutations on $[1, n] = \{1, \ldots, n\}$. In the following problems, a permutation $\pi \in S_n$ is given by the list $\pi(1), \pi(2), \ldots, \pi(n)$. The size n of the domain (also called the degree of the permutations) is part of the input. For $A \subseteq S_n$ we write $d(A)$ for $d(\langle A \rangle, A)$. We then define the following computational problems:

▶ **Problem** (UNARY LENGTH).
Input: a set of permutations $A \subseteq S_n$, an element $\pi \in \langle A \rangle$, and a unary encoded number k.[1]
Question: Is $|\pi|_A \leq k$?

▶ **Problem** (UNARY DIAMETER).
Input: a set of permutations $A \subseteq S_n$ and a unary encoded number k.
Question: Is $d(A) \leq k$?

The problems BINARY LENGTH and BINARY DIAMETER are defined in the same way, except that the input number k is given in binary encoding.

Goldreich and Even [11] were the first who obtained results on the complexity of these problems. They proved that UNARY LENGTH is NP-complete and UNARY DIAMETER is NP-hard[2] but the exact complexity of UNARY DIAMETER remained open. A parameterized variant of UNARY LENGTH (with k as the parameter) is studied under the name PERMUTATION GROUP FACTORIZATION in [10] and shown to be W[1]-hard and in W[P]. The binary setting was first studied by Jerrum [15]. He proved that BINARY LENGTH is PSPACE-complete.

We also study exact versions of the above problems:

▶ **Problem** (UNARY EXACT LENGTH).
Input: a set of permutations $A \subseteq S_n$, a permutation $\pi \in \langle A \rangle$, and a unary encoded number k.
Question: Is $|\pi|_A = k$?

[1] It is well-known that there is a polynomial time algorithm that checks whether $\pi \in \langle A \rangle$ holds [12].

[2] The problem UNARY LENGTH is called MGS for "minimum generator sequence" in [11], whereas UNARY DIAMETER is called MBGS for "minimum upper bound on generator sequences". We believe that UNARY LENGTH and UNARY DIAMETER are more suggestive. Another point is that Even and Goldreich do not specify the encoding of integers in their paper, but from the NP-completeness result for UNARY LENGTH, it can deduced that they have the unary encoding of integers in mind.

▶ **Problem** (UNARY EXACT DIAMETER).
Input: a set of permutations $A \subseteq S_n$ and a unary encoded number k.
Question: Is $d(A) = k$?

Again, there are corresponding problems BINARY EXACT LENGTH and BINARY EXACT DIAMETER, where the input number k is given in binary notation.

The first main result of this paper solves the open problem left in [11]: UNARY DIAMETER is Π_2^{P}-complete, where Π_2^{P} is the second universal level of the polynomial time hierarchy. This result also holds for the restriction, where all permutations in the set $A \subseteq S_n$ pairwise commute and have order two (and hence $\langle A \rangle$ is an abelian group of exponent two). Moreover, we also show that BINARY DIAMETER with a set A of pairwise commuting permutations is Π_2^{P}-complete.

The complexity of BINARY DIAMETER for general permutation groups remains open. The problem is easily seen to be in PSPACE. The above mentioned result of Jerrum (PSPACE-completeness of BINARY LENGTH for a binary encoded number k) seems to have no implications for the complexity of BINARY DIAMETER. Nevertheless, we conjecture that BINARY DIAMETER is PSPACE-complete.

We then proceed to show that UNARY EXACT DIAMETER is complete for the complexity class DP_2, which is the class of all intersections of a Π_2^{P}-language and a Σ_2^{P}-language. Hardness for DP_2 already holds for the restriction of UNARY EXACT DIAMETER to abelian permutation groups of exponent two. To get DP_2-hardness, we use the fact that our Π_2^{P}-hardness proof of UNARY DIAMETER already holds for inputs $A \subseteq S_n$ and $k \in \mathbb{N}$ with the promise that the diameter of $\langle A \rangle$ is either k or $k+1$. Using similar techniques we can also show that UNARY EXACT LENGTH is DP-complete, where DP is the class of all intersections of an NP-language and a coNP-language.

1.2 Equality and universality for finite automata over permutation groups

In the second part of the paper (Section 5), we consider problems related to finite automata over permutation groups. The setting is as follows: Consider a nondeterministic finite automaton (NFA) $\mathcal{A}$ over a finite alphabet Σ of input letters and a mapping $h : \Sigma \to S_n$ to a symmetric group. The mapping h extends to a morphism $h : \Sigma^* \to S_n$ from the free monoid Σ^* to the group S_n (we use the same letter h for this extension). We may then ask whether a given permutation π belongs to $h(L(\mathcal{A}))$. This is the *rational subset membership problem for permutation groups*, where the input consists of the NFA $\mathcal{A}$, the mapping $h : \Sigma \to S_n$ (n is also part of the input) and the permutation π. It is shown in [16, 18] that the rational subset membership problem for permutation groups is NP-complete.[3]

To simplify notation, we omit the mapping $h : \Sigma^* \to S_n$ in the following, and replace in the NFA $\mathcal{A}$ every transition label $a \in \Sigma$ by the corresponding permutation $h(a) \in S_n$. Thus, we consider NFAs, where the transitions are labelled with elements of a symmetric group S_n. The set $L(\mathcal{A})$ accepted by $\mathcal{A}$ is then directly interpreted as a subset of S_n. Clearly, every subset of S_n is of the form $L(\mathcal{A})$ for an NFA $\mathcal{A}$ over S_n, but in general the number of

[3] In [18] stronger results are shown: (i) NP-hardness already holds for membership in sets $\pi^*\sigma^*\tau^*$, where π, σ, τ are input permutations, and (ii) membership in NP holds for black-box groups and a restricted class of context-free languages (where terminal symbols are again replaced by permutations). The general membership problem for context-free sets of permutations is PSPACE-complete [18].

transitions of $\mathcal{A}$ must be exponential in n (this follows from a simple counting argument). Note that for a finite set $A \subseteq S_n$ it is straightforward to come up with an automaton $\mathcal{A}$ over S_n with a single state and $|A|$ many transitions such that $L(\mathcal{A}) = \langle A \rangle$.

In Section 5, we consider the *rational equality problem for permutation groups*, RATIONAL EQUALITY for short:

▶ **Problem** (RATIONAL EQUALITY).
Input: two NFAs $\mathcal{A}$ and $\mathcal{B}$ over S_n (n is as usual part of the input).
Question: Does $L(\mathcal{A}) = L(\mathcal{B})$ hold?

As before, we also consider the abelian variant of this problem, where all permutations labelling the transitions of $\mathcal{A}$ and $\mathcal{B}$ pairwise commute. Moreover, we consider the following restriction of RATIONAL EQUALITY.

▶ **Problem** (RATIONAL UNIVERSALITY).
Input: an NFA $\mathcal{A}$ over S_n.
Question: Does $L(\mathcal{A}) = S_n$ hold?

Note that for RATIONAL UNIVERSALITY, the restriction where the permutations appearing in $\mathcal{A}$ pairwise commute is not interesting, since S_n is not abelian for $n \geq 3$.

We show that RATIONAL EQUALITY and RATIONAL UNIVERSALITY are both Π_2^{P}-complete and that Π_2^{P}-hardness for RATIONAL EQUALITY already holds for the abelian case. For the lower bounds we use reductions from UNARY DIAMETER.

Let us finally remark that our upper bound proofs do not use any specific properties of permutation groups. In particular, all upper bounds shown in this paper also hold for the black-box-group setting, where elements of a black-box group G are encoded by bit strings and there are oracles for (i) multiplying two elements of G, (ii) inverting an element of G, and (iii) checking whether two bit strings represent the same element of G (see [23] for more details on black-box groups).

2 Preliminaries

2.1 Background from complexity theory

We assume that the reader has some basic background from complexity theory, see e.g. [1] for more information. The levels Σ_k^{P} and Π_k^{P} of the *polynomial time hierarchy* [24] are defined as follows:

- $\Sigma_0^{\mathsf{P}} = \Pi_0^{\mathsf{P}} = \mathsf{P}$
- $\Sigma_{k+1}^{\mathsf{P}}$ is the set of all languages L such that there exists a language $K \in \Pi_k^{\mathsf{P}}$ and a polynomial p with $L = \{x \mid \exists y \in \{0,1\}^{p(|x|)} : x\#y \in K\}$ (here $\#$ is a separator symbol).
- Π_{k+1}^{P} is the set of all languages L such that there exists a language $K \in \Sigma_k^{\mathsf{P}}$ and a polynomial p with $L = \{x \mid \forall y \in \{0,1\}^{p(|x|)} : x\#y \in K\}$.

In particular, we have $\Sigma_1^{\mathsf{P}} = \mathsf{NP}$ and $\Pi_1^{\mathsf{P}} = \mathsf{coNP}$. We will make use of the computational problem $\forall\exists$SAT, where the input is a $\forall\exists$-formula

$$\Psi = \forall x_1 \cdots \forall x_n \exists y_1 \cdots \exists y_m F(x_1, \ldots, x_n, y_1, \ldots, y_m), \tag{1}$$

where F is a boolean formula in conjunctive normal form built from the boolean variables $x_1, \ldots, x_n, y_1, \ldots, y_m$. The question is whether Ψ holds. This problem is Π_2^{P}-complete [24].

The complexity class DP_k is defined as

$$\mathsf{DP}_k = \{L_1 \cap L_2 \mid L_1 \in \Sigma_k^{\mathsf{P}} \text{ and } L_2 \in \Pi_k^{\mathsf{P}}\}.$$

The class $\mathsf{DP}_1 = \{L_1 \cap L_2 \mid L_1 \in \mathsf{NP} \text{ and } L_2 \in \mathsf{coNP}\}$ is usually denoted by DP. It was defined in [21]. The only mentioning of the classes DP_k for $k \geq 2$ we are aware of is the stack exchange post [20].

2.2 Some notations for permutation groups

Recall that a permutation group is a subgroup of the *symmetric group* S_n for some n, where S_n is the group of all permutations on $[1, n] = \{1, \ldots, n\}$. We will use standard notations for permutation groups; see e.g., [23]. Permutations will be often written by their decomposition into disjoint cycles, called the *disjoint cycle decomposition*. A cycle of length two is a *transposition*. A product of permutations will be evaluated from left to right. For $a \in [1, n]$ and $\pi \in S_n$ we will also write a^π for $\pi(a)$. This fits nicely to the left-to-right evaluation order: $a^{\pi\tau} = (a^\pi)^\tau$. For a permutation π we denote by $\mathrm{ord}(\pi)$ the order of π, i.e., the smallest $k \geq 1$ such that π^k is the identity permutation.

Most of the hardness results in Sections 3 and 4 will be shown for the group $\mathbb{Z}_2^n$ (the n-fold direct product of the group $\mathbb{Z}_2$) for an $n \geq 0$. Clearly, this is an abelian group of exponent two (i.e., every element has order two). The group $\mathbb{Z}_2^n$ is isomorphic to the subgroup of S_{2n} generated by all transpositions $(2i-1, 2i)$ for $i \in [1, n]$. For a finite set V of size n, we will identify $\mathbb{Z}_2^n$ with the group $\mathbb{Z}_2^V$ of all mappings $f : V \to \mathbb{Z}_2$ with the group operation to be pointwise addition modulo 2. We write this abelian group additively. For a function $f : V \to \mathbb{Z}_2 = \{0, 1\}$ we define its support as $\mathrm{supp}(f) = \{v \in V \mid f(v) = 1\}$. For a subset $U \subseteq V$ we denote by $[U] \in \mathbb{Z}_2^V$ the unique group element with $\mathrm{supp}([U]) = U$.

3 Complexity of diameter for permutation groups

We come to the first main result of the paper: UNARY DIAMETER is Π_2^{P}-hard. In the following theorem, the additional statement that the diameter $d(A)$ is either k or $k+1$ will be needed later when we consider UNARY EXACT DIAMETER.

▶ **Theorem 3.1.** *There is a logspace reduction ϕ from $\forall\exists$SAT to* UNARY DIAMETER *such that for every $\forall\exists$-formula Ψ with $\phi(\Psi) = (A, k)$ we have: $A \subseteq \mathbb{Z}_2^n$ for some n, $\langle A \rangle = \mathbb{Z}_2^n$ and $d(A) \in \{k, k+1\}$.*

Proof. Let us fix a $\forall\exists$-formula Ψ as in (1). We can write F as $F = \bigwedge_{c \in C} c$, where C is a set of clauses (disjunctions of variables and negated variables). We start with several transformations that ensure some additional properties for Ψ.

Step 1. In order to bound the diameter of the group from above by $k+1$ we replace Ψ by the formula

$$\forall x_1 \cdots \forall x_n \exists y_1 \cdots \exists y_m \exists y^* \, G(x_1, \ldots, x_n, y_1, \ldots, y_m, y^*),$$

where y^* is a new variable and

$$G = \neg y^* \wedge \bigwedge_{c \in C} (y^* \vee c).$$

By this it is ensured that for every truth assignment of the universally quantified variables $x_1, \ldots, x_n$, there is a truth assignment of the existentially quantified variables $y_1, \ldots, y_m, y^*$ such that exactly one clause in G is unsatisfied (simply set y^* to the true value 1).

Step 2. Next, it is necessary to ensure that every variable appears in at most d clauses for a fixed constant d. This can be ensured similarly to [25] for 3SAT: for every variable $z \in \{x_1, \ldots, x_n, y_1, \ldots, y_m, y^*\}$ that appears in $l \geq 4$ clauses in G we introduce new variables $z_1, \ldots, z_l$ and replace the i-th occurrence of z by z_i. Then we add the clauses

$$(\neg z_1 \vee z_2), (\neg z_2 \vee z_3), \ldots, (\neg z_{l-1} \vee z_l), (\neg z_l \vee z_1) \tag{2}$$

which enforce that $z_1, \ldots, z_l$ must get the same truth value. If $z \in \{x_1, \ldots, x_n\}$ then we universally quantify z_1 and existentially quantify $z_2, \ldots, z_l$. If $z \in \{y_1, \ldots, y_m, y^*\}$ then all new variables z_i get existentially quantified. In the resulting formula, every variable occurs in at most 3 clauses.

Step 3. Finally, for our later arguments, it is necessary to add for every universally quantified variable x the trivial clause

$$c_x = (x \vee \neg x).$$

Of course, this trivial clause does not change the truth value of the formula. Now every variable occurs in at most 4 clauses. We still have the property that every truth assignment for the universally quantified variables can be extended by a truth assignment for the existentially quantified variables such that exactly one clause becomes unsatisfied. To see this, consider an arbitrary truth assignment for the universally quantified variables. The clauses c_x that we added in Step 3 are always satisfied. We now assign the truth value 1 to all variables y_i^* that replaced in Step 2 the variable y^* from Step 1. This ensures that all clauses that were derived from clauses $y^* \vee c$ with $c \in C$ in Step 2 are satisfied. All remaining clauses of the form (2) (with $z \neq y^*$) can be easily satisfied. If z_1 is universally quantified (so its truth value is already fixed) then all $z_2, \ldots, z_l$ are existentially quantified and we assign to these variables the truth value of z_1. Otherwise $z_1, \ldots, z_l$ are all existentially quantified and we can assign the truth value 1 to all of them (the truth value 0 would also work). At this point, only the single clause derived from $\neg y^*$ in Step 2 is not satisfied. Finally, note that each of the three steps preserves the truth value of the $\forall\exists$-formula.

This concludes the preprocessing of the $\forall\exists$-formula Ψ. To simplify notation, we denote the resulting formula again with

$$\Psi = \forall x_1 \cdots \forall x_n \exists y_1 \cdots \exists y_m F(x_1, \ldots, x_n, y_1, \ldots, y_m). \tag{3}$$

Let $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$. For a variable $z \in X \cup Y$ we denote with $\tilde{z}$ one of the literals z or $\neg z$. Moreover we denote by C the set of clauses of F. A clause is viewed as a set of literals. For a literal $\tilde{z}$ let $C(\tilde{z})$ be the set of all clauses containing $\tilde{z}$. Note that every set $C(\tilde{z})$ has size at most 4. Let

$$V = X \cup Y \cup C$$

(we assume that X, Y, C are pairwise disjoint). We will work in the group $\mathbb{Z}_2^V$ introduced in Section 2.2 and use the notation introduced there.

The logspace reduction ϕ from $\forall\exists$SAT to UNARY DIAMETER is defined by $\phi(\Psi) = (A, k)$ with $k = n + m$ and

$$A = \bigcup_{x \in X} A_x \cup \bigcup_{y \in Y} A_y,$$

where for all universally quantified variables $x \in X$,

$$A_x = \{[\{x\} \cup U] \mid U \subseteq C(x)\} \cup \{[U] \mid U \subseteq C(\neg x), U \neq \emptyset\}$$

and for all existentially quantified variables $y \in Y$,

$$A_y = \{[U] \mid U \subseteq \{y\} \cup C(y), U \neq \emptyset\} \cup \{[U] \mid U \subseteq \{y\} \cup C(\neg y), U \neq \emptyset\}.$$

Since every set $C(\tilde{z})$ has size at most 4, we can construct the instance (A, k) in logspace.

▷ Claim 3.2. $\langle A \rangle = \mathbb{Z}_2^V$.

Proof of Claim 3.2. It suffices to show that every $[\{v\}]$ for $v \in V$ belongs to $\langle A \rangle$. From the definition of A we immediately get $[\{x\}] \in A_x$ for all $x \in X$ and $[\{y\}] \in A_y$ for all $y \in Y$. Consider now a clause $c \in C$ and fix a literal $\tilde{z} \in c$. We have $c \in C(\tilde{z})$. If $z \in Y$ then $[\{c\}] \in A_z$. If $z \in X$ and $\tilde{z} = \neg z$, then again $[\{c\}] \in A_z$. Finally, if $z \in X$ and $\tilde{z} = z$ then $[\{z\}], [\{z, c\}] \in A_z$. Hence, $[\{c\}] = [\{z\}] + [\{z, c\}] \in \langle A \rangle$. ◁

▷ Claim 3.3. Let $f_X : X \to \{0, 1\}$ and $f : V \to \{0, 1\}$ be elements of $\mathbb{Z}_2^V$ with

$$f(x) = f_X(x)$$

for all $x \in X$ and

$$f(v) = 1$$

for all $v \in Y \cup C$. If $f \in A^{\leq n+m}$, then there is a function $f_Y : Y \to \{0, 1\}$ such that $f_X + f_Y$ is a satisfying truth assignment for F.

Proof of Claim 3.3. Suppose that $f \in A^{\leq n+m}$ and hence

$$f = f_1 + \cdots + f_s \tag{4}$$

for an $s \leq n + m$ with $f_i \in A$. Then the right-hand side of (4) must contain for every $x \in X$ a generator from A_x since $f(c_x) = 1$ (recall that we added the clause $c_x = \{x, \neg x\}$ since x is universally quantified) and only generators from A_x set the c_x-value to 1. Moreover, the right-hand side of (4) must contain for every $y \in Y$ a generator from A_y since $f(y) = 1$. Thus, the right-hand side of (4) must contain at least $|X| + |Y| = n + m$ generators. We get $s = n + m$ and obtain

$$f = \sum_{x \in X} g_x + \sum_{y \in Y} g_y \tag{5}$$

with $g_x \in A_x$ and $g_y \in A_y$. For $z \in X \cup Y$ let $C_z = \mathrm{supp}(g_z) \cap C$ be the set of clauses that appear in g_z in the sum (5). For all $y \in Y$ we must have $g_y = [\{y\} \cup C_y]$ since $f(y) = 1$.

We define the function $f_Y : Y \to \{0, 1\}$ by

$$f_Y(y) = \begin{cases} 1 & \text{if } C_y \subseteq C(y), \\ 0 & \text{otherwise.} \end{cases}$$

Note that if $f_Y(y) = 0$ then we must have $C_y \subseteq C(\neg y)$.

We claim that $f_X + f_Y$ satisfies F. Consider a clause $c \in C$. Since $f(c) = 1$ there must exist $z \in X \cup Y$ such that $c \in C_z$. If $z = y \in Y$, then one of the following two cases holds:

- $f_Y(y) = 1$, $g_y = [\{y\} \cup C_y]$, and $c \in C_y \subseteq C(y)$, i.e., $y \in c$,
- $f_Y(y) = 0$, $g_y = [\{y\} \cup C_y]$, and $c \in C_y \subseteq C(\neg y)$, i.e., $\neg y \in c$.

In both cases f_Y set a literal from c (either y or $\neg y$) to 1. Now, assume that $z = x \in X$. Then one of the following two cases holds:

- $f(x) = f_X(x) = 1$, $g_x = [\{x\} \cup C_x]$ and $c \in C_x \subseteq C(x)$, i.e., $x \in c$,
- $f(x) = f_X(x) = 0$, $g_x = [C_x]$ and $c \in C_x \subseteq C(\neg x)$, i.e., $\neg x \in c$.

Again, in both cases f_X sets a literal from c (either x or $\neg x$) to 1. ◁

Our proof of Claim 3.3 also shows that $d(A) \leq k$ implies $d(A) = k$: if $d(A) \leq k$ then for any of the functions f from Claim 3.3 we have $|f|_A = k$.

From Claims 3.2 and 3.3 it follows that if $\langle A \rangle = A^{\leq n+m}$ then for every $f_X : X \to \{0,1\}$ there must exist $f_Y : Y \to \{0,1\}$ such that $f_X + f_Y$ satisfies F. Hence, the formula Ψ from (3) holds.

Now suppose that Ψ holds. Let $f \in \mathbb{Z}_2^V$. We want to show $f \in A^{\leq n+m}$. First observe that there are unique functions $f_X : X \to \{0,1\}$ and $g : Y \cup C \to \{0,1\}$ such that $f = f_X + g$. Since Ψ holds, there is a partial truth assignment $f_Y : Y \to \{0,1\}$ such that $f_X + f_Y$ satisfies F. We define for every variable $z \in X \cup Y$ the set $U(z) \subseteq V$ as follows:

- if $z = x \in X$ and $f_X(x) = 1$ then $U(x) = \{x\} \cup C(x)$,
- if $z = x \in X$ and $f_X(x) = 0$ then $U(x) = C(\neg x)$,
- if $z = y \in Y$ and $f_Y(y) = 1$ then $U(y) = \{y\} \cup C(y)$,
- if $z = y \in Y$ and $f_Y(y) = 0$ then $U(y) = \{y\} \cup C(\neg y)$.

Note that $[U(z)] \in A_z$ for every variable $z \in X \cup Y$, except for the case that $z = x \in X$, $f_X(x) = 0$ and $C(\neg x) = \emptyset$ (then, $U(z) = \emptyset$). Define

$$U = \bigcup_{z \in X \cup Y} U(z).$$

Since all clauses evaluate to 1 under $f_X + f_Y$, we have $C \cup Y \subseteq U$. Moreover, $x \in U$ if and only if $x \in \mathrm{supp}(f)$ for all $x \in X$. We therefore have

$$\mathrm{supp}(f) \subseteq U.$$

We can choose pairwise disjoint (possibly empty) subsets $U'(z) \subseteq U(z)$ such that

$$U = \bigcup_{z \in X \cup Y} U'(z)$$

is a partition of U. It follows that

$$\mathrm{supp}(f) = \bigcup_{z \in X \cup Y} (U'(z) \cap \mathrm{supp}(f))$$

is a partition of $\mathrm{supp}(f)$. Let $Z \subseteq X \cup Y$ be the set of all $z \in X \cup Y$ such that $U'(z) \cap \mathrm{supp}(f) \neq \emptyset$. Then we have

$$f = \sum_{z \in Z} [U'(z) \cap \mathrm{supp}(f)]$$

and $|Z| \leq n + m$. It remains to show that $[U'(z) \cap \mathrm{supp}(f)]$ is a generator from A_z. This is clear if $z = y \in Y$ or ($z = x \in X$ and $U(x) = C(\neg x)$). In those case, for every non-empty subset $U' \subseteq U(z)$, $[U']$ belongs to A_z. Finally, if $z = x \in X$ and $U(x) = \{x\} \cup C(x)$ then also $x \in U'(x)$ must hold because $x \in U(x) \subseteq U = \bigcup_{z \in X \cup Y} U'(z)$ and $x \notin U'(z)$ for $z \neq x$. Moreover, $U(x) = \{x\} \cup C(x)$ implies $f(x) = f_X(x) = 1$, i.e., $x \in \mathrm{supp}(f)$. Therefore, $[U'(x) \cap \mathrm{supp}(f)]$ is of the form $[\{x\} \cup C']$ for some $C' \subseteq C(x)$, which belongs to A_x. We obtain $f \in A^{\leq n+m}$, which shows that ϕ is indeed a logspace reduction from $\forall\exists$SAT to UNARY DIAMETER.

▷ Claim 3.4. $d(A) \leq k+1$.

Proof of Claim 3.4. Our preprocessing ensured that every partial truth assignment of the universally quantified variables can be extended by a truth assignment for the existentially quantified variables such that exactly one clause $c \in C$ is unsatisfied. Let $f \in \mathbb{Z}_2^Y$. Then there are functions $f_X : X \to \{0,1\}$ and $g : Y \cup C \to \{0,1\}$ such that $f = f_X + g$. Moreover, there is a partial truth assignment $f_Y : Y \to \{0,1\}$ such that $f_X + f_Y$ satisfies all clauses from $C \setminus \{c\}$. As above we define for every variable $z \in X \cup Y$ the set $U(z) \subseteq V$ as follows:

- if $z = x \in X$ and $f_X(x) = 1$ then $U(x) = \{x\} \cup C(x)$,
- if $z = x \in X$ and $f_X(x) = 0$ then $U(x) = C(\neg x)$,
- if $z = y \in Y$ and $f_Y(y) = 1$ then $U(y) = \{y\} \cup C(y)$,
- if $z = y \in Y$ and $f_Y(y) = 0$ then $U(y) = \{y\} \cup C(\neg y)$.

Note that $c = \{\neg y\}$ for an existentially quantified variable $y \in Y$ (see Step 1 in our preprocessing). Hence, we have $c \in C(\neg y)$ and $[\{c\}] \in A_y$ is a generator. We define $U(c) = \{c\}$ and

$$U = \bigcup_{z \in X \cup Y \cup \{c\}} U(z).$$

The rest of the argument is the same as above: Since all clauses except for c evaluate to 1 under $f = f_X + f_Y$, we have $C \cup Y \subseteq U$. Moreover, $x \in U$ if and only if $x \in \mathrm{supp}(f)$ for all $x \in X$. We therefore have

$$\mathrm{supp}(f) \subseteq U.$$

Then there are pairwise disjoint subsets $U'(z) \subseteq U(z)$ such that

$$U = \bigcup_{z \in X \cup Y \cup \{c\}} U'(z) \quad \text{and} \quad \mathrm{supp}(f) = \bigcup_{z \in X \cup Y \cup \{c\}} (U'(z) \cap \mathrm{supp}(f))$$

are partitions of U and $\mathrm{supp}(f)$, respectively. Let $Z \subseteq X \cup Y \cup \{c\}$ be the set of all $z \in X \cup Y \cup \{c\}$ such that $U'(z) \cap \mathrm{supp}(f) \neq \emptyset$. Then we have

$$f = \sum_{z \in Z} [U'(z) \cap \mathrm{supp}(f)]$$

and $|Z| \leq n + m + 1$. As above it can easily be shown that $[U'(z) \cap \mathrm{supp}(f)]$ is a generator. Hence $f \in A^{\leq n+m+1}$. ◁

It now follows that $d(A)$ is either k or $k+1$: if $d(A) \leq k$ then $d(A) = k$ (see the remark after the proof of Claim 3.3), and if $d(A) > k$ then $d(A) = k+1$ by Claim 3.4. ◀

▶ **Corollary 3.5.** *The following problems are all Π_2^{P}-complete:*

(i) *UNARY DIAMETER (without a restriction on the permutation group $\langle A \rangle$),*

(ii) *UNARY DIAMETER restricted to abelian permutation groups $\langle A \rangle$ of exponent two,*

(iii) *BINARY DIAMETER restricted to abelian permutation groups $\langle A \rangle$.*

Proof. In all cases the lower bound follows from Theorem 3.1. It remains to show the upper bound in cases (i) and (iii). For (i), this is straightforward: Let $G = \langle A \rangle$ where $A \subseteq S_n$ is a set of permutations and take a unary encoded $k > 0$. First of all we universally guess an element $\pi \in G$. More precisely, we guess an arbitrary permutation $\pi \in S_n$ and then check in polynomial time (using [12]) whether $\pi \in \langle A \rangle$. If this does not hold, we immediately accept, otherwise we proceed with existentially guessing a sequence $a_1 a_2 \cdots a_l$ with $a_i \in A$ and $l \leq k$. We accept if and only if $a_1 a_2 \cdots a_l = \pi$.

The upper bound in case (iii) can be shown in a similar way. We follow the procedure for UNARY DIAMETER up to the point where we guess a sequence $a_1 a_2 \cdots a_l$ with $a_i \in A$ and $l \leq k$. Since k is given in binary encoding this is not feasible. Instead, we guess for each $a \in A$ a binary encoded number $k_a \geq 0$ whose bit length is bounded by the bit length of k. We accept if and only if the following two conditions hold:

- $\sum_{a \in A} k_a \leq k$,
- $\prod_{a \in A} a^{k_a} = \pi$.

Both conditions can be checked in polynomial time. For the second point note that a^{k_a} can be computed in time $\mathcal{O}(n \log k_a)$ by iterated squaring. ◀

▶ **Theorem 3.6.** *UNARY EXACT DIAMETER is DP_2-complete for general permutation groups as well as abelian permutation groups of exponent two.*

Proof. Let $A \subseteq S_n$ be a set of permutations and let k be a unary encoded number. Then we have $d(A) = k$ if and only if $d(A) \leq k$ and $d(A) > k - 1$. This is the intersection of a Π_2^{P}-property and a Σ_2^{P}-property. Hence, UNARY EXACT DIAMETER belongs to DP_2.

Now let $L = L_1 \cap L_2$ be a language from DP_2 with $L_1 \in \Sigma_2^{\mathsf{P}}$ and $L_2 \in \Pi_2^{\mathsf{P}}$. By Theorem 3.1 we can compute from x two pairs (A_1, k_1) and (A_2, k_2) (with $A_i \subseteq S_{n_i}$ and k_i a unary encoded natural number) such that

- $x \in L_1$ if and only if $d(A_1) = k_1 + 1$ if and only if $d(A_1) \neq k_1$, and
- $x \in L_2$ if and only if $d(A_2) = k_2$ if and only if $d(A_2) \neq k_2 + 1$.

Hence, $x \in L$ if and only if $d(A_1) = k_1 + 1$ and $d(A_2) = k_2$.

Consider the subgroup of $S_{n_2} \times S_{n_2} \leq S_{2n_2}$ generated by

$$B := (A_2 \times \{1\}) \cup (\{1\} \times A_2).$$

Here, 1 denotes the identity permutation. Since we have either $d(A_2) = k_2$ or $d(A_2) = k_2 + 1$ we obtain either $d(B) = 2k_2$ or $d(B) = 2k_2 + 2$.

Finally, consider the subgroup of $S_{n_1} \times S_{n_2} \times S_{n_2} \leq S_{n_1 + 2n_2}$ generated by

$$A := (A_1 \times \{(1,1)\}) \cup (\{1\} \times B).$$

There are four cases for the diameter of the group generated by A:

$$d(A) = \begin{cases} k_1 + 2k_2 & \text{if } d(A_1) = k_1 \text{ and } d(A_2) = k_2 \\ k_1 + 2k_2 + 1 & \text{if } d(A_1) = k_1 + 1 \text{ and } d(A_2) = k_2 \\ k_1 + 2k_2 + 2 & \text{if } d(A_1) = k_1 \text{ and } d(A_2) = k_2 + 1 \\ k_1 + 2k_2 + 3 & \text{if } d(A_1) = k_1 + 1 \text{ and } d(A_2) = k_2 + 1. \end{cases}$$

Thus, we have $x \in L$ if and only if $d(A) = k_1 + 2k_2 + 1$, which shows the DP_2-hardness of UNARY EXACT DIAMETER. ◀

4 Complexity of computing the length in permutation groups

Recall that Even and Goldreich [11] proved that UNARY LENGTH is NP-complete. We present below an alternative proof for the NP-hardness, where the reduction has additional properties (similar to Theorem 3.1) that will be needed in order to settle the complexity of UNARY EXACT LENGTH. Our techniques are similar to those from Section 3.

▶ **Theorem 4.1.** *There is a logspace reduction ϕ from SAT to UNARY LENGTH such that for every CNF formula F with $\phi(\Psi) = (A, \pi, k)$ we have: $A \subseteq \mathbb{Z}_2^n$ for some n, $\pi \in \langle A \rangle = \mathbb{Z}_2^n$ and $|\pi|_A \in \{k, k+1\}$.*

Proof. Let $F = \bigwedge_{c \in C} c$ be a conjunction of clauses $c \in C$ with boolean variables from the set X. We preprocess F as in the proof of Theorem 3.1. First, we replace F by $F' = \neg y^* \wedge \bigwedge_{c \in C}(y^* \vee c)$, where $y^* \notin X$ is a new variable ensuring that there is a truth assignment such that exactly one clause in F' is unsatisfied. Moreover, F is satisfiable if and only if F' is satisfiable.

Then we apply the construction of [25] that we also used in the proof of Theorem 3.1 in order to ensure that every variable occurs in at most three clauses. We replace the occurrences of every variable $z \in X \cup \{y^*\}$ that occurs in $l \geq 4$ clauses (negated or unnegated) by new variables $z_1, \ldots, z_l$ and add the clauses $z_l \vee \neg z_1$ and $z_i \vee \neg z_{i+1}$ for $i \in [1, l-1]$. Let F'' be the resulting CNF formula. It still has the property that there is a truth assignment such that exactly one clause in F'' is unsatisfied. One can take the truth assignment that sets all variables of F'' to 1. Moreover, F is satisfiable if and only if F'' is satisfiable.

From this consideration, it follows that we can assume that our input CNF formula F has the following two properties:

- Every variable occurs in at most three clauses.
- There is a truth assignment for F such that exactly one clause of F is not satisfied.

Let X be the variables that occur in F and let C be the set of clauses in F. Moreover, let $V = X \cup C$. With $L = X \cup \{\neg x \mid x \in X\}$ we denote the set of all literals.

We reuse several notations that we have introduced in the proof of Theorem 3.1. For a literal $\tilde{x} \in L$ we denote with $C(\tilde{x}) \subseteq C$ the set of all clauses containing $\tilde{x}$. Note that we have $|C(x)| + |C(\neg x)| \leq 3$. For the reduction we work with the group $\mathbb{Z}_2^V$ and use the notations from Section 2.2. Now we define the set A of generators by

$$A = \bigcup_{c \in C} A_c \cup \bigcup_{\tilde{x} \in L} A_{\tilde{x}},$$

where for $x \in X$ and $c \in C$ we take

$$\begin{aligned} A_x &= \{[\{x\} \cup U] \mid U \subseteq C(x)\}, \\ A_{\neg x} &= \{[\{x\} \cup U] \mid U \subseteq C(\neg x)\}, \\ A_c &= \{[\{c\}]\}. \end{aligned}$$

Note that $\langle A \rangle = \mathbb{Z}_2^V$.

We define $\pi = [V]$ and $k = |X|$. This defines our logspace reduction $\phi : F \mapsto (A, \pi, k)$. To compute ϕ in logspace, it is important that all sets $C(\tilde{x})$ have constant size.

Now we show that $|\pi|_A \leq k$ if and only if F is satisfiable. Suppose $|\pi|_A \leq k$. Since $X \subseteq \mathrm{supp}(\pi)$, we need a generator from every $A_x \cup A_{\neg x}$ ($x \in X$) to produce π. This implies $|\pi|_A = k$ and we can write

$$\pi = \sum_{x \in X} \pi_x$$

with $\pi_x \in A_x \cup A_{\neg x}$. Let $\pi_x = [\{x\} \cup U_x]$ with $U_x \subseteq C$. We define a truth assignment by

$$\sigma(x) = \begin{cases} 1 & \text{if } \pi_x \in A_x, \\ 0 & \text{if } \pi_x \in A_{\neg x} \setminus A_x. \end{cases}$$

for all $x \in X$. From $C \subseteq \mathrm{supp}(\pi)$ it follows that for every clause $c \in C$ there must exist a variable $x \in X$ such that $c \in U_x$. If $\pi_x \in A_x$ (i.e., $\sigma(x) = 1$) then $c \in U_x \subseteq C(x)$, i.e., x appears in the clause c. Hence, π satisfies c. Similarly, if $\pi_x \in A_{\neg x}$ (i.e., $\sigma(x) = 0$) then $\neg x$ appears in the clause c. Therefore, σ satisfies F.

Now suppose that F is satisfiable and let σ be a satisfying truth assignment. Hence, every clause is satisfied. Let $X_0 = \{x \in X \mid \sigma(x) = 0\}$ and $X_1 = \{x \in X \mid \sigma(x) = 1\}$. Then we have

$$\operatorname{supp}(\pi) = V = X \cup C = \bigcup_{x \in X_0} \{x\} \cup C(\neg x) \cup \bigcup_{x \in X_1} \{x\} \cup C(x).$$

Then we can choose for every $x \in X_0$ a subset $U_x \subseteq C(\neg x)$ and for every $x \in X_1$ a subset $U_x \subseteq C(x)$ such that

$$\operatorname{supp}(\pi) = \bigcup_{x \in X} \{x\} \cup U_x$$

is a partition of $\operatorname{supp}(\pi)$. Hence, we have

$$\pi = \sum_{x \in X} [\{x\} \cup U_x].$$

Since $[\{x\} \cup U_x] \in A_x \cup A_{\neg x}$, we finally obtain $|\pi|_A = k$. This show that ϕ is indeed a logspace reduction from SAT to UNARY LENGTH.

We have already noted that $|\pi|_A \leq k$ implies $|\pi|_A = k$. The converse implication is trivially true. Therefore, we have $|\pi|_A \leq k$ if and only if $|\pi|_A = k$. It remains to show that $|\pi|_A \in \{k, k+1\}$. For this it suffices to show $|\pi|_A \leq k+1$.

