PH.D. THESIS IN
INDUSTRIAL AND INFORMATION ENGINEERING

# Decentralized approaches for admission, routing and flow problems

CANDIDATE
Francesca Rosset

SUPERVISOR
Prof. Franco Blanchini

CO-SUPERVISOR
Prof. Pier Luca Montessoro

Cycle XXXV — Year 2023

Institute Contacts
Dipartimento Politecnico di Ingegneria e Architettura
Università degli Studi di Udine
Via delle Scienze, 206
33100 Udine — Italia
+39 0432 558253
https://www.dpia.uniud.it/

Author's Contacts
Francesca Rosset
Via delle Scienze, 206
33100 Udine — Italia
rosset.francesca@spes.uniud.it

# Acknowledgements

Firstly, I am extremely grateful to my supervisor Franco Blanchini for his invaluable advice, continuous support, guidance, and patience, and for all the opportunities he gave me to work on many different interesting topics.

My gratitude extends to my co-supervisor Pier Luca Montessoro for his useful advice and support.

Many special thanks to Raffaele Pesenti for the support and very useful discussions.

Many thanks to Babak Jafarpisheh for his collaboration and advice.

Many thanks to my co-authors Daniele Casagrande, Giulia Giordano, Carlos Andrès Devia, Simone Milanesi, Marta Colaneri, Kenneth Pesenti, Paolo Bolzern, Patrizio Colaneri, Paolo Sacchi, Giuseppe De Nicolao, Raffaele Bruno, and Carlo Drioli for the opportunity of working together.

I would like to extend my thanks to Alessandro Pilotto and Pier Paolo Palestri: their advice during the brief period working together just before my Ph.D. was also greatly useful during these last three years.

Last but not least, I would like to thank my parents, my sister Eleonora, my grandparents and my whole family for their encouragement and support over all these years.

# Abstract

Online control of large dynamic networks is challenging because little data are in general available about the environment, because decentralized strategies have to be employed, which rely on the knowledge of local data only, and because faults can occur. In this regard, three problems are addressed in this thesis.

The first problem concerns the scheduling of some requests of a limited resource occurring at different times, from a supplier with limited capacity. The goal is that of minimizing the average waiting time for these requests.
The problem is formulated as an optimal control one, in which the control is the supply strategy, specified by some constraints, and the state variables are associated with the waiting times of the demands. The exact optimal problem requires mixed-integer linear programming; some relaxed versions are also formulated and, in particular, one of these is based on linear programming and efficiently provides some lower bound. Some online heuristics are analyzed, both centralized and decentralized, for which, in general, no a-posteriori optimality of the solution is obtained.

The second problem is an agent-based minimum path one. Some tokens (agents) are injected in the network, in some source nodes, and must travel in the network to find an exit, a sink. A simple decentralized policy is proposed. This policy allows or denies the transitions of the tokens along the arcs on the basis of a simple local threshold mechanism. In particular, a transition occurs through a directed arc if the amount of tokens present in the origin node minus the amount tokens present in the end node exceeds the arc cost. Despite the very simple local mechanism, in the long run, all the injected tokens leave the network by the closest sink through the shortest path, although some tokens are, unavoidably, lost during the initial transient exploring phase. This issue can be avoided by enhancing the policy allowing the generation of some virtual tokens. Some constraints to the maximum number of transitions can also be imposed to all tokens. In fact, this is equivalent to applying the policy proposed for the unconstrained case to the so-called expanded network.

The third problem considers flow networks with buffers in the nodes containing an amount of a continuous resource at a given level (state), which is transferred between nodes by controlled flows along the arcs. A decentralized control is formulated to meet a given flow demand, stabilize the system and asymptotically minimize the $p$-norm of the flow. This control is specified at the arc level and depends only on the value of $p$ and on the difference of the states of the two arc's endnodes. After an initial transient, the unique optimal desired flow is obtained when $1 < p < \infty$. When $p = 1$ sparsity of the solution tends to emerge, while when $p = \infty$ fairness is promoted; however, no optimality or uniqueness of solution is achieved in these two cases: suboptimal solutions can be obtained by applying the control with $p \to 1$ and $p \to \infty$, respectively. Enhancements can be applied to support uncontrollable flows governed by unknown dynamics depending on the buffer levels and buffer level control to a desired set-point. In this case, a decentralized proportional-integral control is adopted.

# Contents

# List of Tables

# List of Figures

# Introduction

Online control is fundamental in real-world systems in which little data is available about the future. In large, dynamic, and unknown environments, little data might also be available about the environment itself: decentralized strategies can be employed, which rely on the knowledge of local data only, and are also proven to be fault-tolerant and robust. In this regard, in this thesis, three problems are addressed, whose results are the main results I've worked on during the three years of my Ph.D.; in particular, this thesis is divided into three Parts, each of which is based on a specific topic.

In Part I the problem of scheduling some requests requiring a given amount of a limited resource is considered, to minimize the average waiting time required for each request to be completely served. Different supply strategies can be considered (at maximum supply rate, with or without interruptions, or with supply rate variations). An optimal control framework is introduced to provide some optimal solutions, where different supply strategies are supported by means of some additional constraints. Both offline and online solutions, possibly decentralized, are presented. Online heuristics do not require the knowledge of future data, but are not necessarily optimal (when evaluated a-posteriori). Solutions can be analyzed periodically, based on some historical recorded data, to improve the performance of the above-mentioned heuristics, by tuning some parameters.

This Part is based on [1], whose results come from the collaboration between Daniele Casagrande, Babak Jafarpisheh, Pier Luca Montessoro, Franco Blanchini, and me.

After an introduction including a simple exposition of the reasons behind the problem considered in this Part and a literature review (Chapter 1), and a description of the setup (Chapter 2), the above-mentioned optimal control framework is introduced in Chapter 3 and studied. A formulation yielding the exact solution is presented, which however results in mixed integer linear programming implementation. Integral (buffer) variables are introduced, which account for the waiting time of each request. Some relaxations are considered, which result in efficient linear programming implementation and provide some lower bound to the optimal cost.

In Chapter 4, some online heuristics are presented and analyzed, both centralized and decentralized, which can be employed to scheduling the requests without assuming any knowledge of future requests.

An application to the case of electric vehicle battery charging scheduling is reported in Chapter 5, in which the proposed methodology is applied by comparing a-posteriori the performance of the above-mentioned heuristics with the optimal and possibly relaxed solutions obtained by utilizing the batch problems deriving from the proposed framework.

In Part II networks with discrete moving agents are considered. In particular, it is assumed that these agents, named tokens, are injected in some source nodes of a weighted network and have to travel from node to node (explore the network) to reach some unknown sink nodes to leave the network. Tokens might stop in the nodes: the state of the node is introduced as the number of tokens bufferized in there. No information is available to the tokens for their routing decisions, except for the information about the state of the nodes in their neighborhood. A simple decentralized policy is introduced, which not only ensures that the above-mentioned behavior is achieved in the long run, but also that eventually the tokens are able to reach the closest sink through some shortest path, even in the case in which some constrains are applied to it.

This Part is based on [2] (under review), whose results come from the collaboration between Franco Blanchini, and Raffaele Pesenti, and me.

After an introduction including a literature review and a simple introduction to the proposed strategy (Chapter 6), and a description of the setup (Chapter 7), in Chapter 8 the unconstrained problem is studied, in which the travelled paths of the tokens have no restrictions. A decentralized policy is proposed, which uses only local information to determine whether a token in a given node should keep moving to a neighbor node or not: a transition is admitted if the difference between the states of the two nodes is greater than the cost of the connecting arc. After an initial transient in which tokens are possibly forced to stop in some intermediate nodes, a steady-state is reached, in which all newly injected tokens reach the closest sink through the shortest path. An enhancement to the proposed policy is also presented, which allows tokens to proceed moving, instead of stopping in the nodes, during the initial transitory; the performance of the policy is improved and, under the assumption of strongly connected network, it is guaranteed that all the tokens injected in the network reach a sink, although this does not necessarily occur along the shortest paths initially.

Then, in Chapter 9, the constrained problem is considered, in which a secondary cost is added to each arc: the total secondary cost of a path cannot exceed a given value, otherwise, the token is forced to stop. It is shown that this problem can be traced back to the unconstrained problem in the so-called expanded network, which is however larger than the original one. Based on this fact, a decentralized policy is built, on the basis of the one from the unconstrained case, considering multi-component states. Again, in the long run, the optimality (and feasibility) of the paths is achieved.

In Chapters 10 to 12, the proposed policy is applied in different conditions. A very simple network is considered in Chapter 10, which allows to easily analyze the step-by-step evolution of the network. A larger network is considered in Chapter 11, analyzing the effects of the control on the states and the traveled paths. Finally, in Chapter 12 a class of small-world networks is considered and the effects of changing some network characteristics are evaluated and compared.

In Part III networks with continuous flows are considered. Such flows can be continuous in nature (e.g, like in a fluid network) or approximated as such when there is a large number of agents. A network-decentralized flow control stabilizing the network must be found to asymptotically minimize the p-norm of the controlled flow meeting a given demand. The main idea is similar to that from Part II, and a decentralized control based on buffers is introduced. The discussion presented in this Part is based on [3], whose results come from the collaboration between Franco Blanchini, Carlos Andrés Devia, Giulia Giordano, Raffaele Pesenti, and me.

After an introduction including a simple exposition of the main idea behind this Part and a literature review (Chapter 13), and a description of the setup (Chapter 14), networks with uncontrolled demands with buffers on the nodes are considered (Chapter 15): an online decentralized control is proposed, which uses only local information about the buffer levels (representing the *state* of the network) and minimizes the $p$-norm of the arc flows vector. Choosing $p = 1$ tends to concentrate the flow in few arcs, along some shortest paths, usually getting a *sparse* solution, while having $p = +\infty$ tends to distribute the flow, getting a *fair* solution.

In the case in which $1 < p < +\infty$, this control is continuous, and the unique optimal solution is eventually reached at steady-state. This is the most important result of Part III: similarly to Part, by using a local control, a global optimality result is achieved. When considering $p = 1$ (respectively, $p = +\infty$), the control is no longer continuous and the solution might not be unique; then, the proposed control must be applied for $p \to 1$ (respectively, $p \to +\infty$): it is shown that the solution gets closer to the desired optimal one.

Some linear quadratic programming problem formulations are also introduced to efficiently compute offline the optimal control to be used, which can be useful for validating the proposed decentralized online strategies.

Then, some enhancements to the control are introduced (Chapter 16); in particular, the support for networks with uncontrolled dynamics depending on the buffer levels and the support for buffer level control (i.e., driving the buffer levels to a desired set point) is reached by adopting a proportional-integral control. The support for weighted $p$-norm minimization is introduced by scaling the control.

An example of application of such decentralized controls to a fluid network is reported at the end of this Part (Chapter 17), which demonstrates its efficacy in finding the optimal $p$-norm flow. Different scenarios are considered and the effects of the choice of $p$ are assessed.

The final Conclusion chapter at Page 179 summarizes the main findings from the Parts I to III.

Finally, in the Appendices, the proofs of the Theorems, Corollaries, Propositions, and Lemmas from Parts I and II are reported.

Parts II and III are somehow related. Both consider a directed network, although in the former there are discrete traveling agents moving along its arcs, while in the latter some continuous flows are considered. In both cases it is assumed that there is a buffer in each node whose state is roughly defined as the "amount" of a given quantity; in both cases, the adopted local policy/control for each arc assumes that only the information about the states of its two extreme nodes is known, in particular their difference, as well as possibly some information about the connecting arc.

Part I is more independent from the other two. However, again, a point in common is the employment of a controlled buffer system, although the context is different. Here, this buffer integrates a time, while in Parts II and III a physical quantity is accounted; in fact, different problems are considered: in Part I a "time" optimization is performed, while in Parts II and III a somehow "spatial" optimization is considered, at least to a first approximation. In any case, in both cases, the information about these buffers is used in the optimization, possibly in a decentralized way.

As a final remark, recall that the three Parts I to III are based on three works [1, 2, 3] which are a collaboration between different authors. While the theorems, corollaries, propositions, and lemmas from [1, 2, 3] are also reported here almost integrally, for completeness, some more details are also given in their explanation. Some more new aspects are also investigated and discussed.

# Part I
# Admission control optimization for waiting time minimization

Based on [1], published: Francesca Rosset, Daniele Casagrande, Babak Jafarpisheh, Pier Luca Montessoro, and Franco Blanchini. "Optimal Control Approach to Scheduling Power Supply Facilities: Theory and Heuristics". In: *IEEE Transactions on Control of Network Systems* 9.4 (2022). © 2022 IEEE, pp. 1679–1691. DOI: 10.1109/TCNS.2022.3165019

# Chapter 1 Introduction to Part I

This Part presents the work introduced in [1], which is the result of the collaboration between Daniele Casagrande, Babak Jafarpisheh, Pier Luca Montessoro, Franco Blanchini, and me.

Consider a scenario in which there is a supplier facility that can provide a given resource. Some users make some requests for an amount of that resource, at different times. Each one has a maximum rate at which it can receive this resource: as each user would like to be served as soon as possible, following its immediate interest, the supply should be performed at the maximum constant rate. The supplier, however, can only supply a limited total maximum rate. Hence, when the system is congested and there are not enough available resources, the supply of some requests is to be delayed.

However, benefits might emerge for all the users if they stop following their immediate interests and allow for possible interruptions or rate reduction of the supply. Clearly, this possibility depends on the type of requests and the flexibility of the users. Despite these operations increase the supply time of each user, overall, the total waiting time to be served is lower compared to the case in which only delays of the supply are possible, because the available power can be better exploited.

Then, the problem of scheduling the supply of a limited resource at a given rate for a specific time interval facing some resource requests [4] is addressed, with the objective of minimizing the minimum average waiting time. To simplify, three simple classes of supply strategies are considered: i) maximum rate without interruptions, ii) maximum rate with interruptions, iii) variable rate.

The “resource” considered here is generic. Without loss of generality, the specific case in which energy is requested from power networks is considered, which is of fundamental importance nowadays, given the rising electricity demand due to the increasing number of electric vehicles (EVs), for instance. Much research focuses on this topic and, in particular, on *demand-response* methods [5, 6, 7, 8, 9], *smart grids* [10, 11], *home energy management systems* [12, 13, 14, 15], and *smart charging* of EVs batteries [16, 17, 18, 19, 20], with the aim at preventing overloads on the power networks. It is assumed that there are enough supplying points, as in [21, 22], so that the total supplied power is the only constraint.

An important aspect is that some electric loads requesting energy have some flexibility, both in terms of when to supply the requested energy, and how. Timing and supplying rates can therefore be optimized considering different aspects and objectives. For the specific case of EVs batteries charging this is called *smart charging*.

The scheduling can be applied at any level, from the local smart home/EV charging site to the community level, up to the context of smart cities and smart cities networks. Clearly, the broader the context, the larger the amount of data to be considered, so that the scheduler must be able to handle large-scale data; otherwise, a better solution is managing the network at a local level in a decentralized way.

Moreover, usually data about future requests are not known in advance; then, online heuristics are necessary.

However, such online heuristics, both centralized and decentralized, are not optimal, in general, when evaluated a-posteriori. Indeed, the optimality of a strategy can be evaluated only once all the requests are known; then, in a real-time scenario, where future requests are not known in advance, the level of a-posteriori sub-optimality should be assessed. An optimal control framework is here proposed, where the scheduling problem is formulated as an optimal control problem and for different classes of supply strategies, by means of some additional constraints.

As the exact problem formulations for the three considered supply strategies result in mixed-integer linear programming, for some large instances of the problem, it might be difficult to find a solution, possibly within a reasonable time.

A controlled buffer system [23, 24] is introduced, in which an integral variable (virtual buffer) $x_i(t)$ accounts for the individual delay of user $i$ in fulfilling the request. Then, a relaxed problem that can be approached efficiently via Linear Programming (LP) is formulated for the case in which power can be supplied at a variable rate. A greedy approach can also be applied, leading to a (non-optimal, in general) heuristic solution for the variable rate case, which also provides some lower bounds for the exact optimal cost minimizing the average waiting time.

To summarize, two kinds of strategies are considered: off–line (batch) strategies, in which all data are available (regarding all the current and future requests), and on–line strategies, in which data become available in real-time. In the latter case, both centralized or decentralized schedulers are considered, to decide when and how to supply power.

Real-time strategies perform the actual scheduling. Assuming that data about the requests is recorded over time, an optimization could be performed periodically by some centralized element, using the batch strategies and the known collected data. The results can be used to evaluate (a-posteriori) and possibly re-tune the real–time scheduling strategies.

In the rest of this Chapter, first, the intuitive idea behind the considered problem is presented in Section 1.1, and after that, the literature review is reported in Section 1.2 and the main contributions summarized in Section 1.3. Then, in Chapter 2 the setup is described, and the considered problem is stated. In Chapter 3 the proposed optimal control framework is introduced and studied for the non-interruptible, interruptible, and variable rate supply strategies. In Chapter 4 some online heuristics are proposed, which are both centralized and decentralized, for the three above-mentioned supply strategies. Finally, in Chapter 5, the proposed optimal control framework and the online heuristics are evaluated (a-posteriori) and compared, considering the specific application to EVs' battery charging scheduling, using real data from the ACN-Data dataset [25]. The proofs are reported in Appendix A.

## 1.1 The main idea

Consider a power supply facility in which there are four users, labeled $A$, $B$, $C$, and $D$, which require, at different times, a given amount of energy to be supplied at a given rate (power) and for a certain duration. The requested power profiles $R_A(t), R_B(t), R_C(t), R_D(t)$ describe these requests and are represented in the four graphs on the top-left of Fig. 1.1.

Assuming that all these requests are served immediately as required, the overall power supplied by the facility is given by the sum of these profiles $R_A(t) + R_B(t) + R_C(t) + R_D(t)$, see the bottom-left of Fig. 1.1. As no optimization is performed, the system must be able to support this supply: the peak supplied power, however, might be very large. If there is a limit on the overall power capacity $P$ of the system, such peaks result in overloading of the system, so that it is not possible to serve all these requests as required.

Some flexibility can be exploited: it might be acceptable fulfilling such requests with some delay. Then, the actual power supply can be scheduled to avoid overloading the system: this determines the power supply profiles of the requests $S_A(t), S_B(t), S_C(t), S_D(t)$, see the graphs on the right of Fig. 1.1. For instance, it might be possible to just delay the supply (see request $D$), to interrupt and then resume it (see request $A$), or even reduce the power rate (see request $C$): all of these possibilities, however, introduce the above-mentioned additional delay. The supply must provide exactly the required amount of energy, so that the area below each $R_i(t)$ and the corresponding $S_i(t)$ must be the same.

The supply of these requests can be scheduled in many ways. For instance, one might decide to supply the requests one at a time; however, this results in a waste of the non-supplied system capacity and in large additional delays. Then, an optimization is required, so that the available capacity is exploited at best, while avoiding overloading. To this aim, the average additional delay minimization is a reasonable objective to consider.

In practice, there are many requests to schedule and at a given time $t$ it might be not possible to know in advance what requests will be made in the future. Hence, there is the need for online strategies to schedule such requests. These online strategies are in general not optimal (when evaluated a-posteriori), but their performance can be evaluated (a-posteriori) by solving some global optimization problems over a horizon using the recorded known past data, and this can be used to improve them.

## 1.2 Literature review

As already mentioned, *demand-response* methods [5, 6, 7, 8, 9], *smart grids* [10, 11], *home energy management systems* [12, 13, 14, 15], and *smart charging* of electric vehicles' (EVs) batteries [16, 17, 18, 19, 20] are gaining a lot of importance lately and aim at reducing the overload on the system by properly scheduling the timing and rate of the supply of power requests.

Different objectives could be considered, like monetary cost minimization, green energy exploitation maximization, and so on. The objective considered here considers the delay each user experiences to get served, which must be minimized.

In [26, 27, 28] a penalty is imposed for not fulfilling the requests before some due date. In [29] a multi-queuing model is introduced for average waiting time minimization with non-interruptible supply.

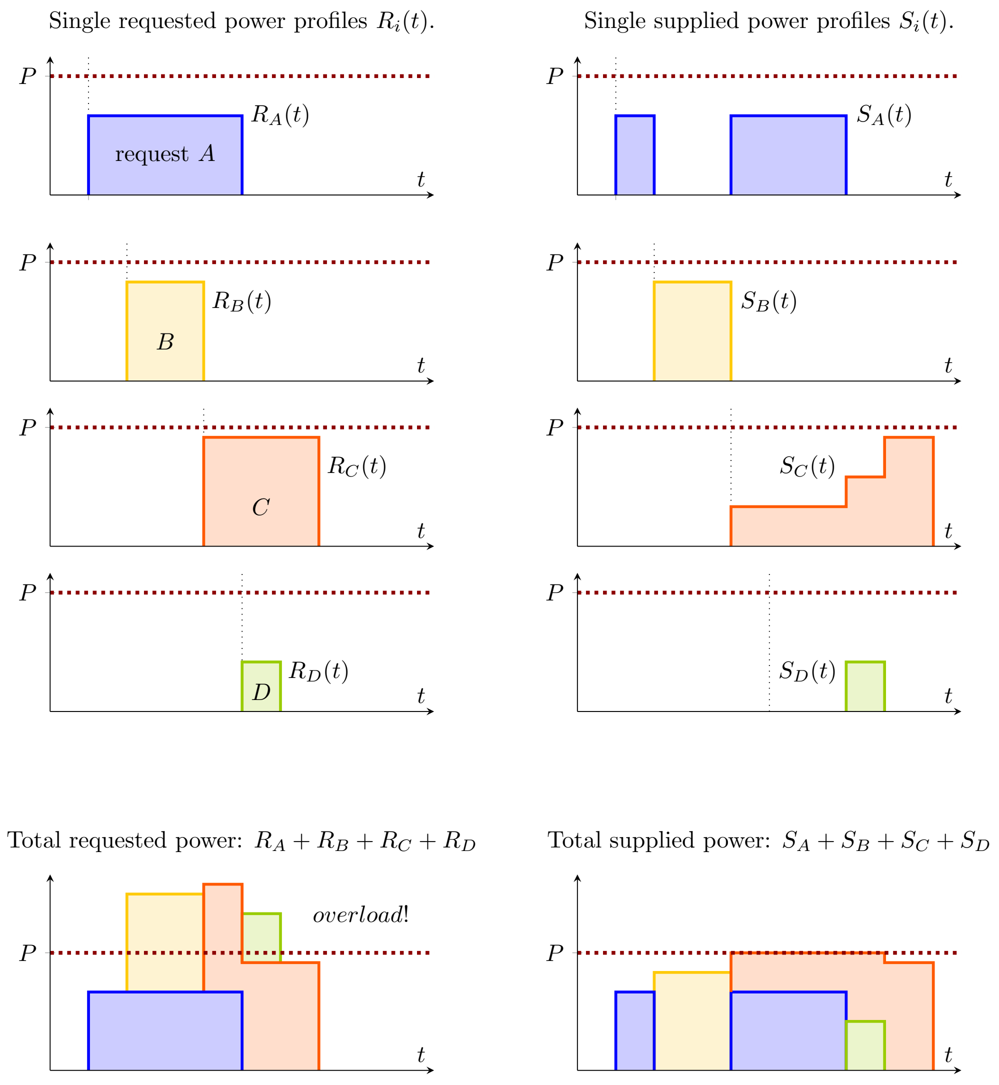


Figure 1.1: Supplying the requests exactly as required might result in overloading of the system if the total supply exceeds the system capacity $P$. By properly scheduling the supplied power profile, it is possible to avoid overloading on the system, at the cost of some additional delays. The scheduler should minimize such additional delays.

A constrained stochastic optimization problem is introduced in [30]. When admitting possible interruptions of the supplying, a multi-objective evolutionary algorithm is studied in [31], and some multi-objective nature-inspired optimization techniques are exposed in [32]. Some other strategies aim to minimize, among others, the maximum waiting time [33, 34], or to supply the resource as quickly as possible [21, 35, 22, 36]. Finally, in some works traveled paths are also taken into account [37, 38, 39, 40, 41].

Here, the specific problem of scheduling to minimize the average waiting time of the users is addressed, or equivalently the average completion time, considering three different supply strategies: i) maximum rate without interruptions; ii) maxiumum rate with interruptions; iii) variable rate.

The problem can also be studied as a scheduling problem [42, 43, 44]. The classes of problems presented here are variants of the *charge scheduling problem* [45], which is quite different from classical scheduling ones. They can be classified, according to [46], as *identical parallel machines problems of independent tasks with release times and resource constraints*, as well as *preemption* and/or *time-varying resource allocation* (depending on the specific version), minimizing the *total completion time*.

As it will be seen, the considered problem is NP-hard, so it might be not so trivial to find the exact optimal solution for the minimum average waiting time and a heuristic approach might need to be considered, instead, especially for large instance of the problem.

Remind that when demands are all unitary and their rate cannot be varied, which occurs in CPU scheduling, the optimal strategies consist in scheduling the shortest job first for the non-interruptible case, and the shortest remaining processing time for the interruptible one [47]. In these conditions (interruptible and unitary requests),

[48] proposes a Mixed-Integer Linear Programming problem for single-machine scheduling for average completion time minimization. The problem considered here is different, as different rates and generic capacities are considered.

