Università degli studi di Udine

Separating sets of strings by finding matching patterns is almost always hard

(Article begins on next page)

10 May 2024

Separating Sets of Strings by Finding Matching Patterns is Almost Always Hard

Giuseppe Lancia

Dipartimento di Matematica e Informatica, University of Udine, Via delle Scienze 206, 33100 Udine, Italy

Luke Mathieson

School of Electrical Engineering and Computer Science, University of Newcastle, Callaghan, NSW 2308, Australia

Pablo Moscato

School of Electrical Engineering and Computer Science, University of Newcastle, Callaghan, NSW 2308, Australia

Abstract

We study the complexity of the problem of searching for a set of patterns that separate two given sets of strings. This problem has applications in a wide variety of areas, most notably in data mining, computational biology, and in understanding the complexity of genetic algorithms. We show that the basic problem of finding a small set of patterns that match one set of strings but do not match any string in a second set is difficult (NP-complete, W[2]-hard when parameterized by the size of the pattern set, and APX-hard). We then perform a detailed parameterized analysis of the problem, separating tractable and intractable variants. In particular we show that parameterizing by the size of pattern set and the number of strings, and the size of the alphabet and the number of strings give FPT results, amongst others.

Keywords: pattern identification, parameterized complexity, computational complexity

1. Introduction

Finding patterns in a collection of data is one of the fundamental problems in data mining, data science, artificial intelligence, bioinformatics and many other areas of both theoretical and applied computer science. Accordingly

there are a large number of formulations of this problem. In this paper we develop a particular formulation, drawn from two central motivations:

1. multiparent recombination in genetic and evolutionary algorithms, and
2. the construction of explanatory patterns in single-nucleotide polymorphisms related to disease.

It should not be construed however that these motivations are limitations on the applicability of the problem we develop and study. As will be seen, the underlying computational problem is a general one that occurs as a fundamental component of many other computational problems.

1.1. The Central Problem

Before expanding upon the motivations, we briefly introduce the core computational problem to provide a semi-formal context and some unifying vocabulary. For full definitions we refer the reader to Section 2. Central to the problem is the notion of *pattern*, a string over an alphabet Σ which has been augmented with a special symbol $*$. A pattern matches a string over Σ if the pattern and the string are the same length and each character of the pattern is the same as the character of the string at that position, or the pattern has an $*$ at that position, *i.e.* $*$ 'matches' any symbol from the alphabet. The fundamental problem is then, given two sets, G and B, of strings over Σ, can we find a set of patterns of size at most k such that every string in G matches one of our patterns, and none of the strings in B match any of our patterns.

1.2. Separating Healthy Patterns from Diseased

A significant portion of bioinformatics and computational medicine efforts are focused on developing diagnostic tools. The identification of explanatory genes, uncovering of biomarkers, metabolic network analysis and protein interaction analysis all have as a key (but not sole) motivation the identification of differential markers of disease and consequently routes to treatment. Consider the following problem as a motivating archetypal example: we have two sets of individuals, healthy and diseased and for each example we are given a string that encodes the single-nucleotide polymorphism (SNPs) states across the two copies of each genome, giving us two sets of strings G and B[1]. A SNP has several alleles of which an individual has two. The individual may

---

[1] Whether healthy is G and diseased is B or vice versa depends on what information we wish the set of patterns to extract.

thus be homozygous in any of the alleles, or heterozygous with any choice of pairs of alleles, giving the underlying alphabet Σ.

It is easy to see that if we can identify patterns of SNPs that separate the healthy from the diseased individuals, we have a source of genetic information that may assist in explaining and treating the disease.

This problem is even more apparent in its computational form when considering a biologically motivated form of computation, *i.e.*, evolutionary algorithms.

1.3. Patterns in Multiparent Recombination

The central mechanism for optimization in Genetic Algorithms (GAs) is the recombination of parent solutions to produce a new child solution which ideally retains the positive aspects of the parents. The mechanism derives from an analogy with sexual reproduction in biological evolution and hence typically combines two existing solutions to produce the offspring. In the optimization setting however, there's no conceptual reason for this restriction. Given that recombination can be viewed as a local move in the search space from one individual solution to another as mediated by a third individual solution, a natural generalization of this is to employ multiple parents in the hope of further refining the components of the solution that promote quality, while producing new solutions that effectively cover the search space.

The central theoretical formalization for describing this process is that of *schemata*[2]. An individual solution in a (simple) GA is described by an array, which we can represent as a string, of length n over a given alphabet Σ. A *schema* is a string of length n over the same alphabet augmented with the special "wild card" character $*$, *i.e.*, a pattern. A schema can then be thought of as representing a portion of the search space. The preservation of desirable shared characteristics of two or more parent individuals can then be viewed as the problem of defining a suitable schema. We can define a set G using the individuals selected as parents for a recombination operation, and, if desired, a set B from any individuals whose characteristics we may wish to avoid. The child individual(s) can then be generated from this schema with the wild cards replaced in whichever manner is chosen. Thus we can use schemata to model the basic operation of genetic recombination operators.

This idea not only models multiparent recombination but also multi-child recombination. When considering simply a set of parents from which we wish to generate a set of children, constructing schemata that are compatible with

---

[2]We use the third declension neuter form of *schema*, as it matches better the Greek roots of the word.

the parents is straightforward. A single schema that is a string of n many $*$ symbols would suffice as a trivial solution and the natural solution where for each position, if all the parents agree on the same symbol, the schema has that symbol and $*$ otherwise also provides a simple solution. However in these cases it is reasonably easy to see that the schemata generated can easily be under-specified, leading to a loss of useful information, rendering the recombination operation ineffective. One solution to this problem is to ask for a small set of schemata that are compatible with the parents, but are incompatible with a set of forbidden strings – akin to the list of forbidden elements in Tabu search. In this paper, we elaborate upon and examine this idea.

Some further complexity issues surrounding multiparent recombination have been examined in [10].

1.4. Our Contribution

In this paper we formalize the problem of finding a small set of patterns that match a set of strings, without matching a set of forbidden strings, as discussed in the introduction and examine its complexity. We call the general form of the problem PATTERN IDENTIFICATION and introduce some useful variants. In most cases this problems turn out to be hard. We naturally then consider the problem from a Parameterized Complexity perspective. The problem has a rich parameter ecology and also provides an interesting example of a non-graph theoretic problem. Unfortunately for many parameterizations the problem turns out to be hard in this setting as well. The natural parameterization by the number of desired schemata is W[2]-hard. Even if we take the length of the strings as the parameter, the problem is para-NP-complete. Table 1 gives a summary of the parameterized results, and some key open problems. It is also inapproximable and for some cases we obtain parameterized inapproximability results as well. The only case for which we are able to obtain fixed-parameter tractability relies on a small number of input strings which have a limited number of symbols which are different from a given "base" symbol.

1.5. Related Work

The identification of patterns describing a set of strings forms a well studied family of problems with a wide series of applications. Although, as best as we can determine, the precise problems we studied here have not yet been considered, a number of interesting related problems are explored in the literature. We present here a selection of some of the more relevant and interesting results, however these can at best form a basis for further exploration by the interested reader.

Parameter	Complexity	Theorem
$k + |\Sigma| + |B|$	W[2]-hard	3.1
$k + |\Sigma| + s + |B|$	W[2]-complete	3.5
$n + d + |B|$	para-NP-complete	3.18
$|\Sigma| + d + r + |B|$	para-NP-complete	3.9
$|\Sigma| + d + s + |B|$	para-NP-complete	3.9
$d + |G| + |B|$	FPT	4.2
$|\Sigma| + n$	FPT	4.1
$k + n$	FPT	4.3
$|G| + n$	FPT	4.4
$k + |\Sigma| + d + r + |B|$	FPT	4.5
$k + |G| + |B|$	Open	
$|\Sigma| + |G| + |B|$	Open	
$k + |\Sigma| + d$	Open	

Table 1: Summary of the parameterized results of the paper. $|\Sigma|$ is the size of the alphabet, n is the length of the strings and patterns, $|G|$ and $|B|$ are the sizes of the two input string sets, k is the number of patterns, r is the maximum number of $*$ symbols in a pattern, s is the maximum number of non-$*$ symbols in a pattern and d is the number of 'non-base' elements in each string. Of course the usual inferences apply: tractable cases remain tractable when expanding the parameter and intractable cases remain intractable when restricting the parameter. We note that a number of cases remain open, of which we include some of the more pertinent here, however given the number of parameters under consideration, we refer the reader to Sections 5.1 and 6 for a proper discussion of the open cases.