We know that there is a truth assignment σ such that exactly one clause $c \in C$ is unsatisfied. As above we can choose generators $\pi_x \in A_x \cup A_{\neg x}$ for all $x \in X$ such that

$$\operatorname{supp}(\pi) \setminus \{c\} = \bigcup_{x \in X} \operatorname{supp}(\pi_x)$$

is a partition of $\operatorname{supp}(\pi) \setminus \{x\}$. From this we obtain

$$\pi = [\{c\}] + \sum_{x \in X} \pi_x$$

and hence $|\pi|_A \leq k+1$, which concludes the proof. ◀

▶ **Theorem 4.2.** *UNARY EXACT LENGTH is* DP*-complete for general permutation groups as well as abelian permutation groups of exponent two.*

Proof. Let A be a set of generators of a permutation group, k a unary encoded integer, and $\pi \in \langle A \rangle$ a permutation. Then we have $|\pi|_A = k$ if and only if $|\pi|_A \leq k$ and $|\pi|_A > k-1$. This is the conjunction of an NP-property and a coNP-property. Thus UNARY EXACT LENGTH is the intersection of a language in NP with a language in coNP and therefore belongs to DP.

We show DP-hardness of UNARY EXACT LENGTH by a reduction from SAT-UNSAT. The input for the latter problem is a pair (F, G) of two CNF formulas and the question is whether F is satisfiable and G is unsatisfiable. This problem is known to DP-complete, see [21].

Let (F, G) be an input for SAT-UNSAT. By Theorem 4.1 we can compute from (F, G) in logspace two triples (A_1, π_1, k_1) and (A_2, π_2, k_2) (with $A_i \subseteq S_{n_i}$, $\pi_i \in \langle A_i \rangle$ and k_i a unary encoded natural number) such that

- F is satisfiable if and only if $|\pi_1|_{A_1} = k_1$ if and only if $|\pi_1|_{A_1} \neq k_1 + 1$ and
- G is unsatisfiable if and only if $|\pi_2|_{A_2} \neq k_2$ if and only if $|\pi_2|_{A_2} = k_2 + 1$.

Hence, (F, G) is a positive instance of SAT-UNSAT if and only if $|\pi_1|_{A_1} = k_1$ and $|\pi_2|_{A_2} = k_2 + 1$.

Consider the subgroup of $S_{n_2} \times S_{n_2} \leq S_{2n_2}$ with the generating set

$$B := (A_2 \times \{1\}) \cup (\{1\} \times A_2).$$

Since we have $|\pi_2|_{A_2} \in \{k_2, k_2+1\}$ we obtain $|(\pi_2, \pi_2)|_B \in \{2k_2, 2k_2+2\}$.

Finally, consider the group $\langle A_1 \rangle \times \langle A_2 \rangle \times \langle A_2 \rangle \leq S_{n_1+2n_2}$ with the generating set

$$A := (A_1 \times \{(1,1)\}) \cup (\{1\} \times B).$$

For the length $|(\pi_1, \pi_2, \pi_2)|_A$ we obtain

$$|(\pi_1, \pi_2, \pi_2)|_A = \begin{cases} k_1 + 2k_2 & \text{if } |\pi_1|_{A_1} = k_1 \text{ and } |\pi_2|_{A_2} = k_2 \\ k_1 + 2k_2 + 1 & \text{if } |\pi_1|_{A_1} = k_1 + 1 \text{ and } |\pi_2|_{A_2} = k_2 \\ k_1 + 2k_2 + 2 & \text{if } |\pi_1|_{A_1} = k_1 \text{ and } |\pi_2|_{A_2} = k_2 + 1 \\ k_1 + 2k_2 + 3 & \text{if } |\pi_1|_{A_1} = k_1 + 1 \text{ and } |\pi_2|_{A_2} = k_2 + 1. \end{cases}$$

Hence, (F, G) is a positive instance of SAT-UNSAT if and only if $|(\pi_1, \pi_2, \pi_2)|_A = k_1 + 2k_2 + 2$, which concludes the proof. ◀

Since BINARY LENGTH is PSPACE-complete [15], one might expect that also BINARY EXACT LENGTH is PSPACE-complete. The following result confirms this.

▶ **Theorem 4.3.** *BINARY EXACT LENGTH is* PSPACE*-complete.*

Proof of Theorem 4.3. Since PSPACE is closed under complement, and $|\pi|_A = k$ if and only if $\pi \in A^{\leq k}$ and $\pi \notin A^{\leq k-1}$, it follows that also BINARY EXACT LENGTH belongs to PSPACE.

For the lower bound let $A \subseteq S_n$ be a set of permutations on $[1, n]$, $\pi \in \langle A \rangle$ and k be a binary encoded number. We construct from A, π, k in logspace a new instance B, τ, k such that $\pi \in A^{\leq k}$ if and only if $|\tau|_B = k$ holds. This proves that BINARY EXACT LENGTH is PSPACE-complete.

Clearly, $S_n \leq S_m$ for $n \leq m$. In the following, we will identify a permutation $\pi \in S_n$ with a permutation from S_m by defining $a^\pi = a$ for $a \in [n+1, m]$.

Let d be the number of bits of k. Then $\log_2(k) < d \leq \log_2(k)+1$. Let $p_1 = 2, p_2 = 3, \ldots, p_d$ be the first d primes. Note that since d is polynomially bounded in the input length, the primes p_i are so too and therefore can be stored in logarithmic space. Let $m = \sum_{i=1}^{d} p_i$ and let $\alpha_1, \ldots, \alpha_d$ be permutations with pairwise disjoint support on $[n+1, n+m]$ such that α_i is a cycle of length p_i. Moreover let $r_1, \ldots, r_d \in [0, p_i - 1]$ such that

$$k \equiv r_i \bmod p_i.$$

These numbers can be computed in logspace; see e.g. [14]. Moreover let $\alpha = \alpha_1 \cdots \alpha_d$, $\beta = \alpha_1^{r_1} \cdots \alpha_d^{r_d}$ and $\tau = \pi\beta$. We have

$$\mathrm{ord}(\alpha) = \prod_{i=1}^{d} p_i \geq 2^d > 2^{\log_2(k)} = k.$$

Finally, we define the set of permutations

$$B = \{\gamma\alpha \mid \gamma \in A\} \cup \{\alpha\} \subseteq S_{n+m}.$$

Since $\beta = \alpha^k$, i.e., $\tau = \pi\alpha^k$ and $\mathrm{ord}(\alpha) > k$, we obtain $\pi \in A^{\leq k}$ if and only if $|\tau|_B = k$, which concludes the reduction. ◀

5 Complexity of equality and universality for NFAs over permutation groups

In this section we determine the complexity of RATIONAL EQUALITY and RATIONAL UNIVERSALITY (defined in Section 1.2).

▶ **Theorem 5.1.** *The following problems are Π_2^{P}-complete for permutation groups:*

(i) *RATIONAL EQUALITY*

(ii) *RATIONAL EQUALITY restricted to the case where all permutations in the two input NFAs $\mathcal{A}$ and $\mathcal{B}$ pairwise commute and have order two.*

(iii) *RATIONAL UNIVERSALITY*

Proof. For the upper bounds, we only have to consider RATIONAL EQUALITY. Membership of RATIONAL EQUALITY in Π_2^{P} follows from the fact that the rational subset membership problem for permutation groups (see Section 1.2) is in NP. More precisely, the following formula expresses the equality $L(\mathcal{A}_0) = L(\mathcal{A}_1)$ for two NFAs $\mathcal{A}_0$ and $\mathcal{A}_1$ over S_n:

$$\forall i \in \{0,1\} \forall \pi \in S_n : \pi \notin L(\mathcal{A}_i) \vee \pi \in L(\mathcal{A}_{1-i}).$$

Since the rational subset membership problem for permutation groups is in NP, the above formula is equivalent to a statement of the form

$$\forall i \in \{0,1\} \forall \pi \in S_n \forall u \exists v : u \text{ is not a witness for } \pi \in L(\mathcal{A}_i) \vee v \text{ is a witness for } \pi \in L(\mathcal{A}_{1-i}).$$

Here u and v are bit strings of size polynomial in the input length.

The lower bound in (ii) is a direct consequence of Corollary 3.5, since for a finite set $A \subseteq S_n$ and a unary encoded number k both $\langle A \rangle$ and $A^{\leq k}$ can be defined by logspace computable NFAs.

It remains to show Π_2^{P}-hardness of RATIONAL UNIVERSALITY. For this we give a reduction from UNARY DIAMETER to RATIONAL UNIVERSALITY. Before we come to the actual reduction, let us explain an auxiliary construction. Fix an $n \geq 1$ and consider the symmetric group S_{2n} on the domain $\Omega = [1, 2n]$. We define the following sets of transpositions:

$$T_i = \{(a,b) \mid a,b \in \Omega \setminus \{i\}, a \neq b\} \subseteq S_{2n} \text{ for all } i \in \Omega, \tag{6}$$

$$Z = \{(2i-1, 2i) \mid 1 \leq i \leq n\} \subseteq S_{2n}. \tag{7}$$

Note that $\langle Z \rangle \cong \mathbb{Z}_2^n$ and $\langle T_i \rangle$ is the set of permutations that fix i.

For every $1 \leq i < j \leq 2n$ with $(i,j) \notin Z$ we can construct in space $\mathcal{O}(\log n)$ three automata $\mathcal{A}_{i,j}, \mathcal{B}_{i,j}, \mathcal{C}_{i,j}$ over S_{2n} such that the following hold:

$$L(\mathcal{A}_{i,j}) = \bigcup_{\ell \in \Omega \setminus \{i,j\}} (i,j)(j,\ell)\langle T_i \cap T_j \rangle \tag{8}$$

$$L(\mathcal{B}_{i,j}) = \bigcup_{\ell \in \Omega \setminus \{i,j\}} (j,i)(i,\ell)\langle T_i \cap T_j \rangle \tag{9}$$

$$L(\mathcal{C}_{i,j}) = (i,j)\langle T_i \cap T_j \rangle \tag{10}$$

▷ Claim 5.2. We have

$$\langle Z \rangle \cap \bigcup_{\substack{1 \leq i < j \leq 2n \\ (i,j) \notin Z}} (L(\mathcal{A}_{i,j}) \cup L(\mathcal{B}_{i,j}) \cup L(\mathcal{C}_{i,j})) = \emptyset.$$

Proof of Claim 5.2. Suppose there is a $\tau \in \langle Z \rangle$ such that $\tau \in L(\mathcal{A}_{i,j}) \cup L(\mathcal{B}_{i,j}) \cup L(\mathcal{C}_{i,j})$ for some $1 \leq i < j \leq 2n$ with $(i,j) \notin Z$. For every $a \in [1,n]$ we have either $(2a-1)^\tau = 2a-1$ and $(2a)^\tau = 2a$ or $(2a-1)^\tau = 2a$ and $(2a)^\tau = 2a-1$.

Case 1. $\tau \in L(\mathcal{A}_{i,j})$. Then we can write $\tau = (i,j)(j,\ell)\pi$ with $\ell \in \Omega \setminus \{i,j\}$ and $\pi \in \langle T_i \cap T_j \rangle$. Then we obtain

$$j^\tau = j^{(i,j)(j,\ell)\pi} = i^{(j,\ell)\pi} = i^\pi = i.$$

We can exclude the case $j = j^\tau = i$, since $i < j$. Hence, we have $j^\tau \in \{j+1, j-1\}$. If j is odd we obtain $j+1 = j^\tau = i$, which is a contradiction since $i < j$. If j is even we obtain $j-1 = j^\tau = i$, and hence $(i,j) \in Z$, which is also a contradiction.

Case 2. $\tau \in L(\mathcal{B}_{i,j})$. Then we can write $\tau = (j,i)(i,\ell)\pi$ with $\ell \in \Omega \setminus \{i,j\}$ and $\pi \in \langle T_i \cap T_j \rangle$. In this case we obtain

$$i^\tau = i^{(j,i)(i,\ell)\pi} = j^{(i,\ell)\pi} = j^\pi = j.$$

We can exclude the case $i = i^\tau = j$, since $i < j$. Hence, we have $i^\tau \in \{i+1, i-1\}$. If i is odd we obtain $i+1 = i^\tau = j$ and hence $(i,j) \in Z$, which is a contradiction. If i is even we obtain $i-1 = i^\tau = j$, which contradicts $i < j$.

Case 3. $\tau \in L(\mathcal{C}_{i,j})$. Then we can write $\tau = (i,j)\pi$ with $\pi \in \langle T_i \cap T_j \rangle$ and get

$$i^\tau = i^{(i,j)\pi} = j^\pi = j.$$

We obtain a contradiction in the same way as in Case 2. ◁

▷ Claim 5.3. We have

$$\bigcup_{\substack{1 \le i < j \le 2n \\ (i,j) \notin Z}} (L(\mathcal{A}_{i,j}) \cup L(\mathcal{B}_{i,j}) \cup L(\mathcal{C}_{i,j})) = S_{2n} \setminus \langle Z \rangle. \tag{11}$$

Proof of Claim 5.3. By Claim 5.2 it suffices to show

$$S_{2n} \setminus \langle Z \rangle \subseteq \bigcup_{\substack{1 \le i < j \le 2n \\ (i,j) \notin Z}} (L(\mathcal{A}_{i,j}) \cup L(\mathcal{B}_{i,j}) \cup L(\mathcal{C}_{i,j})). \tag{12}$$

Let $\tau \in S_{2n} \setminus \langle Z \rangle$. We have to show that τ belongs to the union on the right-hand side of (12). Let $\tau = \gamma_1 \cdots \gamma_m$ be the disjoint cycle decomposition of τ. Since $\tau \notin \langle Z \rangle$ we can assume that w.l.o.g. $\gamma_1 \notin \langle Z \rangle$. Let $\alpha = \gamma_1$ and let $\beta = \gamma_2 \cdots \gamma_m$. Then we can write $\tau = \alpha\beta = \beta\alpha$ in which $\alpha = (i_d, \ldots, i_2, i_1)$ is a cycle of length $d \ge 2$. Note that $i_q \ne i_p$ for all $q \ne p$.

Case 1. $d = 2$. W.l.o.g. we can assume $i_1 < i_2$. Then $(i_1, i_2) \notin Z$ and by this the NFA $\mathcal{C}_{i_1,i_2}$ is defined. We have $(i_1,i_2)\pi \in L(\mathcal{C}_{i_1,i_2})$ for all $\pi \in \langle T_{i_1} \cap T_{i_2} \rangle$. Since β fixes i_1 and i_2, we have $\beta \in \langle T_{i_1} \cap T_{i_2} \rangle$ and hence $\tau = (i_1,i_2)\beta \in L(\mathcal{C}_{i_1,i_2})$.

Case 2. $d \ge 3$ and $(i_1,i_2) \notin Z$. Then, $\mathcal{A}_{i_1,i_2}$ is defined if $i_1 < i_2$ and $\mathcal{B}_{i_2,i_1}$ is defined if $i_2 < i_1$. We have

$$\alpha = (i_d, \ldots, i_1) = (i_1,i_2)(i_2,i_3)(i_3,i_4)\cdots(i_{d-1},i_d).$$

Let $\gamma = (i_3,i_4)\cdots(i_{d-1},i_d)\beta$. We get $\tau = \alpha\beta = (i_1,i_2)(i_2,i_3)\gamma$. Moreover, $\gamma \in \langle T_{i_1} \cap T_{i_2} \rangle$, since β fixes i_1 and i_2 and $i_q \ne i_p$ for $q \ne p$. If $i_1 < i_2$ we have $(i_1,i_2)(i_2,\ell)\pi \in L(\mathcal{A}_{i_1,i_2})$ for all $\ell \in \Omega \setminus \{i_1,i_2\}$ and $\pi \in \langle T_{i_1} \cap T_{i_2} \rangle$. Hence we obtain $\tau = (i_1,i_2)(i_2,i_3)\gamma \in L(\mathcal{A}_{i_1,i_2})$. If $i_2 < i_1$ we have $(i_1,i_2)(i_2,\ell)\pi \in L(\mathcal{B}_{i_2,i_1})$ for all $\ell \in \Omega \setminus \{i_1,i_2\}$ and $\pi \in \langle T_{i_1} \cap T_{i_2} \rangle$. Thus we analogously obtain $\tau = (i_1,i_2)(i_2,i_3)\gamma \in L(\mathcal{B}_{i_2,i_1})$.

Case 3. $d \geq 3$ and $(i_1, i_2) \in Z$. We then have $(i_2, i_3) \notin Z$ (otherwise, we would get $i_3 = i_1$) and $\mathcal{A}_{i_2,i_3}$ is defined if $i_2 < i_3$ and $\mathcal{B}_{i_3,i_2}$ is defined if $i_3 < i_2$. We have

$$\alpha = (i_d, \ldots, i_1) = (i_1, i_d, i_{d-1} \ldots, i_2) = (i_2, i_3)(i_3, i_4)(i_4, i_5) \cdots (i_{d-1}, i_d)(i_d, i_1).$$

Let $\gamma = (i_4, i_5) \cdots (i_{d-1}, i_d)(i_d, i_1)\beta$. Then we obtain $\tau = \alpha\beta = (i_2, i_3)(i_3, i_4)\gamma$ (if $d = 3$ we have $\gamma = \beta$ and $i_4 = i_1$). Analogously to Case 2, we obtain $(i_2, i_3)(i_3, i_4)\gamma \in L(\mathcal{A}_{i_2,i_3})$ if $i_2 < i_3$ and $(i_2, i_3)(i_3, i_4)\gamma \in L(\mathcal{B}_{i_3,i_2})$ if $i_3 < i_2$. ◁

Now we come to the reduction from UNARY DIAMETER to RATIONAL UNIVERSALITY. The proof of Theorem 3.1 shows that we can start with an input instance (A, k) of UNARY DIAMETER, where $A \subseteq \mathbb{Z}_2^n$ for some $n \in \mathbb{N}$ and $k \in \mathbb{N}$ is given in unary encoding. We can therefore assume that $\langle A \rangle = \langle Z \rangle$ for the above Z from (7). From A and k we can easily construct in logspace an NFA $\mathcal{A}$ such that

$$L(\mathcal{A}) = A^{\leq k} \cup \bigcup_{\substack{1 \leq i < j \leq 2n \\ (i,j) \notin Z}} (L(\mathcal{A}_{i,j}) \cup L(\mathcal{B}_{i,j}) \cup L(\mathcal{C}_{i,j})) = A^{\leq k} \cup (S_{2n} \setminus \langle A \rangle),$$

where the second equality follows from Claim 5.3. It is also important that k is given in unary encoding, which allows to construct in logspace an NFA for $A^{\leq k}$. We have $L(\mathcal{A}) = S_{2n}$ if and only if $d(A) \leq k$ which concludes the reduction. ◀

Note that in the above proof we write the complement $S_{2n} \setminus \langle A \rangle = S_{2n} \setminus \langle Z \rangle$ as a union of a polynomial number of cosets (see (8)–(10) and (11)). One might ask why we do not write $S_{2n} \setminus \langle A \rangle$ simply as union of cosets of $\langle A \rangle$. The problem is that the latter would require $|S_{2n}|/|\langle A \rangle| = (2n!)/2^n - 1$ cosets, which is not polynomial in n.

6 Open problems

The main open problem that remains is the complexity of BINARY DIAMETER. We conjecture that this problem is PSPACE-complete. Recall that we proved BINARY DIAMETER to be Π_2^{P}-complete for abelian permutation groups. We conjecture that this result can be extended to nilpotent permutation groups (and maybe even solvable permutation groups).

We conjecture that UNARY DIAMETER is Π_2^{P}-complete for input instances (A, k), where A generates the full symmetric group S_n. The Π_2^{P}-completeness of RATIONAL UNIVERSALITY would directly follow from this. Moreover, we mentioned in the introduction the conjecture according to which the diameter of S_n (with respect to any generating set) is bounded by a polynomial in n. This conjecture would imply that BINARY DIAMETER belongs to Π_2^{P} for input instances (A, k), where A generates the full symmetric group S_n.

References

1 Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. doi:10.1017/CBO9780511804090.

2 László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016*, pages 684–697. ACM, 2016. doi:10.1145/2897518.2897542.

3 László Babai, Robert Beals, and Ákos Seress. On the diameter of the symmetric group: polynomial bounds. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004*, pages 1108–1112. SIAM, 2004. URL: https://dl.acm.org/doi/10.5555/982792.982956.

4 László Babai and Thomas P. Hayes. Near-independence of permutations and an almost sure polynomial bound on the diameter of the symmetric group. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005*, pages 1057–1066. SIAM, 2005. URL: http://dl.acm.org/citation.cfm?id=1070432.1070584.

5 László Babai and Gábor Hetyei. On the diameter of random Cayley graphs of the symmetric group. *Combinatorics, Probability & Computing*, 1:201–208, 1992. doi:10.1017/S0963548300000237.

6 László Babai, Gábor Hetyei, William M. Kantor, Alexander Lubotzky, and Ákos Seress. On the diameter of finite groups. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science, FOCS 1990, Volume II*, pages 857–865. IEEE Computer Society, 1990. doi:10.1109/FSCS.1990.89608.

7 László Babai, William M. Kantor, and A. Lubotsky. Small-diameter Cayley graphs for finite simple groups. *European Journal of Combinatorics*, 10(6):507–522, 1989. doi:10.1016/S0195-6698(89)80067-8.

8 László Babai and Ákos Seress. On the diameter of Cayley graphs of the symmetric group. *Journal of Combinatorial Theory, Series A*, 49(1):175–179, 1988. doi:10.1016/0097-3165(88)90033-7.

9 László Babai and Ákos Seress. On the diameter of permutation groups. *European Journal of Combinatorics*, 13(4):231–243, 1992. doi:10.1016/S0195-6698(05)80029-0.

10 Liming Cai, Jianer Chen, Rodney G. Downey, and Michael R. Fellows. On the parameterized complexity of short computation and factorization. *Archive for Mathematical Logic*, 36(4-5):321–337, 1997. doi:10.1007/s001530050069.

11 Shimon Even and Oded Goldreich. The minimum-length generator sequence problem is NP-hard. *Journal of Algorithms*, 2(3):311–313, 1981. doi:10.1016/0196-6774(81)90029-8.

12 Merrick L. Furst, John E. Hopcroft, and Eugene M. Luks. Polynomial-time algorithms for permutation groups. In *Proceedings of the 21st Annual Symposium on Foundations of Computer Science, FOCS 1980*, pages 36–41. IEEE Computer Society, 1980. doi:10.1109/SFCS.1980.34.

13 Harald A. Helfgott and Ákos Seress. On the diameter of permutation groups. *Annals of Mathematics*, 179(2):611–658, 2014. doi:10.4007/annals.2014.179.2.4.

14 William Hesse, Eric Allender, and David A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *Journal of Computer and System Sciences*, 65(4):695–716, 2002. doi:10.1016/S0022-0000(02)00025-9.

15 Mark Jerrum. The complexity of finding minimum-length generator sequences. *Theoretical Computer Science*, 36:265–289, 1985. doi:10.1016/0304-3975(85)90047-7.

16 Arthur A. Khashaev. On the membership problem for finite automata over symmetric groups. *Discrete Mathematics and Applications*, 32(6):389–395, 2022. doi:10.1515/dma-2022-0033.

17 D. Kornhauser, G. Miller, and P. Spirakis. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *Proceedings of the 25th IEEE Symposium on Foundations of Computer Science, FOCS 1984*, pages 241–250. IEEE Computer Society Press, 1984. doi:10.1109/SFCS.1984.715921.

18 Markus Lohrey, Andreas Rosowski, and Georg Zetzsche. Membership problems in finite groups. In *Proceedings of the 47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022*, volume 241 of *LIPIcs*, pages 71:1–71:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.MFCS.2022.71.

19 Pierre McKenzie. Permutations of bounded degree generate groups of polynomial diameter. *Information Processing Letters*, 19(5):253–254, 1984. doi:10.1016/0020-0190(84)90062-0.

20 not A or B (https://cstheory.stackexchange.com/users/38066/not-a-or b). An analog of DP for the second level of the polynomial hierarchy. Theoretical Computer Science Stack Exchange. URL: https://cstheory.stackexchange.com/q/38776.

21 Christos H. Papadimitriou and Mihalis Yannakakis. The complexity of facets (and some facets of complexity). *Journal of Computer and System Sciences*, 28(2):244–259, 1984. doi:10.1016/0022-0000(84)90068-0.

22 Tomas Rokicki, Herbert Kociemba, Morley Davidson, and John Dethridge. The diameter of the Rubik's cube group is twenty. *SIAM Journal of Discrete Mathematics*, 27(2):1082–1105, 2013. doi:10.1137/120867366.

23 Ákos Seress. *Permutation Group Algorithms*. Cambridge Tracts in Mathematics. Cambridge University Press, 2003. doi:10.1017/CBO9780511546549.

24 Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976. doi:10.1016/0304-3975(76)90061-X.

25 Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984. doi:10.1016/0166-218X(84)90081-7.

Canonical Decompositions in Monadically Stable and Bounded Shrubdepth Graph Classes

Pierre Ohlmann
Institute of Informatics, University of Warsaw, Poland

Michał Pilipczuk
Institute of Informatics, University of Warsaw, Poland

Wojciech Przybyszewski
Institute of Informatics, University of Warsaw, Poland

Szymon Toruńczyk
Institute of Informatics, University of Warsaw, Poland

Abstract

We use model-theoretic tools originating from stability theory to derive a result we call the Finitary Substitute Lemma, which intuitively says the following. Suppose we work in a stable graph class $\mathscr{C}$, and using a first-order formula φ with parameters we are able to define, in every graph $G \in \mathscr{C}$, a relation R that satisfies some hereditary first-order assertion ψ. Then we are able to find a first-order formula φ' that has the same property, but additionally is *finitary*: there is finite bound $k \in \mathbb{N}$ such that in every graph $G \in \mathscr{C}$, different choices of parameters give only at most k different relations R that can be defined using φ'.

We use the Finitary Substitute Lemma to derive two corollaries about the existence of certain canonical decompositions in classes of well-structured graphs.

- We prove that in the Splitter game, which characterizes nowhere dense graph classes, and in the Flipper game, which characterizes monadically stable graph classes, there is a winning strategy for Splitter, respectively Flipper, that can be defined in first-order logic from the game history. Thus, the strategy is canonical.
- We show that for any fixed graph class $\mathscr{C}$ of bounded shrubdepth, there is an $\mathcal{O}(n^2)$-time algorithm that given an n-vertex graph $G \in \mathscr{C}$, computes in an isomorphism-invariant way a structure H of bounded treedepth in which G can be interpreted. A corollary of this result is an $\mathcal{O}(n^2)$-time isomorphism test and canonization algorithm for any fixed class of bounded shrubdepth.

2012 ACM Subject Classification Theory of computation → Finite Model Theory

Keywords and phrases Model Theory, Stability Theory, Shrubdepth, Nowhere Dense, Monadically Stable

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.135

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `https://arxiv.org/abs/2303.01473`

Funding This paper is part of a project that have received funding from the European Research Council (ERC) (grant agreement No 948057 – BOBR).

Acknowledgements We are indebted to Pierre Simon for enlightening discussions about classic results in stability theory that greatly helped in deriving the results presented in this work. We also thank Jakub Gajarský, Rose McCarty, and Marek Sokołowski for their contribution in discussions around the topic of this work.

1 Introduction

Stability theory is a well-established branch of model theory devoted to the study of stable theories, or equivalently classes of structures that are models of such theories. Here, we say that a formula[1] $\varphi(\bar{x};\bar{y})$ is *stable* on a class of relational structures $\mathscr{C}$ if there is an integer $k \in \mathbb{N}$ such that for every $\mathbf{M} \in \mathscr{C}$, one cannot find tuples $\bar{u}_1, \ldots, \bar{u}_k \in \mathbf{M}^{\bar{x}}$ and $\bar{v}_1, \ldots, \bar{v}_k \in \mathbf{M}^{\bar{y}}$ such that for all $i, j \in \{1, \ldots, k\}$,

$$\mathbf{M} \models \varphi(\bar{u}_i, \bar{v}_j) \qquad \text{if and only if} \qquad i < j.$$

Then $\mathscr{C}$ is stable if every formula is stable on $\mathscr{C}$. Intuitively, this means that using a fixed formula, one cannot interpret arbitrarily long total orders in structures from $\mathscr{C}$. We refer to the textbooks of Pillay [24] or of Tent and Ziegler [27] for an introduction to stability.

The goal of this paper is to use certain classic results of stability theory, particularly the understanding of *forking* in stable theories, to derive statements about the existence of canonical decompositions in certain classes of well-structured graphs. Here, we model graphs as relational structures with one binary adjacency relation that is symmetric.

Finitary Substitute Lemma. Our main model-theoretic tool is the Finitary Substitute Lemma, which we state below in a simplified form; see Lemma 10 for a full statement.

To state the lemma, we need some definitions. A formula $\varphi(\bar{x};\bar{y})$ is *finitary* on a class of structures $\mathscr{C}$ if there exists $k \in \mathbb{N}$ such that for every $\mathbf{M} \in \mathscr{C}$, we have

$$\left|\left\{\varphi(\mathbf{M}^{\bar{x}}, \bar{v}) \colon \bar{v} \in \mathbf{M}^{\bar{y}}\right\}\right| \leqslant k,$$

where $\varphi(\mathbf{M}^{\bar{x}}, \bar{v}) = \{\bar{u} \in \mathbf{M}^{\bar{x}} \mid \mathbf{M} \models \varphi(\bar{u}, \bar{v})\}$. In other words, $\varphi(\bar{x};\bar{y})$ is finitary on $\mathscr{C}$ if by substituting different parameters for $\bar{y}$ in any model $\mathbf{M} \in \mathscr{C}$, one can define only at most k different relations on $\bar{x}$-tuples. Next, a sentence ψ is *hereditary* if for every model $\mathbf{M}$ and its induced substructure $\mathbf{M}'$, $\mathbf{M} \models \psi$ implies $\mathbf{M}' \models \psi$. Finally, for a relation $R(\bar{x})$ present in the signature, formula $\varphi(\bar{x})$ (possibly with parameters), and sentence ψ, by $\psi[R(\bar{x})/\varphi(\bar{x})]$ we mean the sentence derived from ψ by substituting every occurrence of R with formula φ.

▶ **Lemma 1** (Finitary Substitute Lemma, simplified version)**.** *Let $\mathscr{C}$ be a stable class of structures. Suppose $\varphi(\bar{x};\bar{y})$ is a formula and ψ a hereditary sentence such that for every $G \in \mathscr{C}$,*

$$\text{there exists } \bar{s} \in G^{\bar{y}} \text{ such that } G \models \psi[R(\bar{x})/\varphi(\bar{x};\bar{s})]. \tag{1}$$

Then there exists a formula $\varphi'(\bar{x},\bar{z})$ that also satisfies (1)*, but is additionally finitary on $\mathscr{C}$.*

Thus, intuitively, the Finitary Substitute Lemma says that in stable classes, every relation that is definable with parameters can be replaced by a finitary one, as long as we care that the relation satisfies some hereditary first-order assertion. The main observation of this paper is that this can be used in the context of various graph decompositions. Intuitively, if every step in decomposing the graph can be defined by a first-order formula with parameters, and the validity of the step can be verified using a hereditary first-order sentence, then we can use the Finitary Substitute Lemma to derive an equivalent definition of a step that is finitary. This yields only a bounded number of different steps that can be taken, making it possible to construct a decomposition that, in a certain sense, is canonical.

[1] All formulas considered in this paper are first-order, unless explicitly stated.

Classes of bounded shrubdepth. Our first application concerns classes of *bounded shrubdepth*. The concept of shrubdepth was introduced by Ganian et al. [17] to capture dense graphs that are well-structured in a shallow way. On one hand, classes of bounded shrubdepth are exactly those that can be interpreted, using first-order formulas with two free variables, in classes of forests of bounded depth. On the other hand, graphs from any fixed class of bounded shrubdepth admit certain decompositions, called *connection models*, which are essentially clique expressions of bounded depth. (See Section 5.1 for a definition of a connection model.) Thus, in particular every graph class of bounded shrubdepth has bounded cliquewidth, but classes of bounded shrubdepth are in addition stable [17].

Shrubdepth is a dense counterpart of *treedepth*, defined as follows: the treedepth of a graph G is the smallest integer d such that G is a subgraph of the ancestor/descendant closure of a rooted forest of depth at most d. In particular, every class of graphs of bounded treedepth has bounded shrubdepth; boundedness of treedepth and of shrubdepth is in fact equivalent assuming that the class excludes some biclique $K_{t,t}$ as a subgraph [17]. In essence, treedepth is a bounded-depth counterpart of treewidth in the same way as shrubdepth is a bounded-depth counterpart of cliquewidth.

In spite of the above, the combinatorial properties of shrubdepth are still much less understood than those of treedepth. For instance, a good understanding of subgraph obstacles allows one to construct suitable canonical decompositions for graphs of bounded treedepth. This allowed Bouland et al. [4] to design a graph isomorphism test that works in fixed-parameter time parameterized by the treedepth, or more precisely, in time $f(d) \cdot n^3 \log n$, where f is a computable function. While it is known that every class of bounded shrubdepth can be characterized by a finite number of forbidden induced subgraphs [17], it is unclear how to use just this result to design any kind of canonical decompositions for classes of bounded shrubdepth. Consequently, so far it was unknown whether the graph isomorphism problem can be solved in fixed-parameter time on classes of bounded shrubdepth[2]. The most efficient isomorphism test in this context is the one designed by Grohe and Schweitzer [20] for the cliquewidth parameterization: it works in XP time, that is, in time $n^{f(k)}$ where k is the cliquewidth and f is a computable function. See also the later work of Grohe and Neuen [19], which improves the XP running time and applies to the more general canonization problem.

We show that the Finitary Substitute Lemma can be used to bridge this gap by proving the following result.

▶ **Theorem 2.** *Let $\mathscr{C}$ be a class of graphs of bounded shrubdepth. Then there is a class $\mathscr{D}$ of binary structures of bounded treedepth and a mapping $\mathcal{A}\colon \mathscr{C} \to \mathscr{D}$ such that:*

- *For each $G \in \mathscr{C}$, the vertex set of G is contained in the domain of $\mathcal{A}(G)$ and the mapping $G \mapsto \mathcal{A}(G)$ is isomorphism-invariant.*
- *Given an n-vertex graph $G \in \mathscr{C}$, the structure $\mathcal{A}(G)$ has $\mathcal{O}(n)$ elements and can be computed in time $\mathcal{O}(n^2)$.*
- *There is a simple first-order interpretation I such that $G = \mathsf{I}(\mathcal{A}(G))$, for every $G \in \mathscr{C}$.*

Here, by isomorphism-invariance we mean that every isomorphism between $G, G' \in \mathscr{C}$ extends to an isomorphism between $\mathcal{A}(G)$ and $\mathcal{A}(G')$. Further, by a *simple* interpretation we mean a first-order interpretation that is 1-dimensional: vertices of G are interpreted in

[2] This statement is somewhat imprecise, as shrubdepth is defined as a parameter of a graph class, rather than of a single graph. By this we mean that there is a universal constant c such that for every graph class $\mathscr{C}$ of bounded shrubdepth, the isomorphism of graphs from $\mathscr{C}$ can be tested in $\mathcal{O}(n^c)$ time, with the constant hidden in the $\mathcal{O}(\cdot)$ notation possibly depending on $\mathscr{C}$.

single elements of $\mathcal{A}(G)$ (actually, every vertex is interpreted in itself). Thus, $\mathcal{A}(G)$ can be regarded as a canonical – obtained in an isomorphism-invariant way – sparse decomposition of G that encodes G faithfully and takes the form of a structure of bounded treedepth. We remark that certain logic-based sparsification procedures for classes of bounded shrubdepth were proposed in [8, 14], but these are insufficient for our applications, which we explain next.

The third point above together with the fact that $\mathcal{A}$ is isomorphism-invariant imply the following: for all $G, G' \in \mathscr{C}$, G and G' are isomorphic if and only if $\mathcal{A}(G)$ and $\mathcal{A}(G')$ are. We can now combine Theorem 2 with the approach of Bouland et al. [4] to give a fixed-parameter isomorphism test on classes of bounded shrubdepth.

▶ **Theorem 3.** *For every graph class $\mathscr{C}$ of bounded shrubdepth there is an $\mathcal{O}(n^2)$-time algorithm that given n-vertex graphs $G, G' \in \mathscr{C}$, decides whether G and G' are isomorphic.*

In fact, our algorithm solves also the general canonization problem, see Section 5.4.

We remark that the algorithm of Theorem 3 is non-uniform, in the sense that we obtain a different algorithm for every class $\mathscr{C}$. Despite the existence of parameters such as rankdepth [22] or SC-depth [17] that are suited for the treatment of single graphs and are equivalent in terms of boundedness on classes to shrubdepth, we do not know how to make our algorithm uniform even for the rankdepth or SC-depth parameterizations.

Finally, we believe that the construction behind our proof of Theorem 2 can be used to obtain an alternative proof of a result of Hliněný and Gajarský [13], later reproved by Chen and Flum [8]: the expressive power of first-order and monadic second-order logic coincide on classes of bounded shrubdepth. This direction will be explored in future research.

Nowhere dense and monadically stable classes. Second, we use the Finitary Substitute Lemma to provide canonical strategies in game characterizations of two important concepts in structural graph theory: nowhere dense classes and monadically stable classes. In both cases, a strategy in the game can be regarded as decompositions of the graph in question.

We start with some definitions. A *unary lift* of a class of graphs $\mathscr{C}$ is any class of structures $\mathscr{C}^+$ such that every member of $\mathscr{C}^+$ is obtained from a graph belonging to $\mathscr{C}$ by adding any number of unary predicates on vertices. A class of graphs $\mathscr{C}$ is *monadically stable* if every unary lift of $\mathscr{C}$ is stable. On the other hand, a class of graphs $\mathscr{C}$ is *nowhere dense* if for every $d \in \mathbb{N}$ there exists t such that no graph in $\mathscr{C}$ contains the d-subdivision of K_t as a subgraph.

Nowhere denseness is the most fundamental concept of uniform sparsity in graphs considered in the theory of Sparsity; see the monograph of Nešetřil and Ossona de Mendez [23] for an introduction to this area. A pinnacle result of this theory was derived by Grohe et al. [18]: the model-checking problem for first-order logic is fixed-parameter tractable on any nowhere dense graph class. As observed by Adler and Adler [2] using earlier results of Podewski and Ziegler [25], monadically stable classes are dense counterparts of nowhere dense classes in the following sense: every nowhere dense class is monadically stable, and nowhere denseness and monadic stability coincide when we assume the class to be sparse, for instance to exclude some biclique $K_{t,t}$ as a subgraph. This motivated the following conjecture [1], which is an object of intensive studies for the last few years: The model-checking problem for first-order logic is fixed-parameter tractable on every monadically stable class of graphs $\mathscr{C}$.