## 1.3 Contributions

One of the main contributions of this Part (and [1]) is proposing an alternative way of treating scheduling problems, which are typical combinatorial problems: these can indeed be reduced to the formulation and solving of some MILP or LP problems. The latter case, in particular, is of great interest: despite not providing the exact optimal solutions for the considered problems, they can be solved quite efficiently even for large instances of the problem and still provide some lower bound to such costs. To summarize, the contributions are:

- the formulation of the considered scheduling problem as an optimal control problem, capable of supporting different classes of supplying strategies by means of some additional constraints;
- while the exact problems result in mixed-integer linear programming, some relaxed version of the problems could be solved exactly quite efficiently and provide some good lower bounds to the optimal costs;
- the formulation of some online heuristics, both centralized and decentralized, to deal with the considered problem without needing information about future requests. Some of the former are based on the above-mentioned framework, while the latter have the advantage of not requiring the knowledge of any data about other current requests, too. A decentralized reservation-variable based heuristic is introduced, which can create a priority ordering among requests in a decentralized way.
- the definition of a procedure to evaluate such heuristics a-posteriori: the results obtained from the optimal control framework based on historical recorded data about the requests can be exploited to improve their performance.

Chapter 2

# Problem setup

## 2.1 Resource supply facilities

Consider a resource supplier facility shared between $n$ users that make requests at specific time instants, requiring an amount of a limited resource for a specific time interval. In particular, consider a generic user $i \in \{1, 2, \dots, n\}$; then,

- user $i$ requires an amount of resource $r_i$, beginning at *release time* $t_i$ and for an interval of duration $\tau_i$ (minimum processing time, assuming maximum rate $r_i$); no strict deadline is assumed;
- user $i$ might receive the required resource at a rate not greater than the maximum $r_i$;
- user $i$ may be immediately admitted to the supply at time $t_i$, or delayed to the *admittance time* $t_i + \Delta_i$;
- different supply strategies can be considered: once the user $i$ is admitted to the supplying, the supplier may decide whether and when to interrupt and resume the supply, and/or to provide the resource at a time-varying rate, not greater than the nominal one $r_i$.
- the user receives all the requested amount of resource at time $t_i + \delta_i$, with $\delta_i \geq \Delta_i + \tau_i$, in general.

Multiple users can be supplied simultaneously. However, the total amount of resource that can be supplied by the supplier at a given time is limited by the system capacity $P$. Hence, not all the requests can be admitted immediately.

The problem investigated in the following is to determine an admission control strategy that minimizes the time the user has to wait to receive all the requested amount of resource.

Note that if the capacity of the system $P$ and the maximum supply rate $r_i$ are unlimited, the problem would trivialize, as each user could be satisfied instantaneously.

The following assumption is needed for feasibility.

**Assumption 2.1:**
The number of requests is finite and equal to $n$. At some time $T > 0$ all the requests are over. All the demands fulfill the constraint $r_i \leq P$.

In the following Sections, the supplying process is characterized more formally, introducing several specific constraints to be included in the formulation of the problem. Then, it is formulated as a dynamic system, introducing an integral variable that accounts for the delay of each request. Finally, the problem statement is formulated.

## 2.2 Request and supply characterization

The dynamics of the supplying process for the generic request $i$ are modeled using the quantities introduced in Fig. 2.1; a detailed description is given next.

The *normalized demand* variables $d_i$, defined by

$$d_i(t) = \begin{cases} 1 & \text{if } t_i \leq t \leq t_i + \tau_i\,, \\ 0 & \text{otherwise,} \end{cases} \tag{2.1}$$

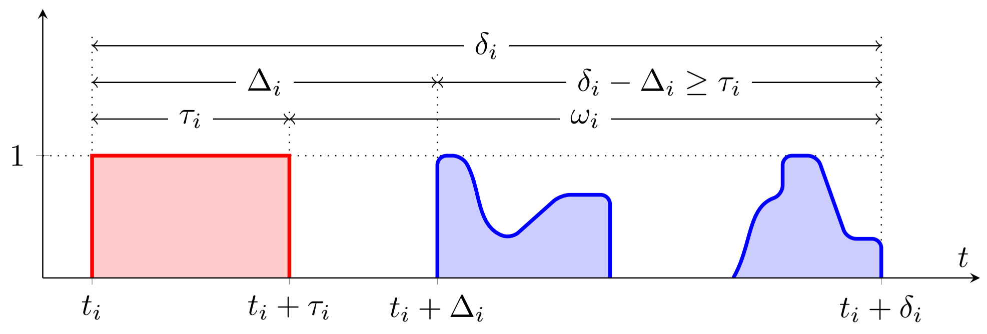


Figure 2.1: Pictorial representation of the quantities characterizing a request $i$. Red: its demand profile $d_i(t)$; blue: a generic example of its supply function $u_i(t)$ (with an interruption/resumption and power rate variations); $t_i$: release time; $\tau_i$: requested processing time; $\Delta_i$: initial delay; $\delta_i$: overall delay; $t_i + \Delta_i$: admission time; $t_i + \delta_i$: completion time; $\omega_i = \delta_i - \tau_i$: waiting time. The areas under $d_i(t)$ (in light red) and $u_i(t)$ (in light blue) are the same, and equal to $\tau_i$, meaning that at time $t_i + \delta_i$ all the requested power has been supplied. (*source:* [1], © 2022 IEEE)

represent the *desired supply time profile*, normalized with respect to the maximum rate $r_i$, see the red line in Fig. 2.1. The requesting user would like to be immediately supplied at the maximum rate $r_i$ in the minimum time $\tau_i$, which is indeed represented by a rectangular-shaped time profile.

However, if there are too many requesting users, there might be not enough resource to be supplied, so the supply might need to be delayed, or interrupted, or possibly its rate might be reduced. Then, the *actual supplied power profile* $y_i(t)$,

$$0 \leq y_i(t) \leq r_i,$$

is the fraction of the maximum power rate $r_i$ supplied at time $t$, and, in general, it is different from the desired one. The corresponding *normalized decision* variables $u_i(t)$ associated with the supply are therefore defined as

$$u_i(t) = \frac{y_i(t)}{r_i}, \quad 0 \leq u_i(t) \leq 1,$$

see the blue line in Fig. 2.1. Note that $u_i(t) = 0$ if the request is not being supplied at time $t$, and it is 1 if it is being supplied at the maximum rate $r_i$.

The integrals of both $d_i$ and $u_i$ over the time horizon, which represent the amount of requested and supplied resource, respectively, normalized by $r_i$, must be equal, eventually. Indeed, to fulfill a request, the total requested resource $E_i$ must be the same as the supplied one:

$$E_i = r_i \int_0^\infty d_i(t)\ dt = r_i \int_{t_i}^{t_i+\tau_i} 1\ dt = r_i\tau_i \equiv r_i \int_0^\infty u_i(t)\ dt,$$

which is equivalent to the following constraint:

$$\int_0^\infty u_i(t)\ dt \equiv \tau_i. \tag{2.2}$$

Note that it is implicitly assumed that the supplied energy cannot be larger than the requested one.

Recalling that the *release time* $t_i$ is the time in which the request $i$ starts, so that the demand/*desired* supply $d_i(t) = 0$ for all $t < t_i$ and $d_i(t_i) > 0$, and that the *minimum processing time* $\tau_i$ is the duration of the *desired* supply, assuming a constant maximum power rate $r_i$ supplying, so that $d_i(t_i + \tau_i) > 0$ and $d_i(t) = 0$ for all $t > t_i + \tau_i$, the next definitions are introduced. Refer to Fig. 2.1 for a representation of these quantities.

**Definition 2.1:**
The *admission delay* $\Delta_i$ is the time the user waits before the *actual* supply $u_i(t)$ starts, that is

$$\Delta_i = \inf\{\Delta^* \geq 0 : \quad u_i(t_i + \Delta^*) \neq 0\}, \tag{2.3}$$

and the *admission time* $t_i + \Delta_i$ is the corresponding time in which that occurs.

A request is said *fulfilled* at time $t^*$ if and only if the requested energy $E_i = r_i\tau_i$ has been fully supplied at that time, i.e., if the following holds

$$\int_0^{t^*} u_i(t)\ dt = \tau_i, \tag{2.4}$$

and *unfulfilled* otherwise, if $0 \leq \int_0^{t^*} u_i(t)\ dt < \tau_i$.

The *completion time* $c_i$ is the time in which the request first becomes fulfilled, that is

$$c_i \doteq \inf\{t^* > 0 : \text{Eq. (2.4) holds}\},$$

and the *overall delay* $\delta_i = c_i - t_i$ is the corresponding time the user has to wait before that occurs.

Finally, the *waiting time* $\omega_i$ is the additional time needed to fulfill the request compared to $\tau_i$, i.e., the delay the user can complain about, and is defined as

$$\omega_i = \delta_i - \tau_i. \tag{2.5}$$

Two obvious constraints are to be imposed to $u(t)$:

$$u_i(t) = 0,\ \ t < t_i, \tag{2.6a}$$
$$u_i(t) = 0,\ \ t > c_i = t_i + \delta_i; \tag{2.6b}$$

that is, the supply cannot start before the request is made Eq. (2.6a), and must stop once the request is first fulfilled Eq. (2.6b). Then, Eq. (2.2) becomes

$$\int_{t_i}^{t_i+\delta_i} u_i(t)\ dt \equiv \tau_i. \tag{2.7}$$

## 2.2.1 Supply strategies

Three types of simple supply strategies are considered:

- The *non-interruptible* (*NI*) case in which the power is supplied at constant rate $r_i$, without interruptions and with delay $\Delta_i \geq 0$:

$$u_i(t) = d_i(t - \Delta_i) = \begin{cases} 1\,, & \text{if } t \in [t_i + \Delta_i, t_i + \Delta_i + \tau_i]\,, \\ 0\,, & \text{otherwise}\,. \end{cases} \tag{2.8}$$

  In this specific case $\delta_i = \Delta_i + \tau_i$, by construction. See Fig. 2.2, left.

- The *interruptible* (*IT*) case, in which the supplying occurs at a constant rate $r_i$ but can be switched on and off, i.e., interrupted and resumed:

$$u(t) \in \{0, 1\},\ \ t \in [t_i, t_i + \delta_i]. \tag{2.9}$$

  See Fig. 2.2, center.

- The *variable rate* (*VR*) case, in which the supplying rate (power) can vary over time between 0 and $r_i$:

$$0 \leq u_i(t) \leq 1,\ \ t \in [t_i, t_i + \delta_i]. \tag{2.10}$$

  See Fig. 2.2, right.

Note that the non-interruptible *NI* case is a special case of the interruptible *IT* one, which in turn is a special case of the variable-rate *VR* one. Temporarily suspending the supply or reducing its intensity clearly increases the waiting time, but provides a benefit for the overall average delay, because this allows for better exploitation of the capacity of the system.

Moreover, in the *NI* case, the waiting time is equal to the admission delay, $\omega_i \equiv \Delta_i$, by construction. In the more general *IT* and *VR* cases, it might be larger, $\omega_i \geq \Delta_i$, because an additional delay due to the interruption of the supply or the reduction of the supplied power rate must be taken into account.

**Remark 2.1:**
Many more constraints can be included regarding the way in which the supply could be interrupted and resumed (e.g., the minimum or maximum time before interruption or resumption), or its rate could be varied (e.g., the minimum and maximum rate variation over time). Here, only the simpler *NI*, *IT*, and *VR* general classes are considered.

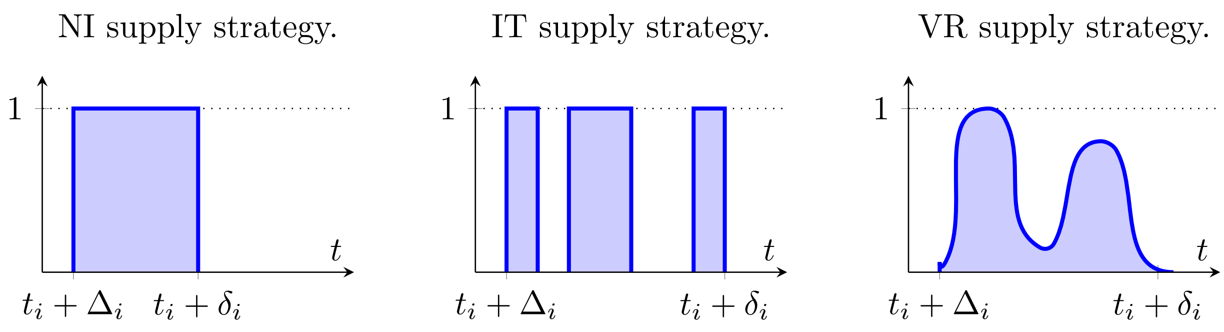


Figure 2.2: The possible supplying strategies.

### 2.2.2 The waiting time as a function of the supply profile

To determine the waiting time $\omega_i$ as a function of the supply $u_i(t)$, the binary variable $z_i(t)$ is defined, which is non-zero and equal to 1 only when the request is still active and some resource needs to be supplied (i.e., between the start of the request at time $t_i$ and the completion of the supplying at time $c_i = t_i + \delta_i$):

$$z_i(t) = \begin{cases} 1\,, & \text{if } t \in [t_i, t_i + \delta_i]\,, \\ 0\,, & \text{otherwise}\,. \end{cases} \tag{2.11}$$

Since, by definition, for any supply strategy $u_i$ is active only in the interval $[t_i, t_i + \delta_i]$, where it can only assume values in $[0, 1]$, and is 0 outside that interval, then, for all $t$,

$$0 \le u_i(t) \le z_i(t) \le 1. \tag{2.12}$$

**Theorem 2.1:**
The waiting time $\omega_i$ can be computed as:

$$\omega_i = \int_0^\infty [z_i(t) - u_i(t)]\, dt. \tag{2.13}$$

### 2.2.3 A state variable accounting for the delay

To study the supplying process within the classical theory of optimal control problems, a dynamic system, representing the state of the system, is introduced, where the following integral state variable $x_i(t)$ accounts for the delay of the supplying. Its dynamics are given by

$$x_i(t) = \frac{1}{\tau_i} \int_0^t \, [d_i(\xi) - u_i(\xi)] d\xi\,, \tag{2.14}$$

or, taking the time derivative,

$$\dot{x}_i(t) = \frac{1}{\tau_i}\, [d_i(t) - u_i(t)]\,, \quad x_i(0) = 0\,. \tag{2.15}$$

$x_i(t)$ represents the unfulfilled request in terms of delay, normalized with respect to $\tau_i$. Note that in the two above expressions Eqs. (2.14) and (2.15) the demand $d_i(t)$ and possibly the supply $u_i(t)$ (depending on the imposed supply strategy) are discontinuous: this is not really a problem, as there are just time discontinuities, and can be faced by considering the right derivative and just using the integral form Eq. (2.14).

**Assumption 2.2: Standing assumption.**
It is imposed that

$$x_i(t) \ge 0, \quad \text{for all} \quad t \ge 0, \quad \text{for all} \quad i. \tag{2.16}$$

Imposing Assumption 2.2 is equivalent to impose that the supply cannot start before $t_i$ and must stop when the required energy has been supplied.

**Proposition 2.1:**
Let Assumption 2.2 be satisfied. The constraint Eq. (2.16) is equivalent to constraints Eqs. (2.6a) and (2.6b).

An advantage of imposing Eq. (2.16) is that, differently from Eq. (2.6b), the knowledge of the completion time $c_i$, which is not known a priori, is not required. The following Proposition follows.

**Proposition 2.2:**
Let Assumption 2.2 be satisfied. The demand is fulfilled at time $t^*$ if and only if $x_i(t) = 0$ for all $t \geq t^* \geq t_i + \tau_i$. Moreover, the overall delay $\delta_i$ is the smallest value for which $x_i(t) = 0$, for $t \geq t_i + \delta_i$,

$$\delta_i \doteq \inf\{\delta^\star \geq \tau_i : \quad x_i(t_i + \delta^\star) = 0\}\,. \tag{2.17}$$

Then, $x_i(t) = 0$ for all $t < t_i$ and $t > t_i + \delta_i$.

Moreover, the relation between variable $x_i(t)$ and with $\Delta_i$, $\omega_i$, and $\delta_i$ is given in the next Theorem.

**Theorem 2.2:**
For any supply strategy $u_i$ such that $u_i \in [0, 1]$, starting at time $t_i + \Delta_i$ and terminating at time $t_i + \delta_i$,

$$\Delta_i \leq \int_0^\infty x_i(t)dt \leq \omega_i < \delta_i\,, \tag{2.18}$$

where equalities hold only in the *NI* case.

This Theorem can be intuitively explained as follows. Consider first the *NI* case: unless $\Delta_i = 0$ (in which case $x_i(t) \equiv 0$), $x_i(t)$ has a trapezoidal profile, see Fig. 2.3. Then, $\int_0^\infty x_i(t)dt$ is simply the area of this shape, and results equal to $\Delta_i$ (see the proof of Theorem in Appendix A).

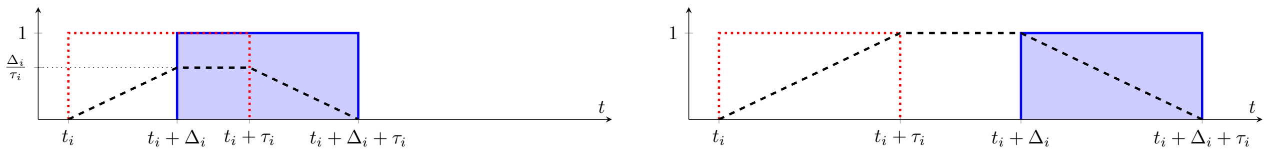


Figure 2.3: NI supply. Red: $d_i(t)$, blue: $u_i(t)$, black: $x_i(t)$. On the left, $\Delta_i \leq \tau_i$; on the right, $\Delta_i \geq \tau_i$. (*source:* [1], © 2022 IEEE)

In a generic supply, $x_i(t)$ has a different profile, see, e.g., Fig. 2.4. Consider the two hypothetical *NI* supplies of duration $\tau_i$ and rate $r_i$ starting at $t_i + \Delta_i$ and $t_i + \omega_i$, respectively (see the yellow and green lines). These correspond to two trapezoidal profiles of the integral variable $x_i$, whose areas are $\Delta_i$ and $\omega_i$, respectively. The actual $x_i(t)$ profile can only lie between these two trapezoids, and so does its area $\int_0^\infty x_i(t)dt$.

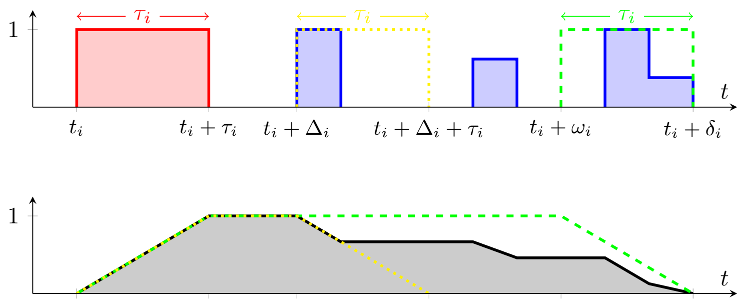


Figure 2.4: Generic supply. Red: $d_i(t)$, blue: the actual supply, $u_i(t)$, black: $x_i(t)$ for the actual supply, yellow: hypothetical supply $d(t - \Delta_i)$, green: hypothetical supply $d(t - \omega_i)$. In the bottom, the corresponding $\int_0^\infty x_i(t)dt$ are reported for such supplies.

## 2.3 The capacity of the system

In the system, there are $n$ requests of supplying occurring at any given time. It is assumed that there is no limitation in the number of possible demands either being requesting or supplied simultaneously. However, there is a limit on the total amount of resource that can be supplied at any time, which indirectly limits those which are being supplied.

Let $y_{tot}(t)$ be the total power supply at a given time,

$$y_{tot}(t) \doteq \sum_{i=1}^{n} r_i u_i(t) = \sum_{i \in \mathcal{A}(t)} r_i u_i(t), \tag{2.19}$$

where $\mathcal{A}(t)$ is the set of unfulfilled requesting users at time $t$, which are being supplied at $t$,

$$\begin{aligned}\mathcal{A}(t) &= \{i : t_i \le t \le c_i\} && (2.20a)\\ &= \{i : x_i(t) > 0 \text{ for } t \ge t + \tau_i\} && (2.20b)\\ &= \{i : \text{ either } x_i(t) > 0 \text{ or } d_i(t) > 0\}. && (2.20c)\end{aligned}$$

Note that since the constraint Eq. (2.16), $x_i(t) \ge 0$, implies that $u_i(t) = 0$ when the demand is inactive, i.e., when $d_i(t) = 0$ and $x_i(t) = 0$, the sum in Eq. (2.19) could be limited to the active demands described by the set $\mathcal{A}(t)$. Recall that the condition $d_i(t) > 0$ means that request $i$ is active at time $t$, hence started no more than $\tau_i$ time units before; $x_i(t) > 0$ means that request i is still unfulfilled.

Then, at any time, the total power supply $y_{tot}(t)$ is limited by a given maximum value $P$, which is the capacity of the system,

$$\sum_{i=1}^{n} r_i u_i(t) \le P\,,\ \forall t \ge 0\,. \tag{2.21}$$

If the total available power $P$ is unlimited, or, more realistically, it is very large, e.g.

$$P \ge \max\left(\sum_{i\in\mathcal{A}(t)} r_i\right),$$

any demand could be trivially admitted with $u_i(t) \equiv d_i(t)$, so that $x_i(t) \equiv 0$ for all $t$, and this is possible for any of the *NI*, *IT*, *VR* cases. Hence, the supply start immediately and is performed at the maximum rate without interruptions, so that $\delta_i = \tau_i$ is the minimum (the admission delay and the waiting time are zero, $\Delta_i = \omega_i = 0$).

**Remark 2.2:**
Constraint Eq. (2.21) is a hard constraint, where $P$ is constant over time.
Two variations are possible:

- the capacity of the system might be time-varying, i.e.,

$$\sum_{i=1}^{n} r_i u_i(t) \le P(t)\,,\ \forall t \ge 0\,;$$

- the constraint is a soft constraint, i.e., $y_{tot}(t)$ could exceed $P$, but a penalization is applied.

## 2.4 The minimization of the delay

When scheduling the supply of a limited resource, different objectives could be considered. Here, the minimization of the *total overall delay* is considered. This is simply defined as

$$\delta = \sum_{i=1}^{n} \delta_i.$$

Then, the objective function to be minimized is

$$J = \delta = \sum_{i=1}^{n} \delta_i.$$

Note that, as of Assumption 2.1, $J$ is finite if all demands are fulfilled.

A different objective function could be considered; indeed, one might want to optimize one of the following quantities, instead of $\delta$:

- the *average overall delay* $\bar{\delta} = \dfrac{1}{n}\displaystyle\sum_{i=1}^{n} \delta_i = \dfrac{\delta}{n}$;
- the *total waiting time* $\omega = \displaystyle\sum_{i=1}^{n} \omega_i = \sum_{i=1}^{n} [\delta_i - \tau_i] = \delta - \sum_{i=1}^{n} \tau_i$;

- the *average waiting time* $\bar{\omega} = \dfrac{1}{n}\sum_{i=1}^{n} \omega_i = \dfrac{\omega}{n} = \dfrac{\delta - \sum_{i=1}^{n} \tau_i}{n}$;
- the *total completion time* $c = \sum_{i=1}^{n} c_i = \sum_{i=1}^{n} [t_i + \delta_i] = \delta + \sum_{i=1}^{n} t_i$;
- the *average completion time* $\bar{c} = \dfrac{1}{n}\sum_{i=1}^{n} c_i = \dfrac{c}{n} = \dfrac{\delta + \sum_{i=1}^{n} t_i}{n}$.

Still, minimizing each of the above-mentioned quantities is equivalent to minimizing $\delta$, because $n$, $\tau_i$, and $t_i$ are given and constant, for all $i$,

$$\arg\min_{\delta_1,\dots,\delta_n} \delta = \arg\min_{\delta_1,\dots,\delta_n} \bar{\delta} = \arg\min_{\delta_1,\dots,\delta_n} \omega = \arg\min_{\delta_1,\dots,\delta_n} \bar{\omega} = \arg\min_{\delta_1,\dots,\delta_n} c = \arg\min_{\delta_1,\dots,\delta_n} \bar{c}.$$

### 2.4.1 Linear Programming and Mixed-Integer Linear Programming problems

In the problem considered here, the solution minimizing the objective function must satisfy some given constraints. If the objective function is linear, and the constraints can be expressed by linear relations, after discretization the problem becomes a Linear Programming (LP) problem.

A generic linear programming problem with $q$ variables described by vector $\eta \in \mathbb{R}^{q\times 1}$, $r$ inequality constraints and $s$ equality constraints, can be written in the form

$$\begin{aligned} \min_{\eta} \quad & f^\top \eta, && (2.22a)\\ \text{s.t.} \quad & A \cdot \eta \le b, && (2.22b)\\ & A_{eq} \cdot \eta = b_{eq}, && (2.22c)\\ & lb \le \eta \le ub, && (2.22d) \end{aligned}$$

where the matrices $A \in \mathbb{R}^{r\times q}$, $A_{eq} \in \mathbb{R}^{s\times q}$ and the vectors $f, lb, ub \in \mathbb{R}^{q\times 1}$, $b \in \mathbb{R}^{r\times 1}$, $b_{eq} \in \mathbb{R}^{s\times 1}$ are to be defined. A problem of this kind can be solved very efficiently by existing tools, even for large instances of the problem.