One of most immediately similar variants is that where pattern variables are allowed. In contrast to the work here, these variables can act as substrings of arbitrary length. Keans and Pitt [23] give a family of polynomial time algorithms for learning the language generated by a single such pattern with a given number k of pattern variables. Angluin [1] studies the inverse problem of generating a pattern, with a polynomial time algorithm for the case where the pattern contains a single pattern variable being the central result. We note that a central difference here is the repeated use of variables, allowing the same undefined substring to be repeated. The properties of these pattern languages have since been studied in some detail, far beyond the scope of this paper.

Bredereck, Nichterlein and Niedermeier [4] employ a similar, but not identi-

cal, formalism to that employed here, but study the problem of taking a set of strings and a set of patterns and determining whether the set of strings can be altered to match the set of patterns. In their formalism patterns are strings over the set $\{\square, \star\}$. We note in particular though that their definition of matching differs from our definition of compatibility in that a string matches a pattern if and only if the string has the special symbol $\star$ exactly where the pattern does. They show this problem to be NP-hard, but in FPT when parameterized by the combined parameter of the number of patterns and the number of strings. They also present an ILP based implementation and computational results. Bredereck *et al.* [3] examine forming teams, *i.e.*, mapping the set of strings to the set of patterns in a consistent manner. They use a similar basis, excepting that the special $\star$ symbol in a pattern now matches any symbol in a string and that the $\square$ symbol requires homogeneity of the matched strings (*i.e.* the symbol it matches is not specified, but all matching strings must have the same symbol at that point). They give a series of classification results, with the problem mostly being intractable, but in FPT for the number of input strings, the number of *different* input strings and the combined parameter of alphabet size with the length of the strings. Gramm, Guo and Niedermeier [18] study another similar problem, DISTINGUISHING SUBSTRING SELECTION, where the input is two sets of strings ("good" and "bad"), and two integers d_g and d_b with the goal of finding a single string of length L whose Hamming distance from all length L substrings of every "good" string is at least d_g and from at least one length L substring for each "bad" string is at most d_b. An extension of the CLOSEST STRING [19, 24] and CLOSEST SUBSTRING [15] problems, the problem has a ptas [12] but they show that it is W[1]-hard when parameterized by any combination of the parameters d_g, d_b and the number of "good" or "bad" strings. Under sufficient restriction they demonstrate an FPT result, requiring a binary alphabet, a 'dual' parameter $d_g' = L - d_g$ and that d_g' is optimal in the sense that it is the minimum possible value. We note that, in relation to the problems studied here, although the number of $*$ symbols in the patterns provides an upper-bound for the Hamming distance, the Hamming distance for a set of strings may be much lower; consider a set of strings with one position set to 1 and all others to 0 such that for every possible position there is a string with a 1 at that point, then the string (or indeed substring) of all 0 has Hamming distance at most one from each input string, but a single pattern would need to be entirely $*$ symbols to match the entire set. Hermelin and Rozenberg introduce a further variant of the CLOSEST STRING problem [21], the CLOSEST STRING WITH WILDCARDS problem. The input is a set of strings $\{s_i\}$, which may include wildcard characters, and an integer d. The goal is to find a string with hamming distance at most d to each s_i.

The solution is required to have no wildcard characters. The examine a number of parameters: the length n of the input strings, the number m of input strings, d, the number $|\Sigma|$ of characters in the alphabet, and the minimum number k of wildcard characters in any input string. They show that the problem is in FPT (with varying explicit running times) when parameterized by m, $m+n$, $|\Sigma|+k+d$ and $k+d$. They also show that the special case where $d = 1$ can be solved in polynomial time, whereas the problem is NP-hard for every $d \geq 2$.

Bulteau *et al.* [5] also give a survey of the parameterized complexity of a variety of more distantly related string problems, with similar multivariate parameterizations as in other work in this area. They cover, amongst others, CLOSEST STRING, CLOSEST SUBSTRING, LONGEST COMMON SUBSEQUENCE, SHORTEST COMMON SUPERSEQUENCE, SHORTEST COMMON SUPERSTRING, MULTIPLE SEQUENCE ALIGNMENT and MINIMUM COMMON STRING.

Introduced by Cannon and Cowen [6], the CLASS COVER problem is a geometric relative of PATTERN IDENTIFICATION where the input is two sets of points colored red and blue, with the goal of selecting a minimum set of blue points (centers) that "cover" the full set of blue points, in the sense that any blue point is closer to its nearest center than any red point. It is NP-hard with an $O(\log n + 1)$-factor approximation algorithm, bearing a close similarity to DOMINATING SET.

2. Preliminaries and Definitions

We now give the relevant definitions for the complexity analysis that follows. In the reductions we use the well known DOMINATING SET and VERTEX COVER problems. The graphs taken as input for these problems are simple, undirected and unweighted. To assist with notation and indexing, we take the vertex set $V(\mathfrak{G})$ of a graph $\mathfrak{G}$ to be the set $\{1, \ldots, n\}$. The edge set $E(\mathfrak{G})$ is then a set of pairs drawn from $V(\mathfrak{G})$ and we denote the edge between vertices i and j by ij $(= ji)$. The SET COVER and k-FEATURE SET problems are also employed. The problems are defined as follows:

> DOMINATING SET:
>   *Instance:*   A graph $\mathfrak{G}$ and an integer k.
>   *Question:*   Is there a set $V' \subseteq V(\mathfrak{G})$ with $|V'| \leq k$ such that for every $u \in V(\mathfrak{G})$ there exists a $v \in V'$ with $u \in N(v)$?

VERTEX COVER:
  Instance: A graph $\mathfrak{G}$ and an integer k.

  Question: Is there a set $V' \subseteq V(\mathfrak{G})$ with $|V'| \leq k$ such that for every $uv \in E(\mathfrak{G})$ we have $u \in V'$ or $v \in V'$?

SET COVER:
  Instance: A base set U, a set $S \subseteq \mathcal{P}(U)$ and an integer k.

  Question: Is there a set $S' \subseteq S$ with $|S'| \leq k$ such that $\bigcup S' = U$?

k-FEATURE SET:
  Instance: An $n \times m$ 0-1 matrix M, an $n \times 1$ 0-1 vector f and an integer k.

  Question: Is there a set of indices $I \subseteq \{1, \dots, m\}$ with $|I| \leq k$ such that for all a, b where $f_a \neq f_b$ there exist $i \in I$ such that $M_{a,i} \neq M_{b,i}$?

We note the following key classification results:

- DOMINATING SET is NP-complete, $O(\log n)$-APX-hard[3] and W[2]-complete when parameterized by k, the size of the dominating set.

- VERTEX COVER is NP-complete and APX-hard, and remains NP-complete when the input is a planar graph [17].

- SET COVER is W[2]-complete when parameterized by the size of the set cover.

- k-FEATURE SET is W[2]-complete when parameterized by the size of the feature set [9].

We also employ a parameterized version of the MODEL CHECKING problem, which takes as input a finite structure and a logical formula and asks the question of whether the structure is a model of the formula, *i.e.* whether there is a suitable assignment of elements of the universe of the structure to variables of the formula such that the formula evaluates to true under that assignment. The parameter is the length of the logic formula. While

---

[3]That is there exists some $c > 0$ such that DOMINATING SET has no $c \cdot \log n$-factor approximation algorithm unless $P = NP$.

we informally introduce the finite structural elements as needed, we briefly describe here the fragments of first-order logic we employ. Let $\Sigma_0 = \Pi_0$ be the set of unquantified Boolean formulae. The classes Σ_t and Π_t for $t > 0$ can be defined recursively as follows:

$$\Sigma_t = \{\exists x_1 \dots \exists x_k \varphi \mid \varphi \in \Pi_{t-1}\}$$
$$\Pi_t = \{\forall x_1 \dots \forall x_k \varphi \mid \varphi \in \Sigma_{t-1}\}$$

The class $\Sigma_{t,u}$ is the subclass of Σ_t where each quantifier block after the first existential block has length at most u. We note that trivially $\Pi_{t-1} \subset \Sigma_t$. We note that these classes are specified in prenex normal form, and are, in general, not robust against Boolean combinations of formulae. In general, the process of converting a formula to prenex normal form (where all quantifiers are "out the front") increases the number of quantifier alternations. An analog of the Σ classes is $\Sigma_{t,u}^*$. Let $\Theta_{0,u}$ be the set of quantifier free formulae, and $\Theta_{t,u}$ for $t > 0$ be the set of Boolean combinations of formulae where each leading quantifier block is existential and quantifies over a formula in $\Theta_{t-1,u}$, where the length of each quantifier block is at most u. That is, the formulae in $\Theta_{t,u}$ are not required to be in prenex normal form, and Boolean connectives may precede some quantifiers. We can deal with leading universal quantification by the normal expedient of the introduction of a trivial existential block. Then $\Sigma_{t,u}^*$ is the class of formulae of the form $\exists x_1 \dots \exists x_k \varphi$ where $\varphi \in \Theta_{t-1,u}$ and where k may be greater than u.