To approach this conjecture, it is imperative to obtain a better structural understanding of graphs from monadically stable classes. This is the topic of several very recent works [5, 6, 7, 11, 15]. In this work we are particularly interested in the results of Gajarský et al. [15], who characterized monadically stable classes of graphs through a game model called the *Flipper game*, which reflects the characterization of nowhere dense classes through the *Splitter game*, due to Grohe et al. [18].

The radius-r Splitter game is played on a graph G between two players: *Splitter* and *Connector*. In every round, Connector first chooses any vertex u and the current *arena* – graph on which the game is played – gets restricted to a ball of radius r around u. Then Splitter removes any vertex of the graph. The game finishes, with Splitter's win, when the arena becomes empty. Splitter's goal is to win the game as quickly as possible, while Connector's goal is to avoid losing for as long as possible. The *Flipper game* is defined similarly, except that the moves of *Flipper* – who replaces Splitter – are as follows. Instead of removing a vertex, Flipper selects any subset of vertices F and performs a *flip*: replaces all edges with both endpoints in F with non-edges, and vice versa. Also, the game finishes when the arena consists of one vertex.

Grohe et al. [18] proved that a class of graphs $\mathscr{C}$ is nowhere dense if and only if for every radius $r \in \mathbb{N}$ there exists $\ell \in \mathbb{N}$ such that on every graph from $\mathscr{C}$, Splitter can win the radius-r Splitter game within at most ℓ rounds. This characterization is the backbone of their model-checking result for nowhere dense classes, as a strategy in the Splitter game provides a shallow decomposition of the graph in question, useful for understanding its first-order properties. Very recently, Gajarský et al. [15] proved an analogous characterization of monadically stable classes in terms of the Flipper game, and subsequently Dreier et al. [10] used this characterization to prove fixed-parameter tractability of the model-checking first order logic on monadically stable classes of graphs which possess so-called sparse neighborhood covers. Given this state-of-the-art, it is clear that a better understanding of strategies for Splitter and Flipper in the respective games may lead to a deeper insight into decompositional properties of nowhere dense and monadically stable graph classes.

In the Splitter game, we prove using just basic compactness, that in any arena there is only a bounded number of possible Splitter's moves that are *progressing*: lead to an arena where the Splitter can win in one less round. (See Theorem 6 for a formal statement.) So this gives a transparent canonical strategy for Splitter: just play all progressive moves one by one, in any order. Obtaining a similar canonicity result for strategies in the Flipper game requires the full power of our Finitary Substitute Lemma, discussed above.

In the interest of space, we have omitted from this version our results about canonical strategies in the Flipper game, as well as most proofs. For a complete exposition, we refer to the full version of the paper.

2 Preliminaries

Models. We work with first-order logic over a fixed signature Σ that consists of (possibly infinitely many) constant symbols and relation symbols. A *model* is a Σ-structure, and is typically denoted $\mathbf{M}, \mathbf{N}$, etc. We usually do not distinguish between a model and its domain, when writing, for instance, $m \in \mathbf{M}$ or $X \subseteq \mathbf{M}$. A graph G is viewed as a model over the signature consisting of one binary relation denoted E, indicating adjacency between vertices.

If $\bar{x}$ is a finite set of variables, then we write $\varphi(\bar{x})$ to denote a first-order formula φ with free variables contained in $\bar{x}$. We may also write $\varphi(\bar{x}_1, \ldots, \bar{x}_k)$ to denote a formula whose free variables are contained in $\bar{x}_1 \cup \ldots \cup \bar{x}_k$. We will write x instead of $\{x\}$ in case of a singleton set of variables, e.g. $\varphi(x, y)$ will always refer to a formula with two free variables x and y. We sometimes write $\varphi(\bar{x}; \bar{y})$ to distinguish a partition of the set of free variables of φ into two parts, $\bar{x}$ and $\bar{y}$; this partition plays an implicit role in some definitions. A Σ-formula $\varphi(\bar{x})$ *with parameters* from a set $A \subseteq \mathbf{M}$ is a formula $\varphi(\bar{x})$ over the signature $\Sigma \uplus A$, where the elements of A are treated as constant symbols (which are interpreted by themselves).

If U is a set and $\bar{x}$ is a set of variables, then $U^{\bar{x}}$ denotes the set of all *$\bar{x}$-tuples* $\bar{a}\colon \bar{x} \to U$ of $\bar{x}$ in U. For a formula $\varphi(\bar{x})$ (with or without parameters) and an $\bar{x}$-tuple $\bar{m} \in \mathbf{M}^{\bar{x}}$, we write $\mathbf{M} \models \varphi(\bar{m})$ if the valuation $\bar{m}$ satisfies the formula $\varphi(\bar{x})$ in $\mathbf{M}$. For a formula $\varphi(\bar{x};\bar{y})$ and a tuple $\bar{b} \in \mathbf{M}^{\bar{y}}$ we denote by $\varphi(\mathbf{M}^{\bar{x}};\bar{b})$ the set of all $\bar{a} \in \mathbf{M}^{\bar{x}}$ such that $\mathbf{M} \models \varphi(\bar{a};\bar{b})$.

Theories and compactness. A *theory* T (over Σ) is a set of Σ-sentences. The theory of a class of structures $\mathscr{C}$ is the set of sentences that hold in every structure $\mathbf{M} \in \mathscr{C}$. For instance, the theory of a class of graphs $\mathscr{C}$ contains sentences expressing that the relation E is symmetric and irreflexive. A model of a theory T is a structure $\mathbf{M}$ such that $\mathbf{M} \models \varphi$ for all $\varphi \in T$. When a theory has a model, it is said to be *consistent*.

▶ **Theorem 4** (Compactness). *A theory T is consistent if and only if every finite subset of T is consistent.*

Elementary extensions. Let $\mathbf{M}$ and $\mathbf{N}$ be two structures with $\mathbf{M} \subseteq \mathbf{N}$, that is, the domain of $\mathbf{M}$ is contained in the domain of $\mathbf{N}$. Then $\mathbf{N}$ is an *elementary extension* of $\mathbf{M}$, written $\mathbf{M} \prec \mathbf{N}$, if for every formula $\varphi(\bar{x})$ (without parameters) and tuple $\bar{m} \in \mathbf{M}^{\bar{x}}$, the following equivalence holds:

$$\mathbf{M} \models \varphi(\bar{m}) \qquad \text{if and only if} \qquad \mathbf{N} \models \varphi(\bar{m}).$$

We also say that $\mathbf{M}$ is an *elementary substructure* of $\mathbf{N}$. In other words, $\mathbf{M}$ is an elementary substructure of $\mathbf{N}$ if $\mathbf{M}$ is an induced substructure of $\mathbf{N}$, where we imagine that $\mathbf{M}$ and $\mathbf{N}$ are each equipped with every relation R_φ of arity k (for $k \in \mathbb{N}$) that is defined by any fixed first-order formula $\varphi(x_1, \ldots, x_k)$. In this intuition, formulas of arity 0 correspond to Boolean flags, with the same valuation for both $\mathbf{M}$ and $\mathbf{N}$.

Interpretations and transductions. A *simple interpretation* I between signatures Σ and Γ is specified by a domain formula $\delta(x)$ and a formula $\alpha_R(x_1, \ldots, x_k)$ for each relation symbol $R \in \Gamma$ of arity k, with δ and the α_R's being in the signature Σ. For a given Σ-structure $\mathbf{M}$, the interpretation outputs the Γ-structure $\mathsf{I}(\mathbf{M})$ whose domain is $\delta(\mathbf{M})$ and in which the interpretation of each relation R of arity k consists of the tuples $\bar{m}$ such that $\mathbf{M} \models \alpha_R(\bar{m})$. In this paper, we only consider simple interpretations, and therefore we will call them interpretations for conciseness.

For an integer $k \in \mathbb{N}$ and a structure $\mathbf{M}$, we define $k \times \mathbf{M}$ to be the structure consisting of k disjoint copies of $\mathbf{M}$, together with a new symmetric binary relation S containing all pairs (m, m') such that m and m' originate from the same element of $\mathbf{M}$. A *transduction* from Σ to Γ consists of an integer k, unary symbols $U_1, \ldots, U_\ell$ and an interpretation I from $\Sigma \cup \{S, U_1, \ldots, U_\ell\}$ to Γ.

For a transduction T and an input Σ-structure $\mathbf{M}$, the output $\mathsf{T}(\mathbf{M})$ consists of all Γ-structures $\mathbf{N}$ such that there exists a coloring $\widehat{\mathbf{M}}$ of $k \times \mathbf{M}$ with fresh unary predicates $U_1, \ldots, U_\ell$ such that $\mathbf{B} = \mathsf{I}(\widehat{\mathbf{M}})$. We say that a class of Σ-structures $\mathscr{C}$ *transduces* a class of Γ-structure $\mathcal{D}$ if there exists a transduction T such that for every structure $\mathbf{N} \in \mathscr{D}$ there is $\mathbf{M} \in \mathscr{C}$ satisfying $\mathbf{N} \in \mathsf{T}(\mathbf{M})$.

Graphs. We use standard graph theory notation. For a graph parameter π, we say that a graph class $\mathscr{C}$ has *bounded* π if there exists $k \in \mathbb{N}$ such that $\pi(G) \leqslant k$ for all $G \in \mathscr{C}$. Similarly, a class of structures $\mathscr{C}$ has bounded π if the class of Gaifman graphs of structures in $\mathscr{C}$ has bounded π.

3 Canonical Splitter-strategies in nowhere dense graphs

In this section, we show how compactness can be used to derive canonical decompositions for nowhere dense classes. More precisely, we will show that in the Splitter game, which characterizes nowhere dense classes [18], there is a constant k (depending only on the graph class $\mathscr{C}$) such that for any graph in $\mathscr{C}$ there are at most k optimal Splitter moves. This will allow us to illustrate the general methodology used in the paper.

Splitter game. First, we recall the rules of the Splitter game. The radius-r Splitter game is played on a graph G by two players, Splitter and Connector, in rounds $i = 1, 2, \dots$ as follows. Initially the arena G_1 is the whole graph G. In the i-th round,

- Connector chooses a vertex $c_i \in G_i$;
- Splitter chooses a vertex $s_i \in G_i$ and we let $G_{i+1} = G_i[B^r_{G_i}(c_i)] - s_i$;
- Splitter wins if G_{i+1} is the empty graph, otherwise the game continues.

Here, $B^r_H(u) = \{v \in V(H) \mid \operatorname{dist}_H(u, v) \leqslant r\}$ denotes the ball of radius r around u in H.

The following result is instrumental in the celebrated proof of model-checking on nowhere dense classes [18].

▶ **Theorem 5** (Theorems 4.2 and 4.5 in [18]). *A class of graphs $\mathscr{C}$ is nowhere dense if and only if for every r, there exists ℓ such that on every graph $G \in \mathscr{C}$, Splitter can win the radius-r game in at most ℓ rounds.*

The *r-Splitter number* of a graph G is the minimal ℓ such that Splitter wins the radius-r game in ℓ rounds. Fix a nowhere dense class $\mathscr{C}$ and a radius r, and let ℓ be as in the theorem (hence ℓ is an upper bound to all r-Splitter numbers of graphs in $\mathscr{C}$). Observe that for a given $\ell' \leqslant \ell$ there is a first-order sentence expressing that Splitter wins the radius-r game in $\leqslant \ell'$ rounds, and therefore, there is a first-order sentence expressing that G has Splitter number ℓ'. Given a Connector move $c \in V(G)$, we say that a Splitter move $s \in V(G)$ is *r-progressing against c* if the r-Splitter number of $G[B_r(c)] - s$ is strictly smaller than the r-Splitter number of $G[B_r(c)]$. In other words, playing s is strictly better for Splitter than not playing any vertex. Again, since an upper bound to Splitter numbers depends only on $\mathscr{C}$, this can be expressed by a formula $\varphi_r(s; c)$. This leads to the following result.

▶ **Theorem 6.** *Let $\mathscr{C}$ be a nowhere dense class of graphs, and $r \in \mathbb{N}$. There is a constant k such that for every graph $G \in \mathscr{C}$, and every Connector move c, there are at most k progressing moves against c in G.*

In particular, this gives an isomorphism-invariant strategy for Splitter: simply play all progressing moves (either one by one, in any order, or all at once in an extended variant of the game considered in [18], where Splitter can remove a bounded number of vertices in each turn, instead of just one.) The idea of the proof is to extend, by compactness, progressive moves towards outside the model (in an elementary extension), and conclude by observing that "being a progressive move" is a definable and hereditary property.

Proof. Let T be the theory of $\mathscr{C}$. Note that T contains the sentence "Splitter wins the radius-r game in $\leqslant \ell$ rounds". Our aim is to prove that for some k, it contains the sentence "for all connector moves c, there are at most k progressing Splitter moves against c". We show that for any model of T and any connector move c, there are finitely many progressing Splitter moves against c; the result then follows from an easy application of compactness.

Assume for contradiction that there is a model $\mathbf{M}$ of T and a connector move $c \in \mathbf{M}$ such that Splitter has infinitely many progressing moves against c. We now let T' be the theory over the signature extended by a constant corresponding to each element $m \in \mathbf{M}$ and an additional constant s, such that T' consists of:

- all sentences in T,
- all sentences (with parameters in $\mathbf{M}$) which hold in $\mathbf{M}$,
- a sentence expressing that s is a progressing move against c, and
- for each $m \in \mathbf{M}$, the sentence $s \neq m$.

Since every finite subset T'' of T' mentions finitely many $m \in \mathbf{M}$, one can construct a model of T'' by starting from $\mathbf{M}$ and setting s to be one of those progressing moves that are not mentioned. We conclude from compactness (Theorem 4) that T' is consistent.

Let $\mathbf{N}$ be a model of T'. By construction $\mathbf{N}$ is an elementary extension of $\mathbf{M}$ – in particular, $\mathbf{N}[B_r(c)]$ has the same Splitter number ℓ' as $\mathbf{M}[B_r(c)]$ – and contains a progressing move $s \in \mathbf{N} - \mathbf{M}$ against c. This means that $\mathbf{N}[B_r(c)] - s$ has Splitter number $< \ell'$. But $\mathbf{M}[B_r(c)]$ is a subgraph of $\mathbf{N}[B_r(c)] - s$ with Splitter number ℓ': this is absurd. ◀

The next section presents more elaborate tools from stability theory that will allow us to extend the above idea to different settings.

4 Stability, forking, and Finitary Substitution

This section collects notions and a few basic results from stability theory. The purpose is to give a self-contained exposition culminating in our Finitary Substitution Lemma; for more context and explanations we refer to [27].

4.1 Stability and definability of types

We say that a formula $\varphi(\bar{x}; \bar{y})$ defines a *ladder* of order k in a model $\mathbf{M}$ if there are sequences $\bar{a}_1, \dots \bar{a}_k \in \mathbf{M}^{\bar{x}}$ and $\bar{b}_1, \dots, \bar{b}_k \in \mathbf{M}^{\bar{y}}$ satisfying

$$\mathbf{M} \models \varphi(\bar{a}_i; \bar{b}_j) \qquad \text{if and only if} \qquad i < j, \qquad \text{for } 1 \leqslant i, j \leqslant k.$$

For a formula $\varphi(\bar{x}; \bar{y})$ we call the largest k such that φ defines a ladder of order k the *ladder index* of φ in $\mathbf{M}$. If no such k exists, we say that the ladder index of φ is ∞.

We say that φ is *stable* in $\mathbf{M}$ if its ladder index is finite. We say that φ is stable in a theory T if it is stable in all models of T. Moreover, we say that a model (or a theory) is stable if every formula is stable.

We now state a fundamental result about stable formulas; it states that sets definable by stable formulas with parameters in some elementary extension can actually be defined from the model itself.

▶ **Theorem 7** (Definability of types). *Let $\mathbf{M} \prec \mathbf{N}$ be two models and $\varphi(\bar{x}; \bar{y})$ be a stable formula of ladder index d in $\mathbf{M}$. For every $\bar{n} \in \mathbf{N}^{\bar{y}}$ there is a formula $\psi(\bar{x})$, which is a positive boolean combination of formulas of the form $\psi(\bar{x}; \bar{m})$ using a tuple $\bar{m}$ of $2d + 1$ parameters from $\mathbf{M}$, such that for every $\bar{a} \in \mathbf{M}^{\bar{x}}$,*

$$\mathbf{N} \models \varphi(\bar{a}; \bar{n}) \qquad \textit{if and only if} \qquad \mathbf{M} \models \psi(\bar{a}).$$

4.2 Forking in stable theories

We move on to the definition of forking, which was first defined by Shelah in order to study stable theories [26], and later grew to become the central notion of stability theory. In stable theories, forking coincides with the simpler notion of dividing, so by a slight abuse we will only work with dividing (and call it forking). We first need to formally introduce types, then we give a definition of forking in stable theories and a few useful properties.

Types. Fix a model $\mathbf{M}$ over a signature Σ. A set π of formulas in variables $\bar{x}$ with parameters from $A \subseteq \mathbf{M}$ is called a *partial type over* A if it is *consistent*: for every finite subset $\pi' \subseteq \pi$ there is $\bar{m} \in \mathbf{M}^{\bar{x}}$ which satisfies all the formulas from π' (i.e. for every formula $\varphi(\bar{x}) \in \pi'$ we have $\mathbf{M} \models \varphi(\bar{m})$). We sometimes write $\pi(\bar{x})$ to explicitly mention free variables. Partial types p which are maximal are called *types*; this amounts to stating that for every formula $\varphi(\bar{x})$ with parameters from A, either $\varphi(\bar{x}) \in p$ or $\neg\varphi(\bar{x}) \in p$. Observe that for sets $A \subseteq B \subseteq \mathbf{M}$ every type p over A can be seen as a partial type over B. We denote the set of types over A in variables $\bar{x}$ by $S_{\bar{x}}(A)$.

For a tuple $\bar{a} \in \mathbf{M}^{\bar{x}}$ and a set $A \subseteq \mathbf{M}$ of parameters, the type of $\bar{a}$ over A, denoted $\mathrm{tp}(\bar{a}/A) \in S_{\bar{x}}(A)$, is the set of all formulas $\varphi(\bar{x})$ with parameters from A such that $\mathbf{M} \models \varphi(\bar{a})$. It follows from compactness that for every $p \in S_{\bar{x}}(\mathbf{M})$ there is some $\mathbf{N} \succ \mathbf{M}$ and an $\bar{x}$-tuple $\bar{n} \in \mathbf{N}^{\bar{x}}$ such that $\mathrm{tp}(\bar{n}/\mathbf{M}) = p$.

Forking. Fix a stable model $\mathbf{M}$ over a signature Σ and a set $A \subseteq \mathbf{M}$. Let $\varphi(\bar{x};\bar{y})$ be a formula without parameters and let $\bar{b} \in \mathbf{M}^{\bar{y}}$. We say that $\varphi(\bar{x};\bar{b})$ *forks over* A if there is an elementary extension $\mathbf{N} \succ \mathbf{M}$, a sequence $\bar{b}_1, \bar{b}_2, \ldots \in \mathbf{N}^{\bar{y}}$ satisfying $\mathrm{tp}(\bar{b}_i/A) = \mathrm{tp}(\bar{b}/A)$ for every i and an integer k such that $S = \{\varphi(\bar{x};\bar{b}_i) : i \in \mathbb{N}\}$ is k-inconsistent: no k-element subset of S is consistent. For a type $p \in S_{\bar{x}}(B)$ over a set $B \subseteq \mathbf{M}$, we say that p forks over A if there is a formula $\varphi(\bar{x};\bar{b}) \in p$ which forks over A.

We will make use of the following important property of forking which is often called *(full) existence*.

▶ **Theorem 8** (See [27, Corollary 7.2.7]). *Let $\mathbf{M}$ be a stable model and let $A \subseteq B \subseteq \mathbf{M}$. For every $p \in S_{\bar{x}}(A)$ there is some $q \in S_{\bar{x}}(B)$ such that $p \subseteq q$ and q does not fork over A.*

Finitary formulas. We say that a formula $\varphi(\bar{x};\bar{y})$ is *finitary* in a theory T if for every model $\mathbf{M}$ of T, the set $\{\varphi(\mathbf{M}^{\bar{x}};\bar{m}) : m \in \mathbf{M}^{\bar{y}}\}$ is finite. By compactness, this is equivalent to the following assertion: there exists $k \in \mathbb{N}$ such that $|\{\varphi(\mathbf{M}^{\bar{x}};\bar{m}) : m \in \mathbf{M}^{\bar{y}}\}| \leqslant k$ for every model $\mathbf{M}$ of T. We now relate forking and finitary formulas.

▶ **Theorem 9** (Special case of [27, Theorem 8.5.1][3]). *Let $\mathbf{M}$ be a stable model, $\mathbf{N}$ an elementary extension of $\mathbf{M}$, $\varphi(\bar{x};\bar{y})$ a formula, $\bar{n} \in \mathbf{N}^{\bar{y}}$, and $A \subseteq \mathbf{M}$. If $\mathrm{tp}(\bar{n}/\mathbf{M})$ does not fork over A, then there is a finitary formula $\varphi'(\bar{x};\bar{z})$ and a tuple $\bar{r} \in \mathbf{M}^{\bar{z}}$ such that $\varphi(\mathbf{N}^{\bar{x}};\bar{n}) \cap \mathbf{M}^{\bar{x}} = \varphi'(\mathbf{M}^{\bar{x}};\bar{r})$.*

Combining Theorems 8 and 9 yields the following statement.

▶ **Lemma 10.** *Let $\mathbf{M}$ be a stable model over the signature Σ, $\varphi(\bar{x};\bar{y})$ a Σ-formula, and ψ a sentence over signature $\Sigma \cup \{R\}$, where $R \notin \Sigma$ has arity $|\bar{x}|$. Let $\bar{s} \in \mathbf{M}^{\bar{y}}$ be such that $\mathbf{M} \models \psi[R(\bar{x})/\varphi(\bar{x};\bar{s})]$. Then there is an elementary extension $\mathbf{N}$ of $\mathbf{M}$, a tuple $\bar{s}' \in \mathbf{N}^{\bar{y}}$, a finitary formula $\varphi'(\bar{x};\bar{z})$ and a tuple $\bar{r} \in \mathbf{M}^{\bar{z}}$, such that $\mathbf{N} \models \psi[R(\bar{x})/\varphi(\bar{x};\bar{s}')]$ and $\varphi(\mathbf{N}^{\bar{x}};\bar{s}') \cap \mathbf{M}^{\bar{x}} = \varphi'(\mathbf{M}^{\bar{x}};\bar{r})$.*

Proof. Consider $p = \mathrm{tp}(\bar{s}/\emptyset)$. By Theorem 8, p extends to a type $q \in S_{\bar{y}}(\mathbf{M})$ which does not fork over $\emptyset$. By compactness there is an elementary extension $\mathbf{N} \succ \mathbf{M}$ and a tuple $\bar{s}' \in \mathbf{N}^{\bar{y}}$ such that $\mathrm{tp}(\bar{s}'/\mathbf{M}) = q$. In particular $\mathrm{tp}(\bar{s}'/\emptyset) = p = \mathrm{tp}(\bar{s}/\emptyset)$, and therefore $\mathbf{N} \models \psi[R(\bar{x})/\varphi(\bar{x};\bar{s}')]$ as required. Applying Theorem 9 we get a finitary formula $\varphi'(\bar{x};\bar{z})$ and a tuple $\bar{r} \in \mathbf{M}^{\bar{z}}$ with the wanted properties. ◀

[3] Formally, [27, Theorem 8.5.1] speaks about definability with *imaginaries*, which is known to be equivalent to the existence of finitary formulas (see for instance [9, Lemma 1.3.2 (5), Lemma 1.3.7]).

4.3 Finitary Substitute Lemma

Recall from Section 3 that applying our method requires a mechanism for moving the wanted property ψ back towards the structure $\mathbf{M}$ we started from. This is formalized by the following definition. In a theory T, and given two sentences ψ and ψ' over the signature $\Sigma \cup \{R\}$, we say that a sentence ψ *induces* ψ' *on semi-elementary substructures* if for every model $\mathbf{M}$ of T, for every elementary extension $\mathbf{N}$ and for every $\mathcal{R} \subseteq \mathbf{N}^k$, where k is the arity of R,

$$\mathbf{N}[R/\mathcal{R}] \models \psi \qquad \text{implies} \qquad \mathbf{M}[R/\mathcal{R}|_{\mathbf{M}}] \models \psi'.$$

As an important special case, if ψ is hereditary then ψ induces ψ on semi-elementary substructures. We are now ready to state our main model-theoretic tool.

▶ **Lemma 11** (Finitary Substitute Lemma). *Let T be a theory with signature Σ, $\varphi(\bar{x};\bar{y})$ a stable formula, and ψ, ψ' be sentences over the signature $\Sigma \cup \{R\}$, where $R \notin \Sigma$ has arity $|\bar{x}|$, such that ψ induces ψ' on semi-elementary substructures. Assume that $T \models \exists \bar{s}.\psi[R(\bar{x})/\varphi(\bar{x};\bar{s})]$. Then there is a finitary formula $\varphi'(\bar{x};\bar{z})$ such that $T \models \exists \bar{s}.\psi'[R(\bar{x})/\varphi'(\bar{x};\bar{s})]$.*

The proof follows from Lemma 10 by applying compactness; we refer to the full version for details.

5 Canonization of graphs of bounded shrubdepth

In this section, we prove Theorems 2 and 3 which we now recall for convenience.

▶ **Theorem 2.** *Let $\mathscr{C}$ be a class of graphs of bounded shrubdepth. Then there is a class $\mathscr{D}$ of binary structures of bounded treedepth and a mapping $\mathcal{A}\colon \mathscr{C} \to \mathscr{D}$ such that:*

- *For each $G \in \mathscr{C}$, the vertex set of G is contained in the domain of $\mathcal{A}(G)$ and the mapping $G \mapsto \mathcal{A}(G)$ is isomorphism-invariant.*
- *Given an n-vertex graph $G \in \mathscr{C}$, the structure $\mathcal{A}(G)$ has $\mathcal{O}(n)$ elements and can be computed in time $\mathcal{O}(n^2)$.*
- *There is a simple first-order interpretation I such that $G = \mathsf{I}(\mathcal{A}(G))$, for every $G \in \mathscr{C}$.*

▶ **Theorem 3.** *For every graph class $\mathscr{C}$ of bounded shrubdepth there is an $\mathcal{O}(n^2)$-time algorithm that given n-vertex graphs $G, G' \in \mathscr{C}$, decides whether G and G' are isomorphic.*

The proof is broken into three parts.

- The first part combines insights about classes of bounded shrubdepth with our Finitary Substitute Lemma developed in the previous section, to conclude that the first level in a shrubdepth decomposition (which we will call a dicing, defined below) can be defined using finitary formulas. This result is stated as Theorem 12 below.
- The second part builds on Theorem 12 to propose a canonical transformation from classes of bounded shrubdepth to classes of bounded treedepth. This proves Theorem 2.
- In the third part, we show how Theorem 3 is derived from Theorem 2, and also establish a stronger result about the canonization problem.

We start by recalling a few preliminaries about shrubdepth in Section 5.1, and proceed with the three parts outlined above in Sections 5.2, 5.3 and 5.4.

5.1 Preliminaries on shrubdepth

Shrubdepth. The decomposition notion underlying shrubdepth is that of *connection models*, defined as follows. Let G be a graph. A *connection model* for G consists of:

- a finite set of labels Labels;
- a labelling $\mathsf{label}\colon V(G) \to \mathsf{Labels}$;
- a rooted tree T whose leaf set coincides with the vertex set of G; and
- for every non-leaf node x of T, a symmetric relation $\mathsf{Adj}(x) \subseteq \mathsf{Labels} \times \mathsf{Labels}$, called the *adjacency table* at x.

The rule is as follows: for every distinct vertices u, v of G, u and v are adjacent in G if and only if $(\mathsf{label}(u), \mathsf{label}(v)) \in \mathsf{Adj}(x)$, where x is the lowest common ancestor of u and v in T.

The *depth* of a connection model is the depth of T. The *shrubdepth* of a graph class $\mathscr{C}$ is the least integer d with the following property: there exists a finite set of labels Labels such that every graph $G \in \mathscr{C}$ has a connection model of depth at most d that uses label set Labels.

Dicings. Our inductive proof requires manipulating the first level (just below the root) of a connection model; we will call this a dicing. Formally, for a graph G, a pair $(\mathcal{P}, \mathcal{L})$ of partitions of the vertex set of G is called a *dicing* of G if for every pair of vertices u, v belonging to different parts of $\mathcal{P}$, whether u and v are adjacent in G depends only on the pair of parts of $\mathcal{L}$ that u and v belong to. In other words, there is a symmetric relation $Z \subseteq \mathcal{L} \times \mathcal{L}$ such that for all u, v belonging to different parts of $\mathcal{P}$,

$$u \text{ and } v \text{ are adjacent in } G \qquad \text{if and only if} \qquad (\mathcal{L}(u), \mathcal{L}(v)) \in Z,$$

where $\mathcal{L}(w)$ denotes the part of $\mathcal{L}$ to which w belongs. In the context of a dicing $(\mathcal{P}, \mathcal{L})$, partition $\mathcal{P}$ will be called the *component partition*, and partition $\mathcal{L}$ will be called the *label partition*. The *order* of a dicing $(\mathcal{P}, \mathcal{L})$ is $|\mathcal{L}|$, the number of parts in the label partition.

Dicings appear naturally in connection models for shrubdepth: given a connection model for a graph G, using "having a common ancestor below the root" as component partition $\mathcal{P}$ and label-classes as label partition $\mathcal{L}$ defines a dicing of G.

5.2 Definability of canonical dicings

We say that a formula $\varphi(\bar{x}; \bar{y})$ with $|\bar{x}| = 2$ *defines a partition* if for every graph G and $\bar{b} \in G^{\bar{y}}$, $\varphi(G^{\bar{x}}; \bar{b})$ is an equivalence relation on the vertex set of G. (Note that for different choices of $\bar{b}$, φ can yield different equivalence relations.) Abusing the notation, by $\varphi(G^{\bar{x}}; \bar{b})$ we will also denote the partition of the vertex set into the equivalence classes of $\varphi(G^{\bar{x}}; \bar{b})$. Recall that a formula $\varphi(\bar{x}; \bar{y})$ is said to be finitary in (the theory of) a graph class $\mathscr{C}$ if there exists k such that for all graph $G \in \mathscr{C}$,

$$|\{\varphi(G^{\bar{x}}; \bar{b}) : \bar{b} \in G^{\bar{y}}\}| \leqslant k.$$

This section is focused on establishing the following result.

▶ **Theorem 12.** *Let $\mathscr{C}$ be a class of shrubdepth at most d, where $d > 1$. Then there exists a hereditary class $\mathscr{C}'$ of shrubdepth at most $d - 1$, finitary first-order formulas $\varphi(\bar{x}; \bar{y})$ and $\lambda(\bar{x}; \bar{y})$, each defining a partition, and $\ell \in \mathbb{N}$, such that the following holds: for every graph $G \in \mathscr{C}$ there exists $\bar{s} \in G^{\bar{y}}$ such that*

- *$(\varphi(G^{\bar{x}}; \bar{s}), \lambda(G^{\bar{x}}; \bar{s}))$ is a dicing of G of order at most ℓ; and*
- *for every part A of $\varphi(G^{\bar{x}}; \bar{s})$, we have $G[A] \in \mathscr{C}'$.*

On a high level, this proves that connection models can be defined using first-order formulas $\varphi(\bar{x};\bar{y}), \lambda(\bar{x};\bar{y})$ and parameters $\bar{s} \in G^{\bar{y}}$. While a good start towards sparsification, this alone would be insufficient for our needs, as different choices of $\bar{s}$ may lead to many different connection models, and choosing an arbitrary $\bar{s}$ would not give an isomorphism-invariant construction. This difficulty is overcome by the finitariness of φ and λ: our construction will take into account all of the (boundedly many) possible dicings (see Section 5.3).

The proof of Theorem 12 is broken into three parts as follows.

- The first part consists of proving that the label partition $\mathcal{L}$ can be chosen to be definable as a partition $\lambda(G^{\bar{x}};\bar{s})$ into $\bar{s}$-types. This is achieved thanks to a more general result of Bonnet et al. [3] pertaining to classes of bounded VC-dimension.
- We then show that the component partition $\mathcal{P}$ can be chosen to be definable by a formula $\varphi(G^{\bar{x}};\bar{s})$ using the same parameters $\bar{s}$. This relies on known properties of classes of bounded shrubdepth [16].
- We then apply our Finitary Substitute Lemma (Lemma 11) and prove that φ and λ can be taken to be finitary.

Definability of the label partition. For a subset of vertices S of a graph G we let $\mathcal{L}_S$ denote the partition of the vertex set of G into neighborhood classes with respect to S: u and v belong to the same part of $\mathcal{L}_S$ if and only if

$$\{w \in S \mid u \text{ and } w \text{ are adjacent}\} = \{w \in S \mid v \text{ and } w \text{ are adjacent}\}.$$

Note that we have $|\mathcal{L}_S| \leqslant 2^{|S|}$. It turns out that label partitions can be taken of this form.

▶ **Lemma 13** (follows from Theorem 3.5 of [3]). *Let $\mathscr{C}$ be graph class of bounded shrubdepth. Then for every graph $G \in \mathscr{C}$ and dicing $(\mathcal{P},\mathcal{L})$ of G of order at most t, there exists $S \subseteq V(G)$ with $|S| \leqslant \mathcal{O}(t^2)$ such that $(\mathcal{P},\mathcal{L}_S)$ is also a dicing of G.*

Definability of the component partition. We now show that the component partition $\mathcal{P}$ can also be defined using a first-order formula.

▶ **Lemma 14.** *Let $\mathscr{C}$ be a graph class of bounded shrubdepth and $t \in \mathbb{N}$ be an integer. There exist formulas $\varphi(\bar{x};\bar{y})$ and $\lambda(\bar{x};\bar{y})$, both defining a partition, such that the following holds: for every graph $G \in \mathscr{C}$ and dicing $(\mathcal{P},\mathcal{L})$ of G of order at most t, there exists $\bar{s} \in G^{\bar{y}}$ such that*

$$(\mathcal{P}',\mathcal{L}') = \big(\varphi(G^{\bar{x}};\bar{s}), \lambda(G^{\bar{x}};\bar{s})\big)$$

is a dicing of G of order at most $2^{\mathcal{O}(t^2)}$. Further, every part of $\mathcal{P}'$ is entirely contained in some part of $\mathcal{P}$.

Proof sketch. By Lemma 13, there exists a vertex subset S with $|S| \leqslant \mathcal{O}(t^2)$ such that $(\mathcal{P},\mathcal{L}_S)$ is also a dicing of G, with relation $Z \subseteq \mathcal{L}_S \times \mathcal{L}_S$. We let H denote the graph obtained by "flipping according to the dicing $(\mathcal{P},\mathcal{L}_S)$", meaning that we exchange edges and non-edges between pairs of parts in $\mathcal{L}_S$ that belong to Z. Since $(\mathcal{P},\mathcal{L}_S)$ is a dicing, connected components of H are contained in single parts of $\mathcal{P}$; let $\mathcal{P}'$ denote the partition of $V(G) = V(H)$ into connected components in H. Since H can be transduced from G, it has bounded shrubdepth, and thus we get that each part of $\mathcal{P}'$ have diameter bounded by a constant; this is because every class of bounded shrubdepth does not admit arbitrarily long induced paths [16]. We deduce that there is a formula expressing that two vertices belong to the same $\mathcal{P}'$-component, and the result follows. ◀

Finitariness of the definition. We are now ready to derive the theorem.

Proof sketch for Theorem 12. Let Labels be a large enough set of labels so that graphs in $\mathscr{C}$ admit connection models with labels in Labels, and let $\mathscr{C}'$ be the class of all graphs that admit a connection model of depth at most $d-1$ using the label set Labels. By Lemma 14, there exist formulas $\varphi(\bar{x};\bar{y})$ and $\lambda(\bar{x};\bar{y})$, depending only on $\mathscr{C}$, such that there is $\bar{s} \in G^{\bar{y}}$ for which $(\varphi(G^{\bar{x}};\bar{s}), \lambda(G^{\bar{x}};\bar{s}))$ is a dicing of G of order at most ℓ, where $\ell \in 2^{\mathcal{O}(|\mathsf{Labels}|^2)}$ is a constant depending only on $\mathscr{C}$. Moreover, every part of $\varphi(G^{\bar{x}};\bar{s})$ is entirely contained in a single part of $\mathcal{P}$, which implies that for every part A' of $\varphi(G^{\bar{x}},\bar{s})$ we have $G[A'] \in \mathscr{C}'$. It remains to transform φ and λ into finitary formulas. Let T be the theory of $\mathscr{C}$.

Let R be a relation symbol of arity 4 and consider the following assertion:

> "R is the product of two partitions $\mathcal{P}$ and $\mathcal{L}$ such that $(\mathcal{P},\mathcal{L})$ is a dicing of G of order at most ℓ. Moreover, for every part A of $\mathcal{P}$ it holds that $G[A] \in \mathscr{C}'$".

It follows from [16, Corollary 3.9] that the assertion above can be expressed by a first order sentence ψ over the signature $\{E, R\}$. Moreover, ψ is hereditary, so we may apply the Finitary Substitute Lemma to the formula $\eta(\bar{x}_1,\bar{x}_2;\bar{y}) = \varphi(\bar{x}_1;\bar{y}) \wedge \lambda(\bar{x}_2,\bar{y})$; we get a finitary $\eta'(\bar{x}_1,\bar{x}_2;\bar{y})$ such that

$$T \quad \text{implies} \quad \exists \bar{s}.\psi[R(\bar{x}_1,\bar{x}_2)/\eta'(\bar{x}_1,\bar{x}_2;\bar{s})].$$

Then the formulas

$$\varphi'(\bar{x};\bar{y}) = \exists z.\eta'(\bar{x},z,z;\bar{y}) \quad \text{and} \quad \lambda'(\bar{x};\bar{y}) = \exists z.\eta'(z,z,\bar{x};\bar{y})$$

yield the wanted result. ◀

5.3 Canonical reduction to bounded treedepth

With Theorem 12 in hand, we proceed to the proof of Theorem 2. Fix a class $\mathscr{C}$ of shrubdepth at most d.

Properties of the construction. We describe a construction that, given a graph $G \in \mathscr{C}$, constructs a structure $\mathcal{A}(G)$ of the following shape.