Sometimes, a subset of the variables of vector $\eta$ needs to be integer. In this case, the problem becomes a Mixed-Integer Linear Programming (MILP) problem, which can be written in the form

$$\begin{aligned} \min_{\eta} \quad & f^\top \eta, && (2.23a)\\ \text{s.t.} \quad & A \cdot \eta \le b, && (2.23b)\\ & A_{eq} \cdot \eta = b_{eq}, && (2.23c)\\ & lb \le \eta \le ub, && (2.23d)\\ & \eta(I) \in \mathbb{Z}, && (2.23e) \end{aligned}$$

where the matrices $A \in \mathbb{R}^{r\times q}$, $A_{eq} \in \mathbb{R}^{s\times q}$ and the vectors $f, lb, ub \in \mathbb{R}^{q\times 1}$, $b \in \mathbb{R}^{r\times 1}$, $b_{eq} \in \mathbb{R}^{s\times 1}$ are to be defined, and $I$ denotes the indices of the elements of $\eta$ which are imposed to be integer.

A MILP problem is harder to solve, compared to the LP case.

## 2.5 Problem statement

The main problem is formulated as follows in generic terms.

**Problem 2.1:**
Given $n$ demands starting at times $t_1, t_2, \dots, t_n$ with power requests $r_1, \dots, r_n$, under Assumption 2.1, and durations $\tau_1, \dots, \tau_n$, respectively, find a supply strategy $u_1(t), \dots u_n(t)$, either *NI*, or *IT*, or *VR*, under constraints Eqs. (2.16) and (2.21), which minimizes

$$J(u_1, \dots, u_n) \doteq \sum_{i=1}^{n} \delta_i\,. \tag{2.24}$$

Clearly, the problem solution depends on the type of supplying strategy that is to be imposed.

For simplicity, it is assumed that the supply strategy is the same for each user. Then, three cases are considered:

- solve Problem 2.1 using the *NI* supply strategy for each request;

- solve Problem 2.1 using the $IT$ supply strategy for each request;
- solve Problem 2.1 using the $VR$ supply strategy for each request.

The extension to the case in which requests are partitioned into three groups, each one characterized by a specific supply strategy, $NI$, $IT$ and $VR$, is straightforward.

**Remark 2.3:**
The problem of minimizing $J = \Delta$ is equivalent to minimizing $\omega = \sum_i \omega_i$ or $c = \sum_i c_i = \sum_i \, [t_i + \delta_i]$. Interestingly, the information about $\tau_i$ is fundamental in the optimization. Indeed, the constant $\tau_i$, along with $r_i$ can be used by the scheduler to privilege "modest" requests with respect to the more demanding ones.

# Chapter 3
# The optimal control framework

Under the structure presented in Chapter 2, Problem 2.1 can be formulated as an optimal control problem aimed at minimizing the total delay. A common framework is specified, which is independent from the chosen supply strategy. The specific class of supply *NI*, *IT*, or *VR* is specified by means of some additional constraints on $u_i(t)$.

In particular, this framework is

$$\min_{u_1,\ldots,u_n} \; J(u_1,\ldots,u_n) = \sum_{i=1}^{n} \delta_i \,, \tag{3.1a}$$

$$\text{s.t.} \;\; \dot{x}_i(t) = \frac{1}{\tau_i}\,[d_i(t) - u_i(t)] \,, \tag{3.1b}$$

$$d_i(t) = \begin{cases} 1 & \text{if } \; t_i \le t \le t_i + \tau_i \,, \\ 0 & \text{otherwise,} \end{cases} \tag{3.1c}$$

$$0 \le u_i(t) \le 1 \,, \quad \text{for all } t \ge 0, \tag{3.1d}$$

$$x_i(0) = 0 \,, \tag{3.1e}$$

$$x_i(t) \ge 0 \,, \quad \text{for all } t \ge 0, \tag{3.1f}$$

$$x_i(t) = 0 \,, \quad \text{for all } t \ge t_i + \delta_i, \tag{3.1g}$$

$$\sum_{i=1}^{n} r_i u_i(t) \le P, \quad \text{for all } t \ge 0 \,, \tag{3.1h}$$

$$(\textit{additional constraints specifying the supply strategy}). \tag{3.1i}$$

Constraints Eqs. (3.1b) to (3.1h), are common to all *NI*, *IT*, *VR* cases, and gather all the constraints introduced in Chapter 2. Recall that, for all $i$, Eq. (3.1f) implies that $x_i(t) = 0$ for all $t < t_i$ and for all $t \ge t_i + \delta_i$, and is equivalent to constraints Eqs. (2.6a) and (2.6b), see Propositions 2.1 and 2.2. Also, recall that the discontinuity of the demand $d_i(t)$ and possibly the supply $u_i(t)$ in Eq. (3.1b) can be faced by considering the right derivative or just using the integral form Eq. (2.14).

The difference is the chosen class of input functions, to be specified in Eq. (3.1i), where the optimal control is to be searched. Noting that Eq. (3.1d) holds for any supply strategy considered here, there are three cases:

- in the *NI* case, $u(t) \in \{0, 1\}$, and, in particular, the inputs $u_i$ are constant and equal to 1 on intervals of size $\tau_i$, with no interruptions;
- in the *IT* case, $u_i(t) \in \{0, 1\}$, so that the inputs $u_i$ are off-on functions, with switching;
- in the *VR* case, $u_i(t) \in [0, 1]$, so the inputs $u_i$ are piecewise-continuous functions ( Eq. (3.1d) is enough).

The specifications of these constraints in Eq. (3.1i) are detailed in the next Sections. In general, as different constraints are to be imposed depending on the chosen supply strategy, the optimal solution will be different. Then, the optimal values of the cost functionals for the three problems will be denoted by $J^*_{NI}$, $J^*_{IT}$ and $J^*_{VR}$, respectively.

Recalling that the classes of functions for $u_i(t)$ are nested, i.e., a constant function on an interval is a special case of a switching function in $\{0, 1\}$, which is a piecewise continuous function, it necessarily holds that

$$J^*_{NI} \ge J^*_{IT} \ge J^*_{VR}. \tag{3.2}$$

**Remark 3.1:**
The proposed optimal control framework, if opportunely modified, could support some different features.

A (finite) deadline $\delta_{max,i}$ for each user $i$, i.e., an upper bound on the completion time $t_i + \delta_i$, can be imposed by adding the constraint $\delta_i \le \delta_{max,i}$, that is:

$$u_i(t) = 0, \quad \text{for all } t \ge t_i + \delta_{max,i}.$$

The deadline could be imposed as a soft constraint, i.e., the supply can occur after $t_i + \delta_{max,i}$, but a penalization is imposed, which is to be minimized. In this case, the objective function is modified into $J'$, e.g,

$$J' = \alpha J + (1-\alpha) \sum_{i=1}^{n} \int_{t_i+\delta_{max,i}}^{\infty} u_i^2(t)\ dt,$$

where $0 < \alpha < 1$ is given. The new term minimizes the supply to each user after its deadline.

Time variability of the available power $P$ can be easily handled by considering a time-varying function $P(t)$ in Eq. (3.1h).
A soft constraint on the maximum power, i.e., the supply can exceed $P$, but a penalization is imposed, which is to be minimized, can be handled by modifying the objective function into $J'$, e.g,

$$J' = \alpha J + (1-\alpha) \int_0^{\infty} \left[ \max\left(0, \sum_{i=1}^{n} r_i u_i(t) - P\right)\right]^2 \ dt,$$

with $0 < \alpha < 1$ is given. The new term minimizes the excess total power supplied above $P$.

The main problem of formulation Eq. (3.1) is the characterization of variables $\delta_i$ in Eq. (3.1a) with respect to variables $u_i$. Then, a possibility is to solve Problem 2.1 approximately, by replacing function Eq. (3.1a) with other functionals suitable for computation.

Recall that from Theorem 2.2,

$$\Delta_i \le \int_0^{\infty} x_i(t)dt \le \omega_i < \delta_i\,,$$

and, in particular, for the *NI* case,

$$\Delta_i = \int_0^{\infty} x_i(t)dt = \omega_i < \delta_i \quad (= \Delta_i + \tau_i)\,.$$

Then, the problem of minimizing $J(u)$ can be heuristically faced by minimizing

$$\sum_{i=1}^{n} \int_0^{\infty} x_i(t)dt, \tag{3.3}$$

which is an optimal control problem that is easier to solve and leads to a lower bound for the optimal $\delta$, in general, and to the exact optimal solution $J^*_{NI}$ in the specific *NI* case. Indeed, the values of $\delta_i$ are not needed to be determined.

**Theorem 3.1:**
Any solution with NI supplies that minimizes Eq. (3.3) solves the problem of minimizing Eq. (2.24).

### Some considerations about the complexity

To study the complexity of the problem, consider an equivalent system with $n$ independent jobs to be scheduled (the requests) and, as there is no limitation on the simultaneous number of active requests, $n$ identical machines in parallel.

Then, the three considered problems *NI*, *IT* and *VR* can be classified, according to [46], as *identical parallel machines problems of independent tasks with release times $t_i$ and resource constraints* [49], as well as *preemption* (for *IT* case) and *time-varying resource allocation* (for *VR* case), *minimizing the total completion time* $\sum_i C_i = \sum_i t_i + \delta_i$, which is equivalent to minimizing Eq. (2.24) as $t_i$ is not a decision variable.

The *NI* and *IT* problems are NP-hard, as they can be reduced, respectively, to the *single-machine problem with release times minimizing* $\sum_i C_i$ [43, 50] and to the *two-identical-machine problem with release times and preemption minimizing* $\sum_i C_i$ [43, 51], which are known to be NP-hard.

Instead, for the *VR* problem, a thorough search of the relevant literature yielded that the closest work is [52], which considers a time-varying resource allocation profile for each user, stating that some general problems of minimizing $\sum_i C_i$ are NP-hard. The *VR* problem is indeed a generalization of the *IT* case, thus it is conjectured that the problem remains NP-hard.

The relaxed problem Eq. (3.3) under constraints Eqs. (3.1b) to (3.1h) is convex and linear. It can be solved in polynomial time.

## 3.1 The non-interruptible case

In this Section, the optimal control problem is specified for the non-interruptible ($NI$) case. For each request, the power is supplied at a constant rate $r_i$, without interruptions and with delay $\Delta_i \geq 0$; Eq. (3.1i) is given by Eq. (2.8), reported next in Eq. (3.4i).

In this case, for each request $i$, the supply profile is entirely specified by means of a single parameter, the initial delay $\Delta_i$, and the overall delay is

$$\delta_i = \Delta_i + \tau_i,$$

so that the $\Delta_i$ are the decision variables, or, equivalently, the initial delays $t_i + \Delta_i$ are such. Then, the functional for this version of the problem is $J_{NI}(u_1, \dots, u_n) = J_{NI}(\Delta_1, \dots, \Delta_n)$. Recall that, exploiting Theorem 2.2, it now holds that

$$\Delta = \sum_{i=1}^{\infty} \Delta_i = X = \sum_{i=1}^{\infty} \int_0^{\infty} x_i(t)dt = \omega = \sum_{i=1}^{\infty} \omega_i = \sum_{i=1}^{\infty} [\delta_i - \tau_i] = \delta - \sum_{i=1}^{\infty} \tau_i,$$

so that the optimal cost minimizing the total accounted delay $\sum_{i=1}^{\infty} \int_0^{\infty} x_i(t)dt$, is indeed the optimal waiting time $\omega$, and the optimal initial delay $\Delta$.

Then, the optimal control problem of minimizing the total overall delay $\delta$ can be formulated exploiting Theorem 3.1, assuming the constraints Eqs. (3.1b) to (3.1h) as well as the additional constraint Eq. (2.8) on $u_i$, as

$$\min_{\Delta_1, \dots, \Delta_n} J_{NI}(\Delta_1, \dots, \Delta_n) = \int_0^{\infty} \sum_{i=1}^{n} x_i(t)dt, \tag{3.4a}$$

$$\text{s.t. } \dot{x}_i(t) = \frac{1}{\tau_i}\left[d_i(t) - u_i(t)\right], \tag{3.4b}$$

$$d_i(t) = \begin{cases} 1 & \text{if } t_i \leq t \leq t_i + \tau_i\,, \\ 0 & \text{otherwise,} \end{cases} \tag{3.4c}$$

$$0 \leq u_i(t) \leq 1\,, \quad \text{for all } t \geq 0, \tag{3.4d}$$

$$x_i(0) = 0\,, \tag{3.4e}$$

$$x_i(t) \geq 0\,, \quad \text{for all } t \geq 0, \tag{3.4f}$$

$$x_i(t) = 0\,, \quad \text{for all } t \geq t_i + \delta_i, \tag{3.4g}$$

$$\sum_{i=1}^{n} r_i u_i(t) \leq P, \quad \text{for all } t \geq 0\,, \tag{3.4h}$$

$$u_i(t) = \begin{cases} 1\,, & \text{if } t \in [t_i + \Delta_i, t_i + \Delta_i + \tau_i]\,, \\ 0\,, & \text{otherwise}\,. \end{cases} \tag{3.4i}$$

The major issue in the exact solution is the constraint Eq. (3.4i), which imposes that, once an user is admitted, it is supposed to remain connected for the whole requested period $[t_i + \Delta_i, t_i + \tau_i + \Delta_i]$. A constraint of this kind will be referred to as *contiguity constraint*.

### 3.1.1 Discrete-time implementation: Mixed integer programming using the LC and SU techniques

For computation and to reach a solution, problem Eq. (3.4) must be discretized. Assuming a sampling period of $\theta$ and a horizon of $T = N\theta$ of $N + 1$ time-slots, consider the discrete time instants $t_k = k\theta$, $k = 0, 1, 2, \dots, N$ and denote

$$x_i(k) \doteq x_i(t_k), \quad u_i(k) \doteq u_i(t_k), \quad d_i(k) \doteq d_i(t_k).$$

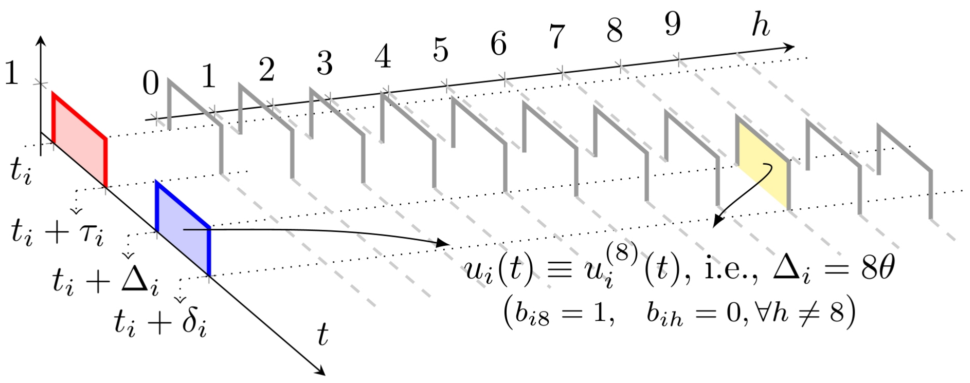

1
$t_i$
$t_i + \tau_i$
$t_i + \Delta_i$
$t_i + \delta_i$
$t$
0 1 2 3 4 5 6 7 8 9 $h$
$u_i(t) \equiv u_i^{(8)}(t)$, i.e., $\Delta_i = 8\theta$
$(b_{i8} = 1, \quad b_{ih} = 0, \forall h \neq 8)$

Figure 3.1: The Linear Combination (LC) technique for implementing contiguity constraint Eq. (3.4i), in the non-interruptible *NI* case. Red: demand profile $d_i(t)$; blue: supply function $u_i(t)$ (with delay $\Delta_i$); gray: the candidate profiles $u_i^{(h)}(t)$ for the optimal control. The candidate profile with $h = 8$ (highlighted in yellow) is to be selected: from Eq. (3.5), setting $b_{i8} = 1$ and $b_{ih} = 0$ for all $h \neq 8$, $u_i(t) = u_i^{(8)}(t)$. In the specific example, $\tau_i = 4\theta, \Delta_i = 8\theta$.

Then, Eq. (3.4b) is discretized with sampling time $\theta$ using the Euler method as

$$x_i(k+1) = x_i(k) + \frac{\theta}{\tau_i}\left[d_i(k) - u_i(k)\right],.$$

Note that as time is discretized, also the *actual* possible values assumed by the release time $t_i$, the processing time $\tau_i$, the delay $\Delta_i$, waiting time $\omega_i$ and overall delay $\delta_i$ are discretized, too: $t_i, \tau_i, \Delta_i, \omega_i, \delta_i \in \{h\theta : h = 0, 1, \dots\}$. Moreover, any sampled–data solution can be seen as a special continuous-time solution, so the ideal continuous-time optimal cost is in general smaller than the sampled-data one.

**Proposition 3.1:**
When the system is sampled with sampling time $\theta$, the worst-case performance loss is not greater than $n(n+1)\theta$.

To ensure the contiguity constraint Eq. (3.4i), two equivalent formulations, i.e., *Linear Combination* and *Start–ups*, are suggested, which however results in a mixed-integer linear programming problem.

## Linear Combination (LC) formulation for the contiguity constraints.

The first alternative form for contiguity constraint Eq. (3.4i) consists in writing $u_i(k)$ as a linear combination of all the possible profiles it can assume, selecting only one of them by means of some new decision variables.

Consider a set of possible values for the delay $\mathcal{H} = \{h\theta \;:\; h = 0, 1, \dots, H_i\}$, where $H_i = N - t_i - \tau_i + 1$ is the maximum one, and define, for each $\Delta_h \in \mathcal{H}$, the continuous-time signal

$$u_i^{(h)}(t) = d_i(t - \Delta_h) = \begin{cases} 1\,, & \text{if } t \in [t_i + \Delta_h, t_i + \Delta_h + \tau_i]\,, \\ 0\,, & \text{otherwise}\,. \end{cases}$$

which is $d(t)$ delayed by $\Delta_h$, so that $u_i^{(0)}(t) = d_i(t)$.

Any $u_i^{(h)}(k)$ is a (discrete-time) candidate profile for the optimal control $u_i(k)$: only one of them must be selected for each request $i$, see Fig. 3.1. Thus, the actual control Eq. (3.4i) is written as a linear convex combination with binary coefficients and is replaced by:

$$u_i(k) = \sum_{h=0}^{H_i} b_{ih} u_i^{(h)}(k), \tag{3.5a}$$

$$\sum_{h=0}^{H_i} b_{ih} = 1, \tag{3.5b}$$

$$b_{ih} \in \{0, 1\}. \tag{3.5c}$$

Note that Eq. (3.5) imposes that only one specific *NI* profile $u_i^{(h)}(k)$ is selected, as there can be only one non-zero $b_{ih}$.

Then, after discretization, problem Eq. (3.4) reduces to the following MILP problem:

$$\min_{\Delta_1,\ldots,\Delta_n} \; J_{NI}(\Delta_1,\ldots,\Delta_n) = \theta \sum_{k=0}^{N} \; \sum_{i=1}^{n} x_i(k), \tag{3.6a}$$

$$\text{s.t.} \;\; x_i(k+1) = x_i(k) + \frac{\theta}{\tau_i}\,[d_i(k) - u_i(k)]\,, \tag{3.6b}$$

$$d_i(k) = \begin{cases} 1 & \text{if } \; t_i \le k\theta \le t_i + \tau_i\,, \\ 0 & \text{otherwise,} \end{cases} \tag{3.6c}$$

$$0 \le u_i(k) \le 1\,, \quad \text{for all } k \ge 0, \tag{3.6d}$$

$$x_i(0) = 0\,, \tag{3.6e}$$

$$x_i(k) \ge 0\,, \quad \text{for all } k \ge 0, \tag{3.6f}$$

$$x_i(k) = 0\,, \quad \text{for all } k\theta \ge t_i + \delta_i, \tag{3.6g}$$

$$\sum_{i=1}^{n} r_i u_i(k) \le P, \quad \text{for all } k \ge 0\,, \tag{3.6h}$$

$$u_i(k) = \sum_{h=0}^{H_i} b_{ih} u_i^{(h)}(k), \tag{3.6i}$$

$$\sum_{h=0}^{H_i} b_{ih} = 1, \tag{3.6j}$$

$$b_{ih} \in \{0,1\}. \tag{3.6k}$$

Problem Eq. (3.6) can be simply implemented as a MILP problem Eq. (2.23) by taking

$$\begin{aligned}
\eta^\top &= \left[\begin{array}{ccc:c:ccc|ccc:c:ccc} x_1(0) & \cdots & x_1(N) & \cdots & x_n(0) & \cdots & x_n(N) & b_{1,0} & \cdots & b_{1,H_1} & \cdots & b_{n,0} & \cdots & b_{n,H_n} \end{array}\right], \\
f^\top &= \left[\begin{array}{ccc:c:ccc|ccc:c:ccc} 1 & \cdots & 1 & \cdots & 1 & \cdots & 1 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \end{array}\right], \\
lb^\top &= \left[\begin{array}{ccc:c:ccc|ccc:c:ccc} 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \end{array}\right], \\
ub^\top &= \left[\begin{array}{ccc:c:ccc|ccc:c:ccc} 1 & \cdots & 1 & \cdots & 1 & \cdots & 1 & 1 & \cdots & 1 & \cdots & 1 & \cdots & 1 \end{array}\right], \\
& b_{ih} \in \mathbb{Z}, \;\; \forall i = 1,\ldots n, \;\; h = 0,\ldots,H_i,
\end{aligned}$$

$$A = \left[\begin{array}{c:c:c|c:c:c} [0] & \cdots & [0] & r_1 U_1 & \cdots & r_n U_n \end{array}\right], \quad b = \begin{bmatrix} P \\ \vdots \\ P \end{bmatrix},$$

$$A_{eq} = \left[\begin{array}{c:c:c|c:c:c}
\tau_1 M & \cdots & [0] & \theta U_1 & \cdots & [0] \\ \hdashline
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \hdashline
[0] & \cdots & \tau_n M & [0] & \cdots & \theta U_n \\ \hline
 & & & 1 \;\cdots\; 1 & \cdots & 0 \;\cdots\; 0 \\
[0] & \cdots & [0] & & \ddots & \\
 & & & 0 \;\cdots\; 0 & \cdots & 1 \;\cdots\; 1
\end{array}\right], \quad b_{eq} = \left[\begin{array}{c} \theta d_1(1) \\ \vdots \\ \theta d_1(N) \\ \hdashline \vdots \\ \hdashline \theta d_n(1) \\ \vdots \\ \theta d_n(N) \\ \hline 1 \\ \vdots \\ 1 \end{array}\right],$$

where $U_i$ is the $(N+1) \times (H_i+1)$ matrix whose columns are the possible $u_i^{(h)}(t)$ profiles for request $i$, which are selected by $b_{i,h}$

$$U_i = \begin{bmatrix} u_i^{(0)}(0) & u_i^{(1)}(0) & \cdots & u_1^{(H_i)}(0) \\ u_i^{(0)}(1) & u_i^{(1)}(1) & \cdots & u_1^{(H_i)}(1) \\ \vdots & \vdots & \ddots & \vdots \\ u_i^{(0)}(N) & u_i^{(1)}(N) & \cdots & u_1^{(H_i)}(N) \end{bmatrix},$$

and $M$ is the $(N+1)\times(N+1)$ matrix defined as

$$M = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ -1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 & 0 \\ & & & \ddots & & & \\ 0 & 0 & 0 & \cdots & -1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & -1 & 1 \end{bmatrix}.$$

Once the solution is found, the actual delay $\Delta_i$ could be computed by

$$\Delta_i = \Delta_h, \text{with } h \text{ such that } b_{ih} = 1.$$

**Remark 3.2:**
This formulation with LC constraints also allows solving the same problem in a more general case in which a predefined non-rectangular shaped fixed power profile $d_i(t)$ is imposed for the supply, that is given by

$$d_i(t) = \begin{cases} f(t) & \text{if } \ t_i \le t \le t_i + \tau_i', \\ 0 & \text{otherwise,} \end{cases}$$

with $\tau_i' > \tau_i$ given and such that

$$\int_0^\infty d_i(t) = \int_{t_i}^{t_i+\tau_i'} f(t) = \tau_i.$$

Then, given the admission delay $\Delta_i$, the supply is $u_i(t) = d_i(t-\Delta_i)$. Hence, in this case,

$$u_i^{(h)}(t) = d_i(t-\Delta_h) = \begin{cases} f(t-\Delta_h)\,, & \text{if } t \in [t_i+\Delta_h, t_i+\delta_h]\,, \\ 0\,, & \text{otherwise}\,. \end{cases}$$

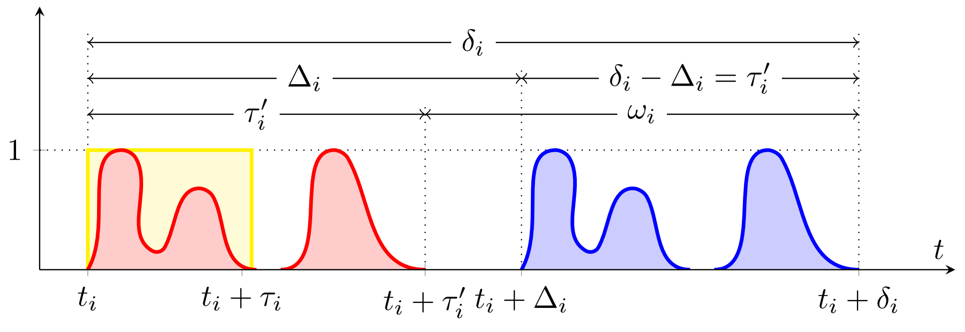


Figure 3.2: A predefined non-rectangular shaped fixed power profile for request $i$. Red: the demand profile $d_i(t)$; blue: the supply function $u_i(t)$, which is a delayed version of $d_i(t)$; $t_i$: release time; $\tau_i'$: requested processing time; $\tau_i < \tau_i'$: minimum requested processing time, assuming maximum constant power supply without interruptions (see the yellow line); $\Delta_i$: initial delay; $\delta_i$: overall delay; $t_i + \Delta_i$: admission time; $t_i + \delta_i$: completion time; $\omega_i = \delta_i - \tau_i'$: waiting time. The areas under $d_i(t)$ (in light red) and $u_i(t)$ (in light blue) are the same, and equal to $\tau_i$, meaning that at time $t_i + \delta_i$ all the requested power has been supplied.