Thus we refer to the MODEL CHECKING problem as $\mathrm{MC}(\Phi)$ where Φ is the first-order fragment employed. In the parameterized setting, $\mathrm{MC}(\Sigma_{t,u})$ is W[t]-complete for every $u \geq 1$, and $\mathrm{MC}(\Sigma_{t,u}^*)$ is W*[t]-complete for every $u \geq 1$. The W*-hierarchy is the hierarchy analogous to the W-hierarchy obtained from using $\mathrm{MC}(\Sigma_{t,u}^*)$ as the complete problem instead of $\mathrm{MC}(\Sigma_{t,u})$. While it is known that W[1] = W*[1] and W[2] = W*[2]; for $t \geq 3$, the best known containment relationship is W[t] $\subseteq$ W*[t] $\subseteq$ W[2t − 2]. For more detail on these results and the full definitions relating to first-order logic and structures we refer the reader to [16]. The W*-hierarchy, introduced by Downey, Fellows and Taylor [14] but more fully explored later [7, 16] is a parameterized hierarchy which takes into account the Boolean combinations of quantified first-order formulae, but is otherwise similar to the more usual W-hierarchy.

In several of our intractability results we make use of the class para-NP, and a useful corollary due to Flum and Grohe with a detailed explanation in [16] (presented as Corollary 2.16). The class para-NP is the direct parameterized complexity translation of NP, where we replace "polynomial-time" with "fixed-parameter tractable time" (or fpt-time in short) in the definition. Flum and Grohe's result states that if, given a parameterized problem

9

(Π, κ), the classical version of the problem Π is NP-complete for at least one fixed value of κ, then (Π, κ) is para-NP-complete. As may be expected, FPT = para-NP if and only if P = NP, thus para-NP-completeness is strong evidence of intractability. We also make reference to parameterized approximation. A parameterized approximation algorithm is, in essence, a standard approximation algorithm, but where we relax the running time to fpt-time, rather than polynomial-time. We refer to [25] for a full introduction to this area.

The other parameterized complexity theory employed is more standard, thus for general definitions we refer the reader to standard texts [13, 16].

We write $A \leq_{FPT} B$ to denote that there exists a parameterized reduction from problem A to problem B, and similarly $A \leq_P B$ to denote the existence of a polynomial-time many-one reduction from problem A to problem B. We also use *strict polynomial-time reductions* to obtain some approximation results. A strict reduction is one that, given two problems A and B, guarantees that the approximation ratio for A is at least as good as that of B. In the cases we present, we employ them for approximation hardness results, so the precise ratio is not discussed. For a full definition of strict reductions (and other approximation preserving reductions) we refer to [11].

Definition 2.1 (Pattern)**.** A *pattern* is a string over an alphabet Σ and a special symbol $*$.

Given a string $s \in \Sigma^*$ and an integer i, we denote the ith symbol of s by $s[i]$.

Definition 2.2 (Compatible)**.** A pattern p is compatible with a string g, denoted $p \rightarrow g$, if for all i such that $p[i] \neq *$ we have $g[i] = p[i]$. If a pattern and string are not compatible, we write $p \nrightarrow g$. We extend this notation to sets of strings, writing $p \rightarrow G$ to denote $\forall g \in G, p \rightarrow g$ and $P \rightarrow G$ for $\forall g \in G \exists p \in P, p \rightarrow g$.

Definition 2.3 (*G-B*-Separated Sets)**.** A set P of patterns *G-B-separates* an ordered pair (G, B) of sets of strings, written $P \rightarrow (G, B)$ if

- $P \rightarrow G$, and

- for every $b \in B$ and $p \in P$ we have $p \nrightarrow b$.

Thus we can state the central problem for this paper:

> PATTERN IDENTIFICATION:
> *Instance:* A finite alphabet Σ, two disjoint sets $G, B \subseteq \Sigma^n$ of strings and an integer k.
>
> *Question:* Is there a set P of patterns such that $|P| \leq k$ and $P \rightarrow (G, B)$?

The complexity analysis of the problem in the parameterized setting leads to the definition of a second, subsidiary problem which allows a convenient examination of sets of strings which are very similar.

Definition 2.4 (Small String). A string s over an alphabet Σ is *d-small* if, given an identified symbol $\sigma \in \Sigma$, for exactly d values of i, $p[i] \neq \sigma$.
We call σ the *base* symbol.
A set of strings S is d-small if, given a fixed base symbol all strings in S are d-small.

This restriction on the structure of the input gives further insight into the complexity of PATTERN IDENTIFICATION and is key to some of the tractability results in Section 4. For convenience we phrase a restricted version of the PATTERN IDENTIFICATION problem:

> PI WITH SMALL STRINGS:
> *Instance:* An alphabet Σ, two disjoint d-small sets $G, B \subseteq \Sigma^n$, an integer k.
>
> *Question:* Is there a set P of patterns with $|P| \leq k$ such that $P \to (G, B)$?

From the perspective of multiparent recombination, minimizing the number of wildcard symbols in each pattern is also an interesting objective:

> PI WITH LARGE PATTERNS:
> *Instance:* An alphabet Σ, two disjoint sets $G, B \subseteq \Sigma^n$, integers k and r.
>
> *Question:* Is there a set P of patterns with $|P| \leq k$ such that $P \to (G, B)$ and for each $p \in P$ the number of $*$ symbols in p is at most r?

We implicitly define the obvious intersection of the two restricted problems, PI WITH LARGE PATTERNS AND SMALL STRINGS.
From a combinatorial perspective, the inverse problem is also interesting:

> PI WITH SMALL PATTERNS:
> *Instance:* An alphabet Σ, two disjoint sets $G, B \subseteq \Sigma^n$, integers k and s.
>
> *Question:* Is there a set P of patterns with $|P| \leq k$ such that $P \to (G, B)$ and for each $p \in P$ the number of non-$*$ symbols in p is at most s?

3. Hard Cases of the Pattern Identification Problem

We first examine the intractable cases of the PATTERN IDENTIFICATION problem. This narrows down the source of the combinatorial complexity of the problem.

Theorem 3.1. PATTERN IDENTIFICATION *is* W[2]-*hard when parameterized by* k, *even if* $|\Sigma| = 2$ *and* $|B| = 1$.

Lemma 3.2. DOMINATING SET $\leq_{FPT}$ PATTERN IDENTIFICATION.
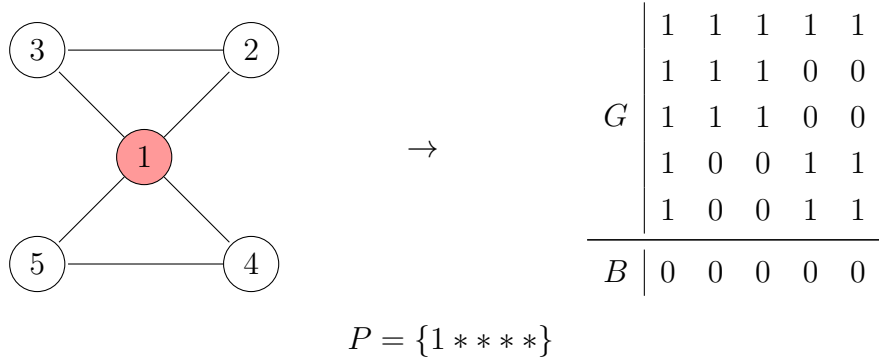

$$P = \{1****\}$$

Figure 1: An example of the reduction used in Lemma 3.2 with $k = 1$. The dominating set is highlighted in red, and the correspond set of patterns (a singleton) is shown.