- $\mathcal{A}(G)$ is a structure over a signature consisting of several unary relations and one binary relation. Thus, we see $\mathcal{A}(G)$ as a vertex-colored directed graph, and we will apply the usual directed graphs terminology to $\mathcal{A}(G)$.
- The vertex set of G is contained in the vertex set of $\mathcal{A}(G)$. The elements of $V(G)$ will be called *leaves* of $\mathcal{A}(G)$. In $\mathcal{A}(G)$ there is a unary predicate marking all the leaves.
- In $\mathcal{A}(G)$ there is a specified vertex, called the *root*, such that for every vertex u of $\mathcal{A}(G)$ there is an arc from u to the root. The root is identified using a unary predicate.

The construction will satisfy the following properties.

- The mapping $G \mapsto \mathcal{A}(G)$ is isomorphism-invariant within the class $\mathscr{C}$.
- For every vertex u of $\mathcal{A}(G)$, there are at most c arcs with tail at u, for some constant c depending only on $\mathscr{C}$.
- There is a transduction T depending on $\mathscr{C}$ such that $\mathcal{A}(G) \in \mathsf{T}(G)$.
- The class $\{\mathcal{A}(G) \mid G \in \mathscr{C}\}$ has bounded treedepth.
- There is an interpretation I depending on $\mathscr{C}$ such that $G = \mathsf{I}(\mathcal{A}(G))$.
- Given G, $\mathcal{A}(G)$ can be computed in time $\mathcal{O}(n^2)$, where n is the vertex count of G.

We proceed by induction on d, the shrubdepth of $\mathscr{C}$; the base case $d = 1$ is obvious.

Preparation for the inductive construction. Suppose $d > 1$. Let $\varphi(\bar{x};\bar{y})$, $\lambda(\bar{x};\bar{y})$, $\ell \in \mathbb{N}$, and $\mathscr{C}'$ be the finitary formulas, the bound, and the class provided by Theorem 12. Since the shrubdepth of $\mathscr{C}'$ is at most $d-1$, by induction assumption we get a suitable mapping $\mathcal{A}'(\cdot)$, constant c', transduction T', and interpretation I' that satisfy the properties stated above for $\mathscr{C}'$.

Call a tuple $\bar{s} \in G^{\bar{y}}$ *good* if $(\varphi(G^{\bar{x}};\bar{s}), \lambda(G^{\bar{x}};\bar{s}))$ is a dicing of G of order at most ℓ satisfying that for every part A of $\varphi(G^{\bar{x}},\bar{s})$ it holds that $G[A] \in \mathscr{C}'$. Define

$$\mathcal{F} = \{\, (\varphi(G^{\bar{x}};\bar{s}), \lambda(G^{\bar{x}};\bar{s})) \;:\; \bar{s} \in G^{\bar{y}} \text{ is a good tuple} \,\}.$$

By Theorem 12, we have

$$1 \leqslant |\mathcal{F}| \leqslant k$$

for some constant $k \in \mathbb{N}$ depending only on $\mathscr{C}$.

Let $\widehat{\mathcal{L}}$ be the coarsest partition that refines all label partitions of the dicings belonging to $\mathcal{F}$; that is, u, v are in the same part of $\widehat{\mathcal{L}}$ if and only if u, v are in the same part of $\mathcal{L}$ for each $(\mathcal{P},\mathcal{L}) \in \mathcal{F}$. Similarly, let $\widehat{\mathcal{P}}$ be the coarsest partition that refines all component partitions of the dicings belonging to $\mathcal{F}$. Since $|\mathcal{F}| \leqslant k$ and $|\mathcal{L}| \leqslant \ell$ for each label partition featured in $\mathcal{F}$, we have

$$|\widehat{\mathcal{L}}| \leqslant \ell^k.$$

Moreover, every part of $\widehat{\mathcal{P}}$ is contained in a single part of any component partition featured in $\mathcal{F}$, hence $G[B] \in \mathscr{C}'$ for every part B of $\widehat{\mathcal{P}}$.

Let $\widehat{\mathcal{F}} = \{(\mathcal{P},\widehat{\mathcal{L}}) \colon (\mathcal{P},\mathcal{L}) \in \mathcal{F}\}$. Since $\widehat{\mathcal{L}}$ refines each label partition featured in $\mathcal{F}$, it follows that every element of $\widehat{\mathcal{F}}$ is a dicing of G. Then, for a component partition $\mathcal{P}$ featured in $\mathcal{F}$, let $Z_{\mathcal{P}} \subseteq \widehat{\mathcal{L}} \times \widehat{\mathcal{L}}$ be the symmetric relation witnessing that $(\mathcal{P},\widehat{\mathcal{L}})$ is a dicing.

Definition of the construction. We now describe the structure $\mathcal{A}(G)$; see Figure 1. Construct:

- a root vertex r;
- for every part $L \in \widehat{\mathcal{L}}$, a vertex x_L;
- for every component partition $\mathcal{P}$ featured in $\mathcal{F}$, a vertex $y_{\mathcal{P}}$;
- for every component partition $\mathcal{P}$ featured in $\mathcal{F}$, and every part $A \in \mathcal{P}$, a vertex $z_{\mathcal{P},A}$; and
- for every component partition $\mathcal{P}$ featured in $\mathcal{F}$, and every (unordered) pair $LL' \in Z_{\mathcal{P}}$, a vertex $q_{\mathcal{P},LL'}$. (Note here that $Z_{\mathcal{P}}$ is symmetric, so we may treat its elements as unordered pairs of elements of $\widehat{\mathcal{L}}$.)

Further, for every part B of $\widehat{\mathcal{P}}$ we have $G[B] \in \mathscr{C}'$, hence we may apply the construction $\mathcal{A}'$ to $G[B]$, yielding a structure $H_B = \mathcal{A}'(G[B])$. Let r_B be the root of H_B. We add all structures H_B obtained in this way to $\mathcal{A}(G)$. We then connect these with the following arcs:

1. for every vertex u of $\mathcal{A}(G)$ there is an arc (u,r);
2. for every vertex of the form $z_{\mathcal{P},A}$ there is an arc $(z_{\mathcal{P},A}, y_{\mathcal{P}})$;
3. for every vertex of the form $q_{\mathcal{P},LL'}$, there are arcs $(q_{\mathcal{P},LL'}, x_L)$, $(q_{\mathcal{P},LL'}, x_{L'})$, and $(q_{\mathcal{P},LL'}, y_{\mathcal{P}})$;
4. for every part B of $\widehat{\mathcal{P}}$ and every component partition $\mathcal{P}$ featured in $\mathcal{F}$, there is an arc $(r_B, z_{\mathcal{P},A})$, where A is the unique part of $\mathcal{P}$ that contains B;
5. for every vertex u of G there is an arc (u, x_L), where L is the unique part of $\widehat{\mathcal{L}}$ that contains u. (Recall that the vertex set of G is the union of the leaf sets of H_B for $B \in \widehat{\mathcal{P}}$.)

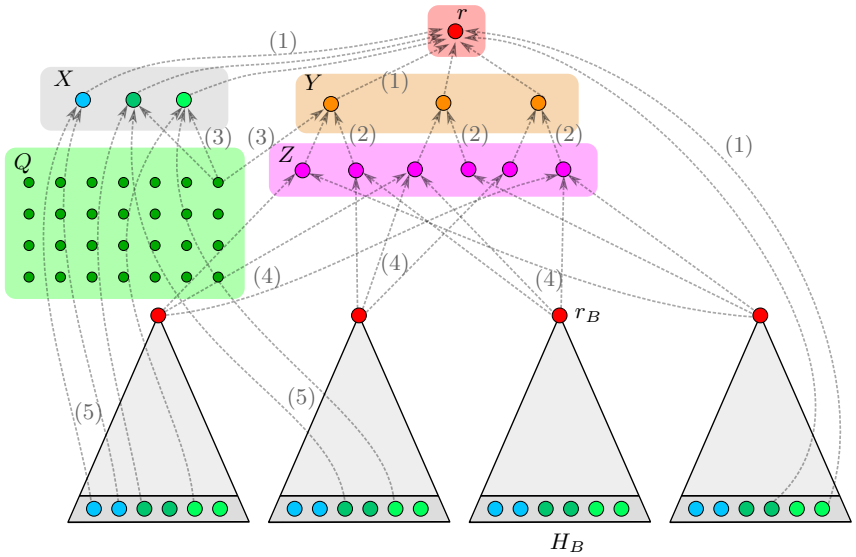


Figure 1 Inductive construction of $\mathcal{A}(G)$. Vertices of the form r, $y_{\mathcal{P}}$, $z_{\mathcal{P},A}$, $q_{\mathcal{P},LL'}$ are depicted in red, orange, violet, and green, respectively. Vertices of the form x_L are depicted in the top-left corner of the figure in different soft colors (which do not correspond to unary predicates), matching the colors of vertices of G that point to them; thus the soft color partition is $\widehat{\mathcal{L}}$. We depict a few representatives for each type of arcs.

Finally, we add five fresh unary predicates, called R, X, Y, Z, Q, respectively selecting the root r, the vertices of the form x_L, the vertices of the form $y_{\mathcal{P}}$, the vertices of the form $z_{\mathcal{P},A}$, and the vertices of the form $q_{\mathcal{P},LL'}$. Note here that H contains more unary predicates: those that come with structures H_B constructed by induction. These include a unary relation selecting the leaves.

This concludes the construction of $\mathcal{A}(G)$. We do not include detailed proofs of the properties listed above, and refer instead to the full version. That the transformation is isomorphism-invariant and that every element is the tail of a bounded number of arcs follows directly from the construction. Also, it is quite straightforward to transduce $\mathcal{A}(G)$ from G, by guessing good tuples $\bar{s}_1, \ldots, \bar{s}_{k'}$; then it follows from the results of Ganian et al. [16] that the class of $\mathscr{D} = \{\mathcal{A}(G) \colon G \in \mathscr{C}\}$ has bounded shrubdepth. This, together with the sparsity of $\mathcal{A}(G)$ following from the bound on outdegrees, implies that $\mathscr{D}$ in fact has bounded treedepth. Further, there is no difficulty in interpreting G in $\mathcal{A}(G)$. To compute $\mathcal{A}(G)$ in quadratic time, we rely on algorithmic meta-theorems over graphs of bounded cliquewidth obtained from combining [12, 21]. Theorem 2 follows.

5.4 Canonization and isomorphism test

We now use Theorem 2 to prove Theorem 3, that is, give a quadratic-time isomorphism test for any class of graphs of bounded shrubdepth. As mentioned in the introduction, in fact we solve the more general canonization problem, defined as follows.

For a class of structures $\mathscr{C}$, a *canonization map* for $\mathscr{C}$ is a mapping $\mathfrak{c}$ with the following property: for every $\mathbf{M} \in \mathscr{C}$, $\mathfrak{c}(\mathbf{M})$ is a total order on elements of $\mathbf{M}$ so that if $\mathbf{M}, \mathbf{M}' \in \mathscr{C}$ are isomorphic, then associating elements with same index in $\mathfrak{c}(\mathbf{M})$ and in $\mathfrak{c}(\mathbf{M}')$ yields an isomorphism between $\mathbf{M}$ and $\mathbf{M}'$. Note that if there is a canonization map $\mathfrak{c}$ for $\mathscr{C}$ that is efficiently computable, then this immediately gives an isomorphism test within the same time complexity.

For classes of bounded treedepth, Bouland et al. [4] gave a relatively easy fixed-parameter isomorphism test. Their techniques can be easily extended to the canonization problem for binary structures of bounded treedepth.

▶ **Theorem 15** (Adapted from [4]). *Let $\mathcal{D}$ be a class of binary structures of bounded treedepth. There exists a canonization map $\mathfrak{c}$ on $\mathcal{D}$ that is computable in time $\mathcal{O}(n \log^2 n)$, where n is the size of the universe of the input structure.*

We can now prove the main result of this section.

▶ **Theorem 16.** *Let $\mathscr{C}$ be a class of graphs of bounded shrubdepth. There exists a canonization map $\mathfrak{c}$ on $\mathscr{C}$ that is computable in time $\mathcal{O}(n^2)$, where n is the vertex count of the input graph.*

Proof. Let $\mathscr{D}$ be the class of bounded treedepth and $\mathcal{A}\colon \mathscr{C} \to \mathscr{D}$ be the mapping provided by Theorem 2 for the class $\mathscr{C}$. Then to get a suitable canonization map for $\mathscr{C}$, it suffices to compose $\mathcal{A}$ with the canonization map for $\mathscr{D}$, provided by Theorem 15, and restrict the output order to the vertex set of the original graph. ◀

As discussed, Theorem 3 follows immediately from Theorem 16.

References

1 Algorithms, Logic and Structure Workshop in Warwick – Open Problem Session. URL: https://warwick.ac.uk/fac/sci/maths/people/staff/daniel_kral/alglogstr/openproblems.pdf, 2016.

2 Hans Adler and Isolde Adler. Interpreting nowhere dense graph classes as a classical notion of model theory. *Eur. J. Comb.*, 36:322–330, 2014. doi:10.1016/j.ejc.2013.06.048.

3 Édouard Bonnet, Jan Dreier, Jakub Gajarský, Stephan Kreutzer, Nikolas Mählmann, Pierre Simon, and Szymon Torunczyk. Model Checking on Interpretations of Classes of Bounded Local Cliquewidth. In *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 54:1–54:13. ACM, 2022. doi:10.1145/3531130.3533367.

4 Adam Bouland, Anuj Dawar, and Eryk Kopczyński. On Tractable Parameterizations of Graph Isomorphism. In *7th International Symposium on Parameterized and Exact Computation, IPEC 2012*, volume 7535 of *Lecture Notes in Computer Science*, pages 218–230. Springer, 2012. doi:10.1007/978-3-642-33293-7_21.

5 Samuel Braunfeld and Michael C. Laskowski. Characterizations of monadic NIP. *Transactions of the American Mathematical Society, Series B*, 8(30):948–970, 2021. doi:10.1090/btran/94.

6 Samuel Braunfeld and Michael C. Laskowski. Existential characterizations of monadic NIP. *CoRR*, abs/2209.05120, 2022. doi:10.48550/arXiv.2209.05120.

7 Samuel Braunfeld, Jaroslav Nešetřil, Patrice Ossona de Mendez, and Sebastian Siebertz. Decomposition horizons: from graph sparsity to model-theoretic dividing lines. *CoRR*, abs/2209.11229, 2022. doi:10.48550/arXiv.2209.11229.

8 Yijia Chen and Jörg Flum. FO-Definability of Shrub-Depth. In *28th EACSL Annual Conference on Computer Science Logic, CSL 2020*, volume 152 of *LIPIcs*, pages 15:1–15:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CSL.2020.15.

9 Artem Chernikov. Lecture notes on stability theory. AMS Open Math Notes (OMN: 201905.110792), 2019.

10 Jan Dreier, Nikolas Mählmann, and Sebastian Siebertz. First-order model checking on structurally sparse graph classes. *CoRR*, abs/2302.03527, 2023. doi:10.48550/arXiv.2302.03527.

11 Jan Dreier, Nikolas Mählmann, Sebastian Siebertz, and Szymon Torunczyk. Indiscernibles and flatness in monadically stable and monadically NIP classes. *CoRR*, abs/2206.13765, 2022. doi:10.48550/arXiv.2206.13765.

12 Fedor V. Fomin and Tuukka Korhonen. Fast FPT-approximation of branchwidth. In *54th Annual ACM SIGACT Symposium on Theory of Computing, STOC '22*, pages 886–899. ACM, 2022. doi:10.1145/3519935.3519996.

13 Jakub Gajarský and Petr Hliněný. Faster Deciding MSO Properties of Trees of Fixed Height, and Some Consequences. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012*, volume 18 of *LIPIcs*, pages 112–123. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPIcs.FSTTCS.2012.112.

14 Jakub Gajarský, Stephan Kreutzer, Jaroslav Nešetřil, Patrice Ossona de Mendez, Michał Pilipczuk, Sebastian Siebertz, and Szymon Toruńczyk. First-Order Interpretations of Bounded Expansion Classes. *ACM Trans. Comput. Log.*, 21(4):29:1–29:41, 2020. doi:10.1145/3382093.

15 Jakub Gajarský, Nikolas Mählmann, Rose McCarty, Pierre Ohlmann, Michał Pilipczuk, Wojciech Przybyszewski, Sebastian Siebertz, Marek Sokolowski, and Szymon Torunczyk. Flipper games for monadically stable graph classes. *CoRR*, abs/2301.13735, 2023. doi:10.48550/arXiv.2301.13735.

16 Robert Ganian, Petr Hliněný, Jaroslav Nešetřil, Jan Obdržálek, and Patrice Ossona de Mendez. Shrub-depth: Capturing Height of Dense Graphs. *Log. Methods Comput. Sci.*, 15(1), 2019. doi:10.23638/LMCS-15(1:7)2019.

17 Robert Ganian, Petr Hliněný, Jaroslav Nešetřil, Jan Obdržálek, Patrice Ossona de Mendez, and Reshma Ramadurai. When Trees Grow Low: Shrubs and Fast MSO_1. In *37th International Symposium on Mathematical Foundations of Computer Science 2012, MFCS 2012*, volume 7464 of *Lecture Notes in Computer Science*, pages 419–430. Springer, 2012. doi:10.1007/978-3-642-32589-2_38.

18 Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding First-Order Properties of Nowhere Dense Graphs. *J. ACM*, 64(3):17:1–17:32, 2017. doi:10.1145/3051095.

19 Martin Grohe and Daniel Neuen. Isomorphism, canonization, and definability for graphs of bounded rank width. *Commun. ACM*, 64(5):98–105, 2021. doi:10.1145/3453943.

20 Martin Grohe and Pascal Schweitzer. Isomorphism Testing for Graphs of Bounded Rank Width. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015*, pages 1010–1029. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.66.

21 Wojciech Kazana and Luc Segoufin. Enumeration of monadic second-order queries on trees. *ACM Trans. Comput. Log.*, 14(4):25:1–25:12, 2013. doi:10.1145/2528928.

22 O-joung Kwon, Rose McCarty, Sang-il Oum, and Paul Wollan. Obstructions for bounded shrub-depth and rank-depth. *J. Comb. Theory, Ser. B*, 149:76–91, 2021. doi:10.1016/j.jctb.2021.01.005.

23 Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity: Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and Combinatorics*. Springer, 2012. doi:10.1007/978-3-642-27875-4.

24 A. Pillay. *Geometric Stability Theory*. Oxford logic guides. Clarendon Press, 1996. URL: https://books.google.pl/books?id=k6FK_Gqal2EC.

25 Klaus-Peter Podewski and Martin Ziegler. Stable graphs. *Fundamenta Mathematicae*, 100(2):101–107, 1978. doi:10.4064/fm-100-2-101-107.

26 Saharon Shelah. *Classification theory – and the number of non-isomorphic models*, volume 92 of *Studies in logic and the foundations of mathematics*. North-Holland Publishing Co., 1978.

27 Katrin Tent and Martin Ziegler. *A Course in Model Theory*. Lecture Notes in Logic. Cambridge University Press, 2012. doi:10.1017/CBO9781139015417.

Probabilistic Guarded KAT Modulo Bisimilarity: Completeness and Complexity

Wojciech Różowski
Department of Computer Science, University College London, UK

Tobias Kappé
Open Universiteit, Heerlen, The Netherlands
ILLC, University of Amsterdam, The Netherlands

Dexter Kozen
Department of Computer Science, Cornell University, Ithaca, NY, USA

Todd Schmid
Department of Computer Science, University College London, UK

Alexandra Silva
Department of Computer Science, Cornell University, Ithaca, NY, USA

Abstract

We introduce Probabilistic Guarded Kleene Algebra with Tests (ProbGKAT), an extension of GKAT that allows reasoning about uninterpreted imperative programs with probabilistic branching. We give its operational semantics in terms of special class of probabilistic automata. We give a sound and complete Salomaa-style axiomatisation of bisimilarity of ProbGKAT expressions. Finally, we show that bisimilarity of ProbGKAT expressions can be decided in $O(n^3 \log n)$ time via a generic partition refinement algorithm.

2012 ACM Subject Classification Theory of computation → Program reasoning

Keywords and phrases Kleene Algebra with Tests, program equivalence, completeness, coalgebra

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.136

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `https://arxiv.org/abs/2305.01755` [51]

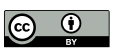
Funding This work was partially supported by ERC grant Autoprobe (no. 101002697; Różowski, Schmid and Silva), the EU's Horizon 2020 research and innovation program under Marie Skłodowska-Curie grant VERLAN (no. 101027412; Kappé), and NSF grant CCF-2008083 (Kozen).

1 Introduction

Randomisation is an important feature in the design of efficient algorithms, cryptographic protocols, and stochastic simulation [9]. For a simple example of randomisation, imagine simulating a three-sided die [64]. There are at least two ways to do this:

- A reference implementation could use a fair coin and a biased coin with probability $\frac{1}{3}$ of landing on `heads`: Toss the biased coin first. If it lands on `heads`, return ⚀, and otherwise toss the fair coin and return ⚁ if it lands on `heads` or ⚂ otherwise.
- Another way to do this is with two consecutive tosses of a fair coin: if the outcome is `heads-heads`, then return ⚀; if it is `heads-tails`, return ⚁; if it is `tails-heads`, return ⚂; and if it is `tails-tails`, repeat the process [34].

These programs can be written using a function `flip`(p) that returns `true` (`heads`) with probability p, and `false` (`tails`) with probability $1-p$, see Figure 1. If we can prove that those programs are equivalent, then we can be certain they implement the same distribution.



```
while true do
    if flip(0.5) then
        if flip(0.5) then
            return ⚀
        else
            return ⚁
    else
        if flip(0.5) then
            return ⚂
        else
            skip
```

```
if flip(1/3) then
    return ⚀
else
    if flip(0.5) then
        return ⚁
    else
        return ⚂
```

Figure 1 On the left, the Knuth-Yao process to simulate a three-sided die with two throws of a fair coin. On the right, direct implementation of the three-sided die (note that the probability of the first else-branch is $\frac{2}{3}$, hence both ⚀ and ⚁ are returned with probability $\frac{1}{3}$).

In this paper, we introduce Probabilistic GKAT (ProbGKAT), a language based on Guarded Kleene Algebra with Tests (GKAT) [39, 58, 53] augmented with extra constructs for reasoning about such randomised programs. The laws of GKAT allow reasoning about the equivalence of uninterpreted programs with deterministic control flow in the form of Boolean branching (`if-then-else`) and looping (`while-do`) constructs. GKAT comes equipped with an automata-theoretic operational semantics, a *nearly linear* decision procedure, and complete axiomatic systems for reasoning about trace equivalence [58] and bisimilarity [53] of expressions, both inspired by Salomaa's axiomatisation of Kleene Algebra [52].

ProbGKAT extends GKAT with three new syntactic constructs: (1) a probabilistic choice operator, representing branching based on a (possibly biased) coin flip; (2) a probabilistic loop operator, representing a generalised Bernoulli process; and (3) return values, which allow a limited form of non-local control flow akin to `return` statements in imperative programming.

The main focus of this paper is the problem of axiomatising bisimilarity of ProbGKAT expressions. We build on an inference system for reasoning about bisimilarity of GKAT expressions [53], which includes a generalisation of Salomaa's axiomatisation of the Kleene star [52] called the Uniqueness of Solutions axiom (UA), also known in the process algebra community as Recursive Specification Principle (RSP) [10]. In the presence of both Boolean guarded and probabilistic branching, axiomatisation becomes significantly more involved. Besides adding intuitive rules governing the behaviour of probabilistic choice and loops, we add axioms capturing the interaction of both kinds of branching when combined with looping constructs. Moreover, in the case of ProbGKAT, showing the soundness of UA becomes highly nontrivial. We do so by exploiting the topological structure of the operational model, namely the *behavioural pseudometric* associated with bisimilarity. Despite the jump in difficulty, our completeness proof follows a similar strategy as the one for GKAT modulo bisimilarity [53].

Our main contributions are as follows.

- We provide an operational semantics of ProbGKAT programs, which relies on a type of automata that have both Boolean guarded and probabilistic transitions (Section 3).
- We concretely characterise bisimulations for our automata using an adaption of the flow network characterisation of bisimilarity of Markov chains [30, 8] (Section 4).
- We give a sound and complete Salomaa-style axiomatisation of bisimulation equivalence of ProbGKAT expressions (Sections 5 and 6).

$$
\begin{array}{rll}
\mathsf{e}, \mathsf{f} \in \mathtt{Exp} ::= & \mathsf{p} \in \mathsf{Act} & \mathsf{do\ p} \\
& \mid \mathsf{b} \in \mathtt{BExp} & \mathsf{assert\ b} \\
& \mid \mathsf{e} +_{\mathsf{b}} \mathsf{f} & \mathsf{if\ b\ then\ e\ else\ f} \\
& \mid \mathsf{e}\,;\mathsf{f} & \\
& \mid \mathsf{e}^{(\mathsf{b})} & \mathsf{while\ b\ do\ e} \\
& \mid v \in \mathsf{Out} & \mathsf{return}\ v \\
& \mid \mathsf{e} \oplus_r \mathsf{f} & \mathsf{if\ flip}(r)\ \mathsf{then\ e\ else\ f} \\
& \mid \mathsf{e}^{[r]} & \mathsf{while\ flip}(r)\ \mathsf{do\ e}
\end{array}
$$

Figure 2 Syntax of ProbGKAT.

- We show that, when the number of tests is fixed, bisimilarity of ProbGKAT expressions can be efficiently decided in $\mathcal{O}(n^3 \log n)$ time, where n is the total size of programs under comparison, by using *coalgebraic partition refinement* [17, 67] (Section 7).

In Section 2 we define the syntax of ProbGKAT. We survey related work in Section 8; conclusions and further work appear in Section 9. Proofs appear in the full version [51].

2 Syntax

ProbGKAT has a two-sorted syntax consisting of a set of *expressions* $\mathtt{Exp}$ that contains a set $\mathtt{BExp}$ of *Boolean assertions* or *tests*. For a fixed finite set T of *primitive tests*, the syntax for tests is denoted $\mathtt{BExp}$ and generated by the grammar

$$\mathsf{b}, \mathsf{c} \in \mathtt{BExp} ::= 0 \mid 1 \mid t \in \mathsf{T} \mid \mathsf{b} + \mathsf{c} \mid \mathsf{bc} \mid \bar{\mathsf{b}}$$

Here, 0 and 1 respectively denote false and true, $\bar{\cdot}$ denotes negation, $+$ is disjunction, and juxtaposition is conjunction. Let $\equiv_{\mathsf{BA}}$ denote Boolean equivalence in $\mathtt{BExp}$. Entailment is a preorder on $\mathtt{BExp}$ given by $\mathsf{b} \leq_{\mathsf{BA}} \mathsf{c} \iff \mathsf{b}+\mathsf{c} \equiv_{\mathsf{BA}} \mathsf{c}$. The quotient of $\mathtt{BExp}$ by $\equiv_{\mathsf{BA}}$ is the free Boolean algebra on the set of generators T, in which entailment – $[\mathsf{b}]_{\equiv_{\mathsf{BA}}} \leq [\mathsf{c}]_{\equiv_{\mathsf{BA}}} \iff \mathsf{b} \leq_{\mathsf{BA}} \mathsf{c}$ – is a partial order, with bottom and top elements being the equivalence classes of 0 and 1 respectively. The minimal non-zero elements of that partial order are called *atoms*, and we will use At to denote the set of atoms. For fixed sets Act of *atomic actions* and Out of *return values*, the set $\mathtt{Exp}$ of ProbGKAT expressions is defined by the grammar in Figure 2.

The syntax of GKAT is captured by the first five cases in Figure 2, and so is a proper fragment of ProbGKAT. There are three new constructs: *return values*, *probabilistic choices*, and *probabilistic loops*. Return values behave like return statements in imperative programs, introducing a form of non-local control flow. The probabilistic choice $\mathsf{e} \oplus_r \mathsf{f}$ flips a biased coin with real bias $r \in [0, 1]$ and depending on the outcome runs e with probability r and f with probability $1 - r$. The probabilistic loop $\mathsf{e}^{[r]}$ also begins with a biased coin flip, and depending on the outcome it either executes e and starts again (probability r) or terminates (probability $1 - r$). A probabilistic loop can be regarded as a generalised Bernoulli processes.

▶ **Example 1.** Recall the two programs from the introduction (Figure 1): one directly implementing a 3-sided die and the other simulating a 3-sided die with a fair coin. We can express these programs using three output values, ⚀, ⚁, and ⚂, to model the possible outcomes of the three-sided die. The first program is the infinite while loop $\mathsf{f} = \mathsf{g}^{(1)}$, where

the loop body is given by $\mathsf{g} = (\boxdot \oplus_{\frac{1}{2}} \boxdot) \oplus_{\frac{1}{2}} (\boxdot \oplus_{\frac{1}{2}} 1)$. In f, $\boxdot$ represents `heads-heads`, $\boxdot$ is `heads-tails`, $\boxdot$ is `tails-heads`, and `tails-tails` prompts a rethrow. The second program encodes the ProbGKAT expression $\mathsf{e} = \boxdot \oplus_{\frac{1}{3}} (\boxdot \oplus_{\frac{1}{2}} \boxdot)$.

3 Operational semantics

In this section, we formally introduce ProbGKAT automata, the operational models of ProbGKAT expressions. We associate a ProbGKAT automaton with each expression via a small-step semantics inspired by Brzozowski derivatives [13]. As we will see, the biggest hurdle is the semantics of the probabilistic loop. Before we provide our small-step semantics, we introduce the notation and operations on probability distributions that we will need.

Preliminary definitions. A function $\nu : X \to [0, 1]$ is called a *(probability) distribution on* X if it satisfies $\sum_{x \in X} \nu(x) = 1$. In case $\sum_{x \in X} \nu(x) \leq 1$ we call ν a *subprobability distribution*, or *subdistribution*. Every (sub)distribution ν in this paper is *finitely supported*, which means that the set $\mathtt{supp}(\nu) = \{x \in X \mid \nu(x) > 0\}$ is finite. Given $A \subseteq X$, we define $\nu[A] = \sum_{x \in A} \nu(x)$. This sum is well-defined because only finitely many summands have non-zero probability.

We use $\mathcal{D}_\omega(X)$ to denote the set of finitely supported probability distributions on the set X. A function $f : X \to Y$ can be *lifted* to a map $\mathcal{D}_\omega(f) : \mathcal{D}_\omega(X) \to \mathcal{D}_\omega(Y)$ between distributions by setting $\mathcal{D}_\omega(f)(\nu) = \nu[f^{-1}(y)]$. Given $x \in X$, its *Dirac delta* is the distribution δ_x; here $\delta_x(y)$ is equal to 1 when $x = y$, and 0 otherwise. Given $f : X \to \mathcal{D}_\omega(Y)$, there is a unique map $\bar{f} \colon \mathcal{D}_\omega(X) \to \mathcal{D}_\omega(Y)$ such that $f = \bar{f} \circ \delta$, called the *convex extension of* f, and explicitly given by $\bar{f}(\nu)(y) = \sum_{x \in X} \nu(x) f(x)(y)$.

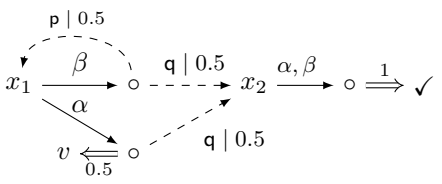
When $\nu, \mu : X \to [0, 1]$ are probability distributions and $r \in [0, 1]$, we write $r\nu + (1 - r)\mu$ for the *convex combination* of ν and μ, which is the probability distribution given by $(r\nu + (1 - r)\mu)(x) = r\nu(x) + (1 - r)\mu(x)$; this operation preserves finite support.

Operational model. Operationally, ProbGKAT expressions denote states in a transition system called a ProbGKAT *automaton*. Below, we write $2 = \{\text{✗}, \text{✓}\}$ for a two element set of symbols denoting rejection and acceptance respectively.

▶ **Definition 2.** *A* ProbGKAT automaton *is a pair* (X, β) *consisting of set of* states X *and a* transition function $\beta : X \times \mathsf{At} \to \mathcal{D}_\omega(2 + \mathsf{Out} + \mathsf{Act} \times X)$.

A state in a ProbGKAT automaton associates each Boolean atom $\alpha \in \mathsf{At}$ (capturing the global state of the Boolean variables) with a finitely supported probability distribution over several possible outcomes. One possible outcome is *termination*, which ends execution and either signals success (✓) or failure (✗), or returns an output value ($v \in \mathsf{Out}$). The other possible outcome is *progression*, performing an action ($\mathsf{p} \in \mathsf{Act}$) and transitioning to a state.

▶ **Example 3.** Let $X = \{x_1, x_2\}$, $\mathsf{At} = \{\alpha, \beta\}$, $\mathsf{Act} = \{\mathsf{p}, \mathsf{q}\}$ and $\mathsf{Out} = \{v\}$. On the right, there is a definition of a transition function $\tau : X \times \mathsf{At} \to \mathcal{D}_\omega(2 + \mathsf{Out} + \mathsf{Act} \times X)$, while on the left there is a visual representation of (X, τ). Given a state $x \in X$ and an atom $\alpha \in \mathsf{At}$, we write $\tau(x)_\alpha$ rather than $\tau(x)(\alpha)$.



$$\tau(x_1)_\alpha = \frac{1}{2}\delta_{(\mathsf{q},x_2)} + \frac{1}{2}\delta_v$$

$$\tau(x_1)_\beta = \frac{1}{2}\delta_{(\mathsf{p},x_1)} + \frac{1}{2}\delta_{(\mathsf{q},x_2)}$$

$$\tau(x_2)_\alpha = \tau(x_2)_\beta = \delta_\text{✓}$$

We use solid lines annotated with (sets of) atoms to denote Boolean guarded branching, dashed lines annotated with atomic actions and probabilities to denote probabilistic labelled transitions to a next state, and double bar arrows pointing at elements of $2 + \mathsf{Out}$ annotated with probabilities to denote probabilistic transitions that result in termination or output.

The following notions of homomorphism and bisimulation describe structure-preserving maps and relations between ProbGKAT automata.

▶ **Definition 4.** *A* homomorphism *between* ProbGKAT *automata* (X, β) *and* (Y, γ) *is a function* $f : X \to Y$ *satisfying for all* $x \in X$ *and* $\alpha \in \mathsf{At}$

1. *For any* $o \in 2 + \mathsf{Out}$, $\gamma(f(x))_\alpha(o) = \beta(x)_\alpha(o)$
2. *For any* $(\mathsf{p}, y) \in \mathsf{Act} \times Y$, $\gamma(f(x))_\alpha(\mathsf{p}, y) = \beta(x)_\alpha[\{\mathsf{p}\} \times f^{-1}(y)]$

▶ **Definition 5.** *Let* (X, β) *and* (Y, γ) *be* ProbGKAT *automata and let* $R \subseteq X \times Y$ *be a relation.* R *is a* bisimulation *if there exists a transition function* $\rho : R \times \mathsf{At} \to \mathcal{D}_\omega(2 + \mathsf{Out} + \mathsf{Act} \times R)$ *such that projection maps* $\pi_1 : R \to X$ *and* $\pi_2 : R \to Y$ *given by* $\pi_1(x, y) = x$ *and* $\pi_2(x, y) = y$ *are homomorphisms from* (R, ρ) *to* (X, β) *and* (Y, γ) *respectively.*

▶ Remark 6. Definitions 4 and 5 are direct translations from the coalgebraic theory of ProbGKAT automata [51, Appendix A]. Coalgebra plays a central role in our proofs, but for purposes of exposition it does not appear in the body of the present paper.

Brzozowski construction. ProbGKAT expressions can be endowed with an operational semantics in the form of a ProbGKAT automaton $\partial : \mathtt{Exp} \times \mathsf{At} \to \mathcal{D}_\omega(2 + \mathsf{Out} + \mathsf{Act} \times \mathtt{Exp})$, which we refer to as the *Brzozowski derivative*, as it is reminiscent of the analogous construction for regular expressions and deterministic finite automata due to Brzozowski [13].

Given $\alpha \in \mathsf{At}$, $\mathsf{e}, \mathsf{f} \in \mathtt{Exp}$, $\mathsf{b} \in \mathtt{BExp}$, $v \in \mathsf{Out}$, $r \in [0, 1]$, and $\mathsf{p} \in \mathsf{Act}$, we define

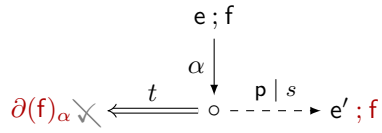
$$\partial(\mathsf{b})_\alpha = \begin{cases} \delta_\checkmark & \alpha \leq_{\mathsf{BA}} \mathsf{b} \\ \delta_\times & \alpha \leq_{\mathsf{BA}} \overline{\mathsf{b}} \end{cases} \qquad \partial(\mathsf{e} +_\mathsf{b} \mathsf{f})_\alpha = \begin{cases} \partial(\mathsf{e})_\alpha & \alpha \leq_{\mathsf{BA}} \mathsf{b} \\ \partial(\mathsf{f})_\alpha & \alpha \leq_{\mathsf{BA}} \overline{\mathsf{b}} \end{cases}$$

$$\partial(v)_\alpha = \delta_v \qquad \partial(\mathsf{p})_\alpha = \delta_{(\mathsf{p},1)} \qquad \partial(\mathsf{e} \oplus_r \mathsf{f})_\alpha = r\partial(\mathsf{e})_\alpha + (1 - r)\partial(\mathsf{f})_\alpha$$

The derivatives of sequential composition and loops are defined below. The outgoing transitions of $\mathsf{b} \in \mathtt{BExp}$ depend on whether or not the input atom $\alpha \in \mathsf{At}$ satisfies b, either outputting ✓ (success) or ✗ (abort) with probability 1. The outgoing transitions of a guarded choice $\mathsf{e} +_\mathsf{b} \mathsf{f}$ consist of the outgoing transitions of e labelled by atoms satisfying b and the outgoing transitions of f labelled by atoms satisfying $\overline{\mathsf{b}}$ (as in GKAT). The output value $v \in \mathsf{Out}$ returns the value v with probability 1 given any input atom. The atomic action $\mathsf{p} \in \mathsf{Act}$ emits p given any input atom and transitions to the expression 1. The outgoing transitions of the probabilistic choice $\mathsf{e} \oplus_r \mathsf{f}$ consist of the outgoing transitions of e with probabilities scaled by r and the outgoing transitions of f scaled by $1 - r$.