Note that in this case, it still holds that

$$\Delta_i = \int_0^\infty x_i(t)dt = \omega_i < \delta_i \quad (= \Delta_i + \tau_i')\,.$$

Indeed, $\Delta_i = \omega_i$ and $\delta_i = \Delta_i + \tau_i'$ by construction. Also, the integral is equal to $\Delta_i$ for the reasons exposed next.

Firstly, it is easy to verify that, if $\Delta_i \geq \tau_i'$,

$$x(t) = \frac{1}{\tau_i}\int_0^t d(\xi) - u(\xi) d\xi = \begin{cases} 0, & \text{if } \ t < t_i, \\ \frac{1}{\tau_i}\int_{t_i}^t d(\xi)d\xi, & \text{if } \ t_i \leq t \leq t_i + \tau_i', \\ 1, & \text{if } \ t_i + \tau_i' < t < t_i + \Delta_i, \\ 1 - \frac{1}{\tau_i}\int_{t_i+\Delta_i}^t u(\xi)d\xi, & \text{if } \ t_i + \Delta_i \leq t \leq t_i + \Delta_i + \tau_i', \\ 0, & \text{if } \ t > t_i + \Delta_i + \tau_i'. \end{cases}$$

Note that

$$\frac{1}{\tau_i}\int_{t_i+\Delta_i}^t u(\xi) \ \ d\xi = \frac{1}{\tau_i}\int_{t_i+\Delta_i}^t d(\xi - \Delta_i) \ \ d\xi = \frac{1}{\tau_i}\int_{t_i}^{t-\Delta_i} d(\psi) \ \ d\psi.$$

Then,

$$\begin{aligned} \int_0^\infty x(t)dt &= \int_0^{t_i} x(t)dt + \int_{t_i}^{t_i+\tau_i'} x(t)dt + \int_{t_i+\tau_i'}^{t_i+\Delta_i} x(t)dt + \int_{t_i+\Delta_i}^{t_i+\Delta_i+\tau_i'} x(t)dt + \int_{t_i+\Delta_i+\tau_i'}^\infty x(t)dt \\ &= 0 + \int_{t_i}^{t_i+\tau_i'} \left[\frac{1}{\tau_i}\int_{t_i}^t d(\xi)d\xi\right] dt + (t_i + \Delta_i - (t_i + \tau_i')) + \int_{t_i+\Delta_i}^{t_i+\Delta_i+\tau_i'} \left[1 - \frac{1}{\tau_i}\int_{t_i}^{t-\Delta_i} d(\xi)d\xi\right] dt + 0 \\ &= \int_{t_i}^{t_i+\tau_i'} \left[\frac{1}{\tau_i}\int_{t_i}^t d(\xi)d\xi\right] dt + (\Delta_i - \tau_i') + (t_i + \Delta_i + \tau_i' - (t_i + \Delta_i)) - \int_{t_i}^{t_i+\tau_i'} \left[\frac{1}{\tau_i}\int_{t_i}^{t'} d(\xi)d\xi\right] dt \\ &= \Delta_i. \end{aligned}$$

Similarly, if $0 \leq \Delta_i < \tau_i'$,

$$x(t) = \frac{1}{\tau_i}\int_0^t d(\xi) - u(\xi) d\xi = \begin{cases} 0, & \text{if } \ t < t_i, \\ \frac{1}{\tau_i}\int_{t_i}^t d(\xi)d\xi, & \text{if } \ t_i \leq t \leq t_i + \Delta_i, \\ \frac{1}{\tau_i}\int_{t_i}^t d(\xi)d\xi - \frac{1}{\tau_i}\int_{t_i+\Delta_i}^t u(\xi)d\xi, & \text{if } \ t_i + \Delta_i < t < t_i + \tau_i', \\ 1 - \frac{1}{\tau_i}\int_{t_i+\Delta_i}^t u(\xi)d\xi, & \text{if } \ t_i + \tau_i' \leq t \leq t_i + \Delta_i + \tau_i', \\ 0, & \text{if } \ t > t_i + \Delta_i + \tau_i'. \end{cases}$$

Then,

$$\begin{aligned} \int_0^\infty x(t)dt = &\int_{t_i}^{t_i+\Delta_i} \left[\frac{1}{\tau_i}\int_{t_i}^t d(\xi)d\xi\right] dt + \int_{t_i+\Delta_i}^{t_i+\tau_i'} \left[\frac{1}{\tau_i}\int_{t_i}^t d(\xi)d\xi - \frac{1}{\tau_i}\int_{t_i+\Delta_i}^t u(\xi)d\xi\right] dt + \\ &\int_{t_i+\tau_i'}^{t_i+\Delta_i+\tau_i'} \left[1 - \frac{1}{\tau_i}\int_{t_i+\Delta_i}^t u(\xi)d\xi\right] dt \\ = &\int_{t_i}^{t_i+\tau_i'} \left[\frac{1}{\tau_i}\int_{t_i}^t d(\xi)d\xi\right] dt - \int_{t_i+\Delta_i}^{t_i+\Delta_i+\tau_i'} \left[\frac{1}{\tau_i}\int_{t_i}^{t-\Delta_i} d(\psi)d\psi\right] dt + \Delta_i \\ = &\int_{t_i}^{t_i+\tau_i'} \left[\frac{1}{\tau_i}\int_{t_i}^t d(\xi)d\xi\right] dt - \int_{t_i}^{t_i+\tau_i'} \left[\frac{1}{\tau_i}\int_{t_i}^{t'} d(\psi)d\psi\right] dt + \Delta_i \\ = &\Delta_i. \end{aligned}$$

**Start–ups (SU) formulation for the contiguity constraints.**

The second alternative form for contiguity constraint Eq. (3.4i) is based on [53] and consists in limiting the number of "start-ups", i.e., transitions from 0 to 1, to one for variable $u_i(k)$.

The additional constrained variable $s_i(k)$ is introduced, as well as the following constraints

$$0 \leq s_i(k) \leq 1, \tag{3.8a}$$

$$s_i(0) \geq u_i(0), \tag{3.8b}$$

$$s_i(k) \geq u_i(k) - u_i(k-1), \ \ k > 0, \tag{3.8c}$$

$$\sum_k s_i(k) = 1, \tag{3.8d}$$

$$u_i(k) = 0, \ \ k < t_i, \tag{3.8e}$$

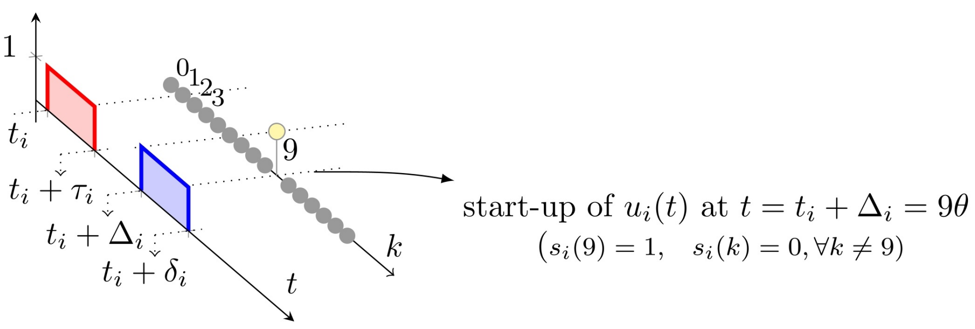

1
0 1 2 3
$t_i$
9
$t_i+\tau_i$
$t_i+\Delta_i$
$t_i+\delta_i$
$k$
$t$
start-up of $u_i(t)$ at $t=t_i+\Delta_i=9\theta$
$(s_i(9)=1,\quad s_i(k)=0,\forall k\neq 9)$

Figure 3.3: The Start-Up (SU) technique for implementing contiguity constraint Eq. (3.4i), in the non-interruptible *NI* case. Red: demand profile $d_i(t)$; blue: supply function $u_i(t)$ (with delay $\Delta_i$); gray: the discrete-time function $s_i(k)$. There is only one non-zero value of $s_i(k)$, located at $t = t_i + \Delta_i = 9\theta$, i.e., when the supply starts at admission time $t_i + \Delta_i$. In the specific example, $\tau_i = 4\theta, \Delta_i = 8\theta$.

$$\theta \sum_k u_i(k) = \tau_i, \tag{3.8f}$$
$$u_i(k) \in \{0,1\}. \tag{3.8g}$$

Since $u_i(k)$ is a binary variable for the *NI* case, the term $u_i(k) - u_i(k-1)$ in Eq. (3.8c) can take only three values: 1 (when there is a start-up), $-1$ (when there is a transition from 1 to 0), or 0 (when there is no transition). Then, by Eqs. (3.8a) to (3.8c), $s_i(k)$ is forced to be 1 when there is a start-up (i.e., at the admission instant $t_i + \Delta_i$), while it is unrestricted in the other cases. Finally, constraint Eq. (3.8d) ensures that there is only one non-zero $s_i(k)$ in the considered time horizon, i.e., only one start-up. See Fig. 3.3.

Then, after discretization, problem Eq. (3.4) reduces to the following MILP problem:

$$\min_{\Delta_1,\dots,\Delta_n} \; J_{NI}(\Delta_1,\dots,\Delta_n) = \theta \sum_{k=0}^{N} \sum_{i=1}^{n} x_i(k), \tag{3.9a}$$
$$\text{s.t.} \quad x_i(k+1) = x_i(k) + \frac{\theta}{\tau_i}\left[d_i(k) - u_i(k)\right], \tag{3.9b}$$
$$d_i(k) = \begin{cases} 1 & \text{if } \; t_i \le k\theta \le t_i + \tau_i\,, \\ 0 & \text{otherwise,} \end{cases} \tag{3.9c}$$
$$0 \le u_i(k) \le 1\,, \quad \text{for all } k \ge 0, \tag{3.9d}$$
$$x_i(0) = 0\,, \tag{3.9e}$$
$$x_i(k) \ge 0\,, \quad \text{for all } k \ge 0, \tag{3.9f}$$
$$x_i(k) = 0\,, \quad \text{for all } k\theta \ge t_i + \delta_i, \tag{3.9g}$$
$$\sum_{i=1}^{n} r_i u_i(k) \le P, \quad \text{for all } k \ge 0\,, \tag{3.9h}$$
$$0 \le s_i(k) \le 1, \tag{3.9i}$$
$$s_i(0) \ge u_i(0), \tag{3.9j}$$
$$s_i(k) \ge u_i(k) - u_i(k-1),\; k > 0, \tag{3.9k}$$
$$\sum_k s_i(k) = 1, \tag{3.9l}$$
$$u_i(k) = 0,\; k < t_i, \tag{3.9m}$$
$$\theta \sum_k u_i(k) = \tau_i, \tag{3.9n}$$
$$u_i(k) \in \{0,1\}. \tag{3.9o}$$

Problem Eq. (3.9) can be simply implemented as a MILP problem Eq. (2.23) by taking

$$\eta^\top = \left[\begin{array}{cccccccc|cccccccc|cccccccc} x_1(0) & \cdots & x_1(N) & \vdots & \cdots & \vdots & x_n(0) & \cdots & x_n(N) & u_1(0) & \cdots & u_1(N) & \vdots & \cdots & \vdots & u_n(0) & \cdots & u_n(N) & s_1(0) & \cdots & s_1(N) & \vdots & \cdots & \vdots & s_n(0) & \cdots & s_n(N) \end{array}\right],$$

$$f^\top = \left[\begin{array}{cccccccc|cccccccc|cccccccc} 1 & \cdots & 1 & \vdots & \cdots & \vdots & 1 & \cdots & 1 & 0 & \cdots & 0 & \vdots & \cdots & \vdots & 0 & \cdots & 0 & 0 & \cdots & 0 & \vdots & \cdots & \vdots & 0 & \cdots & 0 \end{array}\right],$$

$$
\begin{aligned}
lb^\top &= \left[\begin{array}{ccccc|ccccc|ccccc} 0 & \cdots & 0 & \vdots\cdots\vdots & 0 \ \cdots \ 0 & 0 & \cdots & 0 & \vdots\cdots\vdots & 0 \ \cdots \ 0 & 0 & \cdots & 0 & \vdots\cdots\vdots & 0 \ \cdots \ 0 \end{array}\right],\\
ub^\top &= \left[\begin{array}{ccccc|ccccc|ccccc} 1 & \cdots & 1 & \vdots\cdots\vdots & 1 \ \cdots \ 1 & 1 & \cdots & 1 & \vdots\cdots\vdots & 1 \ \cdots \ 1 & 1 & \cdots & 1 & \vdots\cdots\vdots & 1 \ \cdots \ 1 \end{array}\right],\\
&\quad u_i(k) \in \mathbb{Z},\ \forall i = 1, \dots n,\ k = 0, \dots, N,
\end{aligned}
$$

$$
A = \left[\begin{array}{ccc|ccc|ccc}
[0] & \cdots & [0] & M & \cdots & [0] & \begin{matrix} -1 & & \\ & \ddots & \\ & & -1 \end{matrix} & \cdots & [0] \\
\hdashline
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\hdashline
[0] & \cdots & [0] & [0] & \cdots & M & [0] & \cdots & \begin{matrix} -1 & & \\ & \ddots & \\ & & -1 \end{matrix} \\
\hline
[0] & \cdots & [0] & \begin{matrix} r_1 & & \\ & \ddots & \\ & & r_1 \end{matrix} & \cdots & \begin{matrix} r_n & & \\ & \ddots & \\ & & r_n \end{matrix} & [0] & \cdots & [0]
\end{array}\right], \quad b = \left[\begin{array}{c} 0 \\ \vdots \\ 0 \\ \hdashline \vdots \\ \hdashline 0 \\ \vdots \\ 0 \\ \hline P \\ \vdots \\ P \end{array}\right],
$$

$$
A_{eq} = \left[\begin{array}{ccc|ccc|ccc}
M_1 & \cdots & [0] & \begin{matrix} \theta & & \\ & \ddots & \\ & & \theta \end{matrix} & \cdots & [0] & [0] & \cdots & [0] \\
\hdashline
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\hdashline
[0] & \cdots & M_n & [0] & \cdots & \begin{matrix} \theta & & \\ & \ddots & \\ & & \theta \end{matrix} & [0] & \cdots & [0] \\
\hline
[0] & \cdots & [0] & [0] & \cdots & [0] & 1 \ \cdots \ 1 & \ddots & 1 \ \cdots \ 1 \\
\hline
[0] & \cdots & [0] & \theta \ \cdots \ \theta & \ddots & \theta \ \cdots \ \theta & [0] & \cdots & [0]
\end{array}\right], \quad b_{eq} = \left[\begin{array}{c} \theta d_1(0) \\ \vdots \\ \theta d_1(N) \\ \hdashline \vdots \\ \hdashline \theta d_n(0) \\ \vdots \\ \theta d_n(N) \\ \hline 1 \\ \vdots \\ 1 \\ \hline \tau_1 \\ \vdots \\ \tau_n \end{array}\right],
$$

where $M_i$ is the $(N+1) \times (N+1)$ matrix defined as

$$
M_i = \tau_i \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ -1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 & 0 \\ & & & \ddots & & & \\ 0 & 0 & 0 & \cdots & -1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & -1 & 1 \end{bmatrix}.
$$

**Remark 3.3:**
In general, the LC and SU approaches, based on integer programming techniques, can be hard to solve.

## 3.2 The interruptible case

In this Section, the optimal control problem is specified for the interruptible ($IT$) case. The power is still supplied at the maximum rate, but the supply can be interrupted and resumed; Eq. (3.1i) is given by Eq. (2.9), reported next in Eq. (3.11d).

Then, the effect of chattering is to be considered. It is known that given any continuous function $u(t) \in [0, 1]$, the corresponding trajectory $x_i(t)$ in Eq. (2.14) can be achieved as the limit of the response to a suitable switching

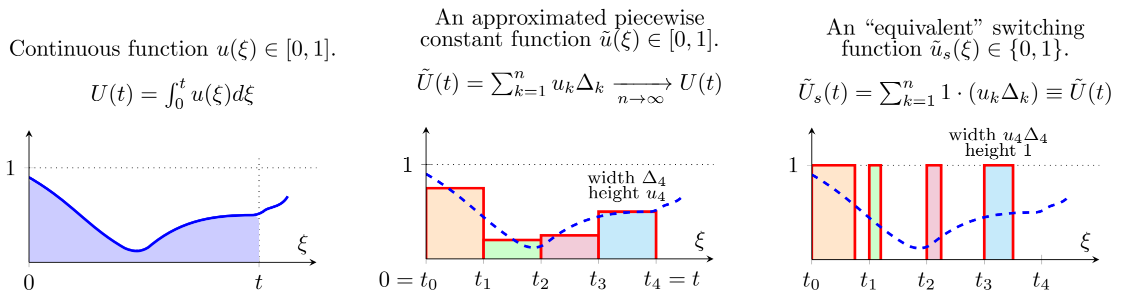

Continuous function $u(\xi) \in [0,1]$.
$U(t) = \int_0^t u(\xi)d\xi$
1
$\xi$
0
$t$
An approximated piecewise constant function $\tilde{u}(\xi) \in [0,1]$.
$\tilde{U}(t) = \sum_{k=1}^n u_k \Delta_k \xrightarrow[n\to\infty]{} U(t)$
1
width $\Delta_4$
height $u_4$
$\xi$
$0 = t_0$
$t_1$
$t_2$
$t_3$
$t_4 = t$
An "equivalent" switching function $\tilde{u}_s(\xi) \in \{0,1\}$.
$\tilde{U}_s(t) = \sum_{k=1}^n 1 \cdot (u_k \Delta_k) \equiv \tilde{U}(t)$
width $u_4\Delta_4$
height 1
1
$\xi$
$t_0$
$t_1$
$t_2$
$t_3$
$t_4$

Figure 3.4: Left: a continuous function $u(\xi) \in [0,1]$, whose integral $U(t) = \int_0^t u(\xi)d\xi$ in the interval $[0,t]$ is highlighted by the blue area. Center: $u(\xi)$ can be approximated by some piecewise constant function $\widetilde{u}(\xi) \in [0,1]$; its integral in $[0,t]$ corresponds to the Riemann sum $\widetilde{U}(t)$: letting the interval $[0,t]$ be partitioned into $n$ sub-intervals $[t_{k-1}, t_k]$, for $k = 1, \dots, n, t_0 = 0, t_n = t$ (here, $n = 4$), this is just the sum the areas of $n$ contiguous rectangles associated with those sub-intervals, each one with width $\Delta_k = t_k - t_{k-1}$ and height $u_k = u(\xi_k) \in [0,1]$ for some $\xi_k \in [t_{k-1}, t_k]$, i.e., $\widetilde{U}(t) = \sum_{k=1}^n u_k \Delta_k$. As $n$ increases, the sub-intervals become smaller and smaller, $\Delta_k \to 0$, and $\widetilde{U}(t) \to U(t)$ as $n \to \infty$. Right: each of these rectangles can be "replaced" by some rectangles with the same area, but height 1 and width $u_k \Delta_k \leq \Delta_k$, which define some switching function $\widetilde{u}_s(\xi) \in \{0,1\}$. Its integral $\widetilde{U}_s(t)$ in $[0,t]$ is equal to $\widetilde{U}(t)$, by construction, hence it tends to $U(t)$ as $n \to \infty$, too, $\widetilde{U}_s(t) \equiv \widetilde{U}(t) \to U(t)$, and the corresponding $\widetilde{u}_s(\xi)$ has an arbitrarily high–frequency switching, as the $\Delta_k$ are arbitrarily small.

function $u_s(t) \in \{0,1\}$ if an arbitrarily high–frequency switching is allowed [54]. Indeed, this fact can be easily derived, see, e.g., Fig. 3.4, where the trajectory of $U(t) = \int_0^t u(\xi)d\xi$ is considered, for simplicity. Clearly, arbitrarily high switching frequency is not realistic. In practice, this problem is automatically solved by using discretization, since the sampling time imposes a bound on the switching frequency.

While the contiguity constraint on $u_i(t)$ in Eq. (3.1i) is not required here, it is not possible to use the total accounted delay $\sum_{i=1}^{\infty} \int_0^{\infty} x_i(t)dt$ as a functional to get the *exact* optimal solution minimizing the total overall delay $\delta$ or, equivalently, the total waiting time $\omega$. Indeed, by Theorem 2.2, it now holds that $\sum_{i=1}^{\infty} \int_0^{\infty} x_i(t)dt \leq \omega = \sum_{i=1}^{\infty} \omega_i$.

Still, Theorem 2.1 from subsection 2.2.2 can be exploited to better specify the functional $J_{IT}(u_1, \dots, u_n, z_1, \dots, z_n)$, by introducing variables $z_i(t)$, see Eqs. (2.11) and (2.12), reported next in Eqs. (3.11b) and (3.11c).

Then, for the $IT$ case, the delay minimization problem is

$$\min_{z_1,\dots,z_n,u_1,\dots,u_n} J_{IT}(u_1, \dots, u_n, z_1, \dots, z_n) = \sum_{i=1}^n \omega_i = \sum_{i=1}^n \int_0^{\infty} [z_i(t) - u_i(t)]\ dt, \tag{3.11a}$$

$$\text{s.t.}\quad z_i(t) = \begin{cases} 1\,, & \text{if } t \in [t_i, t_i + \delta_i]\,, \\ 0\,, & \text{otherwise}\,, \end{cases} \tag{3.11b}$$

$$0 \leq u_i(t) \leq z_i(t) \leq 1, \tag{3.11c}$$

$$u_i(t), z_i(t) \in \{0,1\}, \tag{3.11d}$$

$$\int_0^{\infty} u_i(t)\ dt = \tau_i, \tag{3.11e}$$

$$\sum_{i=1}^n r_i u_i(t) \leq P, \quad \text{for all } t \geq 0\,. \tag{3.11f}$$

As formulated, this is a functional problem of choosing $u_k$. Note that the variables $x_i(t)$ and the corresponding constraints from Eq. (3.1b)–Eq. (3.1h) are not used here, and are replaced by the equivalent constraints associated with $z_i(t)$ in Eqs. (3.11b) to (3.11d) and Eq. (3.11e).

Now, the major issue in the exact solution is that *contiguity constraints* are needed for variables $z_i(t)$, which are used in the objective function.

## 3.2.1 Discrete-time implementation: Mixed integer programming using the LC and SU techniques

For computation, problem Eq. (3.11) and the class of functions $u_i(t) \in \{0,1\}$ needs to be discretized. Assume a sampling time $\theta$ and $N+1$ time-slots, as in subsection 3.1.1.

Then, constraint Eq. (3.11b) can be rewritten based on the $LC$ or the $SU$ forms described in subsection 3.1.1, just using $z_i$ instead of $u_i$ and adapting the formulation.

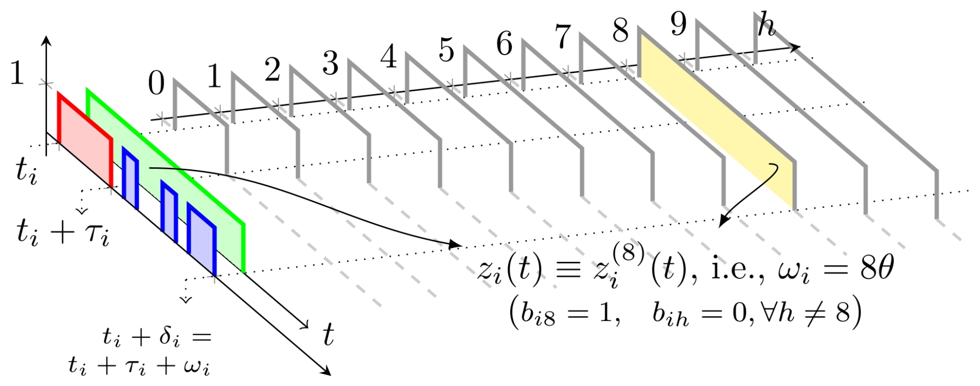

1
$t_i$
$t_i+\tau_i$
$t_i+\delta_i =$
$t_i+\tau_i+\omega_i$
$t$
0
1
2
3
4
5
6
7
8
9
$h$
$z_i(t) \equiv z_i^{(8)}(t)$, i.e., $\omega_i = 8\theta$
$\left(b_{i8}=1, \quad b_{ih}=0, \forall h \neq 8\right)$

Figure 3.5: The Linear Combination (LC) technique for implementing contiguity constraint Eq. (3.11b), in the interruptible $IT$ case. Red: demand profile $d_i(t)$; blue: supply function $u_i(t)$; green: active request profile $z_i(t)$; gray: the candidate profiles $z_i^{(h)}(t)$ for $z_i(t)$. The candidate profile with $h=8$ (highlighted in yellow) is to be selected: from Eq. (3.12), setting $b_{i8}=1$ and $b_{ih}=0$ for all $h \neq 8$, $z_i(t) = z_i^{(8)}(t)$. In the specific example, $\tau_i = 4\theta, \omega_i = 8\theta$.

**Linear Combination (LC) formulation for the contiguity constraints.**

The first alternative form for contiguity constraint Eq. (3.11b) consists in writing $z_i(k)$ as a linear combination of all the possible profiles it can assume, selecting only one of them by means of some new decision variables.