Proof. Let $(\mathfrak{G}, k)$ be an instance of DOMINATING SET. Let $n = |V(\mathfrak{G})|$ and assume $V(\mathfrak{G}) = \{1, \ldots, n\}$. We construct an instance (Σ, G, B, k) as follows:

1. $\Sigma = \{1, 0\}$,
2. $G = \{g_1, \ldots, g_n\}$ where for each i, $g_i \in \Sigma^n$ where for every j, $g_i[j] = 1$ if $ij \in E(\mathfrak{G})$ or $i = j$ and $g_i[j] = 0$ otherwise,
3. $B = \{0^n\}$.

An example of the reduction is given in Figure 1.

Claim 3.3. If (G, k) is a YES instance of DOMINATING SET then (Σ, G, B, k) is a YES instance of PATTERN IDENTIFICATION.

Let $D \subseteq V(\mathfrak{G})$ with $|D| \leq k$ be a dominating set witnessing that $(\mathfrak{G}, k)$ is a YES instance. We can construct a witness set of patterns P with $|P| = |D|$ such that $P \to (G, B)$. For each $i \in D$, we create a pattern p_i where $p_i[i] = 1$ and $p_i[j] = *$ for all $j \neq i$.

As D is a dominating set, for every vertex $j \in V(\mathfrak{G})$ there is a vertex $i \in D$ such that $ij \in E(\mathfrak{G})$. Then for string $g_j \in G$, pattern p_i is compatible with

g_j as by construction $g_j[i] = 1$. Therefore for every string $g \in G$ there exists $p \in P$ such that $p \to g$.

Moreover there is no $p \in P$ such that $p \to b$ where $b \in B$. As B consists of the single string $b = 0^n$ and for each $p \in P$ exactly one element is neither 0 nor $*$ there is one position where the pattern does not match b.

Thus (Σ, G, B, k) is a YES instance.

Claim 3.4. If (Σ, G, B, k) is a YES instance of PATTERN IDENTIFICATION then $(\mathfrak{G}, k)$ is a YES instance of DOMINATING SET

Let P with $|P| \leq k$ be the witness set of patterns that (Σ, G, B, k) is a YES instance. Inductively, we may assume that every $p \in P$ is compatible with at least one element of G, if not, $P \setminus \{p\}$ constitutes an alternate witness.

First we note that no $p \in P$ can consist of only $*$ and 0 symbols, as this would be compatible with the single element of B. Therefore each $p \in P$ has at least one i such that $p[i] = 1$.

Consider a $p \in P$, and the corresponding set of vertices V_p (i.e. each position i where $p[i] = 1$). Let $g_j \in G$ be a string such that $p \to g_j$. By construction for every $i \in V_p$, $ij \in E(\mathfrak{G})$. Let $V_{p \to g}$ be the set of vertices corresponding to the set $G_p \subseteq G$ where $p \to G_p$. Each vertex $j \in V_{p \to g}$ is adjacent (or identical) to every vertex in V_p. Thus we may select arbitrarily a single vertex from V_p to be in the dominating set D.

Thus we have D with $|D| \leq |P| \leq k$, where every vertex in $V(\mathfrak{G})$ is adjacent (or identical) to some vertex in D. Therefore $(\mathfrak{G}, k)$ is a YES instance.

The construction can be clearly performed in fpt time (in fact, polynomial time), and the lemma follows. $\square$

Proof of Theorem 3.1. The theorem follows immediately from Lemma 3.2. $\square$

The structure of the reduction then gives the following:

Corollary 3.5. PI WITH SMALL PATTERNS *is* W[2]*-complete when parameterized by k and* NP*-complete even when $s = 1$, $|\Sigma| = 2$ and $|B| = 1$.*

Proof. The W[2]-hardness is apparent from the proof of Lemma 3.2 (in fact the restriction would make the proof simpler). To show containment in W[2], we reduce PI WITH SMALL PATTERNS to $MC(\Sigma_{2,1})$. The first-order structure is equipped with four unary relations N, Σ, G and B and a binary function symbol C. Each string is represented by an integer, according to an arbitrary fixed ordering. Gi is true if string i is in G, Bi is true if string i is in B. $\Sigma\sigma$ is true if $\sigma \in \Sigma$ and Ni is true if $i \in \mathbb{N}$. The function $C : \mathbb{N} \times \mathbb{N} \to \Sigma$ is defined $Cij = \sigma$ if σ is the jth symbol of string i.

We now provide the first-order formula expressing PI WITH SMALL PATTERNS:

$$\exists i_{1,1}, \ldots, i_{k,s}, c_{1,1}, \ldots, c_{k,s} \forall j ((\bigwedge_{l \in [k], b \in [s]} N i_{l,b}) \wedge$$

$$(\bigwedge_{l \in [k], b \in [s]} \Sigma c_{l,b}) \wedge$$

$$(Gj \rightarrow (\bigvee_{l \in [k]} (\bigwedge_{b \in [s]} C j i_{l,b} = c_{l,b}))) \wedge$$

$$(Bj \rightarrow (\bigwedge_{l \in [k]} (\bigvee_{b \in [s]} C j i_{l,b} \neq c_{l,b}))))$$

The formula states that a solution to PI WITH SMALL PATTERNS consists of k sets of s symbols along with positions such that for each string in G, for at least one set of symbols, the string is compatible and for each string in B no set of symbols is compatible.

Containment in NP can be demonstrated by the usual polynomial verification approach (indeed in much the same format as the above formula). □

Corollary 3.6. PATTERN IDENTIFICATION *has no constant factor fpt-approximation algorithm unless* FPT = W[2] *and there exists a* $c \geq 0$ *such that* PATTERN IDENTIFICATION *has no* $c \cdot \log n$ *polynomial time approximation algorithm unless* P = NP, *even when* $|\Sigma| = 2$ *and the optimization goal is* $\min k$.

Proof. As DOMINATING SET has no constant factor fpt-approximation [8] unless FPT = W[2] and no $c \cdot \log n$ polynomial time approximation [26] for some $c > 0$ unless P = NP and the reduction of Lemma 3.2 is a strict polynomial-time reduction, the corollary follows. □

Given the construction in the proof of Lemma 3.2, we can deduce that one source of complexity might be the freedom (unboundedness) in the alphabet and the structure of the strings. We demonstrate that restricting these parameters is fruitless from a computational complexity perspective.

Corollary 3.7. PI WITH SMALL STRINGS *is* NP *complete even when* $|\Sigma| = 2$, $d = 4$, $s = 1$ *and* $|B| = 1$.

Proof. As DOMINATING SET is NP-complete on planar graphs of maximum degree 3 [17], the number of 1s in each string in the construction of the proof of Lemma 3.2 is at most 4, where we take the base symbol to be 0. □

This result also demonstrates the following:

Lemma 3.8. PI with Large Patterns and Small Strings *and* PI with Large Patterns *are both* NP-*complete even when* $|\Sigma| = 2$, $d = 4$, $r = 9$ *and* $|B| = 1$.

Proof. Following Corollary 3.7, we can see from the construction given in the proof of Lemma 3.2 that for each $p \in P$, instead of setting $p[i] = *$ for each i not in the dominating set, we can choose r to be nine, and set $p[i] := 1$ if i is in the dominating set, $p[j] = *$ for the at most three values of j such that $ij \in E(\mathfrak{G})$ and the at most six additional values of j at distance two[4] from i, and $p[l] = 0$ for all other $l \in \{1, \ldots, n\}$. For the reverse argument, we have similar conditions as before, at least one symbol of each pattern must be a 1 and at most four can be 1s. With at most nine $*$ symbols, the pattern is compatible with all the strings that the corresponding vertex dominates, and all other symbols in these strings are 0. $\square$

Corollary 3.9. *The following are true:*

1. PI with Large Patterns and Small Strings *is* para-NP-*complete when parameterized by* $|\Sigma| + d + r + |B|$.
2. PI with Large Patterns *is* para-NP-*complete when parameterized by* $|\Sigma| + r + |B|$.
3. PI with Small Patterns and Small Strings *is* para-NP-*complete when parameterized by* $|\Sigma| + d + s + |B|$.
4. PI with Small Patterns *is* para-NP-*complete when parameterized by* $|\Sigma| + s + |B|$.

Proof. The result are obtained as follows:

1. Lemma 3.8 gives NP-completeness with fixed $|\Sigma|$, d, r and $|B|$. With Corollary 2.16 from [16], the result follows.
2. The preservation of hardness when taking subsets of a set of parameters gives the result from 1.
3. Corollary 3.7 shows NP-completeness with fixed $|\Sigma|$, d, s and $|B|$. Corollary 2.16 from [16] completes the result.
4. The result follows immediately from 3.