The behaviour of the sequential composition $\mathsf{e}\,;\mathsf{f}$ is more complicated. We need to factor in the possibility that e may accept with some probability t given an input atom α, in which case the α-labelled outgoing transitions of f contribute to the outgoing transitions of $\mathsf{e}\,;\mathsf{f}$. Formally, we write $\partial(\mathsf{e}\,;\mathsf{f})_\alpha = \partial(\mathsf{e})_\alpha \lhd_\alpha \mathsf{f}$, where given $\alpha \in \mathsf{At}$ and $\mathsf{f} \in \mathtt{Exp}$ we define $(- \lhd_\alpha \mathsf{f}) : \mathcal{D}_\omega(2 + \mathsf{Out} + \mathsf{Act} \times \mathtt{Exp}) \to \mathcal{D}_\omega(2 + \mathsf{Out} + \mathsf{Act} \times \mathtt{Exp})$ to be the convex extension of $c_{\alpha,\mathsf{f}} : 2 + \mathsf{Out} + \mathsf{Act} \times \mathtt{Exp} \to \mathcal{D}_\omega(2 + \mathsf{Out} + \mathsf{Act} \times \mathtt{Exp})$ given below on the left.

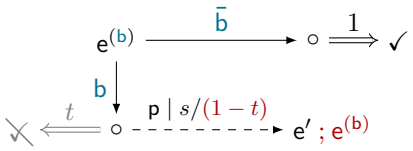
$$c_{\alpha,\mathsf{f}}(x) = \begin{cases} \delta_x & x \in \{\checkmark\} \cup \mathsf{Out} \\ \partial(\mathsf{f})_\alpha & x = \checkmark \\ \delta_{(\mathsf{p},\mathsf{e}';\mathsf{f})} & x = (\mathsf{p},\mathsf{e}') \end{cases}$$



Intuitively, $c_{\alpha,\mathsf{f}}$ reroutes the transitions coming out of e: acceptance (the second case) is replaced by the behaviour of f, and the probability mass of transitioning to e' (the third case) is reassigned to $\mathsf{e}\,;\mathsf{f}$. The branches that output the elements of $\{\checkmark\} + \mathsf{Out}$ are unchanged by this operation. A pictorial representation of the effect on the derivatives of $\mathsf{e}\,;\mathsf{f}$ is given above on the right. Here, we assume that $\partial(\mathsf{e})_\alpha$ can perform a p-transition to e' with probability s; we make the same assumption in the informal descriptions of derivatives for loops, below.

For guarded loops, we consider three cases when defining $\partial\left(\mathsf{e}^{(\mathsf{b})}\right)_\alpha$. If $\alpha \leq_{\mathsf{BA}} \bar{\mathsf{b}}$, then the current state does not satisfy the loop guard and can be skipped: $\partial\left(\mathsf{e}^{(\mathsf{b})}\right)_\alpha = \delta_\checkmark$. If $\alpha \leq_{\mathsf{BA}} \mathsf{b}$ and $\partial(\mathsf{e})_\alpha(\checkmark) = 1$, then the loop body is called, but the inner program e does not perform actions. We identify divergent loops with rejection and so in this case we set $\partial\left(\mathsf{e}^{(\mathsf{b})}\right)_\alpha = \delta_\checkmark$. If $\alpha \leq_{\mathsf{BA}} \mathsf{b}$ and $\partial(\mathsf{e})_\alpha(\checkmark) < 1$, the program executes the loop body and starts again, having to redistribute the probability mass of immediate acceptance $\partial(\mathsf{e})_\alpha(\checkmark)$ through each execution. So, for $\alpha \leq_{\mathsf{BA}} \mathsf{b}$ and $\partial(\mathsf{e})_\alpha(\checkmark) < 1$, the definition of $\partial(\mathsf{e}^{(\mathsf{b})})_\alpha$ is given below on the left: it rejects or returns when e does, and transitions to $\mathsf{e}'\,;\mathsf{e}^{(\mathsf{b})}$ when e transitions to e'.

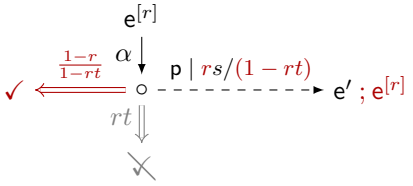
$$\partial(\mathsf{e}^{(\mathsf{b})})_\alpha(x) = \begin{cases} \frac{\partial(\mathsf{e})_\alpha(x)}{1-\partial(\mathsf{e})_\alpha(\checkmark)} & x \in \{\checkmark\} \cup \mathsf{Out} \\ \frac{\partial(\mathsf{e})_\alpha(\mathsf{p},\mathsf{e}')}{1-\partial(\mathsf{e})_\alpha(\checkmark)} & x = \left(\mathsf{p}, \left(\mathsf{e}'\,;\mathsf{e}^{(\mathsf{b})}\right)\right) \\ 0 & \text{otherwise} \end{cases}$$



The reweighing of probabilities used in the definition of the loops comes from defining loops as least fixpoints w.r.t. to an order on distributions, similarly to Stark and Smolka [62].

Finally, we specify the behaviour of the probabilistic loop. In the special case where $\partial(\mathsf{e})_\alpha(\checkmark) = 1$ and $r = 1$, the loop will not terminate; hence we set $\partial\left(\mathsf{e}^{[r]}\right)_\alpha = \delta_\checkmark$. In all other cases, we look at $\partial(\mathsf{e})_\alpha$ to build $\partial(\mathsf{e}^{[r]})_\alpha$ for each $\alpha \in \mathsf{At}$. First, we make sure that the loop may be skipped with probability $1 - r$. Next, we account for the possibility that e may reject or return a value, and we modify the productive branches by adding $\mathsf{e}^{[r]}$ to be executed next, as was done for the guarded loop. The remaining mass is $r\partial(e)_\alpha(\checkmark)$, the probability that we will enter the loop with an atom that can skip over the loop body. As was the case for the guarded loop, we discard this possibility and redistribute it among the remaining branches. The resulting definition of $\partial(\mathsf{e}^{[r]})_\alpha$ is given below on the left.

$$\partial\left(\mathsf{e}^{[r]}\right)_\alpha(x) = \begin{cases} \frac{1-r}{1-r\partial(\mathsf{e})_\alpha(\checkmark)} & x = \checkmark \\ \frac{r\partial(\mathsf{e})_\alpha(x)}{1-r\partial(\mathsf{e})_\alpha(\checkmark)} & x \in \{\checkmark\} \cup \mathsf{Out} \\ \frac{r\partial(\mathsf{e})_\alpha(\mathsf{p},\mathsf{e}')}{1-r\partial(\mathsf{e})_\alpha(\checkmark)} & x = \left(\mathsf{p}, \left(\mathsf{e}'\,;\mathsf{e}^{[r]}\right)\right) \\ 0 & \text{otherwise} \end{cases}$$



As before, we provide an informal visual depiction of the probabilistic loop semantics above on the right, using the same conventions.

Reachable states. For any ProbGKAT automaton (X, β) and any $x \in X$, we denote by $\langle x \rangle_\beta$ the set of states reachable from x via β. Clearly, $(\langle x \rangle_\beta, \beta)$ is a ProbGKAT automaton and is the smallest subautomaton of (X, β) containing x. The canonical inclusion map

$(\langle x\rangle_\beta, \beta) \to (X, \beta)$ is a ProbGKAT automaton homomorphism. In particular, $(\langle \mathsf{e}\rangle_\partial, \partial)$ is the smallest subautomaton of $(\mathtt{Exp}, \partial)$ containing e. We will refer to this subautomaton as the small-step semantics of e. We will often abuse notation and write $\langle \mathsf{e}\rangle_\partial$ for $(\langle \mathsf{e}\rangle_\partial, \partial)$.

The following lemma says that every ProbGKAT expression generates a finite automaton.

▶ **Lemma 7.** *For all* $\mathsf{e} \in \mathtt{Exp}$, $\langle \mathsf{e}\rangle_\partial$ *is finite. In fact, the number of states is bounded above by* $\#(\mathsf{e}) : \mathtt{Exp} \to \mathbb{N}$, *where* $\#(-)$ *is defined recursively by*

$$\#(\mathsf{b}) = 1 \quad \#(v) = 1 \quad \#(\mathsf{p}) = 2 \quad \#(\mathsf{e} +_\mathsf{b} \mathsf{f}) = \#(\mathsf{e}) + \#(\mathsf{f}) \quad \#(\mathsf{e} \oplus_r \mathsf{f}) = \#(\mathsf{e}) + \#(\mathsf{f})$$

$$\#(\mathsf{e}\,;\mathsf{f}) = \#(\mathsf{e}) + \#(\mathsf{f}) \quad \#\left(\mathsf{e}^{(\mathsf{b})}\right) = \#(\mathsf{e}) \quad \#\left(\mathsf{e}^{[r]}\right) = \#(\mathsf{e})$$

4 Bisimulations and their properties

Verifying that a given relation is a bisimulation (Definition 5) requires that we construct a suitable transition structure on the relation. In this section, we give necessary and sufficient conditions for the existence of such a transition structure. We also study properties of the bisimilarity relation $\sim$, the largest bisimulation [50].

Concrete characterisation of bisimulation equivalence. There is a beautiful characterisation of bisimulations between Markov chains in [30], whose proof makes use of the max-flow min-cut theorem. Adapting this work to ProbGKAT automata produces a useful characterisation of *bisimulation equivalences*, bisimulations that are also equivalence relations.

▶ **Lemma 8.** *Let* (X, β) *be a* ProbGKAT *automaton and let* $R \subseteq X \times X$ *be an equivalence relation.* R *is a bisimulation if and only if and only if for all* $(x, y) \in R$ *and* $\alpha \in \mathsf{At}$,

1. *for all* $o \in 2 + \mathsf{Out}$, $\beta(x)_\alpha(o) = \beta(y)_\alpha(o)$, *and*
2. *for all equivalence classes* $Q \in X/R$ *and all* $\mathsf{p} \in \mathsf{Act}$, $\beta(x)_\alpha[\{p\} \times Q] = \beta(y)_\alpha[\{p\} \times Q]$

This lemma can be seen as an extension of Larsen-Skou bisimilarity [40] to systems with outputs. Intuitively, R is a bisimulation equivalence if for any atom $\alpha \in \mathsf{At}$ and $(x, y) \in R$, the transitions assign the same probabilities to any output, and the probability of transitioning into any given equivalence class after emitting p is the same for both x and y.

Bisimilarity and its properties. Given a relation $R \subseteq X \times Y$, define $R^{-1} = \{(y, x) \mid x\ R\ y\}$, and given $A \subseteq X$, write $R(A) = \{y \in Y \mid x\ R\ y, x \in A\}$. The bisimilarity relation $\sim_{\beta,\gamma} \subseteq X \times Y$ between (X, β) and (Y, γ) is the *greatest fixpoint* of the following operator.

▶ **Definition 9.** *Let* (X, β) *and* (Y, γ) *be* ProbGKAT *automata and let* $R \subseteq X \times Y$. *We define the operator* $\Phi_{\beta,\gamma} : 2^{X\times Y} \to 2^{X\times Y}$ *so that* $(x, y) \in \Phi_{\beta,\gamma}(R)$ *if for any given* $\alpha \in \mathsf{At}$,

- *for all* $o \in 2 + \mathsf{Out}$, $\beta(x)_\alpha(o) = \gamma(y)_\alpha(o)$,
- *for all* $A \subseteq X$ *and all* $\mathsf{p} \in \mathsf{Act}$, $\beta(x)_\alpha[\{\mathsf{p}\} \times A] \leq \gamma(y)_\alpha[\{\mathsf{p}\} \times R(A)]$, *and*
- *for all* $B \subseteq Y$ *and* $\mathsf{p} \in \mathsf{Act}$, $\gamma(y)_\alpha[\{\mathsf{p}\} \times B] \leq \beta(x)_\alpha[\{\mathsf{p}\} \times R^{-1}(B)]$.

From now on, we will omit the subscripts from Φ *when the automata are clear from context.*

The operator $\Phi_{\beta,\gamma}$ can also be used to define a behavioural pseudometric. Let (X, β) be a ProbGKAT automaton. A *relation refinement chain* is an indexed family $\{\sim^{(i)}\}_{i\in\mathbb{N}}$ of relations on X defined as: $\sim^{(0)} = X \times X$, $\sim^{(i+1)} = \Phi(\sim^{(i)})$. We can intuitively think of successive elements of this chain as closer approximations of bisimilarity (see also [26]).

▶ **Theorem 10.** *Let* (X, β) *be a* ProbGKAT *automaton. For any* $x, y \in X$, $x \sim y$ *if and only if for all* $i \in \mathbb{N}$, *we have* $x \sim^{(i)} y$.

Table 1 Axiomatisation of ProbGKAT. In the figure $\mathsf{e}, \mathsf{f}, \mathsf{g} \in \mathtt{Exp}$, $\mathsf{b}, \mathsf{c} \in \mathtt{BExp}$, $v \in \mathsf{Out}$, $\mathsf{p} \in \mathsf{Act}$ and $r, s \in [0,1]$. Laws involving division of probabilities apply when the denominator is not zero. To simplify the notation, we write $\mathsf{E}(\mathsf{e}) = 0$ to denote that for all $\alpha \in \mathsf{At}$ it holds that $\mathsf{E}(\mathsf{e})_\alpha = 0$.

Guarded Choice Axioms

(G1) $\mathsf{e} +_\mathsf{b} \mathsf{e} \equiv \mathsf{e}$

(G2) $\mathsf{e} +_\mathsf{b} \mathsf{f} \equiv \mathsf{b} ; \mathsf{e} +_\mathsf{b} \mathsf{f}$

(G3) $\mathsf{e} +_\mathsf{b} \mathsf{f} \equiv \mathsf{f} +_{\bar{\mathsf{b}}} \mathsf{e}$

(G4) $(\mathsf{e} +_\mathsf{b} \mathsf{f}) +_\mathsf{c} \mathsf{g} \equiv \mathsf{e} +_{\mathsf{bc}} (\mathsf{f} +_\mathsf{c} \mathsf{g})$

Distributivity Axiom

(D) $\mathsf{e} \oplus_r (\mathsf{f} +_\mathsf{b} \mathsf{g}) \equiv (\mathsf{e} \oplus_r \mathsf{f}) +_\mathsf{b} (\mathsf{e} \oplus_r \mathsf{g})$

Sequencing Axioms

(S1) $1 ; \mathsf{e} \equiv \mathsf{e}$

(S2) $\mathsf{e} ; 1 \equiv \mathsf{e}$

(S3) $(\mathsf{e} ; \mathsf{f}) ; \mathsf{g} \equiv \mathsf{e} ; (\mathsf{f} ; \mathsf{g})$

(S4) $0 ; \mathsf{e} \equiv 0$

(S5) $(\mathsf{e} +_\mathsf{b} \mathsf{f}) ; \mathsf{g} \equiv \mathsf{e} ; \mathsf{g} +_\mathsf{b} \mathsf{f} ; \mathsf{g}$

(S6) $(\mathsf{e} \oplus_r \mathsf{f}) ; \mathsf{g} \equiv \mathsf{e} ; \mathsf{g} \oplus_r \mathsf{f} ; \mathsf{g}$

(S7) $v ; \mathsf{e} \equiv v$

(S8) $\mathsf{b} ; \mathsf{c} \equiv \mathsf{bc}$

Loop Axioms

(L1) $\mathsf{e}^{(\mathsf{b})} \equiv \mathsf{e} ; \mathsf{e}^{(\mathsf{b})} +_\mathsf{b} 1$

(L2) $\mathsf{e}^{[r]} \equiv \mathsf{e} ; \mathsf{e}^{[r]} \oplus_r 1$

(L3) $(\mathsf{e} +_\mathsf{c} 1)^{(\mathsf{b})} \equiv (\mathsf{c} ; \mathsf{e})^{(\mathsf{b})}$

(L4) $\mathsf{e}^{(1)} \equiv \mathsf{e}^{[1]}$

(L5) $$\frac{\mathsf{e} \equiv (\mathsf{f} \oplus_s 1) +_\mathsf{c} \mathsf{g} \qquad s > 0}{\mathsf{c} ; \mathsf{e}^{(\mathsf{b})} \equiv \mathsf{c} ; \left(\mathsf{f} ; \mathsf{e}^{(\mathsf{b})} +_\mathsf{b} 1\right)}$$

(L6) $$\frac{\mathsf{e} \equiv (\mathsf{f} \oplus_s 1) +_\mathsf{c} \mathsf{g}}{\mathsf{c} ; \mathsf{e}^{[r]} \equiv \mathsf{c} ; \left(\mathsf{f} ; \mathsf{e}^{[r]} \oplus_{\frac{rs}{1-r(1-s)}} 1\right)}$$

Probabilistic Choice Axioms

(P1) $\mathsf{e} \oplus_r \mathsf{e} \equiv \mathsf{e}$

(P2) $\mathsf{e} \oplus_1 \mathsf{f} \equiv \mathsf{e}$

(P3) $\mathsf{e} \oplus_r \mathsf{f} \equiv \mathsf{f} \oplus_{1-r} \mathsf{e}$

(P4) $(\mathsf{e} \oplus_r \mathsf{f}) \oplus_s \mathsf{g} \equiv \mathsf{e} \oplus_{rs} (\mathsf{f} \oplus_{\frac{(1-r)s}{1-rs}} \mathsf{g})$

Fixpoint Rules

(F1) $$\frac{\mathsf{g} \equiv \mathsf{e} ; \mathsf{g} +_\mathsf{b} \mathsf{f} \qquad \mathsf{E}(\mathsf{e}) = 0}{\mathsf{g} \equiv \mathsf{e}^{(\mathsf{b})} ; \mathsf{f}}$$

(F2) $$\frac{\mathsf{g} \equiv \mathsf{e} ; \mathsf{g} \oplus_r \mathsf{f} \qquad \mathsf{E}(\mathsf{e}) = 0}{\mathsf{g} \equiv \mathsf{e}^{[r]} ; \mathsf{f}}$$

Define $\mathsf{E} : \mathtt{Exp} \to \mathsf{At} \to [0,1]$ inductively by

$$\mathsf{E}(\mathsf{p})_\alpha = \mathsf{E}(v)_\alpha = 0$$

$$\mathsf{E}(\mathsf{b})_\alpha = \begin{cases} 1 & \alpha \leq_{\mathsf{BA}} \mathsf{b} \\ 0 & \alpha \leq_{\mathsf{BA}} \bar{\mathsf{b}} \end{cases}$$

$$\mathsf{E}(\mathsf{e} +_\mathsf{b} \mathsf{f})_\alpha = \begin{cases} \mathsf{E}(\mathsf{e})_\alpha & \alpha \leq_{\mathsf{BA}} \mathsf{b} \\ \mathsf{E}(\mathsf{f})_\alpha & \alpha \leq_{\mathsf{BA}} \bar{\mathsf{b}} \end{cases}$$

$$\mathsf{E}(\mathsf{e} \oplus_r \mathsf{f})_\alpha = r\mathsf{E}(\mathsf{e})_\alpha + (1-r)\mathsf{E}(\mathsf{f})_\alpha$$

$$\mathsf{E}(\mathsf{e} ; \mathsf{f})_\alpha = \mathsf{E}(\mathsf{e})_\alpha \, \mathsf{E}(\mathsf{f})_\alpha$$

$$\mathsf{E}\left(\mathsf{e}^{(\mathsf{b})}\right)_\alpha = \mathsf{E}\left(\bar{\mathsf{b}}\right)_\alpha$$

$$\mathsf{E}\left(\mathsf{e}^{[r]}\right)_\alpha = \begin{cases} 0 & r = 1 \text{ and } \mathsf{E}(\mathsf{e})_\alpha = 1 \\ \frac{1-r}{1-r\mathsf{E}(\mathsf{e})_\alpha} & \text{otherwise} \end{cases}$$

Thus, if $x, y \in X$ are not bisimilar, then there exists a maximal $i \in \mathbb{N}$ such that $x \sim^{(i)} y$. In Section 6, we use this to define a pseudometric on the states of any ProbGKAT automaton. Informally speaking, this allows us to quantify *how close* to being bisimilar two states are.

Our main goal is to axiomatise bisimilarity of ProbGKAT expressions with a set of equational laws and reason about equivalence using *equational logic*. For such an axiomatisation to exist, bisimilarity needs to be both an equivalence relation and a congruence with respect to the ProbGKAT operations. The greatest bisimulation on any ProbGKAT automaton is an equivalence [50], but being congruence requires an inductive argument.

▶ **Theorem 11.** *The greatest bisimulation on* $(\mathtt{Exp}, \partial)$ *is a congruence with respect to* ProbGKAT *operations.*

5 Axiomatisation

We turn our attention to *axiomatisation* of bisimilarity of ProbGKAT expressions, using an axiom system based on GKAT modulo bisimilarity [53]. First, we give an overview of the axioms, and establish their soundness. Finally, we show that our axioms are strong enough to decompose every expression into a certain syntactic normal form relating the expressions to their small-step semantics. Completeness is tackled in the next section.

Overview of the axioms. Table 1 contains the axioms, which are either *equational* (of the form $\mathsf{e} \equiv \mathsf{f}$), or *quasi-equational* (of the form $\mathsf{e}_1 \equiv \mathsf{f}_1, \dots \mathsf{e}_n \equiv \mathsf{f}_n \implies \mathsf{e} \equiv \mathsf{f}$). It also holds the definition of the function $\mathsf{E}(-)$, which is necessary to give a side condition to the fixpoint rules. We define $\equiv \subseteq \mathtt{Exp} \times \mathtt{Exp}$ as the smallest congruence relation satisfying the axioms.

Axioms G1–G4 are inherited from GKAT and govern the behaviour of Boolean guarded choice. P1–P4 can be thought of as their analogues, but for the probabilistic choice. The distributivity axiom D states that guarded choice distributes over a probabilistic choice, which reflects the way our operational model resolves both types of branching.

The sequencing axioms S1–S8 are mostly inherited from GKAT. The new axioms include S6 which talks about right distributivity of sequencing over probabilistic choice and S7 which captures the intuitive property that any code executed after a `return` statement is not executed. L1 and L3 come from GKAT, while L2 is a probabilistic loop analogue of L1, which captures the semantics of the probabilistic loop in terms of recursive unrolling. L4 equates the `while(true)` and `while(flip(1))` loops. F1 and F2 are inspired by Salomaa's axioms [52] and provide a partial converse to L1 and L2 respectively, given the loop body cannot immediately terminate. The property that a loop body has a zero probability of outputting ✓ is formally writen using the side condition $\mathsf{E}(\mathsf{e}) = 0$, which can be thought of as *empty word property* from Salomaa's axiomatisation [52].

This leaves us with L5 and L6, which describe the behaviour of guarded and probabilistic loops where parts of the loop body may be skipped. These are quasi-equational, but can be replaced by equivalent equations – see [51, Remark 49]. L5 concerns a loop on an expression e that has probability $1 - s$ of not performing any action, given that c holds. The rule says that, if we start the loop on e given that c holds, then either b holds and we execute f, or it does not, and the loop is skipped. The reason that we can disregard the 1 part of e is that if this branch is taken, then c still holds on the next iteration of the loop, and so the program will have to choose probabilistically between f and 1 once more. Since $s > 0$, it will eventually choose the probabilistic branch f with almost sure probability.

The second rule, L6, is the analogue of L5 for probabilistic choice. In this case, however, a choice for 1 also means another probabilistic experiment to determine whether the loop needs to be executed once more, with probability r. The consequence is that if the loop on e is started given that c holds, some more probability mass will shift towards skipping, as a result executing 1 some number of times before halting the loop.

Soundness with respect to bisimilarity. Using the characterisation from Section 4, we can show that $\equiv$ is a bisimulation equivalence on $(\mathtt{Exp}, \partial)$. The proof is available in the full version of the paper [51, Appendix D].

▶ **Lemma 12.** *$\equiv$ is a bisimulation equivalence on $(\mathtt{Exp}, \partial)$*

We immediately obtain that provable equivalence is contained in bisimilarity.

▶ **Theorem 13** (Soundness). *For all $\mathsf{e}, \mathsf{f} \in \mathtt{Exp}$, if $\mathsf{e} \equiv \mathsf{f}$ then $\mathsf{e} \sim \mathsf{f}$*

Example of equational reasoning. Since our axioms are sound, we can reason about ProbGKAT expressions equationally, without constructing bisimulations by hand. Once again, we revisit the algorithm from Figure 1. To show correctness, we need to prove the equivalence of expressions e and $\mathsf{g}^{(1)}$ from Example 1, as follows:

$$
\begin{aligned}
\mathsf{g}^{(1)} &\equiv \Big(\Big(\boxdot \oplus_{\frac{1}{2}} \boxdot\Big) \oplus_{\frac{1}{2}} \Big(\boxdot \oplus_{\frac{1}{2}} 1\Big)\Big)^{(1)} && \text{(Def. of g)}\\
&\equiv \Big(\Big(\Big(\boxdot \oplus_{\frac{1}{2}} \boxdot\Big) \oplus_{\frac{2}{3}} \boxdot\Big) \oplus_{\frac{1}{2}} 1\Big)^{(1)} && \text{(See below)}\\
&\equiv \Big(\Big(\Big(\Big(\boxdot \oplus_{\frac{1}{2}} \boxdot\Big) \oplus_{\frac{2}{3}} \boxdot\Big) \oplus_{\frac{1}{2}} 1\Big) +_1 0\Big)^{(1)} && \text{(See below)}\\
&\equiv \Big(\Big(\boxdot \oplus_{\frac{1}{2}} \boxdot\Big) \oplus_{\frac{2}{3}} \boxdot\Big)^{(1)} && \text{(Axioms L5 and S1)}\\
&\equiv \Big(\boxdot \oplus_{\frac{1}{3}} \Big(\boxdot \oplus_{\frac{1}{2}} \boxdot\Big)\Big)^{(1)} && \text{(Axiom P4)}\\
&\equiv \Big(\boxdot \oplus_{\frac{1}{3}} \Big(\boxdot \oplus_{\frac{1}{2}} \boxdot\Big)\Big) ; \Big(\boxdot \oplus_{\frac{1}{3}} \Big(\boxdot \oplus_{\frac{1}{2}} \boxdot\Big)\Big)^{(1)} +_1 1 && \text{(Axiom L1)}\\
&\equiv \Big(\boxdot \oplus_{\frac{1}{3}} \Big(\boxdot \oplus_{\frac{1}{2}} \boxdot\Big)\Big) ; \mathsf{e}^{(1)} +_1 1 && \text{(Def. e)}\\
&\equiv \Big(\boxdot \oplus_{\frac{1}{3}} \Big(\boxdot \oplus_{\frac{1}{2}} \boxdot\Big)\Big) ; \mathsf{e}^{(1)} && \text{(See below)}\\
&\equiv \Big(\boxdot ; e^{(1)} \oplus_{\frac{1}{3}} \Big(\boxdot ; e^{(1)} \oplus_{\frac{1}{2}} \boxdot ; e^{(1)}\Big)\Big) && \text{(Axiom S6)}\\
&\equiv \Big(\boxdot \oplus_{\frac{1}{3}} \Big(\boxdot \oplus_{\frac{1}{2}} \boxdot\Big)\Big) && \text{(Axiom S7)}\\
&\equiv \mathsf{e} && \text{(Def. e)}
\end{aligned}
$$

In the second step, we used that for all $\mathsf{e}_1, \mathsf{e}_2, \mathsf{e}_3 \in \mathtt{Exp}$ and $r, s \in [0,1]$ with $(1-r)(1-s) > 0$, we have $\mathsf{e}_1 \oplus_r (\mathsf{e}_2 \oplus_s \mathsf{e}_3) \equiv (\mathsf{e}_1 \oplus_k \mathsf{e}_2) \oplus_l \mathsf{e}_3$, where $k = \frac{r}{1-(1-r)(1-s)}$ and $l = 1-(1-r)(1-s)$. In the third and eighth steps, we used that for all $\mathsf{e}, \mathsf{f} \in \mathtt{Exp}$, we have that $\mathsf{e} +_1 \mathsf{f} \equiv \mathsf{e}$. Both those equivalences follow from the other axioms; see [51, Lemma 50] in the full version of the paper for details.

Fundamental theorem. Every expression in the language of KA (resp. KAT, GKAT) can be reconstructed from its small-step semantics, up to $\equiv$. This property, often referred to as the *fundamental theorem* of (in analogy with the fundamental theorem of calculus and following the terminology of Rutten [50]) is useful in many contexts, and we will need it later on.

▶ **Theorem 14** (Fundamental Theorem). *For every* $\mathsf{e} \in \mathtt{Exp}$ *it holds that*

$$\mathsf{e} \equiv \bigdotplus_{\alpha \in \mathsf{At}} \left(\bigoplus_{d \in \mathtt{supp}(\partial(\mathsf{e})_\alpha)} \partial(\mathsf{e})_\alpha(d) \cdot \mathtt{exp}(d) \right)$$

where $\mathtt{exp}$ *defines a function* $2 + \mathsf{Out} + \mathsf{Act} \times \mathtt{Exp} \to \mathtt{Exp}$ *given by*

$$\mathtt{exp}(\times) = 0 \quad \mathtt{exp}(\checkmark) = 1 \quad \mathtt{exp}(v) = v \quad \mathtt{exp}(\mathsf{a}, \mathsf{f}) = \mathsf{a};\mathsf{f} \qquad (v \in \mathsf{Out}, \mathsf{a} \in \mathsf{Act} \text{ and } \mathsf{f} \in \mathtt{Exp})$$

The proof is given in the appendix. We use a generalised type of guarded and probabilistic choice ranging over indexed collections of expressions, which is defined in [51, Appendix D.2].

6 Completeness

Given the axioms presented in the previous section, a natural question is to ask whether they are *complete* w.r.t. bisimilarity – i.e., whether any bisimilar pair can be proved equivalent using the axioms that make up $\equiv$. The traditional strategy is to develop the idea of *systems of equations* within the calculus, and show that these systems have unique (least) solutions up to provable equivalence. If we can characterise the expressions of interest as solutions to a common system of equations (typically derived from the bisimulation that relates them), then uniqueness of solutions guarantees their equivalence. Unfortunately, the first step of this process, where systems of equations are shown to have unique solutions, does not transfer to ProbGKAT (nor GKAT). Indeed, some systems of equations do not have *any* solution [39, 53]; the lack of a procedure to construct solutions also encumbers a proof of uniqueness.

Instead, we follow the approach from [58] pioneered by Bergstra and Klop [10], and incorporate uniqueness of solutions into the axiomatisation. The *uniqueness axiom* (UA) that accomplishes this is an *axiom scheme*, which is to say it is a template for infinitely many axioms, one for each number of unknowns. In the case of a single unknown, one can show that F1 and F2 are special cases of UA, which moreover give a candidate solution.

With UA in hand, the traditional roadmap towards completeness works out. Before we get there, however, we must expend some energy to properly state this axiom scheme. Moreover, showing soundness of UA requires effort. Both of these take up the bulk of the development in this section; we derive the desired completeness property at the end.

(Salomaa) systems of equations. First, we define formally the idea of *systems of equations* for ProbGKAT automata. The constraints on each variable will be built using the following two-sorted grammar, where X is a finite set of indeterminates.

$$e_1, e_2 \in \mathtt{Exp}(X) ::= p \mid e_1 +_{\mathsf{b}} e_2 \qquad (p \in \mathtt{PExp}(X), \mathsf{b} \in \mathtt{BExp})$$
$$p_1, p_2 \in \mathtt{PExp}(X) ::= \mathsf{f} \mid \mathsf{g}x \mid p_1 \oplus_r p_2 \qquad (\mathsf{f}, \mathsf{g} \in \mathtt{Exp}, x \in X, r \in [0,1])$$

▶ **Definition 15.** *A* system of equations *is a pair* $(X, \tau : X \to \mathtt{Exp}(X))$ *consisting of a finite set* X *of indeterminates and a function* $\tau : X \to \mathtt{Exp}(X)$*. If for all* $x \in X$*, in each of* $\tau(x)$ *all subterms of the form* $\mathsf{g}x$ *satisfy* $\mathsf{E}(\mathsf{g}) = 0$*, then such system is called* Salomaa.[1]

Every finite state ProbGKAT automaton yields a Salomaa system of equations.

▶ **Definition 16.** *Let* (X, β) *be a finite state* ProbGKAT *automaton. A system of equations associated with* (X, β) *is a Salomaa system* (X, τ)*, with* $\tau : X \to \mathtt{Exp}(X)$ *defined by*

$$\tau(x) = \sum_{\alpha \in \mathsf{At}} \left(\bigoplus_{d \in \mathtt{supp}(\beta(x))_\alpha} \beta(x)_\alpha(d) \cdot \mathtt{sys}(d) \right)$$

where $\mathtt{sys} : 2 + \mathsf{Out} + \mathsf{Act} \times \mathtt{Exp} \to \mathtt{PExp}(X)$ *is given by*

$$\mathtt{sys}(\text{✗}) = 0 \quad \mathtt{sys}(\text{✓}) = 1 \quad \mathtt{sys}(v) = v \quad \mathtt{sys}(\mathsf{a}, x) = \mathsf{a}x$$

[1] In process algebra [46], Salomaa systems are usually called *guarded*. We avoid the latter name to prevent confusion with Boolean guarded choice present in (Prob)GKAT.

▶ **Example 17.** In the system associated with the automaton from Example 3, τ is given by

$$x_1 \mapsto (\mathsf{q}x_2 \oplus_{\frac{1}{2}} v) +_\alpha \Big((\mathsf{p}x_1 \oplus_{\frac{1}{2}} \mathsf{q}x_2) +_\beta 0\Big) \qquad x_2 \mapsto 1 +_\alpha (1 +_\beta 0)$$

Given a function $h : X \to \mathtt{Exp}$ that assigns a value to each indeterminate in X, we can derive a ProbGKAT expression $h^\#(e)$ for each $e \in \mathtt{Exp}(X)$ inductively, as follows: $h^\#(\mathsf{f}) = \mathsf{f}$, $h^\#(p_1 \oplus_r p_2) = h^\#(p_1) \oplus_r h^\#(p_2)$, $h^\#(\mathsf{g}x) = \mathsf{g}\,;h(x)$, $h^\#(e_1 +_\mathsf{b} e_2) = h^\#(e_1) +_\mathsf{b} h^\#(e_2)$. We can now state the notion of a *solution* to the Salomaa system. Rather than expecting both sides of equations to be strictly equal, we require them to be related by a relation, which we leave as a parameter to instantiate later.

▶ **Definition 18.** *Let $R \subseteq \mathtt{Exp} \times \mathtt{Exp}$. A* solution *up to R to a system (X, τ) is a map $h : X \to \mathtt{Exp}$ satisfying for all $x \in X$ that $\big(h(x), h^\#(\tau(x))\big) \in R$.*

▶ **Example 19.** A solution up to $\equiv$ to the system from Example 17 would satisfy

$$h(x_1) \equiv (\mathsf{q}\,;h(x_2) \oplus_{\frac{1}{2}} v) +_\alpha \Big((\mathsf{p}\,;h(x_1) \oplus_{\frac{1}{2}} \mathsf{q}\,;h(x_2)) +_\beta 0\Big) \qquad h(x_2) \equiv 1 +_\alpha (1 +_\beta 0)$$

In this case, choosing $h(x_1) = (\mathsf{p} +_\beta v)^{[\frac{1}{2}]}\,;\mathsf{q}$ and $h(x_2) = 1$ fits these constraints.

▶ **Example 20.** Let $\mathsf{r} \in [0, 1]$. The recursive specification on the left below describes a program $\mathsf{randAdd}(\mathsf{m}, \mathsf{r})$ which takes an integer m and bias r. As long as m is strictly below 10, this program flips an r-biased coin to decide between incrementing m followed by a recursive call or termination. That recursive specification can be thought of as a Salomaa system with one unknown; the program on the right is a solution up to $\equiv$.

```
def randAdd(m, r):
    if m < 10 then
        if flip(r) then
            m++;
            randAdd(m, r)
        else
            skip
    else
        skip
```

```
while flip(r) do
    if m < 10 then
        m++;
        randAdd(m, r)
    else
        skip
```

Solutions up to $\equiv$ can be characterised concretely, using Theorem 14.

▶ **Theorem 21.** *Let (X, β) be a finite state* ProbGKAT *automaton. The map $h : X \to \mathtt{Exp}$ is a solution up to $\equiv$ to the system associated with (X, β) if and only if $[-]_\equiv \circ h$ is a* ProbGKAT *automata homomorphism from (X, β) to $(\mathtt{Exp}/{\equiv}, \bar{\partial})$. We write $\bar{\partial}$ to denote the unique transition function on $\mathtt{Exp}/{\equiv}$ which makes the quotient map $[-]_\equiv : \mathtt{Exp} \to \mathtt{Exp}/{\equiv}$ a* ProbGKAT *automaton homomorphism from $(\mathtt{Exp}, \partial)$ [50, Proposition 5.8].*

Uniqueness of Solutions axiom. Informally, UA extends $\equiv$ by stating that solutions to Salomaa systems, if they exist, are unique. Formally, we define $\doteq \subseteq \mathtt{Exp} \times \mathtt{Exp}$ to be the least congruence that contains $\equiv$, and satisfies the following (quasi-equational) axiom:

$$\frac{(X, \tau) \text{ is a Salomaa system} \quad f, g : X \to \mathtt{Exp} \text{ are solutions of } (X, \tau) \text{ up to } \doteq}{f(x) \doteq g(x) \text{ for all } x \in X} \tag{UA}$$

F1 and F2 are instantiations of UA for Salomaa systems with one variable.

Behavioural pseudometric. We now develop the theory necessary to verify soundness of UA. First, note that for every ProbGKAT, we can define a function $d_X : X \times X \to \mathbb{R}^+$:

$$d_X(x,y) = \begin{cases} 2^{-n} & n \text{ is maximal such that } x \sim^{(n)} y \\ 0 & \text{if } x \sim y \end{cases}$$

The above is well-defined by Theorem 10, and is a *pseudometric*, in the following sense.

▶ **Definition 22.** *A* pseudometric space *is a pair* (X, d_X)*, where* $d_X : X \times X \to \mathbb{R}^+$ *is a* pseudometric*, which means that for all* $x, y, z \in X$ *we have*

$$d_X(x,x) = 0 \qquad d_X(x,y) = d_X(y,x) \qquad d_X(x,z) \leq d_X(x,y) + d_X(y,z)$$

Let $k \in \mathbb{R}^+$*. A mapping* $f : X \to Y$ *between pseudometric spaces* (X, d_X) *and* (Y, d_Y) *is called* k-Lipschitz *if for all* $x, y \in X$ $d_Y(f(x), f(y)) \leq k d_X(x,y)$*.*

The behavioural pseudometric satisfies the definition above, in a strong sense.