Consider a set of possible values for the waiting time $\mathcal{H} = \{h\theta \ : \ h = 0, \dots, H_i\}$, where $H_i = N - t_i - \tau_i + 1$ is the maximum one, and define, for each $\omega_h \in \mathcal{H}$, the continuous-time signal

$$z_i^{(h)}(t) = \begin{cases} 1\,, & \text{if } t \in [t_i, t_i+\tau_i+\omega_h]\,, \\ 0\,, & \text{otherwise}\,. \end{cases}$$

Then, the variable $z_i(t)$ Eq. (3.11b) is written as a linear convex combination with binary coefficients $b_{ih}$ where only one of them is non-zero and equal to 1, see Fig. 3.5, and is replaced by:

$$z_i(k) = \sum_{h=1}^{H_i} b_{ih} z_i^{(h)}(k), \tag{3.12a}$$

$$\sum_{h=1}^{H_i} b_{ih} = 1, \tag{3.12b}$$

$$b_{ih} \in \{0,1\}. \tag{3.12c}$$

Then, after discretization, problem Eq. (3.11) reduces to the following MILP problem:

$$\min_{z_1,\dots,z_n,u_1,\dots,u_n} J_{IT}(u_1,\dots,u_n,z_1,\dots,z_n) = \sum_{i=1}^{n} \omega_i = \theta \sum_{k=0}^{N} \sum_{i=1}^{n} [z_i(k) - u_i(k)], \tag{3.13a}$$

$$\text{s.t.} \quad z_i(k) = \sum_{h=1}^{H_i} b_{ih} z_i^{(h)}(k), \tag{3.13b}$$

$$\sum_{h=1}^{H_i} b_{ih} = 1, \tag{3.13c}$$

$$b_{ih} \in \{0,1\}, \tag{3.13d}$$

$$0 \le u_i(k) \le z_i(k) \le 1, \tag{3.13e}$$

$$u_i(k), z_i(k) \in \{0,1\}, \tag{3.13f}$$

$$\theta \sum_{k=0}^{N} \quad u_i(t) = \tau_i, \tag{3.13g}$$

$$\sum_{i=1}^{n} r_i u_i(k) \le P, \quad \text{for all } k \ge 0\,. \tag{3.13h}$$

Problem Eq. (3.13) can be simply implemented as a MILP problem Eq. (2.23) by taking

$$
\begin{aligned}
\eta^\top &= \left[\begin{array}{ccccccc|ccccccc|ccccccc} z_1(0) & \cdots & z_1(N) & \vdots\cdots\vdots & z_n(0) & \cdots & z_n(N) & u_1(0) & \cdots & u_1(N) & \vdots\cdots\vdots & u_n(0) & \cdots & u_n(N) & b_{1,0} & \cdots & b_{1,H_1} & \vdots\cdots\vdots & b_{n,0} & \cdots & b_{n,H_n} \end{array}\right],\\
f^\top &= \left[\begin{array}{ccccccc|ccccccc|ccccccc} 1 & \cdots & 1 & \vdots\cdots\vdots & 1 & \cdots & 1 & -1 & \cdots & -1 & \vdots\cdots\vdots & -1 & \cdots & -1 & 0 & \cdots & 0 & \vdots\cdots\vdots & 0 & \cdots & 0 \end{array}\right],\\
lb^\top &= \left[\begin{array}{ccccccc|ccccccc|ccccccc} 0 & \cdots & 0 & \vdots\cdots\vdots & 0 & \cdots & 0 & 0 & \cdots & 0 & \vdots\cdots\vdots & 0 & \cdots & 0 & 0 & \cdots & 0 & \vdots\cdots\vdots & 0 & \cdots & 0 \end{array}\right],\\
ub^\top &= \left[\begin{array}{ccccccc|ccccccc|ccccccc} 1 & \cdots & 1 & \vdots\cdots\vdots & 1 & \cdots & 1 & 1 & \cdots & 1 & \vdots\cdots\vdots & 1 & \cdots & 1 & 1 & \cdots & 1 & \vdots\cdots\vdots & 1 & \cdots & 1 \end{array}\right],\\
& b_{ih}, u_i(k) \in \mathbb{Z},\ \forall i = 1,\dots n,\ h = 0,\dots,H_i,\ k = 0,\dots,N,
\end{aligned}
$$

$$
A = \left[\begin{array}{ccc|ccc|ccc}
\begin{smallmatrix}-1 & & \\ & \ddots & \\ & & -1\end{smallmatrix} & \cdots & [0] & \begin{smallmatrix}1 & & \\ & \ddots & \\ & & 1\end{smallmatrix} & \cdots & [0] & [0] & \cdots & [0] \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
[0] & \cdots & \begin{smallmatrix}-1 & & \\ & \ddots & \\ & & -1\end{smallmatrix} & [0] & \cdots & \begin{smallmatrix}1 & & \\ & \ddots & \\ & & 1\end{smallmatrix} & [0] & \cdots & [0] \\
\hline
[0] & \cdots & [0] & \begin{smallmatrix}r_1 & & \\ & \ddots & \\ & & r_1\end{smallmatrix} & \cdots & \begin{smallmatrix}r_n & & \\ & \ddots & \\ & & r_n\end{smallmatrix} & [0] & \cdots & [0]
\end{array}\right], \quad
b = \left[\begin{array}{c} 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \\ \hline P \\ \vdots \\ P \end{array}\right],
$$

$$
A_{eq} = \left[\begin{array}{ccc|ccc|ccc}
\begin{smallmatrix}1 & & \\ & \ddots & \\ & & 1\end{smallmatrix} & \cdots & [0] & [0] & \cdots & [0] & -Z_1 & \cdots & [0] \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
[0] & \cdots & \begin{smallmatrix}1 & & \\ & \ddots & \\ & & 1\end{smallmatrix} & [0] & \cdots & [0] & [0] & \cdots & -Z_n \\
\hline
[0] & \cdots & [0] & [0] & \cdots & [0] & 1 \cdots 1 & \ddots & 1 \cdots 1 \\
\hline
[0] & \cdots & [0] & \theta \cdots \theta & \ddots & \theta \cdots \theta & [0] & \cdots & [0]
\end{array}\right], \quad
b_{eq} = \left[\begin{array}{c} 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \\ \hline 1 \\ \vdots \\ 1 \\ \hline \tau_1 \\ \vdots \\ \tau_n \end{array}\right],
$$

where $Z_i$ is the $(N+1) \times (H_i+1)$ matrix whose columns are the possible $z_i^{(h)}(t)$ profiles for request $i$, which are selected by $b_{i,h}$

$$
Z_i = \begin{bmatrix} z_i^{(1)}(0) & z_i^{(2)}(0) & \cdots & z_1^{(H_i)}(0) \\ z_i^{(1)}(1) & z_i^{(2)}(1) & \cdots & z_1^{(H_i)}(1) \\ \vdots & \vdots & \ddots & \vdots \\ z_i^{(1)}(N) & z_i^{(2)}(N) & \cdots & z_1^{(H_i)}(N) \end{bmatrix}.
$$

**Start–ups (SU) formulation for the contiguity constraints.**

The second alternative form for contiguity constraint Eq. (3.11b) is based on [53] and consists in limiting the number of “start-ups”, i.e., transitions from 0 to 1, to one for variable $z_i(k)$.

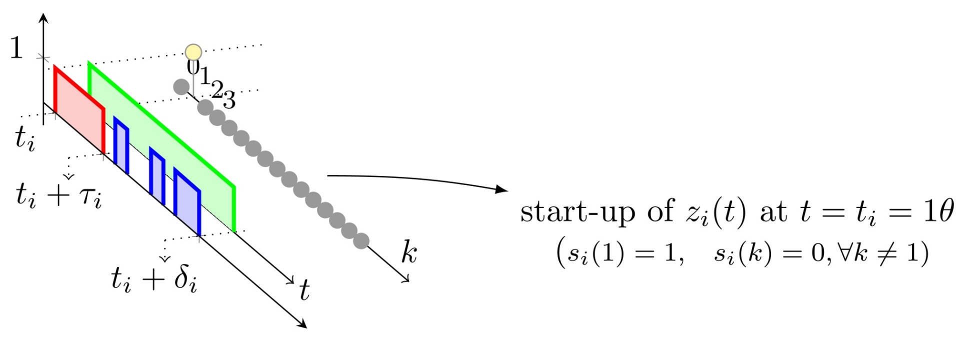

1
$t_i$
$t_i + \tau_i$
$t_i + \delta_i$
$t$
0 1 2 3
$k$
start-up of $z_i(t)$ at $t = t_i = 1\theta$
$(s_i(1) = 1, \quad s_i(k) = 0, \forall k \neq 1)$

Figure 3.6: The Start-Up (SU) technique for implementing contiguity constraint Eq. (3.11b), in the interruptible *IT* case. Red: demand profile $d_i(t)$; blue: supply function $u_i(t)$; green: active request profile $z_i(t)$; gray: the discrete-time function $s_i(k)$. There is only one non-zero value of $s_i(k)$, located at $t = t_i = 1\theta$, i.e., when the request starts. In the specific example, $\tau_i = 4\theta, \Delta_i = 8\theta$.

The additional constrained variable $s_i(k)$ is introduced, as well as the following constraints

$$0 \le s_i(k) \le 1, \tag{3.14a}$$
$$s_i(0) \ge z_i(0), \tag{3.14b}$$
$$s_i(k) \ge z_i(k) - z_i(k-1),\ k > 0, \tag{3.14c}$$
$$\sum_k s_i(k) = 1, \tag{3.14d}$$
$$z_i(k) = 0,\ \theta k < t_i, \tag{3.14e}$$
$$z_i(k) = 1,\ \theta k = t_i, \tag{3.14f}$$
$$z_i(k) \in \{0, 1\}. \tag{3.14g}$$

Note that, differently from the *NI* case, by constraints Eqs. (3.14e) and (3.14f), a start-up is imposed for variable $z_i(t)$ at $k = \theta t_i$, so that by Eqs. (3.14c) and (3.14g), $s_i(k) = 1$ at $k = \theta t_i$. By Eq. (3.14d), this is imposed to be the only start-up instant.

Then, after discretization, problem Eq. (3.11) reduces to the following MILP problem:

$$\min_{z_1,\dots,z_n,u_1,\dots,u_n} J_{IT}(u_1,\dots,u_n,z_1,\dots,z_n) = \sum_{i=1}^{n} \omega_i = \theta \sum_{k=0}^{N} \sum_{i=1}^{n} [z_i(k) - u_i(k)], \tag{3.15a}$$
$$\text{s.t. } 0 \le s_i(k) \le 1, \tag{3.15b}$$
$$s_i(0) \ge z_i(0), \tag{3.15c}$$
$$s_i(k) \ge z_i(k) - z_i(k-1),\ k > 0, \tag{3.15d}$$
$$\sum_k s_i(k) = 1, \tag{3.15e}$$
$$z_i(k) = 0,\ \theta k < t_i, \tag{3.15f}$$
$$z_i(k) = 1,\ \theta k = t_i, \tag{3.15g}$$
$$0 \le u_i(k) \le z_i(k) \le 1, \tag{3.15h}$$
$$u_i(k), z_i(k) \in \{0, 1\}, \tag{3.15i}$$
$$\theta \sum_{k=0}^{N} u_i(t) = \tau_i, \tag{3.15j}$$
$$\sum_{i=1}^{n} r_i u_i(k) \le P, \quad \text{for all } k \ge 0\,. \tag{3.15k}$$

Problem Eq. (3.15) can be simply implemented as a MILP problem Eq. (2.23) by taking

$$\eta^\top = \left[\begin{array}{ccc:c:ccc|ccc:c:ccc|ccc:c:ccc} z_1(0) & \cdots & z_1(N) & \cdots & z_n(0) & \cdots & z_n(N) & u_1(0) & \cdots & u_1(N) & \cdots & u_n(0) & \cdots & u_n(N) & s_1(0) & \cdots & s_1(N) & \cdots & s_n(0) & \cdots & s_n(N) \end{array}\right],$$

$$f^\top = \left[\begin{array}{ccc:c:ccc|ccc:c:ccc|ccc:c:ccc} 1 & \cdots & 1 & \cdots & 1 & \cdots & 1 & -1 & \cdots & -1 & \cdots & -1 & \cdots & -1 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \end{array}\right],$$

$$lb^{\top} = \left[\begin{array}{ccc:c:ccc|ccc:c:ccc|ccc:c:ccc} 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \end{array}\right],$$

$$ub^{\top} = \left[\begin{array}{ccc:c:ccc|ccc:c:ccc|ccc:c:ccc} 1 & \cdots & 1 & \cdots & 1 & \cdots & 1 & 1 & \cdots & 1 & \cdots & 1 & \cdots & 1 & 1 & \cdots & 1 & \cdots & 1 & \cdots & 1 \end{array}\right],$$

$$\begin{array}{ll} u_i(k), z_i(k) \in \mathbb{Z},\ \forall i = 1, \dots n,\ k = 0, \dots, N, & \\ z_i(t_i) = 1 & \rightarrow lb(k) = 1,\ \text{for } k = (i-1)\cdot n + t_i, \forall i = 1, \dots n, \\ z_i(t) = 0, \forall t < t_i & \rightarrow ub(k) = 0, \forall\ 1 \le k < (i-1)\cdot n + t_i,\ \forall i = 1, \dots n, \end{array}$$

$$A = \left[\begin{array}{c:c:c|c:c:c|c:c:c}
M & \cdots & [0] & [0] & \cdots & [0] & \begin{smallmatrix}-1 & & \\ & \ddots & \\ & & -1\end{smallmatrix} & \cdots & [0] \\
\hdashline
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\hdashline
[0] & \cdots & M & [0] & \cdots & [0] & [0] & \cdots & \begin{smallmatrix}-1 & & \\ & \ddots & \\ & & -1\end{smallmatrix} \\
\hline
\begin{smallmatrix}-1 & & \\ & \ddots & \\ & & -1\end{smallmatrix} & \cdots & [0] & \begin{smallmatrix}1 & & \\ & \ddots & \\ & & 1\end{smallmatrix} & \cdots & [0] & [0] & \cdots & [0] \\
\hdashline
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\hdashline
[0] & \cdots & \begin{smallmatrix}-1 & & \\ & \ddots & \\ & & -1\end{smallmatrix} & [0] & \cdots & \begin{smallmatrix}1 & & \\ & \ddots & \\ & & 1\end{smallmatrix} & [0] & \cdots & [0] \\
\hline
[0] & \cdots & [0] & \begin{smallmatrix}r_1 & & \\ & \ddots & \\ & & r_1\end{smallmatrix} & \cdots & \begin{smallmatrix}r_n & & \\ & \ddots & \\ & & r_n\end{smallmatrix} & [0] & \cdots & [0]
\end{array}\right], \quad b = \left[\begin{array}{c} 0 \\ \vdots \\ 0 \\ \hdashline \vdots \\ \hdashline 0 \\ \vdots \\ 0 \\ \hline 0 \\ \vdots \\ 0 \\ \hdashline \vdots \\ \hdashline 0 \\ \vdots \\ 0 \\ \hline P \\ \vdots \\ P \end{array}\right],$$

$$A_{eq} = \left[\begin{array}{c:c:c|c:c:c|c:c:c}
[0] & \cdots & [0] & [0] & \cdots & [0] & 1 \ \cdots \ 1 & & \\
 & & & & & & & \ddots & \\
 & & & & & & & & 1 \ \cdots \ 1 \\
\hline
 & & & \theta \ \cdots \ \theta & & & & & \\
 & & & & \ddots & & & & \\
[0] & \cdots & [0] & & & \theta \ \cdots \ \theta & [0] & \cdots & [0]
\end{array}\right], \quad b_{eq} = \left[\begin{array}{c} 1 \\ \vdots \\ 1 \\ \hline \tau_1 \\ \vdots \\ \tau_n \end{array}\right],$$

where $M$ is the $(N+1) \times (N+1)$ matrix defined as

$$M = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ -1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 & 0 \\ & & & \ddots & & & \\ 0 & 0 & 0 & \cdots & -1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & -1 & 1 \end{bmatrix}.$$

## 3.3 The variable-rate case

In this Section, the optimal control problem is specified for the variable-rate ($VR$) case. The supply $u_i(t)$ can be any (piecewise-continuous) function; Eq. (3.1i) is given by Eq. (2.10), that is $0 \le u_i(t) \le 1$.

A first possibility is minimizing Eq. (3.1a) directly, that is,

$$\min\ J_{VR}(\delta_1, \delta_2 \dots, \delta_n).$$

For fixed $\delta_1, \delta_2 \dots, \delta_n$, the problem of determining $u_i$ from completion times $t_i + \delta_i$ can be formulated as a linear feasibility problem, which can be solved easily.

Another possibility is considering problem Eq. (3.11) with functional $J_{VR}(u_1,\dots,u_n)=\sum_i \omega_i$ and without the $u_i(t)\in\{0,1\}$ constraint, as reported next. Compared to the previous possibility, this is harder because the problem is not convex.

$$
\begin{aligned}
\min_{z_1,\dots,z_n,u_1,\dots,u_n} \quad & J_{VR}(u_1,\dots,u_n,z_1,\dots,z_n)=\sum_{i=1}^{n}\omega_i=\sum_{i=1}^{n}\int_0^{\infty}[z_i(t)-u_i(t)]\,dt, && (3.16a)\\
\text{s.t.} \quad & z_i(t)=\begin{cases}1\,, & \text{if } t\in[t_i,t_i+\delta_i]\,,\\ 0\,, & \text{otherwise}\,,\end{cases} && (3.16b)\\
& 0\le u_i(t)\le z_i(t)\le 1, && (3.16c)\\
& z_i(t)\in\{0,1\}, && (3.16d)\\
& \int_0^{\infty}u_i(t)\,dt=\tau_i, && (3.16e)\\
& \sum_{i=1}^{n}r_iu_i(t)\le P, \quad \text{for all } t\ge 0\,. && (3.16f)
\end{aligned}
$$

### 3.3.1 Discrete-time implementation: Mixed integer programming using the LC and SU techniques

For computation, problem Eq. (3.16) and the class of functions $u_i(t)\in[0,1]$ needs to be discretized. Note that, after discretization, $u_i(t)$ actually becomes a piecewise-constant function: assuming a sampling period $\theta$ and $N+1$ time-slots, the possible discontinuity points are given by $k\theta$, for $k=0,1,\dots,N$.

Again, constraint Eq. (3.16b) can be rewritten based on the $LC$ or the $SU$ forms described in subsection 3.2.1. The formulations are exactly the same of problems Eq. (3.13) and Eq. (3.15), respectively; the only difference is that constraint $u_i(k)\in\{0,1\}$ is replaced by $0\le u_i(k)\le 1$. Hence, these formulations are omitted.

### 3.3.2 A relaxed problem to get a heuristic solution efficiently

As implementing the exact optimal control problems requires Mixed-Integer Linear Programming for any of the considered supply strategies, these problems cannot be solved efficiently for very large instances.

A heuristic solution for the variable rate case is achieved by considering the integral functional

$$
J_{rlx}(u_1,\dots,u_n)=\int_0^{\infty}\sum_{i=1}^{n}x_i(t)dt, \tag{3.17}
$$

under constraints Eqs. (3.1b) to (3.1h), resulting in the so-called *relaxed problem*, that is

$$
\begin{aligned}
\min_{u_1,\dots,u_n} \quad & J_{rlx}(u_1,\dots,u_n)=\int_0^{\infty}\sum_{i=1}^{n}x_i(t)dt,\ , && (3.18a)\\
\text{s.t.} \quad & \dot{x}_i(t)=\frac{1}{\tau_i}\,[d_i(t)-u_i(t)]\,, && (3.18b)\\
& d_i(t)=\begin{cases}1 & \text{if } \ t_i\le t\le t_i+\tau_i\,,\\ 0 & \text{otherwise,}\end{cases} && (3.18c)\\
& 0\le u_i(t)\le 1\,, \quad \text{for all } t\ge 0, && (3.18d)\\
& x_i(0)=0\,, && (3.18e)\\
& x_i(t)\ge 0\,, \quad \text{for all } t\ge 0, && (3.18f)\\
& x_i(t)=0\,, \quad \text{for all } t\ge t_i+\delta_i, && (3.18g)\\
& \sum_{i=1}^{n}r_iu_i(t)\le P, \quad \text{for all } t\ge 0\,, && (3.18h)
\end{aligned}
$$

which is convex and linear. Indeed, if $u_{A1},\dots,u_{An}$ and $u_{B1},\dots,u_{Bn}$ are admissible solutions associated with the costs $J_{rlx,A}$ and $J_{rlx,B}$, respectively, then, for all $\alpha,\beta\ge 0$ with $\alpha+\beta=1$, the solution $\alpha u_{A1}+\beta u_{B1},\dots\alpha u_{An}+\beta u_{Bn}$ is admissible and gives the cost $\alpha J_{rlx,A}+\beta J_{rlx,B}$. Then, it can be implemented as a Linear Programming problem and easily solved even for large instances.

There are two main reasons to solve the relaxed problem:

- the solution is a heuristic solution (in general non–optimal in terms of $\omega$ and $\delta$) for the variable rate $VR$ problem;
- the optimal cost $J^*_{rlx} = \int_0^\infty \sum_{i=1}^n x^*_i(t)dt$ is a lower bound for the optimal cost $\omega^*$ for the $NI$, $IT$, and $VR$ supplying strategies, because of Theorem 2.2. This is specified by the next Proposition.

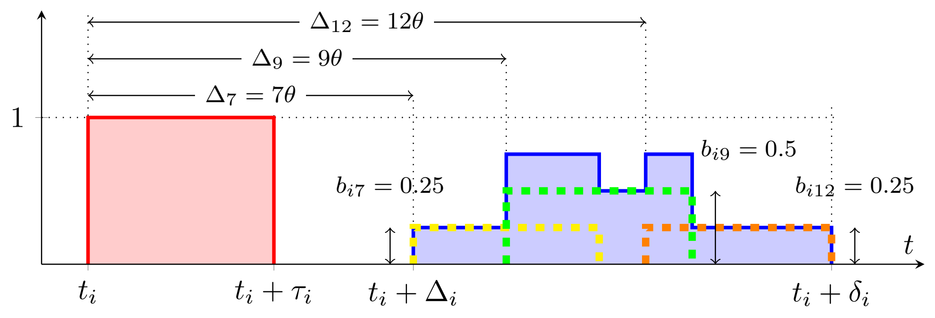
**Proposition 3.2:**
The minimum value of Eq. (3.18a) under constraints Eqs. (3.18b) to (3.18h) is a lower bound for the optimal cost $\sum_i \omega_i$, for all the three cases NI, IT and VR.

**Remark 3.4:**
Some other relaxations could be considered.

- Consider problem Eq. (3.6), where the optimal non-interruptible problem is formulated by means of the Linear-Combination (LC) constraints. Relax Eq. (3.6k) into $0 \le b_{ih} \le 1$ (see Fig. 3.7, top).
- Consider problem Eq. (3.9), where the optimal non-interruptible problem is formulated by means of the Start–ups (SU) constraints. Relax Eq. (3.9i) into $0 \le u_i(t) \le 1$ (see Fig. 3.7, bottom).

In both cases, the supply is a special variable-rate supply and the problem becomes a linear programming one; the solution is a non-optimal heuristic for the $VR$ case, and a lower bound for the optimal waiting time $\omega^*$ of the original $NI$ supply strategy.

*Solution when the LC constraints in the NP optimal problem are relaxed*
$(u_i(t) = b_{i7} \cdot u_i^{(7)}(t) + b_{i9} \cdot u_i^{(9)}(t) + b_{i12} \cdot u_i^{(12)}(t)$, with $b_{i7} + b_{i9} + b_{i12} = 1)$



*Solution when the SC constraints in the NP optimal problem are relaxed*
(with transitions such that $s_i(k) = u_i(k+1) - u_i(k) > 0$ at $k \in \{8, 10, 14\}$,
with $s_i(8) + s_i(10) + s_i(14) = 1$)

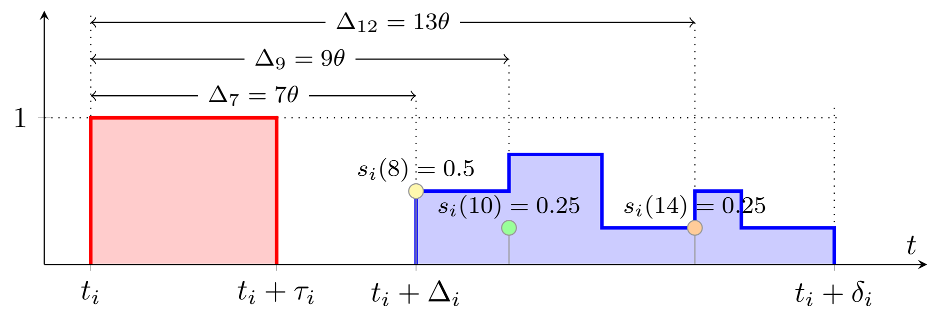


Figure 3.7: Examples of solution when relaxing the binary variables in Eq. (3.6) (top) and Eq. (3.9) (bottom), to continuous variables. Some special types of variable-rate supply are obtained. Red: demand profile $d_i(t)$; blue: supply function $u_i(t)$. On the top image, the yellow, green, and orange dashed lines represent $b_{i7}u_i^{(7)}(t)$, $b_{i9}u_i^{(9)}(t)$, and $b_{i12}u_i^{(12)}(t)$, respectively. On the bottom image, the yellow, green, and orange dots represent $s_i(8), s_i(10)$, and $s_i(14)$, respectively.