$\square$

---

[4]As $\mathfrak{G}$ has maximum degree three, each neighbor of i has at most two other neighbors, so the patterns representing each of these neighbors has a 1 in the ith position, a 1 for its own position and two other 1s. Therefore we need only three $*$ symbols for the neighbors themselves, and two more per neighbor for the distance two neighborhood.

We note that DOMINATING SET is in FPT for graphs of bounded degree, so we do not obtain a W[2]-hardness result. However we can tighten this result a little further:

Theorem 3.10. PATTERN IDENTIFICATION *is* NP-*complete and* APX-*hard even when* $\Sigma = \{0, 1\}$ *and all strings have at most two symbols as* 1 *(equiv. at most two symbols as* 0*) and* $|B| = 1$.

Lemma 3.11. VERTEX COVER $\leq_P$ PATTERN IDENTIFICATION.

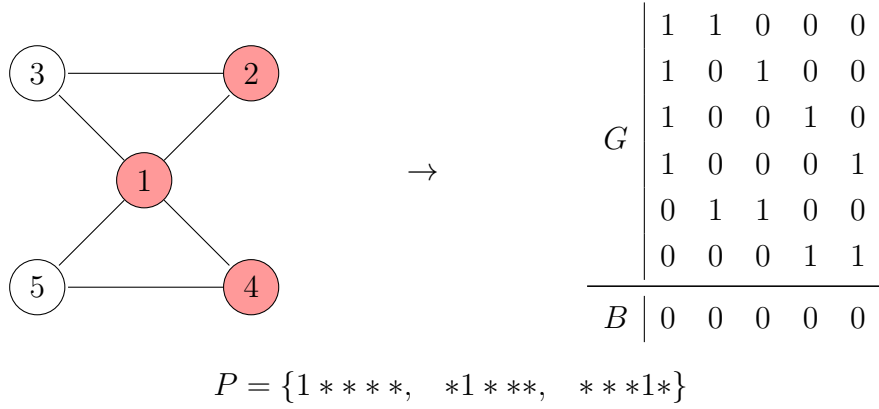$$P = \{1****, \ *1***, \ ***1*\}$$

Figure 2: An example of the reduction used in Lemma 3.11 with $k = 3$. The vertex cover is highlighted in red, and the correspond set of patterns is shown.

Proof. Given an instance $(\mathfrak{G}, k)$ of VERTEX COVER with $V(\mathfrak{G}) = \{1, \ldots, n\}$, we construct an instance (Σ, G, B, k) of PATTERN IDENTIFICATION as follows:

1. $\Sigma = \{0, 1\}$.
2. $G = \{g_{ij} \mid ij \in E(\mathfrak{G})\}$ with $g_{ij} \in \Sigma^n$ where $g_{ij}[i] = g_{ij}[j] = 1$ and $g_{ij}[u] = 0$ for $u \neq i, j$.
3. $B = \{0^n\}$.

Clearly this construction can be performed in polynomial time. The construction is illustrated in Figure 2.

Claim 3.12. If $(\mathfrak{G}, k)$ is a YES instance of VERTEX COVER then (Σ, G, B, k) is a YES instance of PATTERN IDENTIFICATION.

Let $V' \subseteq V(\mathfrak{G})$ where $|V'| \leq k$ be a vertex cover witnessing that $(\mathfrak{G}, k)$ is a YES instance of VERTEX COVER. We construct a set of patterns P with $|P| = |V'|$ that is a solution for (Σ, G, B, k) where for each $i \in V'$ there is a pattern $p_i \in P$ with $p_i[i] = 1$ and $p_i[j] = *$ for $j \neq i$. For each edge

16

$ij \in E(\mathfrak{G})$, either $i \in V'$ or $j \in V'$ (or both). Therefore for the string g_{ij} corresponding to ij, we have either $p_i \in P$ or $p_j \in P$ such that $p_i \to g_{ij}$ or $p_j \to g_{ij}$. Hence $P \to G$. Moreover there is no $p_i \in P$ such that $p_i \to b$ where b is the single element of B as each p_i, by construction, contains a 1, whereas b consists of only 0s. Therefore (Σ, G, B, k) is a YES instance of PATTERN IDENTIFICATION.

Claim 3.13. If (Σ, G, B, k) is a YES instance of PATTERN IDENTIFICATION then $(\mathfrak{G}, k)$ is a YES instance of VERTEX COVER.

Let P with $|P| \leq k$ be the set of patterns witnessing the fact that (Σ, G, B, k) is a YES instance of PATTERN IDENTIFICATION. We may assume without loss of generality that for every $p \in P$, there exists some $g \in G$ such that $p \to g$. Each $p \in P$ must contain at least one 1, otherwise $p \to b$ where b is the single element of B. No $p \in P$ can contain more than two 1s, as there exists $g \in G$ such that $p \to g$, and every such g has exactly two 1s. We note that if a pattern p has two 1s, then there is exactly one $g \in G$ such that $p \to g$.

Let $P_1 \subseteq P$ be the set of patterns with exactly one 1 and $P_2 \subseteq P$ be the set of patterns with exactly two 1s. We have $P_1 \cup P_2 = P$. We construct a vertex cover $V' \subseteq V(\mathfrak{G})$ with $|V'| \leq |P|$ as follows:

1. for each $p \in P_1$ add i to V' where $p[i] = 1$,
2. for each $p \in P_2$ where $p[i] = p[j] = 1$, arbitrarily add one of i or j to V'.

Consider every edge $ij \in E(\mathfrak{G})$, then for the corresponding $g_{ij} \in G$ there exists a $p \in P$ such that $p \to g_{ij}$. As each p has at least one 1, this 1 must be at position i or j (or both). Therefore i or j is in V' (or perhaps both), therefore V' forms a valid vertex cover for $\mathfrak{G}$. $\qed$

Proof of Theorem 3.10. The NP-hardness follows from Lemma 3.11. The containment in NP follows from the usual verification algorithm. The APX-hardness follows as the reduction of Lemma 3.11 is strict and VERTEX COVER is APX-hard [20]. $\qed$

Finally, as restricting the alphabet did not reduce the complexity, we consider the case where the strings themselves are short. Again the problem is hard, but we note that to achieve this reduction we relax the bound on Σ (or in Parameterized Complexity terms, $|\Sigma|$ is no longer a parameter – if $|\Sigma|$ is a parameter, the problem is in FPT).

Theorem 3.14. PI WITH SMALL STRINGS *is* NP-*complete even when* $n = 4$, $d = 4$ *and* $|B| = 1$.

Lemma 3.15. PLANAR VERTEX COVER $\leq_P$ PI WITH SMALL STRINGS *even when the length of strings is restricted to 4.*

$$G \begin{array}{|cccc} \sigma_1 & \sigma_2 & \sigma_6 & \sigma_6 \\ \sigma_1 & \sigma_6 & \sigma_3 & \sigma_6 \\ \sigma_1 & \sigma_4 & \sigma_6 & \sigma_6 \\ \sigma_1 & \sigma_6 & \sigma_5 & \sigma_6 \\ \sigma_6 & \sigma_2 & \sigma_3 & \sigma_6 \\ \sigma_6 & \sigma_4 & \sigma_5 & \sigma_6 \\ \hline B & \sigma_6 & \sigma_6 & \sigma_6 & \sigma_6 \end{array}$$

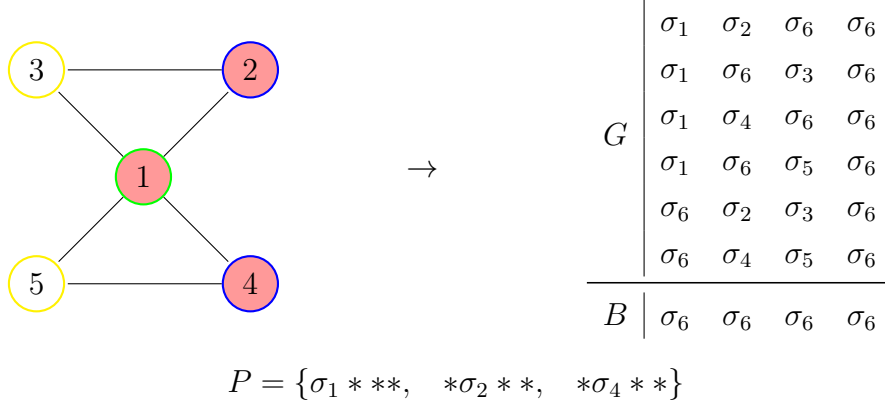$$P = \{\sigma_1 ***, \quad *\sigma_2 **, \quad *\sigma_4 **\}$$

Figure 3: An example of the reduction used in Lemma 3.15 with $k = 3$. The vertex cover is highlighted in red, and the correspond set of patterns is shown. Note the difference with the reduction in Lemma 3.11, here the position encodes the coloring and the symbols encode the edges, whereas previously the string more directly encode the graph.