▶ **Lemma 23.** *For every* ProbGKAT *automaton,* (X, β)*,* (X, d_X) *a pseudometric space that is* ultra*, in the sense that for all* $x, y, z \in X$ *we have* $d_X(x,z) = \max\{d_X(x,y), d_X(y,z)\}$*.*

Let (X, d_X) and (Y, d_Y) be pseudometric spaces. Their *product* is a pseudometric space $(X \times Y, d_{X\times Y})$ where $d_{X\times Y}$ is defined by $d_{X\times Y}((x,y),(x',y')) = \max\{d_X(x,x'), d_Y(y,y')\}$. It is easy to show that if both pseudometric spaces are ultra, then so is their product. Going forward, we will omit subscripts when they are clear from context.

Soundness of the Uniqueness Axiom. Every Salomaa system $(X, \tau : X \to \mathtt{Exp}(X))$ with $X = \{x_1, x_2, \ldots, x_n\}$ induces a mapping $\bar{\tau} : \mathtt{Exp}^n \to \mathtt{Exp}^n$. Intuitively, this mapping takes a vector $\vec{\mathsf{e}} = (\mathsf{e}_1, \ldots, \mathsf{e}_n)$, and produces a new vector where the i-th component is the evaluation of $\tau(x_i)$ when each x_j is substituted by e_j. More formally, given this $\vec{\mathsf{e}}$, we define $e : X \to \mathtt{Exp}$ by $e(x_i) = \mathsf{e}_i$, and set $\bar{\tau}(\vec{\mathsf{e}}) = \big((e^\# \circ \tau)(x_1), \ldots, (e^\# \circ \tau)(x_n)\big)$.

To establish soundness of UA, we first show that $\bar{\tau}$ is $\frac{1}{2}$-Lipschitz on the pseudometric space $(\mathtt{Exp}^n, d)$, where d is the metric that arises from the n-fold product of $(\mathtt{Exp}, \partial)$.

▶ **Lemma 24.** *Given a Salomaa system* $(X, \tau : X \to \mathtt{Exp}(X))$*, the map* $\bar{\tau} : \mathtt{Exp}^n \to \mathtt{Exp}^n$ *from the pseudometric space* $(\mathtt{Exp}^n, d)$ *to itself is* $\frac{1}{2}$*-Lipschitz.*

Finally, we can prove the following.

▶ **Lemma 25.** UA *is satisfied by bisimilarity.*

Proof. Let (X, τ) is a Salomaa system, with $X = \{x_1, \ldots, x_n\}$, and let $f, g : X \to \mathtt{Exp}(X)$ be solutions up to $\doteqdot$ to the system. Finally, let $\vec{\mathsf{f}} = (f(x_1), \ldots, f(x_n))$ and $\vec{\mathsf{g}} = (g(x_1), \ldots, g(x_n))$. Assume that the premises are satisfied by the bisimilarity, $f(x_i) \sim (f^\# \circ \tau)(x_i)$ and $g(x_i) \sim (g^\# \circ \tau)(x_i)$ for all $1 \leq i \leq n$. In other words, we have that $d(\bar{\tau}(\vec{\mathsf{f}}), \vec{\mathsf{f}}) = 0$ and $d(\bar{\tau}(\vec{\mathsf{g}}), \vec{\mathsf{g}}) = 0$

Let $d(\bar{\tau}(\vec{\mathsf{f}}), \bar{\tau}(\vec{\mathsf{g}})) = k$ for some $k \in \mathbb{R}^+$. Then, since d is ultra,

$$d(\vec{\mathsf{f}}, \vec{\mathsf{g}}) = \max\{d(\vec{\mathsf{f}}, \bar{\tau}(\vec{\mathsf{f}})), d(\bar{\tau}(\vec{\mathsf{f}}), \vec{\mathsf{g}})\} = d(\bar{\tau}(\vec{\mathsf{f}}), \vec{\mathsf{g}}) = \max\{d(\bar{\tau}(\vec{\mathsf{f}}), \bar{\tau}(\vec{\mathsf{g}})), d(\bar{\tau}(\vec{\mathsf{g}}), \vec{\mathsf{g}})\} = k$$

By Lemma 24, we find $k = d(\bar{\tau}(\vec{\mathsf{f}}), \bar{\tau}(\vec{\mathsf{g}})) \leq \frac{1}{2} d(\vec{\mathsf{f}}, \vec{\mathsf{g}}) = \frac{1}{2} k$ which implies that $d(\vec{\mathsf{f}}, \vec{\mathsf{g}}) = 0$. Because of the definition of the product pseudometric on $(\mathtt{Exp}, \partial)$, we have $f(x_i) \sim g(x_i)$ for all $1 \leq i \leq n$. Therefore, the conclusion of the UA is satisfied by bisimilarity. ◀

Because of the above lemma and Theorem 13, both UA and the axioms of $\equiv$ are contained in $\sim$, the greatest bisimulation on $(\mathtt{Exp}, \partial)$. Recall that $\doteqdot$ is the least congruence containing those rules. Since $\sim$ on $(\mathtt{Exp}, \partial)$ is a congruence (Theorem 11), we have that $\doteqdot$ is sound.

▶ **Theorem 26** (Soundness with UA)**.** *For all* $\mathsf{e}, \mathsf{f} \in \mathtt{Exp}$ *if* $\mathsf{e} \doteqdot \mathsf{f}$ *then* $\mathsf{e} \sim \mathsf{f}$*.*

Completeness. After all the hard work is done, the proof of completeness follows via the same line of reasoning as the one for GKAT [58, 53].

▶ **Theorem 27** (Completeness). *For all* $\mathsf{e}, \mathsf{f} \in \mathtt{Exp}$ *if* $\mathsf{e} \sim \mathsf{f}$ *then* $\mathsf{e} \doteqdot \mathsf{f}$

Proof. Let $R \subseteq \mathtt{Exp} \times \mathtt{Exp}$ be a bisimulation with a transition structure $\rho : R \times \mathsf{At} \to \mathcal{D}_\omega(2 + \mathsf{Out} + \mathsf{Act} \times R)$ relating ProbGKAT automata $(\langle \mathsf{e} \rangle_\partial, \partial)$ and $(\langle \mathsf{f} \rangle_\partial, \partial)$ such that $(\mathsf{e}, \mathsf{f}) \in R$. Let π_1, π_2 be the projection homomorphisms from (R, ρ) to $(\langle \mathsf{e} \rangle_\partial, \partial)$ and $(\langle \mathsf{f} \rangle_\partial, \partial)$ respectively. Since both $\langle \mathsf{e} \rangle_\partial$ and $\langle \mathsf{f} \rangle_\partial$ are finite (by Lemma 7), so is R.

Let j, k be the inclusion homomorphisms of $(\langle \mathsf{e} \rangle_\partial, \partial)$ and $(\langle \mathsf{f} \rangle_\partial, \partial)$ in $(\mathtt{Exp}, \partial)$. We can construct two homomorphisms $[-]_\equiv \circ j \circ \pi_1$ and $[-]_\equiv \circ k \circ \pi_2$ from (R, ρ) to $(\mathtt{Exp}/\equiv, \bar{\partial})$. By Theorem 21, $j \circ \pi_1$ and $k \circ \pi_2$ are solutions up to $\equiv$ to the Salomaa system associated with (R, ρ). Since $\equiv$ is contained in $\doteqdot$, those are immediately also solutions up to $\doteqdot$.

Because of UA, we have that $(j \circ \pi_1)(\mathsf{g}, \mathsf{h}) \doteqdot (k \circ \pi_2)(\mathsf{g}, \mathsf{h})$ for all $(\mathsf{g}, \mathsf{h}) \in R$. Thus,

$$\mathsf{e} \doteqdot j(\mathsf{e}) \doteqdot (j \circ \pi_1)(\mathsf{e}, \mathsf{f}) \doteqdot (k \circ \pi_2)(\mathsf{e}, \mathsf{f}) \doteqdot k(\mathsf{f}) \doteqdot \mathsf{f}$$ ◀

7 Decidability and Complexity

To decide whether $\mathsf{e} \doteqdot \mathsf{f}$, we need to demonstrate the existence of a bisimulation between the states e and f in $(\mathtt{Exp}, \partial)$. Since bisimulations need only involve reachable states, it suffices to find this bisimulation within $\langle \mathsf{e}, \mathsf{f} \rangle_\partial$, the smallest subautomaton of $(\mathtt{Exp}, \partial)$ containing e and f, which is also the union of $\langle \mathsf{e} \rangle_\partial$ and $\langle \mathsf{f} \rangle_\partial$; this automaton is finite by Lemma 7. We thus focus on the problem of deciding bisimilarity within a single finite ProbGKAT automaton.

Our analysis in this section is facilitated by two simplifying assertions.

1. To avoid having to compare real (infinite-precision) probabilities, we limit ProbGKAT expressions to rational probabilities $r \in [0, 1] \cap \mathbb{Q}$ in this section. This restriction is compatible with the earlier operators on probabilities, which all preserve rationality.
2. Equivalence of GKAT proper is co-NP-hard [58], simply because Boolean unsatisfiability can trivially be encoded in the language of tests. We take a fixed-parameter approach, assuming that At, the set of atoms that can appear on transitions, is fixed beforehand.

Coalgebraic partition refinement. We rely on *partition refinement* [31, 32, 48], which effectively computes the *largest bisimulation* on an automaton, by approximating it from above. In the *coalgebraic* presentation of partition refinement [67], which we instantiate to our setting, automata of various types are encoded as abstract graphs. More specifically, an automaton is encoded in two maps $o : X \to O$ and $\ell : X \to \mathcal{B}(L \times X)$, where

- X is a set of *nodes* that represent (partial) states of the automaton;
- O is a set of *observable values* at each node;
- L is a set representing possible *labels* of edges between nodes;
- $\mathcal{B}(L \times X)$ is a multiset of pairs representing *edges* between nodes.

Subject to a number of coherence conditions on the encoding (omitted here), coalgebraic partition refinement yields an $\mathcal{O}(n \log |X|)$ algorithm to compute the largest bisimulation on an automaton, where $n = \sum_{x \in X} |\ell(x)|$ is the number of *edges* of the automaton.

Encoding ProbGKAT automata. Coalgebraic partition refinement provides suitable encodings for well-known transition types, as well as methods to soundly obtain encodings of composite transition types [67]. The details of these techniques are beyond the scope of this paper, but the underlying idea is fairly intuitive: composite transition types are encoded

by inserting synthetic nodes that represent partially evaluated states – not unlike how our drawings contain intermediate nodes that are the target of α-labelled arrows. More precisely, the nodes of an encoded ProbGKAT automaton (Q, t) are three-sorted:

1. every state of the automaton is a node; and
2. every "intermediate" state (the small circles in our drawings) is a node; and
3. every probabilistic edge gives rise to another node.

Nodes of the third kind separate the dashed arrows in our drawings (labelled with a probability as well as an action) into two arrows, each of which is labelled by one value.

Formally, we choose $X := Q + I + \mathsf{Act} \times Q$ as our set of nodes, where $I := \{t(q)_\alpha : q \in Q, \alpha \in \mathsf{At}\}$. We also set $L := \mathsf{At} + \mathbb{Q} + \mathsf{Act}$. The map $\ell : X \to \mathcal{B}(L \times X)$ is then defined by:[2]

$$\ell(x) := \begin{cases} \{\!|(\alpha, t(q)_\alpha) \mid \alpha \in \mathsf{At}|\!\} & x = q \in Q \\ \{\!|(d(\mathsf{p}, q), (\mathsf{p}, q)) \mid \mathsf{p} \in \mathsf{Act}, q \in Q|\!\} & x = d \in I \\ \{\!|(\mathsf{p}, q)|\!\} & x = (\mathsf{p}, q) \in \mathsf{Act} \times Q \end{cases}$$

In other words, ℓ labels the edges between nodes of the first and second kind with an atom, the edges between nodes of the second and third kind with a probability, and the edges between nodes of the third and first kind with an action.

Observables represent the probabilities assigned to acceptance, rejection, or a return value by nodes of the second kind. Formally, $O := 1 + \mathbb{Q}^{2+\mathsf{Out}}$, where $* \in 1$ means "no observable value", and values from $\mathbb{Q}^{2+\mathsf{Out}}$ assign a probability to each $\xi \in 2 + \mathsf{Out}$. We can then define $o : X \to O$ by setting $o(d)(\xi) := d(\xi)$ when $d \in I$, and $o(x) := *$ otherwise.[3]

Deciding bisimilarity. We can now leverage the encoding given above to decide bisimilarity.

▶ **Theorem 28.** *If all probabilities are rational and* At *is fixed, then bisimilarity of states in a* ProbGKAT *automaton* (Q, t) *is decidable in time* $\mathcal{O}(|Q|^2 |\mathsf{Act}| \log(|\mathsf{Act} \times Q|))$.

Proof. The results from [67] ensure that our encoding of ProbGKAT automata can be equipped with an appropriate interface that allows their algorithm to decide equivalence.

As for the complexity, we instantiate their abstract complexity result by computing the parameters. The number of nodes and edges can be bound from above fairly easily, as follows:

$$|X| = |Q| + |I| + |\mathsf{Act} \times Q| \leq |Q| + 2 \cdot |\mathsf{Act}| \cdot |Q|$$
$$n = \sum_{x \in X} |\ell(x)| \leq |Q| \cdot |\mathsf{At}| + |Q|^2 \cdot |\mathsf{Act}| + |Q| \cdot |\mathsf{Act}|$$

Since At is fixed, the claimed complexity then follows. ◀

This allows us to conclude that bisimilarity of ProbGKAT expressions is also decidable.

▶ **Corollary 29.** *If all probabilities are rational and* At *is fixed, then* ProbGKAT *equivalence of* $\mathsf{e}, \mathsf{f} \in \mathtt{Exp}$ *is decidable in time* $\mathcal{O}(n^3 \log n)$, *where* $n = \#(\mathsf{e}) + \#(\mathsf{f})$.

Proof. By Lemma 7, $\langle \mathsf{e}, \mathsf{f} \rangle_\partial$ is of size at most n, and the number of distinct actions e or f is fixed from above by n as well. The claim then follows by Theorem 28. ◀

[2] Here, $\{\!|- \mid -|\!\}$ denotes multiset comprehension, where each element occurs at most once.
[3] If the coalgebraic approach from [67] is followed to the letter, the observable map for nodes of the third kind behaves slightly differently; we simplify our encoding here for the sake of presentation.

8 Related work

Our work builds on GKAT, a strictly deterministic fragment [39] of Kleene Algebra with Tests (KAT). KAT has been used in several verification tasks, such as cache control [15], compiler optimisations [37], source-to-source translations [3], and network properties [2, 22, 21, 59, 60, 66], and was generalised to include fuzzy logics [23]. GKAT admits a Salomaa-style [52] axiomatisation of trace equivalence [58] and bisimilarity [53], both relying on the Uniqueness of Solutions axiom, and completeness without it remains open, though completeness of a fragment of GKAT was recently proved by [33].

GKAT modulo bisimilarity and Milner's interpretation of regular expressions arise as fragments of the *parametrised processes* framework [55]; this is not the case for ProbGKAT due to a different treatment of loops. The uniqueness axiom was originally introduced by Bergstra and Klop under the name Recursive Specification Principle (RSP) [10] and used in axiomatisations of process calculi [11]. The general pattern of their proofs of completeness is similar to ours, although the key challenge is the extension to the probabilistic setting.

Our paper also builds up the vast line of research on probabilistic bisimulation [40, 56, 18] and the coalgebraic approach to systems with probabilistic transitions [16, 8, 18, 61]. More concretely, we relied on relation refinement characterisation of bisimilarity [63], natural metrics on the final coalgebras for ω-accessible endofunctors [7, 69], coalgebraic completeness theorems [27, 57, 54] and minimisation algorithms for coalgebras [67, 17, 68, 29]. Axiomatisations of probabilistic bisimulation were extensively studied in the process algebra community, including a recursion-free process algebra of Bandini and Segala [6] and recursive calculi of Stark and Smolka [62] and Mislove, Ouaknine and Worrell [47]. Aceto, Ésik and Ingólfsdóttir [1] gave an alternative axiomatisation of Stark and Smolka's calculus by extending Iteration Theories [12, 20] with equational axioms.

Probabilistic Kleene Algebra (pKA) [43] relaxes the axioms of KA to accommodate reasoning about probabilistic predicate transformers; its axioms are complete w.r.t. simulation equivalence of NFAs [44]. pKA was also extended with a probabilistic choice operator and concurrency primitives [45], but completeness this system was not considered. ProbNetKAT [21, 60] is a domain-specific language for reasoning about probabilistic effects in networks based on KAT, which features a probabilistic choice operator, however, axiomatisation of the obtained language was not studied.

9 Conclusion and Future Work

We have presented ProbGKAT, a language for reasoning about uninterpreted programs with branching and loops, with both Boolean and probabilistic guards. We provided an automata-theoretic operational model and characterised bisimilarity for these automata. We gave a sound and complete axiomatisation of bisimilarity, relying on the Uniqueness of Solutions (UA) axiom, and showed bisimilarity can be efficiently decided in $O(n^3 \log n)$ time.

A first natural direction for future work is the question whether the more traditional language semantics of GKAT can be lifted to ProbGKAT and axiomatised. More broadly, we would like to investigate notions of ProbGKAT expression equivalence more permissive than bisimilarity, including the notion of *bisimulation distance* [5] and its possible axiomatisations based on *quantitative equational logic* [42, 4].

A second direction touches on the problem of completeness without UA, which is still open for (Prob)GKAT. In light of recent completeness results for the skip-free fragment of GKAT [33] modulo bisimilarity and trace equivalence, we are interested to study the skip-free fragment of ProbGKAT. The proofs in [33] do not immediately generalise to ProbGKAT as probabilities do not obviously embed into (1-free) regular expressions.

Similarly to GKAT, ProbGKAT is strictly deterministic and thus avoids known complications of combining nondeterminism with probabilistic choice [30, 65, 24]. We are interested if the recent work on combining multisets and probabilities via distributive laws [28, 38] could be applied to extending our developments with nondeterminism.

ProbGKAT can express only uninterpreted programs, hence it cannot be used to reason about programs involving mutable state. An example of a probabilistic program with state is Pólya's urn [41]. One way of adding mutable state [25] to ProbGKAT is by adding *hypotheses* [14]. Unfortunately, adding hypotheses can lead to undecidability or incompleteness [35], although there are forms of hypotheses that retain completeness [36, 19, 49] and exploring this is as an interesting direction for future work.

References

1 Luca Aceto, Zoltán Ésik, and Anna Ingólfsdóttir. Equational axioms for probabilistic bisimilarity. In *AMAST*, pages 239–253, 2002. doi:10.1007/3-540-45719-4_17.

2 Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker. NetKAT: semantic foundations for networks. In *POPL*, pages 113–126, 2014. doi:10.1145/2535838.2535862.

3 Allegra Angus and Dexter Kozen. Kleene algebra with tests and program schematology. Technical Report TR2001-1844, Cornell University, July 2001. URL: https://hdl.handle.net/1813/5831.

4 Giorgio Bacci, Giovanni Bacci, Kim G. Larsen, and Radu Mardare. Complete axiomatization for the bisimilarity distance on Markov chains. In *CONCUR*, pages 21:1–21:14, 2016. doi:10.4230/LIPIcs.CONCUR.2016.21.

5 Paolo Baldan, Filippo Bonchi, Henning Kerstan, and Barbara König. Coalgebraic behavioral metrics. *Log. Methods Comput. Sci.*, 14(3), 2018. doi:10.23638/LMCS-14(3:20)2018.

6 Emanuele Bandini and Roberto Segala. Axiomatizations for probabilistic bisimulation. In *ICALP*, pages 370–381, 2001. doi:10.1007/3-540-48224-5_31.

7 Michael Barr. Terminal coalgebras in well-founded set theory. *Theor. Comput. Sci.*, 114(2):299–315, 1993. doi:10.1016/0304-3975(93)90076-6.

8 Falk Bartels, Ana Sokolova, and Erik P. de Vink. A hierarchy of probabilistic system types. In *CMCS*, pages 57–75, 2003. doi:10.1016/S1571-0661(04)80632-7.

9 Gilles Barthe, Joost-Pieter Katoen, and Alexandra Silva, editors. *Foundations of Probabilistic Programming*. Cambridge University Press, Cambridge, 2020. doi:10.1017/9781108770750.

10 Jan A. Bergstra and Jan Willem Klop. Verification of an alternating bit protocol by means of process algebra. In *Mathematical Methods of Specification and Synthesis of Software Systems*, volume 215 of *LNCS*, pages 9–23. Springer, 1985. doi:10.1007/3-540-16444-8_1.

11 Jan A. Bergstra and Jan Willem Klop. Process theory based on bisimulation semantics. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *LNCS*, pages 50–122, 1988. doi:10.1007/BFb0013021.

12 Stephen L. Bloom and Zoltán Ésik. *Iteration Theories – The Equational Logic of Iterative Processes*. EATCS Monographs on Theoretical Computer Science. Springer, 1993. doi:10.1007/978-3-642-78034-9.

13 Janusz A. Brzozowski. Derivatives of regular expressions. *J. ACM*, 11(4):481–494, 1964. doi:10.1145/321239.321249.

14 Ernie Cohen. Hypotheses in Kleene algebra. Technical report, Bellcore, 1994.

15 Ernie Cohen. Lazy caching in Kleene algebra. Technical report, Bellcore, 1994.

16 Erik P. de Vink and Jan J. M. M. Rutten. Bisimulation for probabilistic transition systems: A coalgebraic approach. *Theor. Comput. Sci.*, 221(1-2):271–293, 1999. doi:10.1016/S0304-3975(99)00035-3.

17 Hans-Peter Deifel, Stefan Milius, Lutz Schröder, and Thorsten Wißmann. Generic partition refinement and weighted tree automata. In *FM*, pages 280–297, 2019. doi:10.1007/978-3-030-30942-8_18.

18 Josée Desharnais. *Labelled Markov processes*. PhD thesis, McGill University, 1999.

19 Amina Doumane, Denis Kuperberg, Damien Pous, and Pierre Pradic. Kleene algebra with hypotheses. In *FoSSaCS*, pages 207–223, 2019. doi:10.1007/978-3-030-17127-8_12.

20 Calvin C. Elgot. Monadic computation and iterative algebraic theories. In H.E. Rose and J.C. Shepherdson, editors, *Logic Colloquium '73*, volume 80 of *Studies in Logic and the Foundations of Mathematics*, pages 175–230. Elsevier, 1975. doi:10.1016/S0049-237X(08)71949-9.

21 Nate Foster, Dexter Kozen, Konstantinos Mamouras, Mark Reitblatt, and Alexandra Silva. Probabilistic NetKAT. In *ESOP*, pages 282–309, 2016. doi:10.1007/978-3-662-49498-1_12.

22 Nate Foster, Dexter Kozen, Mae Milano, Alexandra Silva, and Laure Thompson. A coalgebraic decision procedure for NetKAT. In *POPL*, pages 343–355, 2015. doi:10.1145/2676726.2677011.

23 Leandro Gomes, Alexandre Madeira, and Luís Soares Barbosa. Generalising KAT to verify weighted computations. *Sci. Ann. Comput. Sci.*, 29(2):141–184, 2019. doi:10.7561/SACS.2019.2.141.

24 Alexandre Goy and Daniela Petrisan. Combining probabilistic and non-deterministic choice via weak distributive laws. In *LICS*, pages 454–464, 2020. doi:10.1145/3373718.3394795.

25 Niels Bjørn Bugge Grathwohl, Dexter Kozen, and Konstantinos Mamouras. KAT + B! In *CSL*, pages 44:1–44:10, 2014. doi:10.1145/2603088.2603095.

26 Matthew Hennessy and Robin Milner. On observing nondeterminism and concurrency. In *ICALP*, pages 299–309, 1980. doi:10.1007/3-540-10003-2_79.

27 Bart Jacobs. A bialgebraic review of deterministic automata, regular expressions and languages. In *Algebra, Meaning, and Computation, Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday*, pages 375–404, 2006. doi:10.1007/11780274_20.

28 Bart Jacobs. From multisets over distributions to distributions over multisets. In *LICS*, pages 1–13, 2021. doi:10.1109/LICS52264.2021.9470678.

29 Jules Jacobs and Thorsten Wißmann. Fast coalgebraic bisimilarity minimization. In *POPL*, pages 1514–1541, 2023. doi:10.1145/3571245.

30 Claire Jones. *Probabilistic non-determinism*. PhD thesis, University of Edinburgh, UK, 1990. URL: https://hdl.handle.net/1842/413.

31 Paris C. Kanellakis and Scott A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. In *PODC*, pages 228–240, 1983. doi:10.1145/800221.806724.

32 Paris C. Kanellakis and Scott A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Inf. Comput.*, 86(1):43–68, 1990. doi:10.1016/0890-5401(90)90025-D.

33 Tobias Kappé, Todd Schmid, and Alexandra Silva. A complete inference system for skip-free guarded Kleene algebra with tests. In *ESOP*, pages 309–336, 2023. doi:10.1007/978-3-031-30044-8_12.

34 Donald E. Knuth and Andrew C. Yao. The complexity of nonuniform random number generation. In *Algorithms and Complexity: New Directions and Recent Results*, 1976.

35 Dexter Kozen. On the complexity of reasoning in Kleene algebra. *Inf. Comput.*, 179(2):152–162, 2002. doi:10.1006/inco.2001.2960.

36 Dexter Kozen and Konstantinos Mamouras. Kleene algebra with equations. In *ICALP (Part II)*, pages 280–292, 2014. doi:10.1007/978-3-662-43951-7_24.

37 Dexter Kozen and Maria-Christina Patron. Certification of compiler optimizations using Kleene algebra with tests. In *CL*, pages 568–582, 2000. doi:10.1007/3-540-44957-4_38.

38 Dexter Kozen and Alexandra Silva. Multisets and distributions, 2023. arXiv:2301.10812.

39 Dexter Kozen and Wei-Lung Dustin Tseng. The Böhm-Jacopini theorem is false, propositionally. In *MPC*, pages 177–192, 2008. doi:10.1007/978-3-540-70594-9_11.

40 Kim G. Larsen and Arne Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991. doi:10.1016/0890-5401(91)90030-6.

41 Hosam Mahmoud. *Pólya Urn Models*. Texts in Statistical Science. Chapman & Hall, 2008.

42 Radu Mardare, Prakash Panangaden, and Gordon D. Plotkin. Quantitative algebraic reasoning. In *LICS*, pages 700–709, 2016. doi:10.1145/2933575.2934518.

43 Annabelle McIver, Carlos Gonzalía, Ernie Cohen, and Carroll C. Morgan. Using probabilistic Kleene algebra pKA for protocol verification. *J. Log. Algebraic Methods Program.*, 76(1):90–111, 2008. doi:10.1016/j.jlap.2007.10.005.

44 Annabelle McIver, Tahiry M. Rabehaja, and Georg Struth. On probabilistic Kleene algebras, automata and simulations. In *RAMICS*, pages 264–279, 2011. doi:10.1007/978-3-642-21070-9_20.

45 Annabelle McIver, Tahiry M. Rabehaja, and Georg Struth. Probabilistic concurrent Kleene algebra. In *QAPL*, pages 97–115, 2013. doi:10.4204/EPTCS.117.7.

46 Robin Milner. A complete inference system for a class of regular behaviours. *J. Comput. Syst. Sci.*, 28(3):439–466, 1984. doi:10.1016/0022-0000(84)90023-0.

47 Michael W. Mislove, Joël Ouaknine, and James Worrell. Axioms for probability and nondeterminism. In *EXPRESS*, pages 7–28, 2003. doi:10.1016/j.entcs.2004.04.019.

48 Robert Paige and Robert Endre Tarjan. Three partition refinement algorithms. *SIAM J. Comput.*, 16(6):973–989, 1987. doi:10.1137/0216062.

49 Damien Pous, Jurriaan Rot, and Jana Wagemaker. On tools for completeness of kleene algebra with hypotheses, 2022. arXiv:2210.13020.

50 Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000. doi:10.1016/S0304-3975(00)00056-6.

51 Wojciech Różowski, Tobias Kappé, Dexter Kozen, Todd Schmid, and Alexandra Silva. Probabilistic Guarded KAT Modulo Bisimilarity: Completeness and Complexity, 2023. arXiv:2305.01755.

52 Arto Salomaa. Two complete axiom systems for the algebra of regular events. *J. ACM*, 13(1):158–169, 1966. doi:10.1145/321312.321326.

53 Todd Schmid, Tobias Kappé, Dexter Kozen, and Alexandra Silva. Guarded Kleene algebra with tests: Coequations, coinduction, and completeness. In *ICALP*, pages 142:1–142:14, 2021. doi:10.4230/LIPIcs.ICALP.2021.142.

54 Todd Schmid, Jurriaan Rot, and Alexandra Silva. On star expressions and coalgebraic completeness theorems. In *MFPS*, pages 242–259, 2021. doi:10.4204/EPTCS.351.15.

55 Todd Schmid, Wojciech Różowski, Alexandra Silva, and Jurriaan Rot. Processes parametrised by an algebraic theory. In *ICALP*, 2022. doi:10.4230/LIPIcs.ICALP.2022.132.

56 Roberto Segala and Nancy A. Lynch. Probabilistic simulations for probabilistic processes. In *CONCUR*, pages 481–496, 1994. doi:10.1007/978-3-540-48654-1_35.

57 Alexandra Silva. *Kleene coalgebra*. PhD thesis, University of Nijmegen, 2010.

58 Steffen Smolka, Nate Foster, Justin Hsu, Tobias Kappé, Dexter Kozen, and Alexandra Silva. Guarded Kleene algebra with tests: verification of uninterpreted programs in nearly linear time. In *POPL*, pages 61:1–61:28, 2020. doi:10.1145/3371129.

59 Steffen Smolka, Praveen Kumar, Nate Foster, Dexter Kozen, and Alexandra Silva. Cantor meets Scott: semantic foundations for probabilistic networks. In *POPL*, pages 557–571, 2017. doi:10.1145/3009837.3009843.

60 Steffen Smolka, Praveen Kumar, David M. Kahn, Nate Foster, Justin Hsu, Dexter Kozen, and Alexandra Silva. Scalable verification of probabilistic networks. In *PLDI*, pages 190–203, 2019. doi:10.1145/3314221.3314639.

61 Ana Sokolova. Probabilistic systems coalgebraically: A survey. *Theor. Comput. Sci.*, 412(38):5095–5110, 2011. doi:10.1016/j.tcs.2011.05.008.

62 Eugene W. Stark and Scott A. Smolka. A complete axiom system for finite-state probabilistic processes. In *Proof, Language, and Interaction, Essays in Honour of Robin Milner*, pages 571–596, 2000.

63 Sam Staton. Relating coalgebraic notions of bisimulation. *Log. Methods Comput. Sci.*, 7(1), 2011. doi:10.2168/LMCS-7(1:13)2011.

64 Joseph Aaron Toumanios. Three sided die, 2019. US patent 10384119. URL: https://image-ppubs.uspto.gov/dirsearch-public/print/downloadPdf/10384119.

65 Daniele Varacca and Glynn Winskel. Distributing probability over non-determinism. *Mathematical Structures in Computer Science*, 16(1):87–113, 2006. doi:10.1017/S0960129505005074.

66 Jana Wagemaker, Nate Foster, Tobias Kappé, Dexter Kozen, Jurriaan Rot, and Alexandra Silva. Concurrent NetKAT – Modeling and analyzing stateful, concurrent networks. In *ESOP*, pages 575–602, 2022. doi:10.1007/978-3-030-99336-8_21.

67 Thorsten Wißmann, Ulrich Dorsch, Stefan Milius, and Lutz Schröder. Efficient and modular coalgebraic partition refinement. *Logical Methods in Computer Science*, 16:1:8:1–8:63, 2020. doi:10.23638/LMCS-16(1:8)2020.

68 Thorsten Wißmann, Stefan Milius, and Lutz Schröder. Quasilinear-time computation of generic modal witnesses for behavioural inequivalence. *Log. Methods Comput. Sci.*, 18(4), 2022. doi:10.46298/lmcs-18(4:6)2022.

69 James Worrell. On the final sequence of a finitary set functor. *Theor. Comput. Sci.*, 338(1-3):184–199, 2005. doi:10.1016/j.tcs.2004.12.009.

Action Codes

Frits Vaandrager
Radboud University, Nijmegen, The Netherlands

Thorsten Wißmann
Radboud University, Nijmegen, The Netherlands
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Abstract

We provide a new perspective on the problem how high-level state machine models with abstract actions can be related to low-level models in which these actions are refined by sequences of concrete actions. We describe the connection between high-level and low-level actions using *action codes*, a variation of the prefix codes known from coding theory. For each action code $\mathcal{R}$, we introduce a *contraction* operator $\alpha_{\mathcal{R}}$ that turns a low-level model $\mathcal{M}$ into a high-level model, and a *refinement* operator $\varrho_{\mathcal{R}}$ that transforms a high-level model $\mathcal{N}$ into a low-level model. We establish a Galois connection $\varrho_{\mathcal{R}}(\mathcal{N}) \sqsubseteq \mathcal{M} \Leftrightarrow \mathcal{N} \sqsubseteq \alpha_{\mathcal{R}}(\mathcal{M})$, where $\sqsubseteq$ is the well-known simulation preorder. For conformance, we typically want to obtain an overapproximation of model $\mathcal{M}$. To this end, we also introduce a *concretization* operator $\gamma_{\mathcal{R}}$, which behaves like the refinement operator but adds arbitrary behavior at intermediate points, giving us a second Galois connection $\alpha_{\mathcal{R}}(\mathcal{M}) \sqsubseteq \mathcal{N} \Leftrightarrow \mathcal{M} \sqsubseteq \gamma_{\mathcal{R}}(\mathcal{N})$. Action codes may be used to construct adaptors that translate between concrete and abstract actions during learning and testing of Mealy machines. If Mealy machine $\mathcal{M}$ models a black-box system then $\alpha_{\mathcal{R}}(\mathcal{M})$ describes the behavior that can be observed by a learner/tester that interacts with this system via an adaptor derived from code $\mathcal{R}$. Whenever $\alpha_{\mathcal{R}}(\mathcal{M})$ implements (or conforms to) $\mathcal{N}$, we may conclude that $\mathcal{M}$ implements (or conforms to) $\gamma_{\mathcal{R}}(\mathcal{N})$.

Almost all results, examples, and counter-examples are formalized in Coq.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Automata, Models of Reactive Systems, LTS, Action Codes, Action Refinement, Action Contraction, Galois Connection, Model-Based Testing, Model Learning

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.137

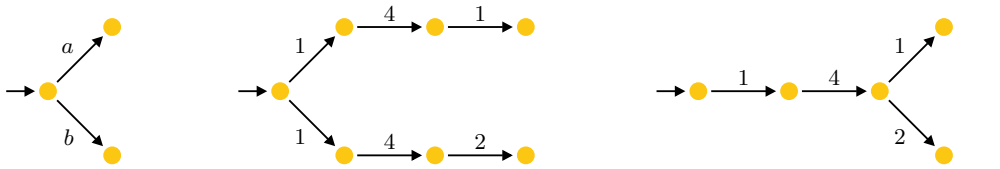
Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version With Appendix*: `https://arxiv.org/abs/2301.00199`

Supplementary Material *Software*: `https://gitlab.science.ru.nl/twissmann/action-codes-coq`, archived at `swh:1:dir:953b24c1e0771ce9ed7961f59f07294e0fd615d2`

Funding *Frits Vaandrager*: Supported by the NWO TOP project 612.001.852.
Thorsten Wißmann: Supported by the NWO TOP project 612.001.852 (until 2022) and the DFG project 419850228 (since 2023).

Acknowledgements As part of an MSc thesis project under supervision of the first author, Timo Maarse studied a different and more restricted type of action codes (called action refinements) [29]. It turned out, however, that for these action codes, the concretization operator is not monotone. The present paper arose from our efforts to fix this problem. We thank the anonymous reviewers for their suggestions, Paul Fiterău-Broştean for examples of the use of action codes in model learning, and Jules Jacobs for helpful discussions about Coq. The first author would like to thank Rocco De Nicola for his hospitality at IMT Lucca during the work on this paper.



(a) Original System. **(b)** Existing action refinements. **(c)** Desired action refinement behavior.

Figure 1 Example for the (lack of) preservation of determinism in action refinement.

1 Introduction

Labeled transition systems (LTSs) constitute one of the most fundamental modeling mechanisms in Computer Science. An LTS is a rooted, directed graph whose nodes represent *states* and whose edges are labeled with *actions* and represent *state transitions*. LTS-based formalisms such as Finite Automata [21], Finite State Machines [25], I/O automata [26], IOTSs [35], and process algebras [4] have been widely used to model and analyze a broad variety of reactive systems, and a rich body of theory has been developed for them.

In order to manage the complexity of computer-based systems, designers structure such systems into hierarchical layers. This allows them to describe and analyze systems at different levels of abstraction. Many LTS-based frameworks have been proposed to formally relate models at different hierarchical levels, e.g. [4, 14, 27, 40]. In most of these frameworks, the states of a high-level LTS correspond to sets of states of a low-level LTS via simulation or bisimulation-like relations. However, the actions are fixed and considered to be atomic. Actions used at a lower level of abstraction can be hidden at a higher level, but higher-level actions will always be available at the lower level. For this reason, Rensink & Gorrieri [18, 31] argue that these (bi)simulations relate systems at the same conceptual level of abstraction, and therefore they call them *horizontal* implementation relations. They contrast them with *vertical* implementation relations that compare systems that belong to conceptually different abstraction levels, and have different alphabets of actions.