This relaxation could be potentially applied to the LC and SU implementations in problems Eq. (3.13) and Eq. (3.15), by converting all the integer variables to continuous variables. However, this is not useful, because the corresponding relaxed optimal solutions would be the ones in which $z_i(t) \equiv u_i(t)$, and this only ensures that the lower bound of $\omega^*$ is 0.

Moreover, consider the exact optimal *IT* problem Eq. (3.11), where $u_i(t)$ are integer and the contiguity constraints on $z_i(t)$ are imposed too, which makes the problem harder to solve. A lower bound for the optimal *NI* cost, which is closer to it compared to the optimum of the relaxed problem Eq. (3.18), could be achieved also by considering the relaxed problem Eq. (3.18) with the additional constraint $u_i(t) \in \{0, 1\}$. While this results in a MILP problem too, it is easier to solve, because fewer integer variables are required.

**Discrete-time implementation: Linear programming**

After discretization, problem Eq. (3.18) can be simply implemented as a LP problem Eq. (2.22) by taking

$$
\begin{aligned}
\eta^\top &= \left[\begin{array}{ccc:c:ccc|ccc:c:ccc} x_1(0) & \cdots & x_1(N) & \cdots & x_n(0) & \cdots & x_n(N) & u_1(0) & \cdots & u_1(N) & \cdots & u_n(0) & \cdots & u_n(N) \end{array}\right],\\
f^\top &= \left[\begin{array}{ccc:c:ccc|ccc:c:ccc} 1 & \cdots & 1 & \cdots & 1 & \cdots & 1 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \end{array}\right],\\
lb^\top &= \left[\begin{array}{ccc:c:ccc|ccc:c:ccc} 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \end{array}\right],\\
ub^\top &= \left[\begin{array}{ccc:c:ccc|ccc:c:ccc} +\infty & \cdots & +\infty & \cdots & +\infty & \cdots & +\infty & 1 & \cdots & 1 & \cdots & 1 & \cdots & 1 \end{array}\right],
\end{aligned}
$$

$$
A = \left[\begin{array}{c:c:c|c:c:c} [0] & \cdots & [0] & \begin{matrix} r_1 & & \\ & \ddots & \\ & & r_1 \end{matrix} & \cdots & \begin{matrix} r_n & & \\ & \ddots & \\ & & r_n \end{matrix} \end{array}\right], \quad b = \begin{bmatrix} P \\ \vdots \\ P \end{bmatrix},
$$

$$
A_{eq} = \left[\begin{array}{c:c:c|c:c:c} \tau_1 M & \cdots & [0] & \begin{matrix} \theta & & \\ & \ddots & \\ & & \theta \end{matrix} & \cdots & [0] \\ \hdashline \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \hdashline [0] & \cdots & \tau_n M & [0] & \cdots & \begin{matrix} \theta & & \\ & \ddots & \\ & & \theta \end{matrix} \end{array}\right], \quad b_{eq} = \left[\begin{array}{c} \theta d_1(0) \\ \vdots \\ \theta d_1(N) \\ \hdashline \vdots \\ \hdashline \theta d_n(0) \\ \vdots \\ \theta d_n(N) \end{array}\right],
$$

where $M$ is the $(N+1) \times (N+1)$ matrix defined as

$$
M = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ -1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 & 0 \\ & & & \ddots & & & \\ 0 & 0 & 0 & \cdots & -1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & -1 & 1 \end{bmatrix}.
$$

### 3.3.3 A greedy strategy for the relaxed problem

Sometimes, even though the relaxed problem Eq. (3.18) can be solved quite efficiently, for very large instances of the problem there might still be some issues to find a solution. Then, the *VR* problem can be approached by applying the so-called greedy strategy, in which the problem is solved iteratively at each time $t$ by finding the control $u(t)$ that minimizes the instantaneous derivative of the cost function inside the integral Eq. (3.17) instead of the integral itself.

While greedy strategies are not optimal, in general, there are still three advantages to adopting this approach:

- the (relaxed) greedy strategy is amenable for real-time implementation, because no knowledge of future data is required;
- it can be used to provide upper and lower bounds for the optimal strategies;
- it has very low complexity and can be implemented recursively, even for very large problems.

The (relaxed) greedy strategy to be solved at generic time $t$ is formulated as follows

$$
\begin{aligned}
&\min_{u_1(t),\dots,u_n(t)} \sum_{i=1}^{n} \frac{1}{\tau_i}\,[d_i(t)-u_i(t)], && (3.19a)\\
&\qquad s.t.\ \dot{x}_i(t) = \frac{1}{\tau_i}\,[d_i(t)-u_i(t)], && (3.19b)\\
&\qquad\quad x_i(t) \geq 0, \quad \text{for all} \quad t>0, && (3.19c)\\
&\qquad\quad x_i(t) = 0, \quad \text{if} \quad t=0, && (3.19d)\\
&\qquad\quad d_i(t) = \begin{cases} 1 & \text{if} \ \ t_i \leq t \leq t_i+\tau_i\,, \\ 0 & \text{otherwise,} \end{cases} && (3.19e)\\
&\qquad\quad 0 \leq u_i(t) \leq 1, && (3.19f)\\
&\qquad\quad \sum_{i=1}^{n} \frac{1}{\tau_i}\ \ r_i u_i(t) \leq P. && (3.19g)
\end{aligned}
$$

Note that Eq. (3.19a) means that the point-wise sum of the derivatives $\sum \dot{x}_i$ is minimized.

Eq. (3.19c) imposes that only active (unfulfilled) requests are considered (see Proposition 2.1); the control is forced to be 0 otherwise. Moreover, constraints Eqs. (3.19b) and (3.19c) imply that it is necessary to keep track of variable $x_i(t)$, to ensure that $\dot{x}_i(t)$ does not make the next $x_i(t+dt)$ become negative.

Overall, the greedy strategy is not optimal for the corresponding integral cost

$$
J_{rlx,grd} = \sum_{i=1}^{n} \int_0^{\infty} x_i(t)dt, \quad \text{s.t. Eqs. (3.19b) to (3.19g).}
$$

Hence, *in this form*, it cannot be considered as a lower bound for the optimal cost of the considered supply strategies. Despite this, from numerical simulations, it turns out that it achieves results quite close to the optimum of Eq. (3.17), so that it can be considered a good "reference value".

### Discrete-time implementation: Linear programming

After discretization, problem Eq. (3.19) can be simply implemented as a LP problem Eq. (2.22) by taking

$$
\begin{aligned}
\eta^\top &= \left[\begin{array}{ccc|ccc} x_1(k) \vdots & \cdots & \vdots x_n(k) & u_1(k) \vdots & \cdots & \vdots u_n(k) \end{array}\right],\\
f^\top &= \left[\begin{array}{ccccc|ccccc} 1 & \vdots & \cdots & \vdots & 1 & 0 & \vdots & \cdots & \vdots & 0 \end{array}\right],\\
lb^\top &= \left[\begin{array}{ccccc|ccccc} 0 & \vdots & \cdots & \vdots & 0 & 0 & \vdots & \cdots & \vdots & 0 \end{array}\right],\\
ub^\top &= \left[\begin{array}{ccccc|ccccc} +\infty & \vdots & \cdots & \vdots & +\infty & 1 & \vdots & \cdots & \vdots & 1 \end{array}\right],
\end{aligned}
$$

$$
A = \left[\begin{array}{c|c} 0 : \cdots : 0 & r_1 : \cdots : r_n \end{array}\right], \qquad b = \left[\ \ P\ \ \right],
$$

$$
A_{eq} = \left[\begin{array}{ccc|ccc} \tau_1 & \vdots & \vdots & \theta & \vdots & \vdots \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \tau_n & \vdots & \vdots & \theta \end{array}\right], \quad b_{eq} = \begin{bmatrix} \theta d_1(k) + \tau_1 x_1(k-1) \\ \vdots \\ \theta d_n(k) + \tau_n x_n(k-1) \end{bmatrix},
$$

### A further relaxation resulting in an optimal greedy strategy

A further relaxation to the exact optimal problem is achieved by replacing the constraint Eq. (3.19f) from the relaxed greedy problem Eq. (3.19) with the condition $u_i(t) \geq 0$. Despite the new problem has no direct relevance, now the results in [55] can be used to claim global optimality of the greedy strategy in this case; then, the optimal value of the corresponding integral cost is a lower bound to the cost of any strategy applied to the admission problem.

While constraint $x_i(t) \geq 0$ is fundamental, because it guarantees that the request is satisfied without supplying more resource than requested, see Propositions 2.1 and 2.2, it is not included in [55]. Then, this constraint $x_i(t) \geq 0$

can be enforced by introducing a penalization term in the cost

$$J_{pen} = \int_0^{\infty} \sum_{i=1}^{n} g(x_i(t))dt = \int_0^{\infty} \sum_{i=1}^{n} \max\{-\mu x_i(t); x_i(t)\}dt, \tag{3.20}$$

with large positive $\mu$.

**Proposition 3.3:**
The greedy strategy Eq. (3.19), after replacing Eq. (3.19f) by $u_i(t) \geq 0$, is globally optimal, since it minimizes both the costs $J_{pen}$ (Eq. (3.20)) and $\int_0^{\infty} \sum_i x_i(t)dt$.

**Remark 3.5:**
In general, penalization introduces an approximation. In this case, the constraint $x_i(t) \geq 0$ is exactly satisfied, since $x_i(t)$ does not become negative.

## 3.4 Recap of the relations between the optimal costs and their lower bounds

To recap, because of Eq. (3.2) and Propositions 3.2 and 3.3, the optimal costs of the presented problems (here marked with superscript $*$) fulfill the following relations:

$$J^*_{rlx,grd(u \geq 0)} = J^*_{rlx(u \geq 0)} \leq J^*_{rlx(0 \leq u \leq 1)} \leq J^*_{VR} \leq J^*_{IT} \leq J^*_{NI}, \tag{3.21}$$

in which the parentheses denote the level of relaxation of $u_i(t)$ in the corresponding integral costs. Specifically, denote $X = \int_0^{\infty} \sum_i x_i(t)dt$ and $\omega = \sum_i \omega_i(t)$ the two considered objective functions. Then, the first equality of Eq. (3.21) results from Proposition 3.3. The first inequality holds because the corresponding problems have the same cost function $X$, but different constraints on $u_i$, and the latter is more restrictive. The second inequality holds because the corresponding problems have the same constraints on $u_i$, but different objective functions: indeed, because of Theorem 2.2 and Proposition 3.2, $X^*_{rlx,(0 \leq u \leq 1)} \leq \widetilde{X}^*_{VR} \leq \omega^*_{VR}$, where $\widetilde{X}^*_{VR}$ is the value of $X$ corresponding to the exact solution minimizing $\omega$ in the $VR$ case, which is not optimal in general. The third and fourth inequalities follow from Eq. (3.2).

Recall that, for the relaxed problem Eq. (3.18) minimizing $X$ with $0 \leq u_i(t) \leq 1$ and the same solved applying a greedy approach, the relation between the corresponding final costs is just

$$J^*_{rlx(0 \leq u \leq 1)} \leq \bar{J}_{rlx,grd(0 \leq u \leq 1)}, \tag{3.22}$$

where $J^*_{rlx(0 \leq u \leq 1)}$ is optimal, but $\bar{J}_{rlx,grd(0 \leq u \leq 1)}$ is not.

Another useful relations consider the relaxations of the exact $NI$ case described in Remark 3.4, using the LC and SU constraints. Denote $J^*_{NI,LC(0 \leq u \leq 1)}$ and $J^*_{NI,SU(0 \leq u \leq 1)}$ the optimal cost of the two relaxed problems. The following relations hold:

$$J^*_{rlx(0 \leq u \leq 1)} \leq J^*_{NI,LC(0 \leq u \leq 1)} \leq J^*_{NI}, \tag{3.23}$$

$$J^*_{rlx(0 \leq u \leq 1)} \leq J^*_{NI,SU(0 \leq u \leq 1)} \leq J^*_{NI}, \tag{3.24}$$

where, in both cases, both inequalities hold because the corresponding problems have the same cost function $X$, but different constraints on $u$, with the latter being a special, more restrictive $VR$ supply (see Remark 3.4).

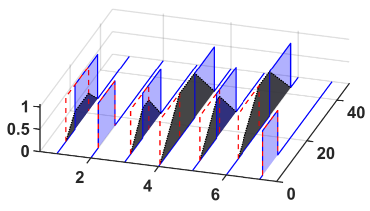
Finally, consider the relaxed problem Eq. (3.18) with the additional constraint $u_i(t) \in \{0, 1\}$, as described in Remark 3.4, and denote $J^*_{rlx(u_i(t) \in \{0,1\})}$ the optimal cost of this relaxed problem. The following relation holds:

$$J^*_{rlx(0 \leq u \leq 1)} \leq J^*_{rlx(u_i(t) \in \{0,1\})} \leq J^*_{IT}, \tag{3.25}$$

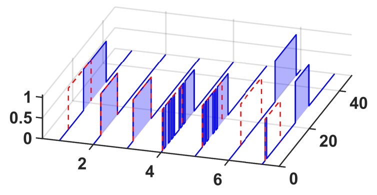
where the first inequality holds because the corresponding problems have the same cost function $X$, but different constraints on $u$, with the latter being more restrictive. The second inequality holds because the corresponding problems have the constraints on $u$, but different cost functions $X$ and $\omega$, respectively: for the same reasons expressed above, $X^*_{rlx,(u_i(t) \in \{0,1\})} \leq \widetilde{X}^*_{IT} \leq \omega^*_{IT}$.

**Example 3.1:**

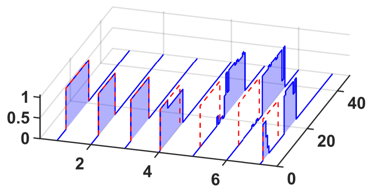
Consider the case of $n = 7$ power requests. Consider an optimization horizon of 8 hours, and a discretization of $\theta = 10$ minutes corresponding to $N = 48$ time slots. The randomly generated demands are described by the sets $r_i \in \{0.6,\ 0.7,\ 0.9,\ 1,\ 1.4,\ 1.7,\ 2.3\}\ kW$, $\tau_i \in \{15,\ 11,\ 12,\ 13,\ 13,\ 14,\ 10\}$ and $t_i \in \{6,\ 5,\ 4,\ 1,\ 5,\ 8,\ 2\}$ time slots. The maximum overall power is $P = 3$ kW. In Fig. 3.8, the optimal solution for each considered problem is reported.



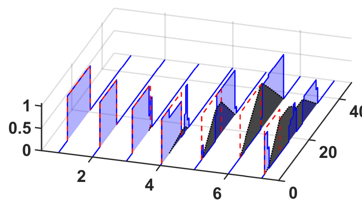
(a) The exact optimal NI solution ($J_{NI} = 69$).



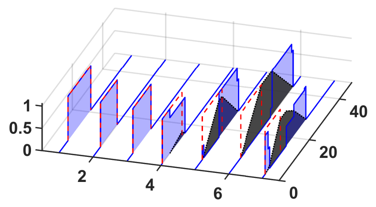
(b) The exact optimal IT solution ($J_{IT} = 64$).



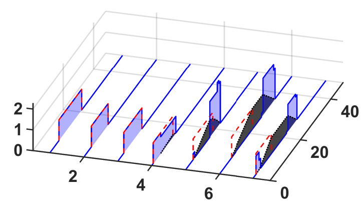
(c) The exact optimal VR solution ($J_{VR} = 48$).



(d) The relaxed optimal solution, for $0 \leq u_i \leq 1$ ($J_{rlx} = 40.7098$).



(e) The relaxed greedy strategy solution, for $0 \leq u_i \leq 1$ ($J_{rlx,grd} = 43.7291$).



(f) The relaxed optimal solution, for $u_i \geq 0$, equal to the greedy strategy ($J_{rlx} = J_{rlx,grd} = 40.2180$).

Figure 3.8: The solutions for Example 1. Dashed red: $d_i(t)$, blue: $u_i(t)$, black: $x_i(t)$. The time axes and costs $J$ are expressed in terms of time slots ($\theta = 10$ minutes).

First of all, note that the relations Eqs. (3.21) and (3.22) are satisfied. The supply rate profiles $u_i(t)$ in subfigure (a) are just a delayed version of the requests $d_i(t)$; also, there are interruptions/resumptions in subfigure (b), and there are rate variations in the other subfigures. Note that in subfigure (f), $u_i(t)$ possibly exceeds 1, as the rate is unlimited: this does not happen at the beginning of the requests, because of the imposition of constraints $x_i(t) \geq 0$. Finally, regarding $x_i(t)$, note that these have a trapezoidal profiles for the case in subplot (a), i.e., in the non-interruptible case, while, from subfigure (d), (e), (f), where $u_i(t)$ can vary over time, the more the supply profiles $u_i(t)$ differ from the requested (possibly delayed) ones, the less the profiles are trapezoidal. In any case, the smaller the waiting times for the fulfillment of the requests are, the smaller are the areas under $x_i(t)$, and these are actually 0 when no delay is applied. $x_i(t)$ are not shown in subfigure (b), (c), as they are not involved in the formulation of the exact IT and VR problems.

# Chapter 4
# Online heuristics: centralized and decentralized solutions

In this Chapter, some online heuristics for the considered *NI*, *IT*, and *VR* problems are presented, which are both centralized and decentralized. The main reason for adopting online heuristics is that in practice the data about the requests are not known in advance. Moreover, possible disturbances or time variability of the available power $P$ and duration $\tau_i$ of the requests can be easily supported. Recall that online strategies cannot guarantee a-posteriori optimality for all possible demand profiles.

For the centralized strategies, all the data of active users at time $t$, either requesting or being supplied, are available for optimization to a central manager. For the decentralized strategies, each supply point is considered as a decision agent that knows only its local information ($t_i$, $r_i$, $\tau_i$) and the global system state ($P$, $y_{tot}(t)$): consequently, these have a low computational burden implementation, and are easily scalable and fault-tolerant.

The following strategies will be considered, which are based on well-known methods and some existing works:

- **Centralized greedy strategies** apply a greedy approach to the presented optimal control problems. They are very simple to implement, as only the current time is optimized;
- **Centralized predictive control strategies** apply a predictive control. A similar approach is adopted in [21, 22], where a LP formulation for the *VR* case is introduced considering many different objectives. Compared to the greedy strategies, the optimization is performed over a finite time window;
- **Centralized priority queue-based strategies** are based on the well-known priority queue-based job scheduling algorithms. This idea has been applied to charging systems by other works, too [56, 57]. They require simple computation and the sorting of the elements of a queue;
- **Decentralized *p-persistent-based* strategies** are inspired by the well-known p-persistent protocols used in CSMA. Multiple attempts are performed to admit a request, which occurs at a given probability. As they mostly require computing probabilities, they are very simple to implement;
- **Decentralized reservation variable-based strategies** are based on a control law inspired by an enhanced version of the p-persistent protocol [58, 59] and applied to home energy management systems in [60, 61]. Other than being very simple to implement, the main advantage is that a priority order is created in a distributed way.

To describe such strategies, the following quantities are introduced:

- The *system saturation* at time $t$: $\sigma(t) = y_{tot}(t)/P \in [0,1]$. No power is supplied at time $t$ if $\sigma(t)$ is 0, while if it is 1, all the system capacity is occupied and no more request can be admitted.
- The portion of the unfulfilled request that has not been supplied yet to user $i$ at time $t$, $\widetilde{\tau}_i(t) = \tau_i - \int_{t_i}^{t} u_i(t')dt'$.
- The set $\mathcal{R}(t) = \{i \mid \widetilde{\tau}_i(t) > 0 \textit{ and } u_i(t) = 0\}$ of users requesting the resource, but not being supplied.
- The set $\mathcal{S}(t) = \{i \mid \widetilde{\tau}_i(t) > 0 \textit{ and } u_i(t) > 0\}$ of currently supplied users.
- The set $\mathcal{A}(t) = \mathcal{R}(t) \cup \mathcal{S}(t)$ of all unfulfilled requests, either being supplied or not.

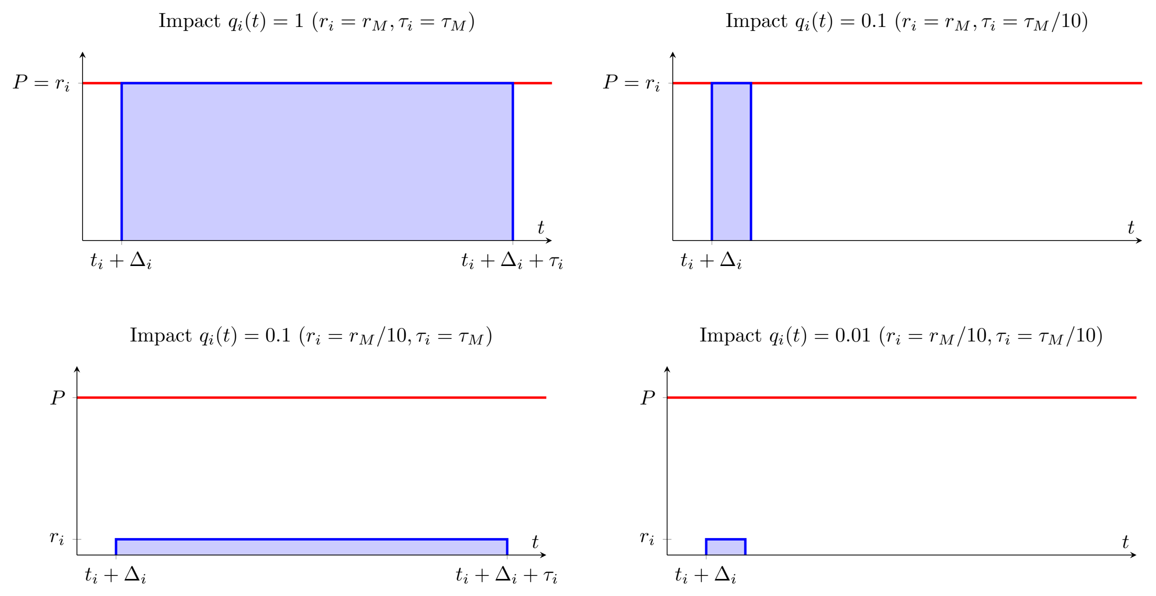


Figure 4.1: The impact of the requests, assuming $\tau_M = P$. The lower the impact, the lower the system is occupied.

- The *impact on the system* of request $i \in \mathcal{A}(t)$ at time $t$

$$q_i(t) = \min\left(\frac{r_i\widetilde{\tau}_i(t)}{r_M\tau_M}, 1\right) \in (0, 1], \tag{4.1}$$

  i.e., the ratio of the energy yet to be supplied $(r_i\widetilde{\tau}_i(t))$ and the maximum possible energy, computed as the product of $r_M$ and $\tau_M$, the maximum possible values for $r_i$ and $\tau_i$ which are assumed to be known from historical data. Admitting a request with a high impact could affect the scheduling of other requests.

  For instance, assume $P = r_M$ and a *NI* power supply strategy and consider Fig. 4.1. Admitting a request with $r_i = r_M$ and $\widetilde{\tau}_i = \tau_M$ $(q_i = 1)$ prevents other requests to be admitted for a large amount of time. Admitting a request with $r_i = r_M$ and $\widetilde{\tau}_i = \tau_M/10$ $(q_i = 0.1)$ prevents other requests to be admitted for a shorter period, while admitting a request with $r_i = \tau_M/10$ and $\tau_i = \tau_M$ $(q_i = 0.1)$ reduces a bit the resource available to others requests for a large amount of time, possibly preventing their admission: in both cases, the impact on the system is reduced. Finally, admitting a request of $r_i = r_M/10$ and $\widetilde{\tau}_i = \tau_M/10$ $(q_i = 0.01)$ only reduces the total available power shortly, so the admission of other requests is barely affected.

- The local *admission priority*, a decreasing function of the impact and the saturation,

$$\pi_i^A(t) = 1 - q_i(t)(\sigma(t))^{\rho_A} \in [0, 1], \ \forall i \in \mathcal{R},$$

  where the exponent $\rho_A$ is a constant that determines how much $\pi_i^A$ is influenced by the saturation. This priority is maximum at full power availability (regardless of the impact), and, as the saturation increases, it differentiates by $q_i(t)$ (higher impact means lower priority).

- The local *interruption priority*, an increasing function of the impact and the saturation,

$$\pi_i^I(t) = q_i(t)(\sigma(t))^{\rho_I} \in [0, 1], \ \forall i \in \mathcal{S},$$

  where the exponent $\rho_I$ is a constant that determines how much $\pi_i^I$ is influenced by the saturation. This priority is minimum at full power availability (regardless of the impact), and, as the saturation increases, it differentiates by $q_i(t)$ (lower impact means lower priority).

A detailed description of the heuristics for the considered supply strategies is reported in the next Sections. For some of them, a basic version is first introduced, which aims at scheduling earlier requests first. As in this way early large power requests might saturate the network capacity increasing the delay of others, an enhanced version is also studied, which takes the impact into account.

## 4.1 The non-interruptible case

In the non-interruptible case, at time $t$, a request $i \in \mathcal{S}(t)$, once admitted, cannot be rescheduled in future optimizations. If a user $i$ is set to switch on at $t'$, i.e., $u_i(t') > 0$, then $u_i(t)$ is forced to be 1 up to $t' + \tau_i$. Thus, only the admission policy is to be specified.

### 4.1.1 The greedy heuristic (*NI* strategy)

For the centralized greedy heuristic for the *NI* supply strategy, an optimization problem is solved at each $t_k = k\theta$, considering only the active requests $i \in \mathcal{A}(t_k)$. Two possibilities are proposed.