Proof. Let $(\mathfrak{G}, k)$ be an instance of PLANAR VERTEX COVER. We assume without loss of generality that $V(\mathfrak{G}) = \{1, \ldots, n\}$. As $\mathfrak{G}$ is planar, we can compute a proper 4-coloring in polynomial time [2]. Let $C : V(\mathfrak{G}) \to \{1, 2, 3, 4\}$ be such a coloring. We construct an instance (Σ, G, B, k', d) of PI WITH SMALL STRINGS as follows:

1. $\Sigma = \{\sigma_1, \ldots, \sigma_{n+1}\}$.
2. $G = \{g_{ij} \mid ij \in E(\mathfrak{G})\}$ where for $k \in \{1, \ldots, 4\}$ we set

$$g_{ij}[k] := \begin{cases} \sigma_i & \text{if } C(i) = k \\ \sigma_j & \text{if } C(j) = k \\ \sigma_{n+1} & \text{otherwise.} \end{cases}$$

3. $B = \{\sigma_{n+1}^4\}$.
4. $d = 4$.

We note that as C is a proper coloring, $C(i) \neq C(j)$ for any $ij \in E(\mathfrak{G})$. Moreover for $i \in V(\mathfrak{G})$, σ_i only appears as the $C(i)$th symbol in any string. The construction can clearly be performed in polynomial time. The construction is illustrated in Figure 3.

Claim 3.16. If $(\mathfrak{G}, k)$ is a YES instance of PLANAR VERTEX COVER then (Σ, G, B, k, d) is a YES instance of PI WITH SMALL STRINGS.

18

Let $V' \subseteq V(\mathfrak{G})$ with $|V'| \leq k$ be a vertex cover witnessing that $(\mathfrak{G}, k)$ is a YES instance of PLANAR VERTEX COVER. We construct a set P with $|P| = |V'| \leq k$ of patterns that forms a solution for (Σ, G, B, k, d) in the following manner: for each $i \in V'$, we add the pattern p_i to P where $p_i[C(i)] = \sigma_i$ and all other symbols in p_i are $*$. No pattern in P is compatible with the singleton element of B, as each has a symbol σ_i with $1 \leq i \leq n$. For every edge $ij \in E(\mathfrak{G})$, at least one of i and j is in V'. Without loss of generality assume that $i \in V'$. By construction the string g_{ij} is compatible with the pattern $p_i \in P$, therefore every string in G is compatible with some pattern in P.

Claim 3.17. If (Σ, G, B, k, d) is a YES instance of PI WITH SMALL STRINGS then $(\mathfrak{G}, k)$ is a YES instance of PLANAR VERTEX COVER.

Let P with $|P| \leq k$ be a set of patterns such that $P \to (G, B)$. As before we may assume that P is minimal in the sense that each pattern is compatible with some string in G. Each $p \in P$ must have at least one symbol drawn from the set $\{\sigma_1, \ldots, \sigma_n\}$, otherwise $p \to B$. No pattern $p \in P$ can have more than two symbols from $\{\sigma_1, \ldots, \sigma_n\}$, otherwise $p \not\to G$. As before, we partition P into P_1, the subset of patterns with one symbol from $\{\sigma_1, \ldots, \sigma_n\}$, and P_2, the subset of patterns with two symbols from $\{\sigma_1, \ldots, \sigma_n\}$. We construct a vertex cover $V' \subseteq V(\mathfrak{G})$ for $\mathfrak{G}$ with $|V'| \leq |P| \leq k$ as follows:

- for each $p \in P_1$ add i to V' if $p[C(i)] = \sigma_i$,

- for each $p \in P_2$ where $p[C(j)] = \sigma_j$ and $p[C(i)] = \sigma_i$, arbitrarily add either i or j to V'.

Consider every edge $ij \in E(\mathfrak{G})$. The string g_{ij} is compatible with some pattern $p \in P$, therefore at least one of i and j is in V', thus V' forms a proper vertex cover for $\mathfrak{G}$. $\qquad\square$

Proof of Theorem 3.14. The construction used in the proof of Lemma 3.15 has the required structural properties. Again containment in NP is apparent from the usual verification algorithm techniques. $\qquad\square$

Corollary 3.18. PI WITH SMALL STRINGS *is* para-NP-*complete when parameterized by* $n + d + |B|$.

Proof. The corollary follows from Theorem 3.10 and Corollary 2.16 from [16]. $\qquad\square$

3.1. Containment

Although the W[2]-hardness reduction is quite direct, containment of PATTERN IDENTIFICATION when parameterized by k is not apparent. In fact it is not clear that the problem lies in W[P] or even XP. As the non-parameterized version of the problem is NP-complete, it is at least contained in para-NP. For PI WITH SMALL PATTERNS we have shown containment in W[2]. In contrast, for PI WITH LARGE PATTERNS we can show containment in W*[5].

Theorem 3.19. PI WITH LARGE PATTERNS $\in$ W*[5] *when parameterized by* $k + r$.

Proof. We reduce the problem to $\mathrm{MC}(\Sigma_{5,1}^*)$, which is complete for W*[5] [7, 16]. We use the same first-order structure as in the proof of Corollary 3.5, and give a suitable first-order formula:

$$
\exists s_1, \ldots, s_k, i_{1,1}, \ldots, i_{k,r} \forall j
$$
$$
(Gj \to (\exists l(\bigvee_{c \in [k]} l = s_c \land \forall b(Cjb = Clb \lor \bigvee_{d \in [r]} b = i_{c,d})))) \land
$$
$$
(Bj \to (\forall l(\bigwedge_{c \in [k]} l = s_c \land \exists b(Cjb \neq Clb \land \bigwedge_{d \in [r]} b \neq i_{c,d})))) \land
$$
$$
(\bigwedge_{c \in [k]} (Ns_c \land \bigwedge_{d \in [r]} Ni_{c,d}))
$$

The formula picks out k indices of strings (implicitly in G, as a choice of a string from B will fail) and for each of these, r indices which will be the location of the $*$ symbols in the patterns. For each index, if the index selects a string in G, then one of the patterns is compatible with the string, if it selects a string in B, no pattern is compatible with the string. We note that the B clause is in $\Pi_{2,1}$, and hence $\Sigma_{3,1}$, giving the final bound of $\Sigma_{5,1}^*$. $\qquad\square$

This also places PI WITH LARGE PATTERNS somewhere between W[5] and W[8] [7]. We note that the above formula could be converted into prenex form, giving a tighter containment, however the central observation is that it will be greater than W[2], in contrast to the hardness result and the containment of PI WITH SMALL PATTERNS.

4. Tractable Cases of Pattern Identification Problem

Guided by the results of Section 3, we identify the following cases where the PATTERN IDENTIFICATION problem is tractable.

Theorem 4.1. PATTERN IDENTIFICATION *is fixed-parameter tractable when parameterized by* $|\Sigma| + n$.

Proof. Taking the alphabet size and the string length as a combined parameter gives an immediate kernelization. The total number of strings of length n over alphabet Σ is $|\Sigma|^n$. Thus $|G| + |B| \leq |\Sigma|^n$. □

Theorem 4.2. PI WITH SMALL STRINGS *is fixed-parameter tractable when parameterized by* $d + |G| + |B|$, *with a kernel of size* $O(d \cdot (|G| + |B|)^2)$ *in both the total number of symbols across all strings and the size of the alphabet.*

Proof. As G and B are d-small, there can be at most $d \cdot (|G| + |B|)$ positions where any pair of strings in $G \cup B$ differ, that is, every other position must be the base symbol uniformly across all strings. The positions where every string is identical cannot be of use in generating patterns, thus we may ignore these positions. This gives restricted sets G' and B' of size $|G'| + |B'| \leq |G| + |B|$ containing strings of length at most $d \cdot (|G| + |B|)$. Furthermore this restricts the number of symbols used from Σ to at most $d \cdot (|G| + |B|)^2$. Thus we can restrict our alphabet to these symbols alone, denote this new alphabet by Σ'. This gives our required kernel size.