A prototypical example of a hierarchical design is a computer network. To reduce design complexity, such a network is organized as a stack of layers or levels, each one built upon the one below it [34]. Examples are the transport layer, with protocols such as TCP and UDP, and the physical layer, concerned with transmitting raw bits over a communication channel. Now consider a host that receives a TCP packet in some state s. If P is the set of possible packets then, in an LTS model of the transport layer, state s will contain outgoing transitions labeled with action $receive(p)$, for each $p \in P$. At the physical layer, however, receipt of a packet corresponds to a sequence of $receive(b)$ actions, with b a bit in $\{0, 1\}$. Only after the final bits have arrived, the host knows which packet was actually received. Mechanisms for transforming high-level actions into sequences (or processes) of low-level actions have been addressed extensively in work on action refinements [18]. These approaches, however, are unable to describe the above scenario in a satisfactory manner and somehow assume that a host upfront correctly guesses the packet that it will receive, even before the first bit has arrived. In order to illustrate this problem, we consider the simplified example of an LTS with a distinguished initial state, displayed in Figure 1a, which accepts either input a or input b. At a lower level of abstraction, input a is implemented by three consecutive inputs 1 4 1, whereas input b is implemented by action sequence 1 4 2 (the ASCII encodings of a and b in octal format). An action refinement operator will replace the a-transition in Figure 1a by a sequence of three consecutive transitions with labels 1, 4 and 1, respectively,

and will handle the b-transition in an analogous manner. Thus, action refinement introduces a nondeterministic choice (Figure 1b), rather than the deterministic behavior that one would like to see (Figure 1c). As a consequence of this and other limitations, refinement operators have not found much practical use [18].

Based on the observation that any action can be modeled as a state change, some authors (e.g. [2, 10, 24]) prefer modeling formalisms in which the term "action" is only used informally, and Kripke structures rather than LTSs are used to model systems. These state-based approaches have the advantage that a distinction between horizontal and vertical implementation relations is no longer needed, and a single implementation relation suffices. Purely state-based approaches, however, are problematic in cases where we need to interact with a black-box system and (by definition) we have no clue about the state of this system. Black-box systems prominently occur in the areas of model-based testing [36] and model learning [37]. In these application areas, use of LTSs makes sense and there is a clear practical need for formalisms that allow engineers to relate actions at different levels of abstraction.

Van der Bijl et al. [7], for instance, observe that in model-based testing specifications are usually more abstract than the System Under Test (SUT). This means that generated test cases may not have the required level of detail, and often a single abstract action has to be translated (either manually or by an adaptor) to a sequence of concrete actions that are applied to the SUT. Van der Bijl et al. [7] study a restricted type of action refinement in which a single input is refined into a sequence of inputs, and implement this in a testing tool.

Also in model learning, typically an adaptor is placed in between the SUT and the learner, to take care of the translation between abstract and concrete actions. For example, in a case study on hand-held smartcard readers for Internet banking, Chalupar et al. [9] used abstract inputs that combine several concrete inputs in order to accelerate the learning process and reduce the size of the learned model. In particular, they introduced a single abstract input COMBINED_PIN corresponding to a USB command, followed by a 4-digit PIN code, followed by an OK command. Fiterău-Broştean et al. [12] used model learning for a comprehensive analysis of DTLS implementations, and found four serious security vulnerabilities, as well as several functional bugs and non-conformance issues. Handshakes in (D)TLS are defined over flights of messages. Hence, (D)TLS entities are often expected to produce multiple messages before expecting a response. During learning, Fiterău-Broştean et al. [12] used an adaptor that contracted multiple messages from the SUT into a single abstract output. Also in other case studies on TLS [32], Wi-Fi [33] and SSH [39, 13], multiple outputs from the SUT were contracted into a single abstract output. Verleg [39] used a single abstract input to execute the entire key re-exchange during learning higher layers of SSH.

In this article, we provide answers to two fundamental questions: (1) How can we formalize the concept of an *adaptor* that translates between abstract and concrete actions?, and (2) Suppose the behavior of an SUT is described by an unknown, concrete model $\mathcal{M}$, and suppose a learner interacts with this SUT through an adaptor and learns an abstract model $\mathcal{N}$. What can we say about the relation between $\mathcal{M}$ and $\mathcal{N}$?

We answer the first question by introducing *action codes*, a variation of the prefix codes known from coding theory [5]. Action codes describe how high-level actions are converted into sequences of low-level actions, and vice versa. This makes them different from action refinements, which specify how high-level actions can be translated into low-level processes, but do not address the reverse translation. Our notion of an action code captures adaptors that are used in practice, and in particular those described in the case studies listed above.

In order to answer the second question we introduce, for each action code $\mathcal{R}$, a *contraction* operator $\alpha_{\mathcal{R}}$ that turns a low-level model $\mathcal{M}$ into a high-level model by contracting concrete action sequences of $\mathcal{M}$ according to $\mathcal{R}$. We also introduce the left adjoint of $\alpha_{\mathcal{R}}$, the *refinement*

operator $\varrho_{\mathcal{R}}$ that turns a high-level model $\mathcal{M}$ into a low-level model by refining abstract actions of $\mathcal{N}$ according to $\mathcal{R}$. This refinement operator, for instance, maps the LTS of Figure 1a to the LTS of Figure 1c. We establish a Galois connection $\varrho_{\mathcal{R}}(\mathcal{N}) \sqsubseteq \mathcal{M} \;\Leftrightarrow\; \mathcal{N} \sqsubseteq \alpha_{\mathcal{R}}(\mathcal{M})$, where $\sqsubseteq$ denotes the simulation preorder. So if an abstract model $\mathcal{N}$ implements contraction $\alpha_{\mathcal{R}}(\mathcal{M})$, then the refinement $\varrho_{\mathcal{R}}(\mathcal{N})$ implements concrete model $\mathcal{M}$, and vice versa.

In practice, we typically want to obtain an overapproximation of concrete model $\mathcal{M}$. To this end, we introduce the right adjoint of $\alpha_{\mathcal{R}}$, the *concretization* operator $\gamma_{\mathcal{R}}$. This operator behaves like the refinement operator, but adds arbitrary behavior at intermediate points (cf. the demonic completion of [6]). We establish another Galois connection: $\alpha_{\mathcal{R}}(\mathcal{M}) \sqsubseteq \mathcal{N} \Leftrightarrow \mathcal{M} \sqsubseteq \gamma_{\mathcal{R}}(\mathcal{N})$. This connection is useful, because whenever we have established that $\alpha_{\mathcal{R}}(\mathcal{M})$ implements (or conforms to) $\mathcal{N}$, it allows us to conclude that $\mathcal{M}$ implements (or conforms to) $\gamma_{\mathcal{R}}(\mathcal{N})$.

We show that, in a setting of Mealy machines (subsuming Finite State Machines), an *adaptor* can be constructed for any action code for which a winning strategy exists in a certain 2-player game. If a learner/tester interacts with an SUT via an adaptor generated from such an action code $\mathcal{R}$, and the SUT is modeled by Mealy machine $\mathcal{M}$, then from the learner/tester perspective, the composition of adaptor and SUT will behave like $\alpha_{\mathcal{R}}(\mathcal{M})$. Thus, if a learner succeeds to learn an abstract model $\mathcal{N}$ such that $\mathcal{N} \approx \alpha_{\mathcal{R}}(\mathcal{M})$ then, using the Galois connections, the learner may conclude that $\varrho_{\mathcal{R}}(\mathcal{N}) \sqsubseteq \mathcal{M} \sqsubseteq \gamma_{\mathcal{R}}(\mathcal{N})$.

The remainder of this article is structured as follows. We start with a preliminary Section 2 that introduces basic notations and results for LTSs. Next, action codes and the contraction operator are introduced in Section 3. After describing the refinement operator, we establish our first Galois connection in Section 4. Next we define concretization and establish our second Galois connection in Sections 5. Section 6 explains how action codes can be composed, and shows that contraction and refinement commute with action code composition. Section 7 describes how adaptors can be constructed from action codes. Finally, Section 8 contains a discussion of our results and identifies directions for future research.

Almost all proofs are formalized in Coq (about 6000 lines of code) and can be accessed via `https://gitlab.science.ru.nl/twissmann/action-codes-coq` and via the ancillary files of the full version on arxiv. We mark formalized results with a clickable Coq icon pointing to the respective location in the HTML documentation. Appendix A (in the full version) contains comments on the Coq formalization and Appendix B contains full proofs (in natural language) and additional remarks.

2 Preliminaries

If Σ is a set of symbols then Σ^* denotes the set of all finite words over Σ, and Σ^+ the set of all non-empty words. We use ε to denote the empty word, so e.g. $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$. Concatenation of words $u, w \in \Sigma^*$ is notated $u \cdot w$ (or simply $u\,w$). We write $u \leq w$ if u is a prefix of w, i.e. if there is $v \in \Sigma^*$ with $u\,v = w$. We write $|w|$ to denote the length of word w.

We use $f\colon X \rightharpoonup Y$ to denote a partial map f from X to Y and write $\mathsf{dom}(f) \subseteq X$ for its domain, i.e. set of $x \in X$ on which f is defined. The *image* $\mathsf{im}(f)$ of a partial map $f\colon X \rightharpoonup Y$ is the set of elements of Y it can reach: $\mathsf{im}(f) := \{f(x) \mid x \in \mathsf{dom}(f)\} \subseteq Y$.

▶ **Definition 2.1** (). *For a set A of action labels, a* labeled transition system (LTS) *is a tuple $\mathcal{M} = \langle Q, q_0, \longrightarrow \rangle$ where Q is a set of states, $q_0 \in Q$ is a starting state, and $\longrightarrow \subseteq Q \times A \times Q$ is a transition relation. We write $\mathsf{LTS}(A)$ for the class of all LTSs with labels from A. We refer to the three components of an LTS $\mathcal{M}$ as $Q^{\mathcal{M}}$, $q_0^{\mathcal{M}}$ and $\longrightarrow_{\mathcal{M}}$, respectively, and introduce the following notation:*

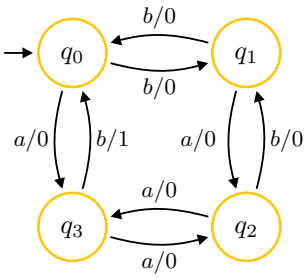


Figure 2 A Mealy machine.

- $q \xrightarrow{a} q'$ *denotes* $(q, a, q') \in \longrightarrow$; $q \xrightarrow{a}$ *denotes that there is some* q' *with* $q \xrightarrow{a} q'$;
- $q \overset{w}{\Longrightarrow} q'$ *for* $w \in A^*$ *denotes that there are finite sequences* $a_1, \ldots, a_n \in A$, $r_0, \ldots, r_n \in Q$ *such that* $w = a_1 \cdots a_n$, *and* $r_0 = q$, $r_n = q'$ *and* $r_{i-1} \xrightarrow{a_i} r_i$ *for all* $1 \leq i \leq n$;
- $q \overset{w}{\Longrightarrow}$ *denotes that there is* q' *such that* $q \overset{w}{\Longrightarrow} q'$;
- $q \in Q$ *is* reachable *if there is* $w \in A^*$ *such that* $q_0 \overset{w}{\Longrightarrow} q$.

A special class of LTSs that is frequently used in conformance testing and model learning are *Mealy machines*. Mealy machines with a finite number of states are commonly referred to as *Finite State Machines*.

▶ **Definition 2.2.** *For non-empty sets of inputs* I *and outputs* O, *a (non-deterministic)* Mealy machine $\mathcal{M} \in \mathsf{LTS}(I \times O)$ *is an LTS where the labels are pairs of an input and an output. We write* $q \xrightarrow{i/o} q'$ *to denote that* $(q, (i, o), q') \in \longrightarrow$. *Whenever we omit a symbol in predicate* $q \xrightarrow{i/o} q'$ *this is quantified existentially. Thus,* $\xrightarrow{i/o}$ *if there are* q *and* q' *s.t.* $q \xrightarrow{i/o} q'$, $q \xrightarrow{i/} q'$ *if there is an* o *s.t.* $q \xrightarrow{i/o} q'$, *and* $q \xrightarrow{i/}$ *if there is a* q' *s.t.* $q \xrightarrow{i/} q'$.

▶ **Example 2.3** (🐓). Figure 2 visualizes a simple Mealy machine with inputs $\{a, b\}$ and outputs $\{0, 1\}$. The machine always outputs 0 in response to an input, except in one specific situation. Output 1 is produced in response to input b if the previous input was a and the number of preceding inputs is odd. The machine has four states q_0, q_1, q_2 and q_3, with starting state q_0 marked by an incoming arrow. In states q_0 and q_2 the number of preceding inputs is always even, whereas in states q_1 and q_3 it is always odd. In states q_2 and q_3 the previous input is always a, whereas in states q_0 and q_1 either the previous input is b, or no input has occurred yet. Thus, only in state q_3 input b triggers output 1.

We introduce some notation and terminology for LTSs.

▶ **Definition 2.4** (🐓). *Let* $\mathcal{M} = \langle Q, q_0, \longrightarrow \rangle \in \mathsf{LTS}(A)$ *be an LTS. We say that*

- $\mathcal{M}$ *is* deterministic *if, whenever* $q \xrightarrow{a}$ *for some* q *and* a, *there is a unique* q' *with* $q \xrightarrow{a} q'$.
- $\mathcal{M}$ *is a* tree-shaped *if each state* $q \in Q$ *can be reached via a unique sequence of transitions from state* q_0.
- $q \in Q$ *is a* leaf, *notated* $q \not\rightarrow$, *if there is no* $a \in A$ *with* $q \xrightarrow{a}$.
- $\mathcal{M}$ *is* grounded *if every state* $q \in Q$ *has a path to a leaf.*

We can now define the set of traces of an LTS:

▶ **Definition 2.5** (🐓). *Let* $\mathcal{M} = \langle Q, q_0, \longrightarrow \rangle \in \mathsf{LTS}(A)$. *A word* $w \in A^*$ *is a* trace *of state* $q \in Q$ *if* $q \overset{w}{\Longrightarrow}$, *and a* trace *of* $\mathcal{M}$ *if it is a trace of* q_0. *We write* $\mathit{trace}(\mathcal{M})$ *for the set* $\{w \in A^* \mid q_0 \overset{w}{\Longrightarrow}\}$ *of all traces of* $\mathcal{M}$.

▶ **Definition 2.6** (Simulation, 🐓). *For $\mathcal{M}, \mathcal{N} \in \mathsf{LTS}(A)$, a* simulation *from $\mathcal{M}$ to $\mathcal{N}$ is a relation $S \subseteq Q^{\mathcal{M}} \times Q^{\mathcal{N}}$ such that*
1. $q_0^{\mathcal{M}}\ S\ q_0^{\mathcal{N}}$ *and*
2. *if $q_1\ S\ q_2$ and $q_1 \xrightarrow{a}_{\mathcal{M}} q_1'$ then there exists a state q_2' such that $q_2 \xrightarrow{a}_{\mathcal{N}} q_2'$ and $q_1'\ S\ q_2'$.*

We write $\mathcal{M} \sqsubseteq \mathcal{N}$ if there exists a simulation from $\mathcal{M}$ to $\mathcal{N}$.

It is a classical result that trace inclusion coincides with the simulation preorder for deterministic labeled transition systems (see e.g. [28]):

▶ **Lemma 2.7** (🐓). *For all $\mathcal{M}, \mathcal{N} \in \mathsf{LTS}(A)$ where $\mathcal{N}$ is deterministic: $\mathit{trace}(\mathcal{M}) \subseteq \mathit{trace}(\mathcal{N})$ iff $\mathcal{M} \sqsubseteq \mathcal{N}$.*

We will often consider LTSs up to isomorphism of their reachable parts:

▶ **Definition 2.8** (Isomorphism, 🐓). *For $\mathcal{M}, \mathcal{N} \in \mathsf{LTS}(A)$, an* isomorphism *from $\mathcal{M}$ to $\mathcal{N}$ is a bijection $f\colon Q_R^{\mathcal{M}} \to Q_R^{\mathcal{N}}$, where:*
1. *$Q_R^{\mathcal{M}} \subseteq Q^{\mathcal{M}}$ and $Q_R^{\mathcal{N}} \subseteq Q^{\mathcal{N}}$ are the subsets of reachable states in $\mathcal{M}$ and $\mathcal{N}$, respectively;*
2. *$f(q_0^{\mathcal{M}}) = q_0^{\mathcal{N}}$, and*
3. *$q \xrightarrow{a}_{\mathcal{M}} q'$ iff $f(q) \xrightarrow{a}_{\mathcal{N}} f(q')$, for all $q, q' \in Q_R^{\mathcal{M}}$, $a \in A$.*

We write $\mathcal{M} \cong \mathcal{N}$ if there exists an isomorphism from $\mathcal{M}$ to $\mathcal{N}$.

Note that $\cong$ is an equivalence relation on $\mathsf{LTS}(A)$, and that $\mathcal{M} \cong \mathcal{N}$ implies $\mathcal{M} \sqsubseteq \mathcal{N}$, since each isomorphism (when viewed as a relation) is trivially a simulation.

3 Action Codes

Adaptors that are used for learning and testing translate sequences of abstract actions into sequences of concrete actions, and vice versa. Action codes describe *how* an adaptor may translate between two action label alphabets, for example from A to B. Intuitively, we understand the first alphabet A as the actions at the lower, concrete level, and the second alphabet B as the actions at the higher, more abstract level. In an action code, a single abstract action $b \in B$ corresponds to a finite, non-empty sequence of concrete actions $a_1 \cdots a_n$ in A. Essentially, action codes are just a special type of *prefix codes* [5], as known from coding theory. Prefix codes have the desirable property that they are *uniquely decodable*: given a sequence of concrete actions, there is at most one corresponding sequence of abstract actions. We provide two equivalent definitions of action codes: one via tree-shaped LTSs and one via partial maps.

▶ **Definition 3.1** (Action code, 🐓). *For sets of action labels A and B, a* (tree-shaped) action code *$\mathcal{R}$ from A to B is a structure $\mathcal{R} = \langle \mathcal{M}, l \rangle$, with $\mathcal{M} = \langle R, r_0, \longrightarrow \rangle \in \mathsf{LTS}(A)$ a deterministic, tree-shaped LTS with L being the set of non-root leaves $L \subseteq R \setminus \{r_0\}$ and an injective function $l\colon L \to B$. We write $\mathsf{Code}(A, B)$ for all action codes from A to B.*

The injectivity of l and the tree-shape ensure that every abstract $b \in B$ is represented by at most one $w \in A^+$.

▶ **Example 3.2.** Figure 3 shows an action code for a fragment of the ASCII encoding in octal format, e.g., 1 1 5 encodes the letter `M`, 1 4 5 encodes the letter `e`, etc.

▶ **Example 3.3.** Figure 4 shows an action code for the activity of getting a cup of coffee or espresso, in the special case of Mealy machines, i.e. where $A = I \times O$ and $B = I' \times O'$ are sets of input/output-pairs. Rather than the full sequence of interventions that is required in order to get a drink, the abstract input/output pair only reports on the type of drink that was ordered and the number of interventions that occurred.

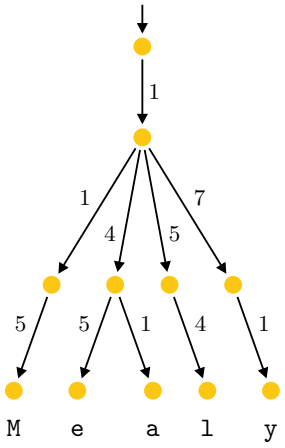


Figure 3 Action code for a fragment of ASCII.

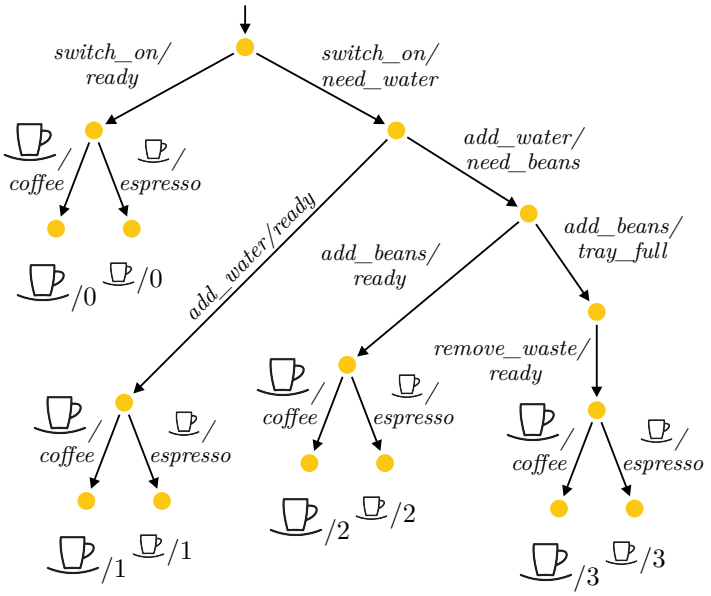


Figure 4 Action code for a coffee machine.

The definition of action codes as LTSs allows an intuitive visualization. For easier mathematical reasoning, we characterize action codes also in terms of maps:

▶ **Definition 3.4** (🐓). *A* (map-based) action code *from A to B is a partial map $f\colon B \rightharpoonup A^+$ which is* prefix-free*, by which we mean that for all $b, b' \in \mathsf{dom}(f)$,*

$$f(b) \leq f(b') \qquad \textit{implies} \qquad b = b'. \tag{1}$$

In the following, we show that these prefix-free partial maps bijectively correspond to the tree-shaped LTSs:

▶ **Lemma 3.5** (🐓). *Every tree-shaped action code $\mathcal{R} \in \mathsf{Code}(A, B)$ induces a unique map-based action code $f\colon B \rightharpoonup A^+$ with the property that for all $b \in B, w \in A^+$:*

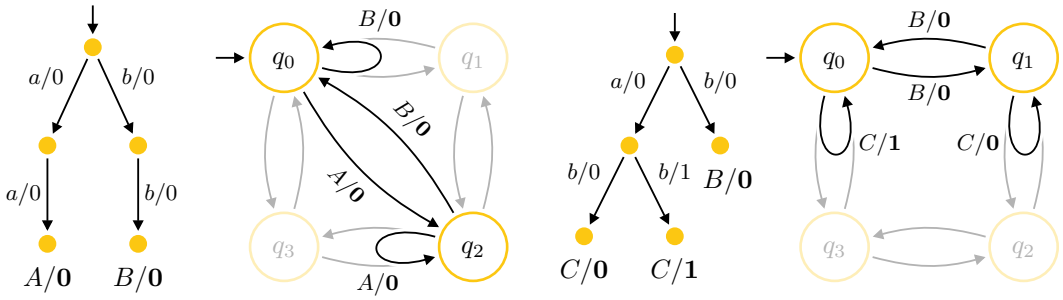
$$f(b) = w \qquad \textit{iff} \qquad \exists r \in L\colon r_0 \xRightarrow{w}_{\mathcal{R}} r, \quad l(r) = b \tag{2}$$

▶ **Lemma 3.6** (🐓). *For each map-based action code $f\colon B \rightharpoonup A^+$, there is (up to isomorphism) a unique tree-shaped action code $\mathcal{R} \in \mathsf{Code}(A, B)$ which is grounded and satisfies* (2)*.*

▶ **Example 3.7** (🐓). For the uniqueness in Lemma 3.6, we use groundedness, because for $A = \{a\}$ and any B, the action codes

$\mathcal{R} :=$  and $\mathcal{S} :=$  .

both have no non-root leaves, and so they both induce the empty partial map $f\colon B \rightharpoonup A^+$ via Lemma 3.5. This f is undefined for all $b \in B$. And indeed, $\mathcal{R}$ and $\mathcal{S}$ are not isomorphic. The issue is that while the finite $\mathcal{S}$ is grounded, the infinite $\mathcal{R}$ is not grounded. So $\mathcal{R}$ contains subtrees which do not contribute anything to the partial map f but which hinder the existence of an isomorphism.



(a) An action code $\mathcal{R}$ together with $\alpha_{\mathcal{R}}(\mathcal{M})$. **(b)** Another code $\mathcal{S}$ together with $\alpha_{\mathcal{S}}(\mathcal{M})$.

Figure 5 The resulting contraction of the LTS $\mathcal{M}$ from Figure 2 for different action codes.

Having shown the correspondence between tree-shaped and map-based action codes $\mathsf{Code}(A, B)$, we can switch between the two views in proofs. Mostly, we use the tree-shaped version for visualization and the map-based version for mathematical reasoning.

Consider a concrete $\mathcal{M} \in \mathsf{LTS}(A)$, together with an action code $\mathcal{R}$ from A to B. We can construct an abstract LTS for the action labels B by walking through $\mathcal{M}$ with seven-league boots, repeatedly choosing input sequences that correspond to runs to some leaf of $\mathcal{R}$, and then contracting this sequence to a single abstract transition.

▶ **Notation 3.8.** In the rest of the paper, we introduce operators $\alpha_{\mathcal{R}}$, $\varrho_{\mathcal{R}}$, $\gamma_{\mathcal{R}}$ on LTSs, involving an action code $\mathcal{R}$. Whenever the action code $\mathcal{R}$ is clear from the context, we omit the index and simply speak of operators α, ϱ, γ for the sake of brevity.

▶ **Definition 3.9** (Contraction, 🐓). *For each action code $\mathcal{R} \in \mathsf{Code}(A, B)$, the* contraction *operator $\alpha_{\mathcal{R}} \colon \mathsf{LTS}(A) \to \mathsf{LTS}(B)$ is defined as follows. For $\mathcal{M} \in \mathsf{LTS}(A)$, the LTS $\alpha_{\mathcal{R}}(\mathcal{M})$ has states $Q^{\alpha(\mathcal{M})} \subseteq Q^{\mathcal{M}}$ and transitions $\longrightarrow_{\alpha(\mathcal{M})}$ defined inductively by the rules (1_α) and (2_α), for all $q, q' \in Q^{\mathcal{M}}$, $b \in B$.*

$$\frac{}{q_0^{\mathcal{M}} \in Q^{\alpha(\mathcal{M})}} \quad (1_\alpha) \qquad \frac{q \in Q^{\alpha(\mathcal{M})}, \quad b \in \mathsf{dom}(\mathcal{R}), \quad q \xRightarrow{\mathcal{R}(b)}_{\mathcal{M}} q'}{q \xrightarrow{b}_{\alpha(\mathcal{M})} q', \qquad q' \in Q^{\alpha(\mathcal{M})}} \quad (2_\alpha)$$

The initial state $q_0^{\alpha(\mathcal{M})} := q_0^{\mathcal{M}}$ is the same as in $\mathcal{M}$.

▶ **Example 3.10.** Figures 5 shows two examples of action codes and the contractions obtained when we apply them to the Mealy machine of Figure 2 (with the original machine shaded in the background). The examples illustrate that by choosing different codes we may obtain completely different abstractions of the same LTS.

The next proposition asserts that we can view $\alpha_{\mathcal{R}}$ as a monotone function $\alpha_{\mathcal{R}} \colon \mathsf{LTS}(A) \to \mathsf{LTS}(B)$ between preordered classes.

▶ **Proposition 3.11** (Monotonicity, 🐓). *For every action code $\mathcal{R} \in \mathsf{Code}(A, B)$, whenever $\mathcal{M} \sqsubseteq \mathcal{N}$ for $\mathcal{M}, \mathcal{N} \in \mathsf{LTS}(A)$, then $\alpha_{\mathcal{R}}(\mathcal{M}) \sqsubseteq \alpha_{\mathcal{R}}(\mathcal{N})$ in $\mathsf{LTS}(B)$.*

4 Refinements

Now that we have introduced the contraction $\alpha_{\mathcal{R}}$ of an LTS for a code $\mathcal{R}$, it is natural to consider an operation in the other direction, which we call the *refinement* $\varrho_{\mathcal{R}}$. Intuitively, refinement replaces each abstract transition $q \xrightarrow{b} q'$ by a sequence of concrete transitions, as prescribed by $\mathcal{R}$.

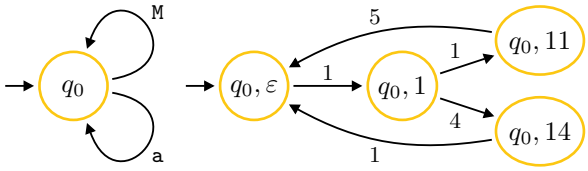


Figure 6 LTS and its refinement w.r.t $\mathcal{R}$ of Figure 3.

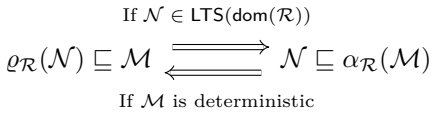


Figure 7 Theorem 4.5.

▶ **Definition 4.1** (Refinement, 🐓). *For each action code $\mathcal{R} \in \mathsf{Code}(A, B)$, we define the* refinement *operator $\varrho_{\mathcal{R}} \colon \mathsf{LTS}(B) \to \mathsf{LTS}(A)$ as follows. For $\mathcal{M} \in \mathsf{LTS}(B)$, the LTS $\varrho_{\mathcal{R}}(\mathcal{M}) \in \mathsf{LTS}(A)$ has a set of states*

$$Q^{\varrho(\mathcal{M})} := \{(q, w) \in Q^{\mathcal{M}} \times A^* \mid w = \varepsilon \text{ or (there is } b \text{ with } q \xrightarrow{b}_{\mathcal{M}} \text{ and } w \lneqq \mathcal{R}(b))\}$$

and the initial state $(q_0^{\mathcal{M}}, \varepsilon)$. The transition relation $\longrightarrow_{\varrho(\mathcal{M})}$ is defined by the following rules:

$$\frac{(q, wa) \in Q^{\varrho(\mathcal{M})}}{(q, w) \xrightarrow{a}_{\varrho(\mathcal{M})} (q, wa)} \ (1_\varrho) \qquad \frac{q \xrightarrow{b}_{\mathcal{M}} q' \quad wa = \mathcal{R}(b)}{(q, w) \xrightarrow{a}_{\varrho(\mathcal{M})} (q', \varepsilon)} \ (2_\varrho)$$

Intuitively, whenever $\varrho(\mathcal{M})$ is in state (q, w), then this corresponds to being in state q in the abstract automaton $\mathcal{M} \in \mathsf{LTS}(B)$ and having observed the actions $w \in A^*$ so far. However, we have insufficiently many actions for finding an abstract transition $q \xrightarrow{b}_{\mathcal{M}} q'$ with $w = \mathcal{R}(b)$ because w is still to short. Nevertheless, whenever $\varrho(\mathcal{M})$ admits a transition to a state (q, w) with $w \neq \varepsilon$, then we know that we can eventually complete w to a sequence corresponding to an abstract transition: there exist at least one $q \xrightarrow{b}_{\mathcal{M}} q'$ for some $b \in \mathsf{dom}(\mathcal{R})$ with $w \leq \mathcal{R}(b)$. If the abstract system $\mathcal{M}$ is non-deterministic, then there may be multiple abstract transitions that match in the final rule (2_ϱ), but the transitions produced by rule (1_ϱ) are deterministic.

▶ **Example 4.2.** Figure 6 shows an example application of a refinement operator that replaces the actions of the LTS $\mathcal{M}$ on the left by their ASCII encoding in octal format, as prescribed by the action code from Figure 3. The initial state is (q_0, ε), corresponding to q_0 in $\mathcal{M}$. Since $\mathcal{M}$ contains abstract labels `M` and `a`, with $\mathcal{R}(\mathtt{M}) = 1\,1\,5$ and $\mathcal{R}(\mathtt{a}) = 1\,4\,1$, we need to introduce additional states for having read 1, 1 1, and 1 4, because those are the sequences of A-actions before we have observed a sequence $\mathcal{R}(b) \in A^+$ for some $b \in B$.

A more visual explanation of $\varrho_{\mathcal{R}}(\mathcal{M})$ is the following: for every state $q \in Q^{\mathcal{M}}$, we consider the outgoing transitions $\{q \xrightarrow{b}_{\mathcal{M}} q' \mid b \in B, q' \in Q^{\mathcal{M}}\}$ and labels $B' \subseteq B$ that appear in it. Then, this outgoing-transition structure is replaced with (a copy of) the minimal subgraph of the tree $\mathcal{R}$ containing all leaves with labels in B'.

Like contraction, the refinement operation also preserves the simulation preorder.

▶ **Proposition 4.3** (Monotonicity, 🐓). *For all action codes $\mathcal{R} \in \mathsf{Code}(A, B)$, if $\mathcal{M} \sqsubseteq \mathcal{N}$ in $\mathsf{LTS}(B)$, then $\varrho_{\mathcal{R}}(\mathcal{M}) \sqsubseteq \varrho_{\mathcal{R}}(\mathcal{N})$ in $\mathsf{LTS}(A)$.*

As $\mathcal{R}$ is deterministic, applying $\varrho_{\mathcal{R}}$ on a deterministic LTS results in a deterministic LTS:

▶ **Proposition 4.4** (Refinement preserves determinism, 🐓). *For every action code $\mathcal{R} \in \mathsf{Code}(A, B)$, if $\mathcal{M} \in \mathsf{LTS}(B)$ is deterministic, then $\varrho_{\mathcal{R}}(\mathcal{M}) \in \mathsf{LTS}(A)$ is deterministic, too.*

▶ **Theorem 4.5** (Galois connection, 🐓). *For $\mathcal{R} \in \mathsf{Code}(A, B)$, $\mathcal{N} \in \mathsf{LTS}(B)$, and $\mathcal{M} \in \mathsf{LTS}(A)$:*
1. *If $\mathcal{N}$ is in the subclass $\mathsf{LTS}(\mathsf{dom}(\mathcal{R})) \subseteq \mathsf{LTS}(B)$, then $\varrho_{\mathcal{R}}(\mathcal{N}) \sqsubseteq \mathcal{M}$ implies $\mathcal{N} \sqsubseteq \alpha_{\mathcal{R}}(\mathcal{M})$.*
2. *If $\mathcal{M}$ is deterministic, then $\mathcal{N} \sqsubseteq \alpha_{\mathcal{R}}(\mathcal{M})$ implies $\varrho_{\mathcal{R}}(\mathcal{N}) \sqsubseteq \mathcal{M}$.*

The condition in the first direction means that $\mathcal{N} \in \mathsf{LTS}(B)$ only makes use of action labels in the subset $\mathsf{dom}(\mathcal{R}) \subseteq B$. Hence, in the proof, we can consider $\mathcal{R}$ to be a total map $\mathsf{dom}(\mathcal{R}) \to A^+$.

▶ Remark 4.6. If we wanted to support non-deterministic $\mathcal{M}$, we can consider a less-pleasant $\varrho'_{\mathcal{R}}$ that replaces every $q \xrightarrow{b} q'$ for $\mathcal{R}(a_1 \cdots a_n) = b$ with literally a sequence $q \xrightarrow{a_1} \cdots \xrightarrow{a_n} q'$. Thus, $\varrho'_{\mathcal{R}}$ would rather create a system as in Figure 1b whereas $\varrho_{\mathcal{R}}$ creates a system as in Figure 1c. However, such an operator $\varrho'_{\mathcal{R}}$ does not preserve determinism.

▶ Remark 4.7. In the proof of the Galois connection, we make use of the fact that our action codes are functional, i.e. that every $b \in B$ is encoded by at most one $w \in A^*$. We would allow multiple, then one can show that α can not have a left-adjoint (details in appendix).

In the first direction, we can even prove a stronger statement for $\mathcal{M} := \varrho_{\mathcal{R}}(\mathcal{N})$, showing a Galois insertion between $\alpha_{\mathcal{R}}$ and $\varrho_{\mathcal{R}}$:

▶ **Theorem 4.8** (Galois insertion, 🐓). *For $\mathcal{R} \in \mathsf{Code}(A, B)$, if $\mathcal{N} \in \mathsf{LTS}(B)$ is in the subclass $\mathcal{N} \in \mathsf{LTS}(\mathsf{dom}(\mathcal{R}))$, then $\mathcal{N} \cong \alpha_{\mathcal{R}}(\varrho_{\mathcal{R}}(\mathcal{N}))$.*

5 Concretizations

In this section, we consider another method of transforming an abstract system into a concrete one: the *concretization* operator. Whereas refinement is the left adjoint of contraction (Theorem 4.5), this section will establish that concretization is the right adjoint (Theorem 5.5) of contraction. Whereas for refinement we omitted transitions for which the action code $\mathcal{R}$ was not defined, for concretization we add transitions to a new *chaos* state [20] in which any action may occur. Essentially, this is the idea of *demonic completion* of [6]. In order to reduce the number of transitions to the chaos state, the concretization operator is parametric in a reflexive relation $\mathcal{I} \subseteq A \times A$ which describes whether two symbols are sufficiently similar. With this relation, we allow transitions to the chaos state only for those symbols that are not similar to any symbol for which the code is defined:

▶ **Definition 5.1** (Concretization, 🐓). *Let $\mathcal{M} \in \mathsf{LTS}(B)$ be an LTS, $\mathcal{R} \in \mathsf{Code}(A, B)$ an action code, and $\mathcal{I} \subseteq A \times A$ a reflexive relation. The* concretization $\gamma_{\mathcal{R},\mathcal{I}}(\mathcal{M}) \in \mathsf{LTS}(A)$ *consists of:*

- $Q^{\gamma(\mathcal{M})} := Q^{\mathcal{M}} \times W \cup \{\chi\}$ *with* $W := \{w \in A^* \mid w = \varepsilon \text{ or } \exists b \in \mathsf{dom}(\mathcal{R}) \colon w \lneq \mathcal{R}(b)\}$.
- $q_0^{\gamma(\mathcal{M})} := (q_0^{\mathcal{M}}, \varepsilon)$
- *Transitions are defined by the following rules, for $a \in A$, $w \in A^*$, $b \in B$:*

$$\frac{wa \in W}{(q,w) \xrightarrow{a}_{\gamma(\mathcal{M})} (q,wa)} \ (1_\gamma) \qquad \frac{q \xrightarrow{b}_{\mathcal{M}} q', \quad \mathcal{R}(b) = wa}{(q,w) \xrightarrow{a}_{\gamma(\mathcal{M})} (q',\varepsilon)} \ (2_\gamma)$$

$$\frac{\forall a' \in A, (a,a') \in \mathcal{I}\colon \quad wa' \notin W \wedge wa' \notin \mathsf{im}(\mathcal{R})}{(q,w) \xrightarrow{a}_{\gamma(\mathcal{M})} \chi} \ (3_\gamma) \qquad \frac{}{\chi \xrightarrow{a}_{\gamma(\mathcal{M})} \chi} \ (4_\gamma)$$

Intuitively, W represents the internal nodes of the tree-representation of action code $\mathcal{R}$. The transitions then try to accumulate a word $w \in A^*$ known to the action code (rule (1_γ)). As soon as we reach $w = \mathcal{R}(b)$ for some b, we use a b-transition in the original $\mathcal{M} \in \mathsf{LTS}(B)$ to jump to a new state (rule (2_γ)). The chaos state χ attracts all runs with symbols unknown to the action code. The corresponding rule (3_γ) involves the relation $\mathcal{I} \subseteq A \times A$. The rule only allows a transition to χ for a symbol $a \in A$ if there is no related symbol $a' \in A$, $(a,a') \in A$ for which the code R could make a transition. For general LTSs, we can simply consider $\mathcal{I}$ to be the identity relation on A. Once transitioned to the chaos state χ, we allow transitions for arbitrary action symbols $a \in A$ (rule (4_γ)).