A first possibility is considering the problem introduced in subsection 3.3.3, adapting it by imposing $u_i(t) \in \{0, 1\}$, and that if $u_i(t)$ switches on at $t'$, then it remains 1 up to $t' + \tau_i$, that is

$$\min_{u_1(t),\dots,u_n(t)} \sum_{i=1}^{n} \dot{x}_i(t) = \sum_{i=1}^{n} \frac{1}{\tau_i} \left[d_i(t) - u_i(t)\right], \tag{4.2a}$$

$$\text{s.t.} \quad \dot{x}_i(t) = \frac{1}{\tau_i} \left[d_i(t) - u_i(t)\right], \tag{4.2b}$$

$$x_i(t) \geq 0, \quad \text{for all} \quad t > 0, \tag{4.2c}$$

$$x_i(t) = 0, \quad \text{if} \quad t = 0, \tag{4.2d}$$

$$d_i(t) = \begin{cases} 1 & \text{if} \;\; t_i \leq t \leq t_i + \tau_i \,, \\ 0 & \text{otherwise,} \end{cases} \tag{4.2e}$$

$$u_i(t) \in \{0, 1\}, \tag{4.2f}$$

$$U_i^k \leq u_i(t) \leq 1, \tag{4.2g}$$

$$\sum_{i=1}^{n} \frac{1}{\tau_i} \; r_i u_i(t) \leq P, \tag{4.2h}$$

where

$$U_i^k = \begin{cases} 1, & \text{if } 0 < \widetilde{\tau}_i(k) < \tau_i, \\ 0, & \text{otherwise,} \end{cases}$$

in Eq. (4.2g), is a known quantity introduced to ensure that the supply, once started, cannot be interrupted until the request is fulfilled. Indeed, before the admission, $\widetilde{\tau}_i = \tau_i$, so that $U_i^k = 0$, and after the request is fulfilled, $\widetilde{\tau}_i = 0$, so that $U_i^k = 0$, again. In between these two events, when the supplying is non-zero, $0 < \widetilde{\tau}_i < \tau_i$, and then $U_i^k = 1$, so that this imposes $u_i = 1$.

**Remark 4.1:**
Constraint Eq. (4.2b) can be discretized as

$$\frac{x_i(k) - x_i(k-1)}{\theta} = \frac{1}{\tau_i} \left[d_i(k) - u_i(k)\right],$$

where $d_i(k)$ and $x_i(k-1)$ are known from the optimization results of the previous time slot $k-1$, and thus are not decision variables.
The objective function Eq. (4.2a) at time $t$ is the sum of the derivative of each $x_i(t)$: considering its discretization, it holds that

$$\arg\min \sum_i \frac{x_i(k) - x_i(k-1)}{\theta} = \arg\min \sum_i x_i(k) = \arg\min \sum_i \frac{1}{\tau_i} \left[d_i(k) - u_i(k)\right] = \arg\min \sum_i -\frac{u_i(k)}{\tau_i}.$$

Problem Eq. (4.2) can be easily discretized and implemented as a MILP problem Eq. (2.23) by taking

$$\eta = \begin{bmatrix} x_1(k) \\ \vdots \\ x_n(k) \\ u_1(k) \\ \vdots \\ u_n(k) \end{bmatrix}, \quad f = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad lb = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ U_i^k \\ \vdots \\ U_i^k \end{bmatrix}, \quad ub = \begin{bmatrix} +\infty \\ \vdots \\ +\infty \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \quad u_i(k) \in \mathbb{Z},$$

$$A = \begin{bmatrix} 0 & \cdots & 0 & r_1 & \cdots & r_n \end{bmatrix}, \quad b = \begin{bmatrix} P \end{bmatrix}, \quad A_{eq} = \left[\begin{array}{ccc|ccc} 1 & \cdots & 0 & \frac{\theta}{\tau_1} & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 & \cdots & \frac{\theta}{\tau_n} \end{array}\right], \quad b_{eq} = \begin{bmatrix} x_1(k-1) + \frac{d_1(k)\theta}{\tau_1} \\ \vdots \\ x_n(k-1) + \frac{d_n(k)\theta}{\tau_n} \end{bmatrix}.$$

A second possibility is applying a greedy approach to problem Eq. (3.11), by minimizing $\sum_{i\in\mathcal{A}} z_i(t) - u_i(t)$ at time $t$, and imposing $u_i(t) \leq \widetilde{\tau}_i(t)$ instead of Eq. (3.11e), to take into account the quantity of resource already supplied in the previous time-slots. Indeed, to consider the lack of knowledge of the future, the integral in Eq. (3.11e) is to be evaluated only up to the current time $t$ and the constraint becomes an inequality, that is

$$u_i(t) + \int_0^t u_i(t')\, dt' \leq \tau_i, \qquad \text{where} \quad \widetilde{\tau}_i(t) = \tau_i - \int_0^t u_i(t')\, dt'.$$

Moreover, note that each $z_i(t)$ is no more a decision variable, as it can be forced to be 1 when $t \geq t_i$ and as long as $\widetilde{\tau}_i(t) > 0$.

Then, the problem becomes

$$\min_{u_1(t),\dots,u_n(t)} \sum_{i\in\mathcal{A}} z_i(t) - u_i(t), \tag{4.3a}$$

$$\text{s.t.} \quad z_i(t) = \begin{cases} 1\,, & \text{if } t \geq t_i \text{ and } \widetilde{\tau}_i(t) > 0\,, \\ 0\,, & \text{otherwise}\,, \end{cases} \tag{4.3b}$$

$$U_i^k \leq u_i(t) \leq z_i(t) \leq 1, \tag{4.3c}$$

$$u_i(t) \in \{0,1\}, \tag{4.3d}$$

$$u_i(t) \leq \widetilde{\tau}_i(t), \tag{4.3e}$$

$$\sum_{i=1}^{n} r_i u_i(t) \leq P, \tag{4.3f}$$

where Eq. (4.3c) imposes non-interruption of the supply, as in the previous case.

**Remark 4.2:**
The objective function Eq. (4.3a) can be easily discretized, and, as $z_i(k)$ is known, it holds that

$$\arg\min \sum_i z_i(k) - u_i(k) = \arg\min \sum_i -u_i(k).$$

Problem Eq. (4.2) can be easily discretized and implemented as a MILP problem Eq. (2.23) by taking

$$\eta = \begin{bmatrix} u_1(k) \\ \vdots \\ u_n(k) \end{bmatrix}, \quad f = \begin{bmatrix} -1 \\ \vdots \\ -1 \end{bmatrix}, \quad lb = \begin{bmatrix} U_i^k \\ \vdots \\ U_i^k \end{bmatrix}, \quad ub = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}, \quad u_i(k) \in \mathbb{Z}, \quad A_{eq}, b_{eq}: \ N/A,$$

$$A = \left[\begin{array}{ccc} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \\ \hline 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \\ \hline r_1 & \cdots & r_n \end{array}\right], \quad b = \left[\begin{array}{c} z_1(k) \\ \vdots \\ z_n(k) \\ \hline \widetilde{\tau}_1(k) \\ \vdots \\ \widetilde{\tau}_1(k) \\ \hline P \end{array}\right], \quad \widetilde{\tau}_i(k) = \tau_i - \sum_{h=0}^{k-1} u_i(h), \quad z_i(k) = \begin{cases} 1, & \text{if } k\theta \geq t_i \text{ and } \widetilde{\tau}_i(k) > 0, \\ 0, & \text{otherwise.} \end{cases}$$

### 4.1.2 The predictive control heuristic (*NI* strategy)

For the predictive control heuristic for the *NI* supply strategy, the optimization problem Eq. (3.4) is solved at each $t_k = k\theta$ over a finite time horizon $T_{pc}$, considering only the non-admitted requests $i \in \mathcal{R}(t_k)$ occurred up to the current time $t_k$. Then, at $\hat{t} + \Theta$ the process is repeated, taking into account the variations of the system.

This is different from the greedy strategy described in subsection 4.1.1, as in that case only the instantaneous value of the cost in a given time slot is optimized. Again, the solution is not guaranteed to be optimal (a-posteriori), considering the entire time horizon.

To distinguish the variables of the sub-problem to be solved at generic time $t_k$, denote by $\hat{u}(t)$ the normalized supply power profile to be optimized in the horizon $[t_k, t_k + T_{pc}]$, and by $u(t)$ the *actual* one, in the entire time horizon $[0, \infty)$. Initially, it is assumed that $u_i(t) = 0$ for all $i = 1, \ldots, n$ and for all $t \geq 0$.

First of all, if the release time $t_i$ of request $i \in \mathcal{R}(t_k)$ occurs before the current time $t_k$ and the supply has not started yet at time $t_k$, the request has to be considered in the optimization. However, its admittance cannot occur in the "past", i.e., for $t < t_k$: the delay $t_k - t_i$ is fixed and is experienced in any case. Then, for computation, considering the same request occurring at release time $\hat{t}_i = t_k$ is equivalent when minimizing the overall delay. Clearly, supposing that the supply is scheduled to start at $t_k + \hat{\Delta}_i$, the term $\hat{t}_i - t_i = t_k - t_i$ is to be considered for the computation of the *actual* delay, which is $\Delta_i = \hat{\Delta}_i + (t_k - t_i)$. Then, all the requests to be considered in the optimization problem at time $t_k$ can use the same release time $\hat{t}_i = t_k$.

Moreover, once the "temporary" optimal solution $\hat{u}_i^*(t)$ is obtained by solving the sub-problem at time $t_k$, if the admittance of request $i \in \mathcal{R}(t_k)$ is scheduled in the $[t_k; t_k + \Theta)$ time interval, i.e., if the optimized admission delay is $\hat{\Delta}_i < \Theta$, the supply is confirmed and the actual supply profile is updated as

$$u_i(t) \leftarrow \hat{u}_i(t) = 1, \text{ for } \ t \in [t_k + \hat{\Delta}_i; t_k + \hat{\Delta}_i + \tau_i), \text{ if } \ \hat{\Delta}_i < \Theta;$$

recall that the actual admission delay is $\hat{\Delta}_i + t_k - t_i$.

A request, if it is scheduled to be admitted before $t_k + \Theta$, is removed from the set $\mathcal{R}(t_k)$ and therefore it will not be considered again for further optimizations, to support the contiguity constraints on $u_i(t)$. Still, its (known) assigned supply profile $u_i(t)$ must be taken into account in Eq. (3.4h) for the successive optimizations, so that the constraint becomes

$$\sum_{i \in \mathcal{R}(t_k)} r_i \hat{u}_i(t) \leq P(t) = P - \sum_{i=1}^{n} r_i u_i(t), \tag{4.4}$$

where the known time-varying $P(t)$ is to be used instead of $P$.

The problem to be solved at each $t_k$ becomes

$$\begin{aligned}
\min \ & \int_{t_k}^{t_k+T_{pc}} \sum_{i \in \mathcal{R}(t_k)} \hat{x}_i(t) dt, \\
\text{s.t.} \ & \dot{\hat{x}}_i(t) = \frac{1}{\tau_i} [\hat{d}_i(t) - \hat{u}_i(t)], \\
& \hat{d}_i(t) = \begin{cases} 1 & \text{if } \ t_k \leq t \leq t_k + \tau_i \, , \\ 0 & \text{otherwise,} \end{cases} \\
& 0 \leq \hat{u}_i(t) \leq 1 \, , \quad \text{for all } t_k \leq t \leq t_k + T_{pc}, \\
& \hat{x}_i(t_k) = 0 \, , \\
& \hat{x}_i(t) \geq 0 \, , \quad \text{for all } t_k \leq t \leq t_k + T_{pc}, \\
& \hat{x}_i(t) = 0 \, , \quad \text{for all } t_k + \hat{\delta}_i \leq t \leq t_k + T_{pc}, \\
& \sum_{i \in \mathcal{R}(t_k)} r_i \hat{u}_i(t) \leq P(t) = P - \sum_{i=1}^{n} r_i u_i(t), \quad \text{for all } t_k \leq t \leq t_k + T_{pc} \, , \\
& \hat{u}_i(t) = \begin{cases} 1 \, , & \text{if } t \in [t_k + \hat{\Delta}_i, t_k + \hat{\Delta}_i + \tau_i] \, , \\ 0 \, , & \text{otherwise} \, . \end{cases}
\end{aligned}$$

Discrete-time implementations using the Linear Combination (LC) and Start–Ups (SU) techniques, as reported in subsection 3.1.1, results in a Mixed-Integer-Linear-Programming problem.

### 4.1.3 The priority queue-based heuristic (*NI* strategy)

In the centralized priority queue-based heuristic for the *NI* supply strategy, new power requests are sent to a central manager and inserted in an *admission queue* (storing requests $i \in \mathcal{R}(t)$).

At the end of each time-slot, the central manager sorts the elements of the queue according to a given criterion: then, as long as there is energy availability, the requests are admitted in the given order.

When evaluating the requests one at a time, it may happen that some of them cannot be fulfilled because there is not enough available power. If a request requires more power than the available one, i.e., if $r_i > P - y_{tot}$, the request is ignored, and the next one in the queues is evaluated.

The ordering method could be

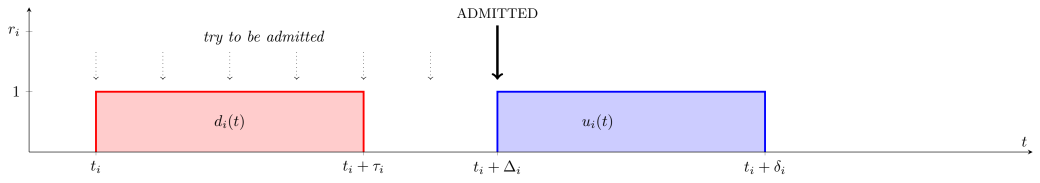


Figure 4.2: The $p$-persistent-based strategy for the non-interruptible *NI* case.

- First In First Out (FIFO);
- Last In First Out (LIFO);
- Larger Requested Power First (LPF);
- Smaller Requested Power First (SPF);
- Larger Requested Duration First (LDF);
- Smaller Requested Duration First (SDF);
- Larger Requested Energy First (LEF);
- Smaller Requested Energy First (SEF);
- Larger Remaining Duration First (LRDF);
- Smaller Remaining Duration First (SRDF);
- Larger Remaining Energy First (LREF);
- Smaller Remaining Energy First (SREF).

Note that, by the definition of impact $q_i(t)$, the SREF sorting method prioritizes requests with a lower impact.

### 4.1.4 The p-persistent-based heuristic (*NI* strategy)

In the decentralized p-persistent-based heuristic for the *NI* supply strategy, each user $i \in \mathcal{R}(t)$ whose request has not started being supplied yet continuously checks the power availability, at every $k$. As soon as there is enough available power for fulfilling its request, i.e., if

$$\sum_j y_j(k) + r_i \leq P,$$

it is eventually admitted with an *admission probability* $p_i^A(t)$, see Fig. 4.2. This can be implemented by generating a real random number $rand \in [0,1]$: if $rand \leq p_i^A(t)$, the supply starts; otherwise, another attempt is performed at the next time-slot. In Fig. 4.3 the flow-chart of the $p$-persistent strategy is reported.

The admission probability $p_i^A(t)$ could be

- constant and equal to 1, i.e., $p_i^A(t) = 1$. In this case, the supply starts immediately, as soon as there is power availability. In CSMA, this variation is known as 1-persistent. The main problem is that, when the system is congested, the immediate admission of requests with high $r_i$ and $\tau_i$, makes it less likely to reduce the saturation, preventing other requests to be admitted, and increasing their waiting times.
- equal to the admission priority of the request, i.e., $p_i^A(t) = \pi_i^A(t)$ (recall that the priority is a number between 0 and 1, hence it could be a probability). In this enhanced version, if the system saturation is low, the request is very likely to be admitted immediately, regardless of its impact on the system; otherwise, if the saturation is high, requests with low impact are way more likely to be admitted compared to those with high impact, for which the admission priority is close to zero.

### 4.1.5 The reservation-variable-based heuristic (*NI* strategy)

In the decentralized reservation variable-based heuristic for the *NI* supply strategy, each user $i \in \mathcal{R}(t)$ whose request has not started being supplied yet is assigned an integral variable $\alpha_i(t) \in \mathbb{R}$.

A local dynamic mechanism processes this variable according to the control law:

$$\begin{cases} \dot{\alpha}_i(t) = \mu_i(t) \cdot (r_i - \alpha_i(t)) \cdot (1-\sigma(t))^\rho, \ \forall i \in \mathcal{R}(t), \\ \alpha_i(t_i) = 0, \end{cases} \tag{4.5}$$

where $\mu_i(t)$ is a non-negative function of time and $\rho$ a given positive parameter.

Variable $\alpha_i(t)$ is initialized with 0 and starts increasing as soon as the request starts at $t_i$. Its growth at each time $t$ depends on $\mu(t)$ and on the saturation $\sigma(t) = y_{tot}(t)/P \in [0,1]$. In particular, on the one hand, the term $(1-\sigma(t))^\rho$ ensures that the more the current total load $y_{tot}(t)$ is high, the less $\alpha_i(t)$ grows. Eventually, when the system is fully loaded, there is no increase. On the other hand, a larger $\mu_i(t)$ means a faster variation; depending

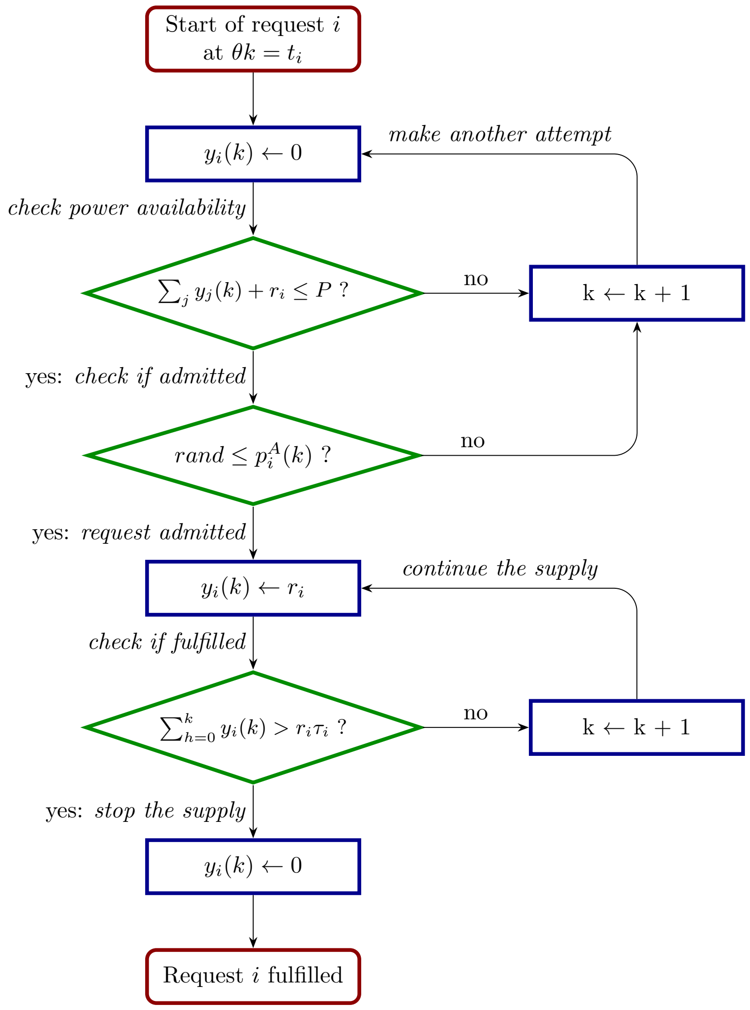


Figure 4.3: Flow-chart of the $p$-persistent-based strategy for the non-interruptible $NI$ case.

on its specification, it could be used to differentiate the behavior of different requests. The term $r_i - \alpha_i(t)$ makes $\alpha_i$ converge to $r_i$, as it makes the derivative become zero as $\alpha_i$ grows.

Let $\xi$ be a constant close and smaller than 1. As soon as $\alpha_i(t)$ reaches a threshold $\xi r_i$, which is close and smaller than $r_i$, the request can be admitted, and the supply starts as soon as there is enough available power, see Fig. 4.4. Note that the time in which the threshold is reached does not depend on $r_i$ (unless $\mu_i(t) = \mu_i(t, r_i)$). In Fig. 4.5 the flow-chart of the reservation-variable-based strategy is reported.

Function $\mu_i(t)$ could be

- constant and equal to some $\mu_0$, i.e., $\mu_i(t) = \mu_0$;
- proportional to the admission priority of the request, i.e., $\mu_i(t) = \mu_0 \pi_i^A(t)$. In this enhanced version, if the system saturation is low, the growth of variable $\alpha_i$ is very fast, regardless of the impact of the request on the system. Otherwise, if the saturation is high, the lower is the impact, the faster $\alpha_i$ grows: if the impact is very high, the growth of $\alpha_i$ is very small, which prevents its admittance, so that requests with lower impact tend to be admitted earlier.

When $\mu_i(t) = \mu_0$, the control law is the same for every demanding user, so that the control variables corresponding to earlier requests have been increased for a longer time, based on the saturation of the system: a priority order (based on the requests' release-times $t_i$) is created among users in a decentralized way and fairness is assured. Time-varying $\mu_0 \pi_i^A(t)$ enhances this behavior by prioritizing requests with lower impact.

For implementation, the above-mentioned control law can be easily discretized via the Euler scheme as:

$$\alpha_i(k+1) = \alpha_i(k) + \hat{\theta} \cdot \mu_i(k) \cdot (r_i - \alpha_i(k)) \cdot (1 - \sigma(k))^{\rho}, \tag{4.6}$$

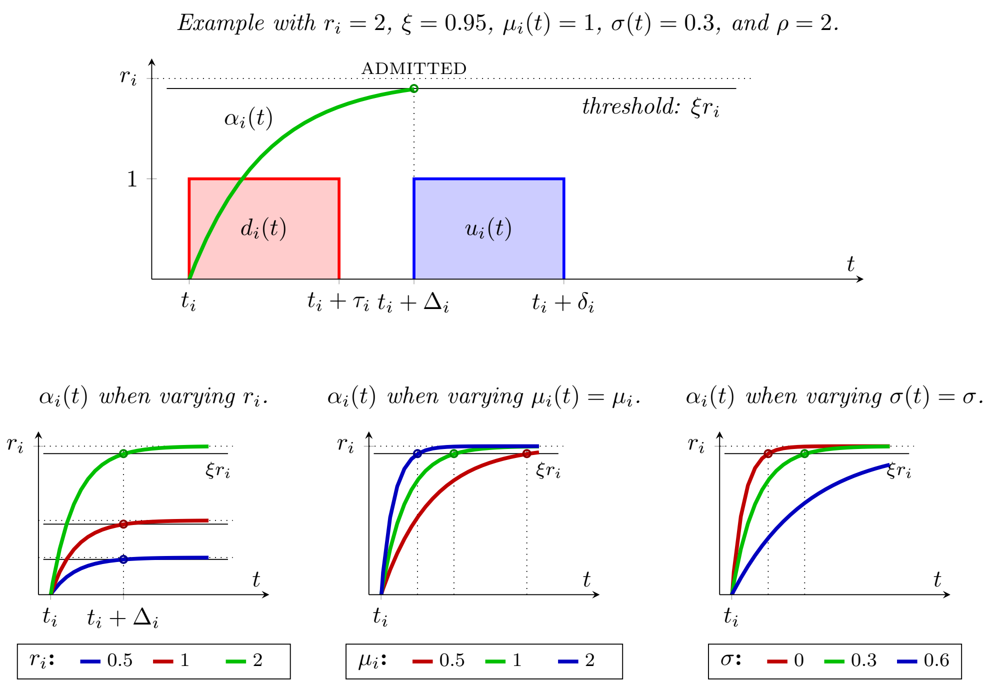


Figure 4.4: The reservation-variable-based strategy for the non-interruptible *NI* case. Assuming $\mu_i(t)$ and $\sigma(t)$ constant, variable $\alpha_i(t)$ for request $i$ evolves according to $\alpha_i(t) = r_i \left(1 - e^{-\mu_i(t)(1-\sigma(t))^\rho (t-t_i)}\right)$, for $t \geq t_i$. The growth is faster as $\mu_i(t)$ increases, and as $\sigma(t)$ decreases. However, the time in which the threshold $\xi r_i$ is reached does not depend on $r_i$. In practice, the saturation $\sigma(t)$ varies over time (as well as $\mu_i(t)$, possibly), so that function $\alpha_i(t)$ has a more complex behavior, but still remains an increasing function approaching $r_i$.

in which $\hat{\theta}$ is the sampling rate at which the control law is updated (e.g. 1 second). Note that $\hat{\theta}$ is independent from the sampling time $\theta$ and $\hat{\theta} \leq \theta$: a finer resolution is needed to get a behavior closer to the one of the continuous-time version.

## 4.2 The interruptible case

In this Section, the real-time heuristics for the interruptible supply strategy are presented.

Now an admitted request can be rescheduled. Indeed, it is imposed that the supply can only be either 0 or $r_i$. However, the supply can be interrupted and resumed later on, even multiple times.

For all the proposed heuristics, the admission procedure is exactly the same as the corresponding one of the *NI* case, with the difference that the control $u_i(t)$ is not forced to be 1 for the entire requested duration. The interruption procedure is to be specified.