The initial determination of which positions to ignore can be computed in $O(n \cdot (|G| + |B|))$ time, thus the kernelization can be performed in polynomial time. □

Theorem 4.3. PATTERN IDENTIFICATION *is fixed-parameter tractable when parameterized by* $k + n$.

Proof. Let (Σ, G, B, k) be an instance of PATTERN IDENTIFICATION. If (Σ, G, B, k) is a YES instance, by definition, there exists a P with $|P| \leq k$ such that every string $g \in G$ must be compatible with at least one $p \in P$. Therefore given g, the compatible p must consist of, at each position, either the $*$ symbol, or the symbol at the same position in g.

This gives a direct bounded search tree algorithm for PATTERN IDENTIFICATION. At each node in the tree we select an arbitrary g from G. We then branch on all possible patterns p that are compatible with g, with a new set $G := G \setminus \{h \in G \mid p \to h\}$ (note that this removes g from further consideration). If there is a $b \in B$ such that $p \to b$, then we terminate the branch. If we reach depth k and $G \neq \emptyset$, we terminate the branch. Otherwise if at any point we have $G = \emptyset$, we answer YES.

Obviously the depth of the search tree is explicitly bounded by k. The branching factor is equal to the number of patterns compatible with a string of length n, which is 2^n. The adjustment of G and checks against B at each node individually take $O(n)$ time, giving $O((|G|+|B|)\cdot n)$ time at each node. Combined the algorithm takes $O(2^{kn}\cdot(|G|+|B|)\cdot n)$ time, and the theorem follows.

$\square$

Theorem 4.4. PATTERN IDENTIFICATION *is fixed-parameter tractable when parameterized by* $|G|+n$.

Proof. The search tree approach used in the proof of Theorem 4.3 can also be adapted to the combined parameter $|G|+n$. Again we select an arbitrary g from G. We branch on all possible patterns p that are compatible with g, of which there are at most 2^n, with the new set $G := G \setminus \{h \in G \mid p \to h\}$. If $p \to b$ for any $b \in B$, the branch is terminated. When we have $G = \emptyset$, we check whether the collected set P of patterns in that branch. If $|P| \le k$ we answer YES, otherwise the branch is terminated. If all branches terminate with no YES answer, we answer NO. $\square$

Theorem 4.5. PI WITH LARGE PATTERNS AND SMALL STRINGS *is fixed-parameter tractable when parameterized by* $k+|\Sigma|+d+r+|B|$.

Proof. As each pattern can have at most r many $*$ symbols, every other symbol in each pattern is fixed. Thus each pattern is compatible with $|\Sigma|^r$ strings. This limits the number of strings in G to $k\cdot|\Sigma|^r$.

The tractability then follows from Theorem 4.2. $\square$

5. Discussion

Complementing the classification results given above, we now discuss some related issues. Firstly (in Section 5.1), given the complex parameter landscape introduced, what problems remain unsolved, and which are the interesting parameterizations for future work? Secondly, we related PATTERN IDENTIFICATION to some similar problems that give some small intuition as to sources of complexity in PATTERN IDENTIFICATION (Section 5.2).

5.1. The Mire of Multivariate Analysis: Untangling the Parameters

The complexity analysis in this work involves a considerable number of parameters and unsurprisingly, there are some relationships between them that can be identified, allowing a better perspective on the sources of complexity

in the problem, and what cases remain open. The immediately obvious relationships, for non-trivial parameter values[5], are $r \leq n$, $s \leq n$ and $d \leq n$. We also note that $k \leq |G|$ and $k \leq (|\Sigma| + 1)^n$, again for non-trivial values of k. This helps to unravel some of the relationships present in the results of this work. We also note that, of course, expanding a list of parameters preserves tractability, while reducing a list of parameters preserves intractability
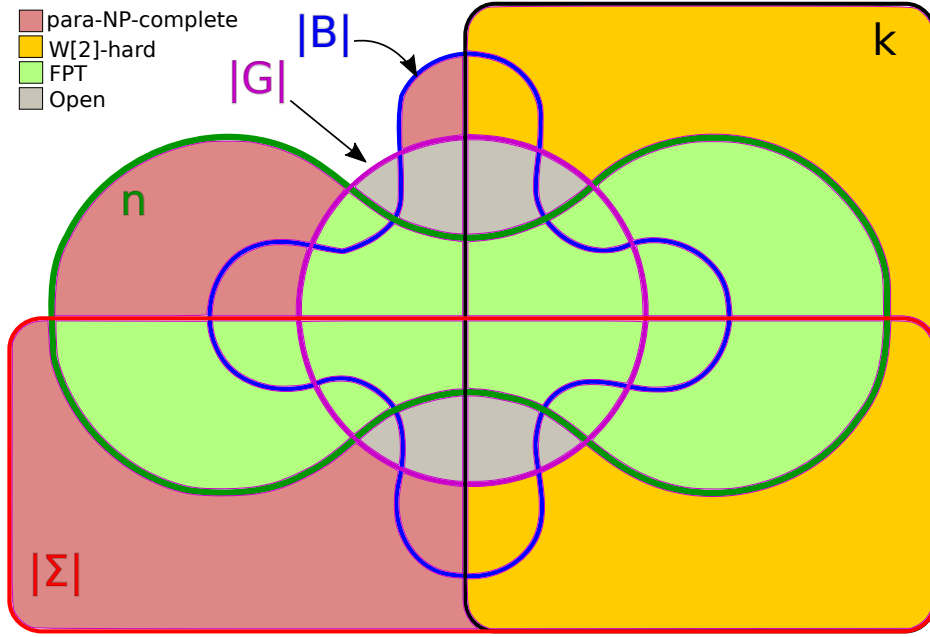

Figure 4: Simplified representation of the parameter space and the complexity results. We note in particular that n or at least one of its related parameters s, r or d seems essential for tractability (though never sufficient). Given the nature of the input as a set of strings, it is perhaps unsurprising that at least two parameters are (apparently) needed for tractability. The obvious open cases are dominated by the parameter $|G|$.

A visual summary of the tractable, intractable and open cases for a simplified parameter space is given in Figure 4. Given the relationships between s, r, d and n, we reduce the parameter space to k, $|\Sigma|$, n, $|G|$ and $|B|$. Although this reduces the accuracy of the space, the broad complexity landscape of the problem becomes more comprehensible.

Speculatively, we may observe that the problem seems to require at least two parameters for tractability. This is perhaps unsurprising, given the nature of

---

[5]By non-trivial we mean values which differentiate the parameters – for example, if $s > n$, s becomes meaningless as any number of $*$ symbols would be allowed, within the limitation of length n strings.

the input – we need some parametric "handle" on the length of the strings and another on the number of strings.

From Figure 4 it is clear that the central parameter in the open cases is $|G|$, though we note that in the full parameter space, there are combinations of s, r and d with other parameters for which the complexity remains open[6].

5.2. Ties to Other Problems

The PATTERN IDENTIFICATION problem, as would be expected, has ties to other problems that (can) model the general search for patterns that separate two sets of data. These ties also illustrate some features of the computational complexity of the problem.

5.2.1. Set Covering

When the length n of the strings is small, PATTERN IDENTIFICATION can be easily reduced to SET COVER. Given an instance (Σ, G, B, k) of PATTERN IDENTIFICATION, we can generate the set P of all patterns that are compatible with some string in G. We know that $|P| \leq |G| \cdot 2^n$. From P we remove any pattern that is compatible with a string in B. Let P' be the set thus obtained. For each $p \in P'$, let $s_p = \{g \in G \mid p \to g\}$, and let $S = \{s_p \mid p \in P'\}$. Taking G as the base set, (G, S, k) forms an instance of SET COVER (parameterized by k). This reduction can be performed in $O((|B| + |G|) \cdot |G| \cdot 2^n n)$ time.

This leads to the following theorem:

Theorem 5.1. PATTERN IDENTIFICATION $\in \mathsf{W}[2]$ *when* $n \leq f(k) \cdot \log |I|$ *where* $|I|$ *is the overall size of the instance and* $f(k)$ *is a computable function of the parameter* k.

Proof. The reduction above is a parameterized reduction if $2^n \in O(g(k) \cdot |I|^c)$ for some computable function g. $\square$

It is not clear that we retain $\mathsf{W}[2]$-hardness in this case however, so we unfortunately do not obtain a $\mathsf{W}[2]$-completeness result.