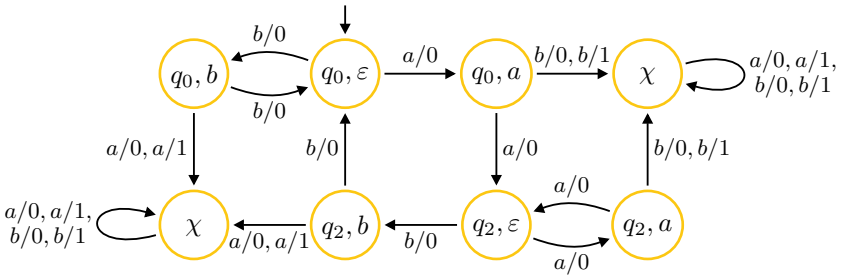


Figure 8 Concretization of the Mealy machine of Figure 5a.

▶ **Example 5.2.** For the special case of Mealy machines $A := I \times O$, we can define $\mathcal{I} \subseteq (I \times O) \times (I \times O)$ to relate (i, o) and (i', o') iff $i = i'$, i.e. two actions are related if they use the same input symbol. Then, we only have transitions to the chaos states if the code can't do any action for the same input symbol $i \in I$. Figure 8 depicts the concretization (for this $\mathcal{I}$) of the Mealy machine of Figure 5a(right) with the action code of Figure 5a(left). To increase readability, we introduced two copies of chaos state χ. Also, multiple labels next to an arrow denote multiple transitions.

Like in the refinement operator, the transition structure of γ is built in such a way that transitions for $b \in B$ in $\mathcal{M}$ correspond to runs of $\mathcal{R}(b)$ in $\gamma(\mathcal{M})$:

$$(q, \varepsilon) \xrightarrow{\mathcal{R}(b)}_{\gamma(\mathcal{M})} \bar{q} \quad \text{iff} \quad \exists q' \colon q \xrightarrow{b}_{\mathcal{M}} q' \text{ and } \bar{q} = (q', \varepsilon).$$

To make γ right adjoint to α, all runs outside the code $\mathcal{R}$ lead to the chaos state. One may think that the many transitions to the chaos state χ would make the construction $\gamma_{\mathcal{R}}$ trivial. However, only those paths lead to χ for which the action code is not defined.

The following technical condition describes that a code $\mathcal{R}$ contains sufficiently many related symbols compared to a given $\mathcal{M} \in \mathsf{LTS}(A)$:

▶ **Definition 5.3** (🐓). *A code $\mathcal{R} \in \mathsf{Code}(A, B)$ is called $\mathcal{I}$-complete for $\mathcal{M} \in \mathsf{LTS}(A)$, if for all $w \in B^*$, $u \in A^*$, $q \in Q^{\mathcal{M}}$, $a, a' \in A$:*

$$r_0 \xrightarrow{u\,a}_{\mathcal{R}} \quad \text{and} \quad (a, a') \in \mathcal{I} \quad \text{and} \quad q_0 \xrightarrow{\mathcal{R}^*(w)\ u}_{\mathcal{M}} q \xrightarrow{a'} \quad \text{implies} \quad r_0 \xrightarrow{u\,a'}_{\mathcal{R}}.$$

Intuitively, $\mathcal{I}$-completeness means that if a state $q \in \mathcal{M}$ can do a transition for $a' \in A$ which is related to similar symbol $a \in A$ defined in the action code, then $a' \in A$ itself is also defined in the action code. However, we do not compare arbitrary transitions of q in $\mathcal{M}$ with arbitrary symbols mentioned in $\mathcal{R}$, but only look at the node in $\mathcal{R}$ reached when 'executing $\mathcal{R}$' zero or more times while following the path $q_0 \Longrightarrow q$.

For example, if $\mathcal{I} \subseteq A \times A$ happens to be the identity relation, then $\mathcal{R}$ is $\mathcal{I}$-complete for any $\mathcal{M} \in \mathsf{LTS}(A)$. In the instance of $\mathcal{I} \subseteq (I \times O) \times (I \times O)$ for Mealy machines, if $\mathcal{R}$ is $\mathcal{I}$-complete for $\mathcal{M}$, then this means: whenever a state $q \in Q^{\mathcal{M}}$ has transitions $q \xrightarrow{i/o}$ and $q \xrightarrow{i/o'}$, then the code $\mathcal{R}$ is defined for either both or none of them.

▶ **Assumption 5.4.** For the rest of the present Section 5, we fix the sets A, B, an action code $\mathcal{R} \in \mathsf{Code}(A, B)$, and a reflexive relation $\mathcal{I} \subseteq A \times A$.

▶ **Theorem 5.5** (Galois connection, 🐓). *For all $\mathcal{N} \in \mathsf{LTS}(A)$, and $\mathcal{M} \in \mathsf{LTS}(B)$, such that $\mathcal{R}$ is $\mathcal{I}$-complete for $\mathcal{N}$, we have*

$$\alpha_{\mathcal{R}}(\mathcal{N}) \sqsubseteq \mathcal{M} \ (\text{in } \mathsf{LTS}(B)) \qquad \Longleftrightarrow \qquad \mathcal{N} \sqsubseteq \gamma_{\mathcal{R},\mathcal{I}}(\mathcal{M}) \ (\text{in } \mathsf{LTS}(A)).$$

▶ **Example 5.6** (🐓). If we instantiate $\mathcal{I}$ to be the identity relation Δ on A, then this means that we simply replace a' with a in rule (3_γ), and then we have above equivalence for all $\mathcal{N} \in \mathsf{LTS}(A)$ and $\mathcal{M} \in \mathsf{LTS}(B)$ (without any side-condition):

$$\alpha_{\mathcal{R}}(\mathcal{N}) \sqsubseteq \mathcal{M} \text{ (in } \mathsf{LTS}(B)) \qquad \Longleftrightarrow \qquad \mathcal{N} \sqsubseteq \gamma_{\mathcal{R},\Delta}(\mathcal{M}) \text{ (in } \mathsf{LTS}(A)).$$

▶ **Example 5.7** (🐓). Consider the instantiation of $\mathcal{I}$ for Mealy machines described in Example 5.2. Let $\mathcal{N}$ be our running example of Figure 2, let $\mathcal{R}$ be the action code from Figure 5a(left), and let $\mathcal{M}$ be the abstract Mealy machine from Figure 5a(right), i.e. $\alpha_{\mathcal{R}}(\mathcal{N}) = \mathcal{M}$. One can verify that $\mathcal{R}$ is $\mathcal{I}$-complete for $\mathcal{N}$. Therefore, application of the Galois connection gives that there is a simulation from $\mathcal{N}$ to the Mealy machine $\gamma_{\mathcal{R},\mathcal{I}}(\mathcal{M})$ of Figure 8.

It is a standard proof that the operators in a Galois connections are monotone. In that proof, one applies the Galois connection also to $\mathcal{M} := \gamma_{\mathcal{R},\mathcal{I}}(\mathcal{N})$, so we first need to show that it satisfies the technical completeness condition:

▶ **Lemma 5.8** (🐓). *$\mathcal{R}$ is always $\mathcal{I}$-complete for $\gamma_{\mathcal{R},\mathcal{I}}(\mathcal{M})$.*

▶ **Corollary 5.9** (🐓). *$\mathcal{M} \sqsubseteq \mathcal{N}$ in $\mathsf{LTS}(B)$ implies $\gamma_{\mathcal{R},\mathcal{I}}(\mathcal{M}) \sqsubseteq \gamma_{\mathcal{R},\mathcal{I}}(\mathcal{N})$ in $\mathsf{LTS}(A)$.*

▶ Remark 5.10. Monotonicity of concretization also follows by observing that the rules in Definition 5.1 all fit the *tyft* format of [19] if we view $(\cdot, w)$ as a unary operator for each sequence $w \in W$. Monotonicity then follows from the result of [19] that the simulation preorder is a congruence for any operator defined using the *tyft* format. Since contraction also can be defined using the *tyft* format, also monotonicity of contraction (Proposition 3.11) follows from the result of [19].

Like refinement, concretization preserves determinism.

▶ **Proposition 5.11** (🐓). *If $\mathcal{M} \in \mathsf{LTS}(B)$ is a deterministic LTS and Δ the identity relation on A, then $\gamma_{\mathcal{R},\Delta}(\mathcal{M})$ is deterministic, too.*

If the code $\mathcal{R} \in \mathsf{Code}(A, B)$ is defined for all labels mentioned in $\mathcal{M} \in \mathsf{LTS}(B)$, then $\gamma_{\mathcal{R}}$ is even the right inverse of $\alpha_{\mathcal{R}}$, that is, we have a Galois insertion:

▶ **Theorem 5.12** (Galois insertion, 🐓). *If $\mathcal{M} \in \mathsf{LTS}(\mathsf{dom}(\mathcal{R}))$, then $\mathcal{M} \cong \alpha_{\mathcal{R}}(\gamma_{\mathcal{R},\mathcal{I}}(\mathcal{M}))$.*

Note that $\mathsf{dom}(\mathcal{R}) \subseteq B$, and so $\mathsf{LTS}(\mathsf{dom}(\mathcal{R})) \subseteq \mathsf{LTS}(B)$. Since we may reach the chaos state χ in the concretization, it is clear that $\gamma_{\mathcal{R}}$ is not a left inverse of $\alpha_{\mathcal{R}}$ in general.

6 Action Code Composition

Since notions of abstraction can be stacked up, it is natural to consider multiple adaptors for multiple action codes. Assume an action code $\mathcal{R} \in \mathsf{Code}(A, B)$ and an action code $\mathcal{S} \in \mathsf{Code}(B, C)$. Then the composition of $\mathcal{R}$ and $\mathcal{S}$ should be an action code from A to C.

▶ **Definition 6.1** (🐓). *Given two map-based action codes $\mathcal{R}\colon B \rightharpoonup A^+$ and $\mathcal{S}\colon C \rightharpoonup B^+$, we define their* (Kleisli) composition $(\mathcal{R} * \mathcal{S})\colon C \rightharpoonup A^+$ *by*

$$(\mathcal{R} * \mathcal{S})(c) = \begin{cases} \mathcal{R}(b_1) \cdots \mathcal{R}(b_n) & \text{if } \mathcal{S}(c) = b_1 \cdots b_n \text{ with } \forall i\colon b_i \in \mathsf{dom}(\mathcal{R}) \\ \text{undefined} & \text{otherwise} \end{cases}$$

The composed action code $\mathcal{R} * \mathcal{S}$ is only defined for $c \in C$ if $\mathcal{S}$ is defined for c and additionally $\mathcal{R}$ is defined for every letter $b_i \in B$ that appears in the word $\mathcal{S}(c) \in B^+$.

▶ Remark 6.2. The defined composition is an instance of *Kleisli composition* for a monad, which is a standard concept in functional programming and category theory.

▶ **Lemma 6.3** (🐓). *Action codes are closed under composition.*

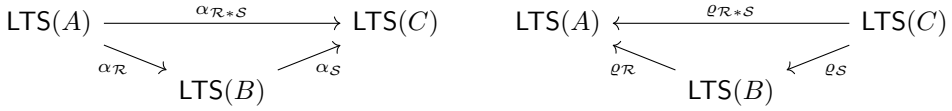
Concretely, given two map-based action codes $\mathcal{R}\colon B \rightharpoonup A^+$ *and* $\mathcal{S}\colon C \rightharpoonup B^+$*, their Kleisli composition* $(\mathcal{R} * \mathcal{S})\colon C \rightharpoonup A^+$ *is again a* prefix-free *partial map.*

Now that we can compose action codes, we can now investigate how the previously defined operators on LTSs behave for composed action codes:

▶ **Theorem 6.4** (🐓, 🐓). *Contraction and refinement commute with action code composition: for action codes* $\mathcal{R} \in \mathsf{Code}(A, B)$, $\mathcal{S} \in \mathsf{Code}(B, C)$,

1. $\alpha_{\mathcal{R}*\mathcal{S}}(\mathcal{M}) = \alpha_{\mathcal{S}}(\alpha_{\mathcal{R}}(\mathcal{M}))$ *for all* $\mathcal{M} \in \mathsf{LTS}(A)$.
2. $\varrho_{\mathcal{R}*\mathcal{S}}(\mathcal{M}) = \varrho_{\mathcal{R}}(\varrho_{\mathcal{S}}(\mathcal{M}))$, *whenever* $\mathsf{im}(\mathcal{S}) \subseteq \mathsf{dom}(\mathcal{R})^+$ *and for all* $\mathcal{M} \in \mathsf{LTS}(C)$.

For the case of refinement, the additional assumption expresses that every word produced by $\mathcal{S}$ only contains letters $b \in B$ for which $\mathcal{R}$ is defined. The equations of Theorem 6.4 equivalently mean that the following diagrams commute:



▶ Remark 6.5 (🐓). Concretization does not commute with action code composition. The reason for that is that the rules (1_γ) and (2_γ) in $\gamma_{\mathcal{R}}(\gamma_{\mathcal{S}}(\mathcal{M}))$ would also be applied to transitions for the chaos state in $\gamma_{\mathcal{S}}(\mathcal{M}) \in \mathsf{LTS}(B)$ (see appendix for details).

7 Adaptors

In this section, we describe how action codes may be used for learning and testing of black-box systems. The general architecture is shown in Figure 9. On the right we see the *system under test (SUT)*, some piece of hardware/software whose behavior can be modeled by a Mealy machine $\mathcal{M}$ with inputs I and outputs O. On the left we see the *learner/tester*, an

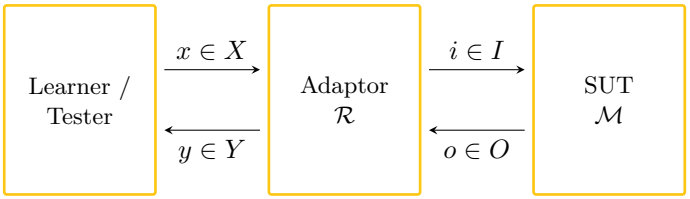


Figure 9 Using action codes for learning/testing.

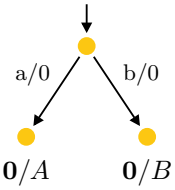


Figure 10 Example 7.4.

agent which either tries to construct a model $\mathcal{N}$ of $\mathcal{M}$ by performing experiments, or already has such a model $\mathcal{N}$ and performs experiments (tests) to find a counterexample which shows that $\mathcal{M}$ and $\mathcal{N}$ behave differently. The learner/tester uses abstract inputs X and outputs Y. In between the learner/tester and the SUT we place an *adaptor*, which uses action code $\mathcal{R}$ to translate between the abstract world of the learner/tester and the concrete world of the SUT. In order to enable the adaptor to do its job, we need to make four (reasonable) assumptions.

Our first assumption, common in model-based testing [35], is that the SUT will accept any input from I in any state, that is, we require that $\mathcal{M}$ is *input enabled*: for all $q \in Q^{\mathcal{M}}$ and $i \in I$, $q \xrightarrow{i/}_{\mathcal{M}}$. Our second assumption is that code $\mathcal{R}$ is $\mathcal{I}$-complete for $\mathcal{M}$ (for $\mathcal{I}$

denoting *same input* as in Example 5.2). This ensures that whenever the adaptor sends a concrete symbol $i \in I$ to the SUT, the adaptor will accept any output $o \in O$ that the SUT may possibly produce in response. Our third assumption is that whenever the adaptor receives an abstract input $x \in X$ from the learner/tester, it can choose concrete inputs from I that drive $\mathcal{R}$ from its initial state to a leaf with label (x, y), for some $y \in Y$. Output y can then be returned as a response to the learner/tester. Reaching such a leaf is nontrivial since the transitions taken in $\mathcal{R}$ are also determined by the outputs provided by the SUT. We may think of the situation in terms of a 2-player game where the adaptor wins if the game ends in an x-leaf, and the SUT wins otherwise. Formally, we require that $\mathcal{R}$ has finitely many states and a winning strategy for every input $x \in X$, as defined below:

▶ **Definition 7.1** (Winning). *Let $\mathcal{R} = \langle R, r_0, \longrightarrow, l\rangle \in \mathsf{Code}(I \times O, X \times Y)$ be an action code with R finite and let $x \in X$. Then*

1. *A leaf $r \in R$ is* winning for x *if $\pi_1(l(r)) = x$.*[1]
2. *An internal state $r \in R$ is* winning for x with input $i \in I$ *if $r \xrightarrow{i/}$ and, for each transition of the form $r \xrightarrow{i/o} r'$, r' is winning for x.*
3. *An internal state $r \in R$ is* winning for x *if it is winning for x with some $i \in I$.*
4. *$\mathcal{R}$ has a* winning strategy for x *if r_0 is winning for x.*

▶ **Example 7.2.** The action codes for Mealy machines that we have seen thus far (Figures 4, 5a and 5b) are winning for all the inputs that label their leaves. The action code of Figure 4 is not winning for the input  (latte macchiato), for the simple reason that this input does not label any leaf. If we remove the transition to the leaf  /2 in Figure 4, then the resulting code is no longer winning for  (espresso), although it is winning for  (coffee).

Our fourth and final assumption is that action code $\mathcal{R}$ is *determinate*. If an action code is determinate then, for each state r and abstract input x, there is at most one concrete input i such that r is winning for x with i.

▶ **Definition 7.3** (Determinate, ). *An action code $\mathcal{R}$ is* determinate *if, for each state r, whenever $r \xrightarrow{i_1/} r_1$, $r \xrightarrow{i_2/} r_2$ and from both r_1 and r_2 there is a path to a leaf labeled with input x, then $i_1 = i_2$.*

▶ **Example 7.4.** All action codes for Mealy machines that we have seen thus far (Figures 4, 5a and 5b) are determinate. Figure 10 shows an action code that is not determinate: in the root two different concrete inputs a and b are enabled that lead to leaves with the same abstract input $\mathbf{0}$. Hence (trivially), this action code does have a winning strategy for input $\mathbf{0}$.

Algorithm 1 shows pseudocode for an adaptor that implements action code $\mathcal{R}$. During learning/testing, the adaptor records the current state of the action code in a variable r. When an abstract input x arrives, it first sets r to r_0. As long as current state r is internal, the adaptor chooses an input i that is winning for x, and forwards it to the SUT. When the SUT replies with an output o, the adaptor sets r to a state r' with $r \xrightarrow{i/o} r'$. When the new r is internal the adaptor chooses again a winning input, and updates its current state after interacting with the SUT, etc. When the new r is a leaf with label (x, y) then the adaptor returns symbol y to the learner/tester and waits for the next abstract input to arrive.

[1] We use projections functions π_1 and π_2 to denote the first and second element of a pair, respectively. So $\pi_1(x, y) = x$ and $\pi_2(x, y) = y$.

Algorithm 1 Pseudocode for an adaptor that implements action code $\mathcal{R}$.

```
1: while true do
2:     x ← Receive-from-learner()
3:     r ← r_0
4:     while r is internal do                ▷ loop invariant: r is winning for x
5:         i ← unique input such that r is winning for x with i
6:         Send-to-SUT(i)
7:         o ← Receive-from-SUT()
8:         r ← unique state r' such that r --i/o--> r'        ▷ R is I-complete for M
9:     end while
10:    Send-to-learner(π_2(l(r)))
11: end while
```

From the perspective of the learner/tester, the combination of the adaptor and SUT behaves the same as the contraction $\alpha_{\mathcal{R}}(\mathcal{M})$. In the appendix, we will formalize this statement by modeling both the combination of adaptor and SUT, as well as contraction $\alpha_{\mathcal{R}}(\mathcal{M})$ as expressions in the process calculus CCS [30], and then establish the existence of *delay simulations* between these expressions. This implies that both expressions have the same traces if we remove all occurrences of the synchronizations between adaptor and SUT, which are invisible from the perspective of the learner.

▶ **Theorem 7.5.** *Let $\mathcal{M} \in \mathsf{LTS}(I \times O)$ be an input enabled Mealy machine and let $\mathcal{R} \in \mathsf{Code}(I \times O, X \times Y)$ be a finite, determinate action code that has a winning strategy for every input in X and that is output enabled for $\mathcal{M}$. Then the composition of an implementation for $\mathcal{M}$ and an adaptor for $\mathcal{R}$ is delay simulation equivalent to an implementation for $\alpha_{\mathcal{R}}(\mathcal{M})$.*

▶ Remark 7.6. Requiring the existence of a determinate action code with a winning strategy for a Mealy machine is not a severe restriction. Definition 7.1 implicitly describes a bottom-up algorithm (linear in the size of the action code) that checks whether a winning strategy exists. Checking whether an action code is determinate is also easy. A sufficient (but not necessary) condition for an action code to be determinate and have a winning strategy is that when we project the action code to the inputs (with concrete inputs labeling the transitions and abstract inputs as label for the leaves) and merge isomorphic subtrees, then the result is still an action code (defined for all the abstract inputs). This is a natural condition that can also be used for the design of determinate action codes with a winning strategy: we start from an action code for the inputs and recursively add output labels starting from the root. Whenever, for a given input i, different outputs may occur, we make a copy of the subtree after i for each possible output o. Finally, the abstract outputs need to be defined in such a way that the labeling of the leaves remains injective.

Active automata learning algorithms and tools for Mealy machines typically assume that the system under learning is *output deterministic*[2]: the output and target state of a transition are uniquely determined by its source state and input.

▶ **Definition 7.7.** *Mealy machine $\mathcal{M}$ is* output deterministic *if, for each state q and input i,*

$$q \xrightarrow{i/o} r \wedge q \xrightarrow{i/o'} r' \quad \Rightarrow \quad o = o' \wedge r = r'.$$

[2] The notion of deterministic that we use in this article is the standard one for LTSs. In the literature on Mealy machines and FSMs, machines that we call output deterministic are called deterministic, and machines that we call deterministic are called observable.

For action codes that are determinate, contraction preserves output determinism. This property makes it possible to use existing automata learning tools to learn models of an output deterministic SUT composed with a determinate adaptor.

▶ **Proposition 7.8** (🐓). *Suppose $\mathcal{M}$ is a Mealy machine and $\mathcal{R}$ is an action code. If $\mathcal{M}$ is output deterministic and $\mathcal{R}$ is determinate then $\alpha_{\mathcal{R}}(\mathcal{M})$ is output deterministic.*

8 Discussion and Future Work

Via the notion of action codes, we provided a new perspective on the fundamental question how high-level state machine models with abstract actions can be related to low-level models in which these actions are refined by sequences of concrete actions. This perspective may, for instance, help with the systematic design of adaptors during learning and testing, and the subsequent interpretation of obtained results. Our theory allows for action codes (such as in Figure 4) that are adaptive in the sense that outputs which occur in response to inputs at the concrete level may determine the sequence of concrete inputs that refines an abstract input. We are not aware of case studies in which such adaptive codes are used, but believe they may be of practical interest. One may, for instance, consider a scenario in which an abstract action AUTHENTICATE is refined by a protocol in which a user is either asked to authenticate by entering a PIN code, or by providing a fingerprint.

Close to our work are the results of Rensink and Gorrieri [31], who investigate vertical implementation relations to link models at conceptually different levels of abstraction. These relations are indexed by a refinement function that maps abstract actions into concrete processes. Within a setting of a CCS-like language, Rensink & Gorrieri [31] list a number of proof rules that should hold for any vertical implementation relation, and propose *vertical bisimulation* as a candidate vertical implementation relation for which these proof rules hold. In the setting of our paper, we can define two vertical implementation relations $\sqsubseteq_{\gamma}^{\mathcal{R}}$ and $\sqsubseteq_{\varrho}^{\mathcal{R}}$, for any action code $\mathcal{R}$, by

$$\mathcal{M} \sqsubseteq_{\gamma}^{\mathcal{R}} \mathcal{N} \;\Leftrightarrow\; \mathcal{M} \sqsubseteq \gamma_{\mathcal{R}}(\mathcal{N}) \quad \text{and} \quad \mathcal{M} \sqsubseteq_{\varrho}^{\mathcal{R}} \mathcal{N} \;\Leftrightarrow\; \mathcal{M} \sqsubseteq \varrho_{\mathcal{R}}(\mathcal{N}).$$

Then $\sqsubseteq_{\varrho}^{\mathcal{R}} \subseteq \sqsubseteq_{\gamma}^{\mathcal{R}}$ and both relations satisfy all language-independent proof rules of [31]. For instance, we have

$$\frac{\mathcal{M} \sqsubseteq \mathcal{M}' \quad \mathcal{M}' \sqsubseteq_{\gamma}^{\mathcal{R}} \mathcal{N}' \quad \mathcal{N}' \sqsubseteq \mathcal{N}}{\mathcal{M} \sqsubseteq_{\gamma}^{\mathcal{R}} \mathcal{N}}$$

(since $\gamma_{\mathcal{R}}$ is monotone and $\sqsubseteq$ is transitive). With the action code $\mathcal{R}$ of Figure 3, both implementation relations relate the LTSs of Figures 1c and 1a. However, the vertical bisimulation preorder of Rensink and Gorrieri [31] does not relate these LTSs, when using a code that maps a to 1 4 1, and b to 1 4 2. This suggests that bisimulations may not be suitable as vertical implementation relations.

Also close to our work are results of Burton et al. [8, 23], who propose a vertical implementation relation in the context of CSP. Instead of action codes, they use *extraction patterns*, a strict monotonic map $extr : Dom \to B^*$, where Dom is the prefix closure of a set $dom \subseteq A^*$ of concrete action sequences that may be regarded as complete. As a mapping from concrete to abstract sequences of actions, extraction patterns are more general than action codes. However, as extraction mappings are not required to have an inverse, establishing interesting Galois connections in this setting may be difficult. With an extraction pattern

defined in the obvious way, the LTSs of Figures 1c and 1a are related by the implementation relation of [8]. We are not aware of any other vertical implementation relation proposed in the literature that handles our basic interface refinement example correctly. We find it surprising that the fundamental problem of refining inputs actions has not been properly addressed in the literature, except in some work that apparently has not been picked up outside Newcastle-upon-Tyne and Catania.

The action refinement operator $\varrho_{\mathcal{R}}$ that we study is similar to the one proposed by [16, 17]. It improves on the one from [16, 17] by not introducing unnecessary nondeterminism, as illustrated in the example of Figure 1. However, it falls short of the approach of [16, 17] by not considering concurrency. Another difference is that in [16, 17] $\mathcal{R}(b)$ can be an arbitrary system (including choice and parallel composition), whereas in our work it must be a sequence. But then [16, 17] did not have the dual contraction operator $\alpha_{\mathcal{R}}$. It would be very interesting to combine both approaches.

Our theory is orthogonal to the one of Aarts et al. [1], which explores the use of so-called *mappers* to formalize adaptors that abstract the large action alphabets of realistic applications into small sets of actions that can be handled by a learning tool. Aarts et al. [1] also describe the relation between abstract and concrete models using a Galois connection. In practical applications of model learning, it makes sense to construct an adaptor that combines a mapper in the sense of [1] with an action code as introduced in this paper. Fiterău-Broştean et al. [11] describe a small domain specific language to specify mapper components, and from which adaptor software can be generated automatically. It would be interesting to extend this domain specific language so that it may also be used to specify action codes.

We developed our theory for LTSs and Mealy machines, using the simulation preorder as the implementation relation. It would be interesting to transfer our results to other modeling frameworks, such as IOTSs [35] timed automata [3] and Markov Decision Processes, and to other preorders and equivalences in the linear-time branching-time spectrum for LTSs [15] and IOTSs [22]. An obvious direction for future work would be to explore how action codes interact with parallel composition. Here the work of [8, 23] may serve as a basis.

Different action codes lead to different contractions, and thereby to different abstract views of a system, see for instance Figures 5a and 5b. We may try to exploit this fact during learning and testing. For instance, if a system $\mathcal{M}$ is too big for state-of-the-art learning algorithms, we may still succeed to learn partial views using cleverly selected action codes. Using our Galois connections we then could obtain various upper and lower bounds for $\mathcal{M}$. Ideally, such an approach may even succeed to uniquely identify $\mathcal{M}$. In particular, learning algorithms such as $L^{\#}$ [38] that use observation trees as their primary data structure may exploit the use of different action codes, since the refinement operator $\varrho_{\mathcal{R}}$ and contraction operator $\alpha_{\mathcal{R}}$ transform observation trees for abstract actions into observation trees for concrete actions, and vice versa. Maarse [29] quantified the quality of a contraction $\alpha_{\mathcal{R}}(\mathcal{M})$ in terms of the graph-theoretic concept of *eccentricity*. If q and q' are states in an LTS $\mathcal{M}$ then $d(q, q')$ is defined as the number of transitions in the shortest path from q to q' (or ∞ if no such path exists). For any set of states $Q \subseteq Q_{\mathcal{M}}$, the *eccentricity* $\varepsilon(Q)$ is defined as $\max_{q' \in Q_{\mathcal{M}}} \min_{q \in Q} d(q, q')$, that is, the maximal distance one needs to travel to visit a state of $\mathcal{M}$, starting from a state of Q. A good contraction has a small set of states Q and a low eccentricity $\varepsilon(Q)$: it only covers a small subset Q of the states of $\mathcal{M}$, but any state from $\mathcal{M}$ can be reached via a few transitions from a Q-state.

References

1 F. Aarts, B. Jonsson, J. Uijen, and F.W. Vaandrager. Generating models of infinite-state communication protocols using regular inference with abstraction. *Formal Methods in System Design*, 46(1):1–41, 2015. doi:10.1007/s10703-014-0216-x.

2 M. Abadi and L. Lamport. Composing specifications. *ACM Transactions on Programming Languages and Systems*, 1(15):73–132, 1993.

3 G. Behrmann, A. David, K. G. Larsen, J. Håkansson, P. Pettersson, W. Yi, and M. Hendriks. Uppaal 4.0. In *Third International Conference on the Quantitative Evaluation of SysTems (QEST 2006),* 11-14 September 2006, Riverside, CA, USA, pages 125–126. IEEE Computer Society, 2006.

4 J.A. Bergstra, A. Ponse, and S.A. Smolka, editors. *Handbook of Process Algebra.* North-Holland, 2001.

5 J. Berstel and D. Perrin. *Theory of codes.* Academic Press, 1985.

6 M. van der Bijl, A. Rensink, and J. Tretmans. Compositional testing with ioco. In A. Petrenko and A. Ulrich, editors, *Formal Approaches to Software Testing, Third International Workshop on Formal Approaches to Testing of Software, FATES 2003, Montreal, Quebec, Canada, October 6th, 2003*, volume 2931 of *Lecture Notes in Computer Science*, pages 86–100. Springer, 2003. doi:10.1007/978-3-540-24617-6_7.

7 M. van der Bijl, A. Rensink, and J. Tretmans. Action refinement in conformance testing. In F. Khendek and R. Dssouli, editors, *Testing of Communicating Systems, 17th IFIP TC6/WG 6.1 International Conference, TestCom 2005, Montreal, Canada, May 31 - June 2, 2005, Proceedings*, volume 3502 of *Lecture Notes in Computer Science*, pages 81–96. Springer, 2005.

8 J. Burton, M. Koutny, and G. Pappalardo. Implementing communicating processes in the event of interface difference. In *2nd International Conference on Application of Concurrency to System Design (ACSD 2001), 25-30 June 2001, Newcastle upon Tyne, UK*, page 87. IEEE Computer Society, 2001. doi:10.1109/CSD.2001.981767.

9 G. Chalupar, S. Peherstorfer, E. Poll, and J. de Ruiter. Automated reverse engineering using Lego. In *Proceedings 8th USENIX Workshop on Offensive Technologies (WOOT'14),* San Diego, California, Los Alamitos, CA, USA, August 2014. IEEE Computer Society.

10 E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking.* MIT Press, Cambridge, Massachusetts, 1999.

11 P. Fiterău-Broştean, R. Janssen, and F.W. Vaandrager. Combining model learning and model checking to analyze TCP implementations. In S. Chaudhuri and A. Farzan, editors, *Proceedings 28th International Conference on Computer Aided Verification (CAV'16),* Toronto, Ontario, Canada, volume 9780 of *Lecture Notes in Computer Science*, pages 454–471. Springer, 2016. doi:10.1007/978-3-319-41540-6_25.

12 P. Fiterău-Broştean, B. Jonsson, R. Merget, J. de Ruiter, K. Sagonas, and J. Somorovsky. Analysis of DTLS implementations using protocol state fuzzing. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2523–2540. USENIX Association, August 2020.

13 P. Fiterău-Broştean, T. Lenaerts, E. Poll, J. de Ruiter, F. Vaandrager, and P. Verleg. Model learning and model checking of SSH implementations. In *Proceedings of the 24th ACM SIGSOFT International SPIN Symposium on Model Checking of Software*, SPIN 2017, pages 142–151, New York, NY, USA, 2017. ACM. doi:10.1145/3092282.3092289.

14 H. Garavel and F. Lang. Equivalence checking 40 years after: A review of bisimulation tools. In N. Jansen, M. Stoelinga, and P. van den Bos, editors, *A Journey from Process Algebra via Timed Automata to Model Learning*, pages 213–265, Cham, 2022. Springer Nature Switzerland. doi:10.1007/978-3-031-15629-8_13.

15 R.J. van Glabbeek. The linear time – Branching time spectrum I. The semantics of concrete, sequential processes. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*, pages 3–99. North-Holland, 2001.

16 R.J. van Glabbeek and U. Goltz. Equivalence notions for concurrent systems and refinement of actions (extended abstract). In A. Kreczmar and G. Mirkowska, editors, *Mathematical Foundations of Computer Science 1989, MFCS'89, Porabka-Kozubnik, Poland, August 28 - September 1, 1989, Proceedings*, volume 379 of *Lecture Notes in Computer Science*, pages 237–248. Springer, 1989. doi:10.1007/3-540-51486-4_71.

17 R.J. van Glabbeek and U. Goltz. Refinement of actions and equivalence notions for concurrent systems. *Acta Informatica*, 37(4/5):229–327, 2001. doi:10.1007/s002360000041.

18 R. Gorrieri and A. Rensink. Action refinement. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*, pages 1047–1147. North-Holland, 2001.

19 J.F. Groote and F.W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, October 1992.

20 C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall International, Englewood Cliffs, 1985.

21 J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

22 R. Janssen, F.W. Vaandrager, and J. Tretmans. Relating alternating relations for conformance and refinement. In W. Ahrendt and S. Lizeth Tapia Tarifa, editors, *Integrated Formal Methods - 15th International Conference, IFM 2019, Bergen, Norway, December 2-6, 2019, Proceedings*, volume 11918 of *Lecture Notes in Computer Science*, pages 246–264. Springer, 2019. doi:10.1007/978-3-030-34968-4_14.

23 M. Koutny and G. Pappalardo. The ERT model of fault-tolerant computing and its application to a formalisation of coordinated atomic actions. Report 636, Department of Computing Science, University of Newcastle upon Tyne, 1998. URL: http://www.cs.ncl.ac.uk/publications/trs/papers/636.pdf.

24 L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, May 1994.

25 D. Lee and M. Yannakakis. Principles and methods of testing finite state machines — a survey. *Proceedings of the IEEE*, 84(8):1090–1123, 1996.

26 N.A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc., San Fransisco, California, 1996.

27 N.A. Lynch and M.R. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proceedings of the 6^{th} Annual ACM Symposium on Principles of Distributed Computing*, pages 137–151, August 1987. A full version is available as MIT Technical Report MIT/LCS/TR-387.

28 N.A. Lynch and F.W. Vaandrager. Forward and backward simulations, I: Untimed systems. *Information and Computation*, 121(2):214–233, September 1995.

29 T. Maarse. *Active Mealy Machine Learning Using Action Refinements*. Master's thesis, Radboud University, Institute for Computing and Information Sciences, August 2020.

30 R. Milner. *Communication and Concurrency*. Prentice-Hall International, Englewood Cliffs, 1989.

31 A. Rensink and R. Gorrieri. Vertical implementation. *Information and Computation*, 170(1):95–133, 2001. doi:10.1006/inco.2001.2967.

32 J. de Ruiter and E. Poll. Protocol state fuzzing of TLS implementations. In *24th USENIX Security Symposium*, pages 193–206, Washington, D.C., August 2015. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/de-ruiter.

33 C. McMahon Stone, T. Chothia, and J. de Ruiter. Extending automated protocol state learning for the 802.11 4-way handshake. In J. López, J. Zhou, and M. Soriano, editors, *Computer Security - 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I*, volume 11098 of *Lecture Notes in Computer Science*, pages 325–345. Springer, 2018. doi:10.1007/978-3-319-99073-6_16.

34 A.S. Tanenbaum and D. Wetherall. *Computer networks, 5th Edition*. Pearson, 2011. URL: https://www.worldcat.org/oclc/698581231.

35 J. Tretmans. Test generation with inputs, outputs, and repetitive quiescence. *Software–Concepts and Tools*, 17:103–120, 1996.

36 J. Tretmans. Model based testing with labelled transition systems. In R.M. Hierons, J.P. Bowen, and M. Harman, editors, *Formal Methods and Testing, An Outcome of the FORTEST Network, Revised Selected Papers*, volume 4949 of *Lecture Notes in Computer Science*, pages 1–38. Springer, 2008.

37 F.W. Vaandrager. Model learning. *Communications of the ACM*, 60(2):86–95, February 2017. doi:10.1145/2967606.

38 F.W. Vaandrager, B. Garhewal, J. Rot, and T. Wißmann. A new approach for active automata learning based on apartness. In D. Fisman and G. Rosu, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I*, volume 13243 of *Lecture Notes in Computer Science*, pages 223–243. Springer, 2022. doi:10.1007/978-3-030-99524-9_12.

39 P. Verleg. *Inferring SSH state machines using protocol state fuzzing.* Master thesis, Radboud University Nijmegen, February 2016. URL: https://www.ru.nl/publish/pages/769526/z07_patrick_verleg.pdf.

40 M. Yannakakis. Hierarchical state machines. In J. van Leeuwen, O. Watanabe, M. Hagiya, P.D. Mosses, and T. Ito, editors, *Theoretical Computer Science: Exploring New Frontiers of Theoretical Informatics*, pages 315–330, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.