This does give us an immediate approximation algorithm for this case however. As SET COVER has a $1 + \log(|S|)$-factor linear time approximation algorithm [22], we obtain a $1 + \log(|G|^2 \cdot \log(|I|) \cdot 2^{f(k)})$-factor fpt-time approximation algorithm.

---

[6]At last hand-count, 72 cases out of the 256 possible parameterizations with these parameters remain open, compared to 8 with the reduced parameter space.

5.2.2. Feature Set

The k-FEATURE SET problem bears a strong resemblance to the PATTERN IDENTIFICATION problem[7], except in the k-FEATURE SET case, the problem asks for a set of features that separate the "good" examples from the "bad" rather than a set of patterns. In fact, given a feasible solution for one problem, we can construct a feasible (but not necessarily optimal) solution to the other.

Given a set $I = \{i_1, \ldots, i_k\}$ of indices of columns forming a feature set, we can construct a set of patterns that separates G and B as follows: for each $g \in G$, let p_g be the pattern where $p_g[i] = g[i]$ if $i \in I$ and $p_g[i] = *$ otherwise. We note that this gives a set of small patterns (*i.e.*, $s = k$), however the number of patterns may be as large as $|G|$.

Conversely, given a set of patterns P with at most s non-$*$ symbols in each pattern, the set $I = \{i \in [n] \mid \exists p \in P(p[i] \neq *)\}$ forms a feature set. Again we note that the size of the feature set may be as large as $|G| \cdot s$.

If we consider a variant of PI WITH SMALL PATTERNS where we relax the constraint on the number of patterns in the solution, it is easy to see that this problem is in $\mathsf{W}[2]$. This suggests that the solution size plays a significant role in raising the complexity of the problem from a parameterized perspective.

6. Conclusion and Future Directions

There are a number of open complexity questions prompted by this paper, three of which we think are particularly interesting.

The central question is of course the precise classification of PATTERN IDENTIFICATION. Although PI WITH SMALL PATTERNS is $\mathsf{W}[2]$-complete, the general problem is only $\mathsf{W}[2]$-hard, and the containment of PI WITH LARGE PATTERNS simply gives a loose upper bound, although does suggest that the problem is harder than PI WITH SMALL PATTERNS. The problem, intuitively, also shares some similarities with p-HYPERGRAPH-(NON)-DOMINATING-SET which is $\mathsf{W}[3]$-complete [7]. p-COLORED-HYPERGRAPH-(NON)-DOMINATING-SET however is $\mathsf{W}^*[3]$-complete [7] and appears "harder" than PATTERN IDENTIFICATION, hence we conjecture:

Conjecture 6.1. PATTERN IDENTIFICATION *is* $\mathsf{W}[3]$-*complete when parameterized by* k.

---

[7]Indeed, variants of k-FEATURE SET have also been considered for use in similar applications as PATTERN IDENTIFICATION [10].

There are also some interesting parameterizations for which the complexity remains open:

- PI WITH SMALL STRINGS parameterized by $k + |\Sigma| + d$, and

- PI WITH LARGE PATTERNS AND SMALL STRINGS parameterized by $k + d + r$.

Turning to the parameter $|G|$, results for the following combinations of parameters would also close some of the significant open cases:

- PATTERN IDENTIFICATION parameterized by $k + |\Sigma| + |G|$,

- PATTERN IDENTIFICATION parameterized by $|G| + |\Sigma| + |B|$, and

- PATTERN IDENTIFICATION parameterized by $k + |B| + |G|$.

As a matter of prognostication, we would guess that the first of these is in FPT, and the latter two are hard for some level of the W-hierarchy, but as yet have no strong evidence for these claims.

7. Acknowledgements

8. References

[1] Dana Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21(1):46–62, 1980.

[2] Kenneth Appel and Wolfgang Haken. *Every Planar Map is Four-Colorable*, volume 98 of *Contemporary Mathematics*. American Mathematical Society, Providence, RI, 1989. With the collaboration of J. Koch.

[3] Robert Bredereck, Thomas Köhler, André Nichterlein, Rolf Niedermeier, and Geevarghese Philip. Using patterns to form homogeneous teams. *Algorithmica*, 71:517–538, 2015.

[4] Robert Bredereck, André Nichterlein, and Rolf Niedermeier. Pattern-guided k-anonymity. *Algorithms*, 6:678–701, 2013.

[5] Laurent Bulteau, Falk Hüffner, Christian Komusiewicz, and Rolf Niedermeier. Multivariate algorithmics for NP-hard string problems. *Bulletin of the EACTS*, 114, 2014.

[6] Adam Cannon and Lenore Cowen. Approximation algorithms for the class cover problem. *Annals of Mathematics and Artificial Intelligence*, 40(3-4):215–224, 2004.

[7] Yijia Chen, Jörg Flum, and Martin Grohe. An analysis of the W*-hierarchy. *The Journal of Symbolic Logic*, 72(2):513–534, 2007.

[8] Yijia Chen and Bingkai Lin. The constant inapproximability of the parameterized dominating set problem. *CoRR*, abs/1511.00075, 2015.

[9] Carlos Cotta and Pablo Moscato. The k-Feature Set problem is W[2]-complete. *Journal of Computer and System Sciences*, 67(4):686–690, 2002.

[10] Carlos Cotta and Pablo Moscato. The parameterized complexity of multiparent recombination. In *Proceedings of the 6th Metaheuristics International Conference (MICS2005)*, pages 237–242, 2005.

[11] Pierluigi Crescenzi. A short guide to approximation preserving reductions. In *Proceedings of the Twelfth Annual IEEE Conference on Computational Complexity*, pages 262–273. IEEE Computer Society, 1997.

[12] Xiaotie Deng, Guojun Li, Zimao Li, Bin Ma, and Lusheng Wang. A ptas for distinguishing (sub)string selection. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, ICALP '02, pages 740–751. Springer-Verlag, 2002.

[13] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[14] Rodney G. Downey, Michael R. Fellows, and Udayan Taylor. The parameterized complexity of relational database queries and an improved characterization of W[1]. In Douglas S. Bridges, Cristian S. Calude, Jeremy Gibbons, Steve Reeves, and Ian H. Witten, editors, *First Conference of the Centre for Discrete Mathematics and Theoretical Computer Science, DMTCS 1996, Auckland, New Zealand, December, 9-13, 1996*, pages 194–213. Springer-Verlag, Singapore, 1996.

[15] Michael R. Fellows, Jens Gramm, and Rolf Niedermeier. On the parameterized intractability of CLOSEST SUBSTRING size and related

problems. In Helmut Alt and Afonso Ferreira, editors, *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2002)*, volume 2285 of *Lecture Notes in Computer Science*, pages 262–273. Springer, 2002.

[16] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[17] M. R. Garey and D. S. Johnson. *Computers and Intractability – A Guide to the Theory of NP–completeness*. Freeman and Company, San Francisco, 1979.

[18] Jens Gramm, Jiong Guo, and Rolf Niedermeier. Parameterized intractability of distinguishing substring selection. *Theory of Computing Systems*, 39(4):545–560, 2006.

[19] Jens Gramm, Rolf Niedermeier, and Peter Rossmanith. Fixed-parameter algorithms for CLOSEST STRING and related problems. *Algorithmica*, 37(1):25–42, 2003.

[20] J. Håstad. Some optimal inapproximability results. In *Proceedings of the 29th ACM Symposium on the Theory of Computing (STOC)*, pages 1–10, 1997.

[21] Danny Hermelin and Liat Rozenberg. Parameterized complexity analysis for the closest string with wildcards problem. *Theoretical Computer Science*, 600:11–18, 2015.

[22] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, pages 256–278, 1974.

[23] Michael Kearns and Leonard Pitt. A polynomial-time algorithm for learning k–variable pattern languages from examples. In *Proceedings of the 2nd Annual ACM Workshop on Computational Learning Theory*, pages 57–71, 1991.

[24] Ming Li, Bin Ma, and Lusheng Wang. On the closest string and substring problems. *Journal of the ACM*, 49(2):157–171, March 2002.

[25] Dániel Marx. Parameterized complexity and approximation algorithms. *The Computer Journal*, 51(1):60–78, 2008.

[26] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP.

In *Proceedings of the 29th ACM Symposium on the Theory of Computing (STOC)*, pages 475–484, 1997.