**Remark 4.3: Fix the interruption instants by dividing the requests into packets of known duration.**
A simple way to support the *IT* case is to divide each request into *packets* of fixed duration $\tau_{pkt}$ or energy $E_{pkt}$, as in [62, 56]. In the former case, there are $P = \text{floor}(\tau_i/\tau_{pkt})$ packets of duration $\tau_{pkt}$, and one of duration $\tau'_{pkt}$, such that $\tau_i = P\tau_{pkt} + \tau'_{pkt}$. In the latter case, there are $P = \text{floor}(\tau_i r_i/E_{pkt})$ packets of duration $E_{pkt}/r_i$, and one of duration $\tau'_{pkt}$, such that $\tau_i = PE_{pkt}/r_i + \tau'_{pkt}$.
Then, each packet is like an independent request: the new "packet request" $p$ is issued just after the previous "packet request" $p-1$ is fulfilled. The main advantage is that in this way the possible interruption instants are known.
Then, the greedy, priority queue-based, p-persistent-based, and reservation-variable-based heuristics for the *NI* supply strategy can be applied to each packet. See, for instance, Fig. 4.6. This case will be denoted by *IT-packets.*

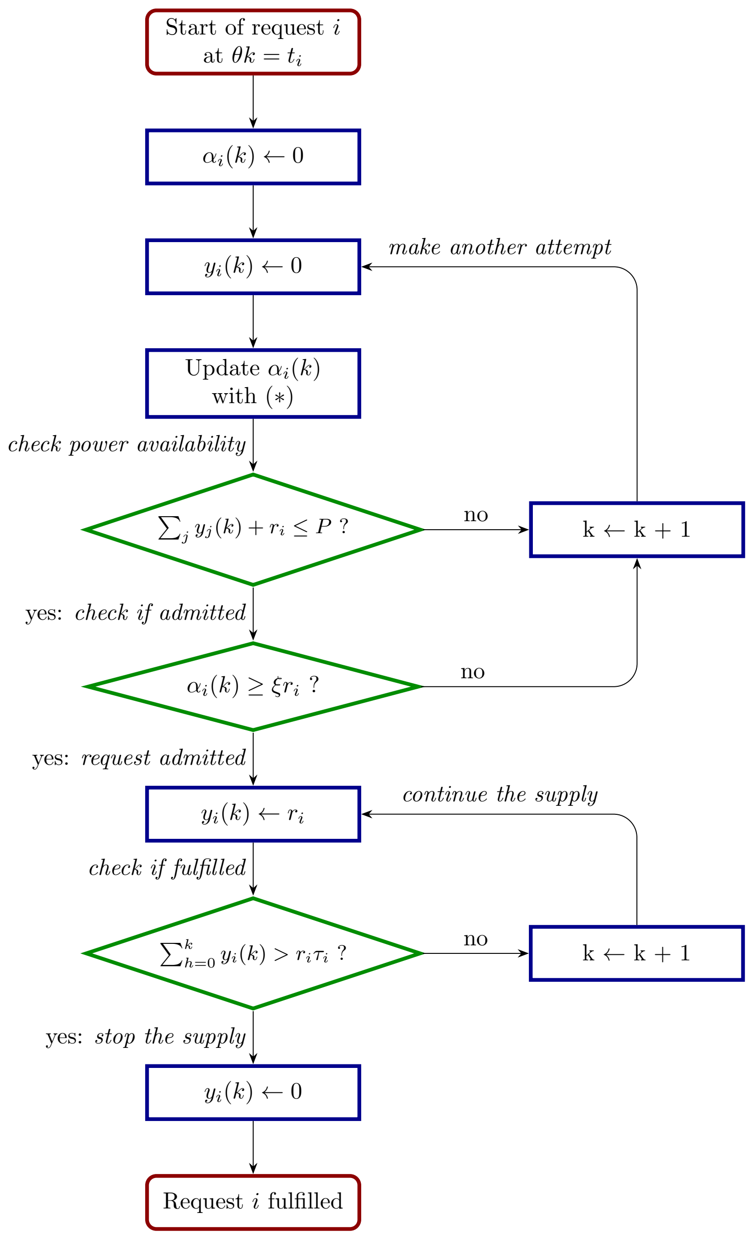


Figure 4.5: Flow-chart of the reservation-variable-based strategy for the non-interruptible *NI* case. $(*)$ refers to control Eq. (4.5).

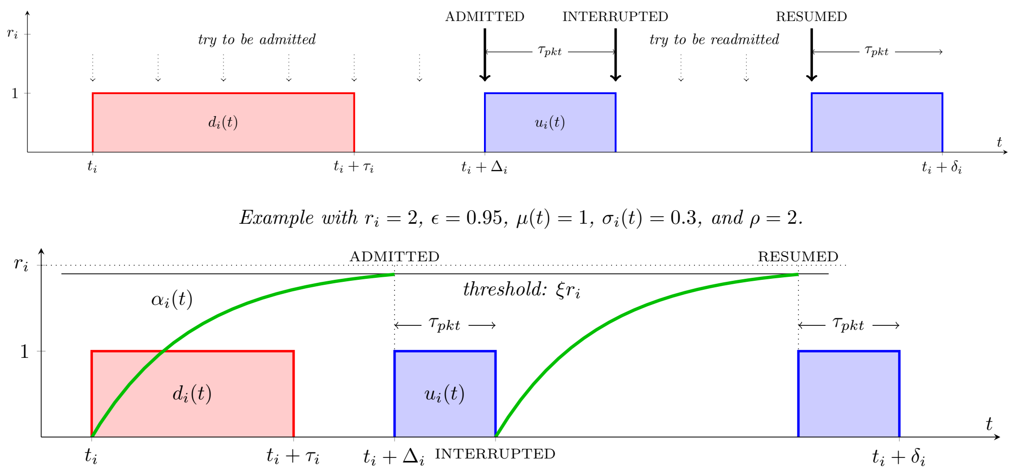


Figure 4.6: The p-persistent-based (top) and reservation-variable-based (bottom) strategies for the interruptible *IT*-packets case, where the requests are divided into packets.

### 4.2.1 The greedy heuristic (*IT* strategy)

The centralized greedy strategy for the *IT* supply strategy can be formulated exactly as in subsection 4.1.1, in two ways: the only difference is that now $U_i^k = 0$ for all $i, k$ as there is no need to impose that once a request is admitted, it must be supplied at maximum rate without interruptions until it is fulfilled.

### 4.2.2 The predictive control heuristic (*IT* strategy)

For the predictive control heuristic for the *IT* supply strategy, an optimization problem, either the exact problem Eq. (3.11) or the relaxed problem Eq. (3.18) with the constraint $u_i(t) \in \{0,1\}$, is solved at each $t_k = k\theta$ over a finite time horizon $T_{pc}$, considering all the active unfulfilled requests $i \in \mathcal{A}(t_k)$ up to the current time $t_k$. Then, at $\hat{t} + \Theta$ the process is repeated, taking into account the variations of the system.

Compared to the *NI* case, the admitted requests can be rescheduled, so the requests in $\mathcal{A}(t_k)$ are considered, instead of just those in $\mathcal{R}(t_k)$. Moreover, there is no need to enforce the modified constraint Eq. (4.4) with time-varying $P(t)$, since the profile of $u_i(t)$ for $t \geq t_k + \Theta$ is modified by the next optimization performed at $t_k + \Theta$. Indeed, once the "temporary" optimal solution $\hat{u}_i^*(t)$ is obtained by solving the sub-problem at time $t_k$, the actual supply is set for the $[t_k; t_k + \Theta)$ time interval as

$$u_i(t) \leftarrow \hat{u}_i(t), \text{ for } \; t \in [t_k; t_k + \Theta).$$

Like in subsection 4.1.2, all the requests to be considered in each optimization problem at time $t_k$ can use the same release time $\hat{t}_i = t_k$. Moreover, to take into account the portion of the request that has possibly already been fulfilled before $t_k$, the duration of the request $\widetilde{\tau}_i(\hat{t})$ is to be used in the computations, instead of $\tau_i$.

The exact problem Eq. (3.11) to be solved at each $t_k$ becomes

$$\begin{aligned}
\min \sum_{i \in \mathcal{A}(t_k)} \omega_i = & \sum_{i \in \mathcal{A}(t_k)} \int_{t_k}^{t_k+\Theta} [\hat{z}_i(t) - \hat{u}_i(t)] \, dt, \\
\text{s.t. } \; \hat{z}_i(t) = & \begin{cases} 1\,, & \text{if } t \in [t_k, t_k + \hat{\delta}_i]\,, \\ 0\,, & \text{otherwise}\,, \end{cases} \\
& 0 \leq \hat{u}_i(t) \leq \hat{z}_i(t) \leq 1, \\
& \hat{u}_i(t), \hat{z}_i(t) \in \{0,1\}, \\
& \int_{t_k}^{t_k+\Theta} \hat{u}_i(t) \, dt = \widetilde{\tau}_i, \\
& \sum_{i=1}^{n} r_i \hat{u}_i(t) \leq P, \quad \text{for all } t_k \leq t \leq t_k + \Theta\,.
\end{aligned}$$

Discrete-time implementations using the Linear Combination (LC) and Start–Ups (SU) techniques, as reported in subsection 3.2.1, result in a Mixed-Integer-Linear-Programming problem.

Alternatively, the relaxed problem Eq. (3.18) to be solved at each $t_k$ becomes

$$\begin{aligned}
\min & \int_{t_k}^{t_k+\Theta} \sum_{i \in \mathcal{A}(t_k)} \hat{x}_i(t) dt, \\
\text{s.t. } & \dot{\hat{x}}_i(t) = \frac{1}{\widetilde{\tau}_i} \left[\hat{d}_i(t) - \hat{u}_i(t)\right], \\
& \hat{d}_i(t) = \begin{cases} 1 & \text{if } \; t_k \leq t \leq t_k + \widetilde{\tau}_i\,, \\ 0 & \text{otherwise,} \end{cases} \\
& 0 \leq \hat{u}_i(t) \leq 1\,, \quad \text{for all } t_k \leq t \leq t_k + \Theta, \\
& \hat{x}_i(0) = 0\,, \\
& \hat{x}_i(t) \geq 0\,, \quad \text{for all } t_k \leq t \leq t_k + \Theta, \\
& \hat{x}_i(t) = 0\,, \quad \text{for all } t_k + \delta_i \leq t \leq t_k + \Theta, \\
& \sum_{i \in \mathcal{A}(t_k)} r_i \hat{u}_i(t) \leq P, \quad \text{for all } t_k \leq t \leq t_k + \Theta\,, \\
& u_i(t) \in \{0,1\}.
\end{aligned}$$

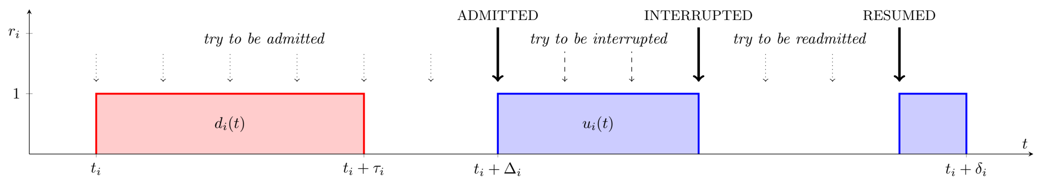


Figure 4.7: The $p$-persistent-based strategy for the interruptible $IT$ case.

Discrete time implementation, as reported in subsection 3.3.3, results in a Mixed-Integer-Linear-Programming problem in both cases, because of $u_i(t) \in \{0, 1\}$.

### 4.2.3 The priority queue-based heuristic ($\boldsymbol{IT}$ strategy)

The centralized priority queue-based heuristic for the $IT$ supply strategy is similar to the corresponding one for the $NI$ case described in subsection 4.1.3.

The main difference is that instead of the admission queue, the *active queue* is used to store the active requests $i \in \mathcal{A}(t)$ in the central manager. In the active queue, there are both the non-admitted requests and the already-admitted requests, which could possibly be rescheduled (interrupted and resumed) at any time.

### 4.2.4 The p-persistent-based heuristic ($\boldsymbol{IT}$ strategy)

The decentralized p-persistent-based strategy for the $IT$ supply strategy is similar to that for the $NI$ strategy from subsection 4.1.4: the same mechanism is used to admit a request; however, an *interruption probability* $p_i^I$ is also introduced for each request $i \in \mathcal{S}(t)$ which is currently being supplied. In the case in which the supply is interrupted before the request is fulfilled, it is readmitted with probability $p_i^A(t)$ as soon as there is power availability, using (again) the same mechanism of the $NI$ case, see Fig. 4.7. Then, this behavior could possibly be repeated multiple times, i.e., there can be multiple interruptions/resumptions before the request is fulfilled. In Fig. 4.8 the flow-chart of the $p$-persistent-based strategy is reported.

The interruption probability $p_i^I$ could be equal to the interruption priority of the request, i.e., $p_i^I = \pi_i^I(t)$ (recall that the priority is a number between 0 and 1, hence it could be a probability). Then, if the system saturation is low, the request is very likely not to be interrupted immediately, regardless of its impact on the system; otherwise, if the saturation is high, requests with low impact are less likely to be interrupted compared to those with high impact, for which the interruption priority is close to one.

### 4.2.5 The reservation-variable-based heuristic ($\boldsymbol{IT}$ strategy)

The decentralized reservation-variable-based strategy for the $IT$ supply strategy is similar to that for the $NI$ strategy from subsection 4.1.5: the same mechanism is used to admit a request.

However, just after the admission, at $t_i + \Delta_i$, for each request $i \in \mathcal{S}(t)$ which is currently being supplied, $\alpha_i(t)$ is updated in a different way, so that it decreases at high saturation:

$$\dot{\alpha}_i(t) = -\nu_i(t) \cdot \alpha_i(t) \cdot (\sigma(t))^\rho, \ \forall i \in \mathcal{S}(t), \tag{4.7}$$

where $\nu_i(t)$ is a non-negative function of time and $\rho$ a given positive parameter.

The decrease of this function at time $t$ now depends on $\nu_i(t)$ and on the saturation $\sigma(t) = y_{tot}(t)/P \in [0, 1]$. In particular, on the one hand, the term $(\sigma(t))^\rho$ ensures that the more the system is saturated, the larger $\alpha_i(t)$ decreases; when no other request is being supplied ($\sigma(t) = 0$), there is no decreasing. On the other hand, a larger $\nu_i(t)$ means a faster decreasing; depending on its specification, it could be used to differentiate the behavior of different requests. The term $\alpha_i(t)$ makes $\alpha_i$ converge to 0 from above, as it makes the derivative become zero.

Let $\epsilon$ be a constant close and smaller than 1. As soon as $\alpha_i(t)$ reaches a threshold $(1 - \epsilon)r_i$, which is positive and close to zero, the supply is interrupted. Note that the time in which the threshold is reached does not depend on $r_i$ (unless $\nu_i(t) = \nu_i(t, r_i)$). In the case in which the supply is interrupted before the request is fulfilled, it is readmitted using again the same mechanism of the $NI$ case, increasing $\alpha_i(t)$ with Eq. (4.5) to readmit it when threshold $\xi r_i$ is reached, see Fig. 4.9. Multiple interruptions/resumptions are possible before the request is fulfilled. In Fig. 4.10 the flow-chart of the reservation-variable-based strategy is reported.

Function $\nu_i(t)$ could be

- constant and equal to some $\nu_0$, i.e., $\nu_i(t) = \nu_0$;

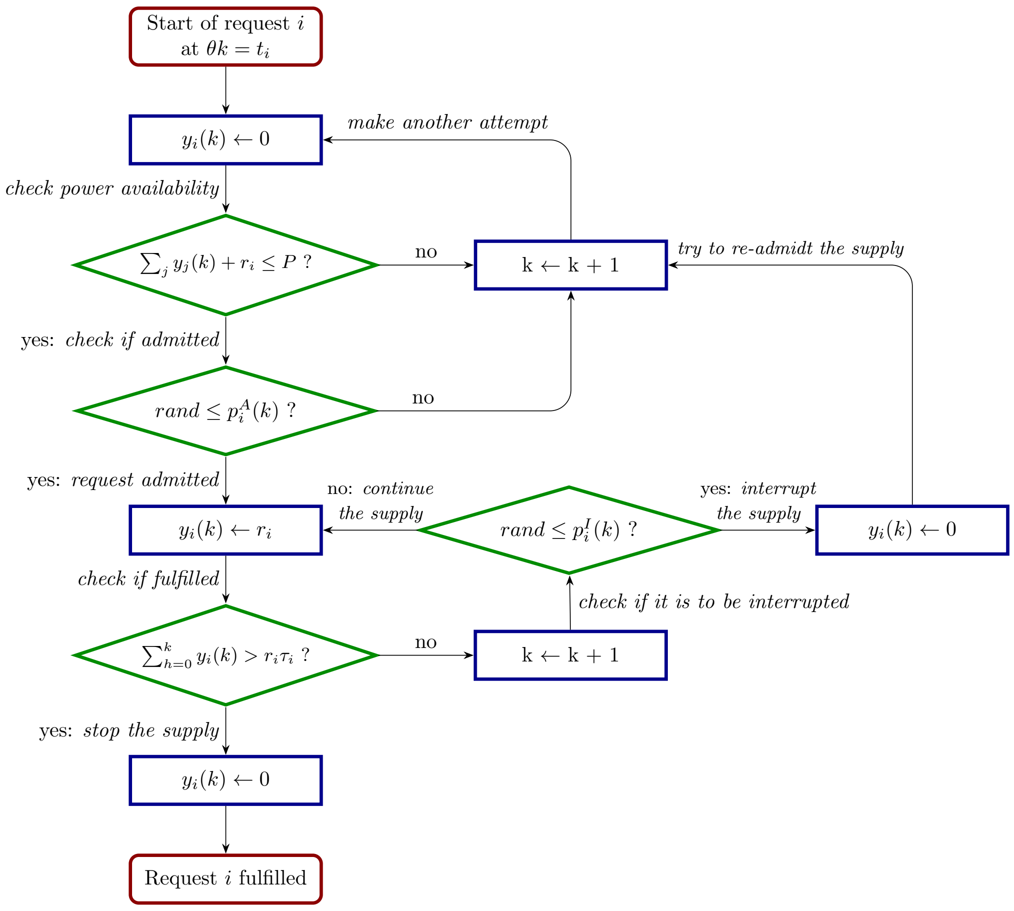


Figure 4.8: Flow-chart of the $p$-persistent-based strategy for the interruptible $IT$ case.

- proportional to the interruption priority of the request, i.e., $\nu_i(t) = \nu_0\pi_i^I(t)$. In this enhanced version, if the system saturation is low, the decrease of variable $\alpha_i$ is very slow, regardless of the impact of the request on the system, so that possible interruptions occur after a long time. Otherwise, if the saturation is high, the higher is the impact, the faster $\alpha_i$ decreases, so that requests with high impact tend to be interrupted earlier.

For the interruption process, note that when $\nu_i(t) = \nu_0$, the control law is the same for every request being supplied, so that the control variables corresponding to earlier requests have been decreased for a longer time, based on the saturation of the system, so that these are interrupted earlier: a priority order (based on the requests' (re)-admittance times) is created among users in a decentralized way and fairness is assured. Time-varying $\nu_0\pi_i^I(t)$ enhances this behavior by prioritizing the interruption of the requests with high impact.

For implementation, control law Eq. (4.7) for $i \in \mathcal{S}(t)$ can be easily discretized via Euler scheme as:

$$\alpha_i(k+1) = \alpha_i(k) - \hat{\theta} \cdot \nu_i(k) \cdot \alpha_i(k) \cdot (\sigma(k))^\rho. \tag{4.8}$$

## 4.3 The variable-rate case

In this Section, the real-time heuristics for the variable rate supply strategy are presented.

In general, now there is no need to impose that the supply can only be either 0 or $r_i$, nor to impose that the supply cannot be interrupted.

### 4.3.1 The greedy heuristic (*VR* strategy)

The centralized greedy strategy for the *VR* supply strategy can be formulated exactly as in subsection 4.1.1: the only difference is that the constraint $u_i(t) \in \{0, 1\}$ is replaced by $0 \leq u_i(t) \leq 1$, and $U_i^k = 0$ for all $i, k$.

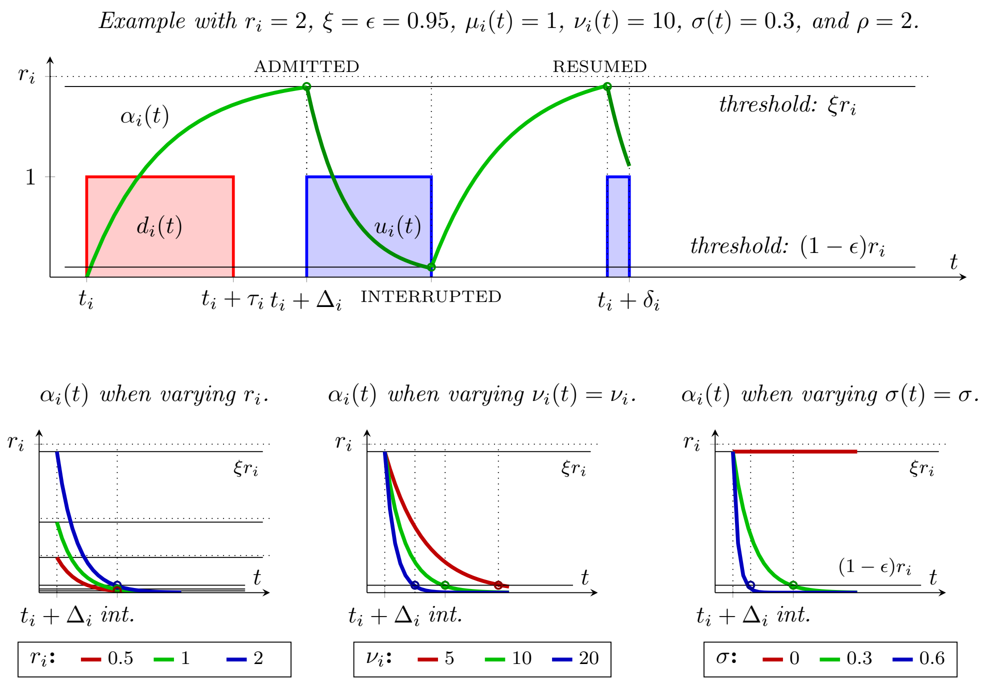


Figure 4.9: The reservation-variable-based strategy for the interruptible *IT* case. Assuming $\mu_i(t), \nu_i(t)$ and $\sigma(t)$ constant, variable $\alpha_i(t)$ for request $i$ evolves according to $\alpha_i(t) = r_i\left(1 - e^{-\mu_i(t)(1-\sigma(t))^\rho(t-t^*)}\right)$, until it is admitted, for $t^* = t_i \le t \le t_i + \Delta_i$. Then, it evolves according to $\alpha_i(t) = \xi r_i e^{-\nu_i(t)(\sigma(t))^\rho(t-t^{**})}$, from $t^{**} = t_i + \Delta_i$ until it is interrupted at $t_{int}$. Then, it evolves again according to $\alpha_i(t) = r_i\left(1 - e^{-\mu_i(t)(1-\sigma(t))^\rho(t-t^*)}\right)$, from $t^* = t_{int}$ until it is resumed, and so on. When the supplying is non-zero, the decreasing is faster as $\nu_i(t)$ increases, and as $\sigma(t)$ increases. However, the time in which the threshold $\epsilon r_i$ is reached does not depend on $r_i$. In practice, the saturation $\sigma(t)$ varies over time (as well as $\mu_i(t)$ and $\nu(t)$, possibly), so that function $\alpha_i(t)$ has a more complex behavior, but still remains an increasing function approaching $r_i$ when there is no supply ($u_i(t) = 0$), and a decreasing function approaching 0 when supply is active ($u_i(t) = 1$).

## 4.3.2 The predictive control heuristic (*VR* strategy)

The centralized predictive control strategy for the *VR* supply strategy can be formulated exactly as in subsection 4.2.2: the only difference is that the constraint $u_i(t) \in \{0, 1\}$ is replaced by $0 \le u_i(t) \le 1$.

## 4.3.3 The priority queue-based heuristic (*VR* strategy)

The centralized priority queue-based strategy for the *VR* supply strategy generalizes the corresponding *NI* and *IT* strategies by allowing rate reduction: instead of possibly assigning either a fixed quantity $r_i$ of resource, or none, according to a given criterion, even a fraction of $r_i$ can be assigned.

New requests are still sent to a central manager and inserted in the *active queue* (storing requests $i \in \mathcal{A}(t)$), whose elements are still sorted according to a given order which reflects the impact $q_i(t)$[1]. Then, at each time $t$, the total available power $P$ is partitioned among all the users in $\mathcal{A}(t)$ according to their impact, so that the lower $q_i(t)$, the higher the amount of resource is assigned, which is between 0 and $r_i$.

In particular, at each $t$, $y_i(t) = 0$ is temporally set for each $i \in \mathcal{A}$. The, by evaluating such requests one at a time in the order given by the queue, the supplied power for each of them is

$$y_i(t) = \min\left(r_i, \frac{P - \sum_{j\in\mathcal{S}(t),\ q_j(t)<q_i(t)} y_j(t)}{\sum_{j\in\mathcal{R}(t),\ q_j(t)\ge q_i(t)} \frac{1}{q_j(t)}} \frac{1}{q_i(t)}\right).$$

The meaning is the following. $\sum_{j\in\mathcal{S}(t),\ q_j(t)<q_i(t)} y_j(t)$ is the total power assigned so far at time $t$ to other requests (whose impact is lower than $q_i(t)$, the one of the request). Then, $P - \sum_{j\in\mathcal{S}(t),\ q_j(t)<q_i(t)} y_j(t)$ is the amount of resource yet to be assigned at time $t$. The amount of assigned resource to request $i$ is a fraction of it, which is given by $(1/q_i(t))/\left(\sum_{j\in\mathcal{R}(t),\ q_j(t)\ge q_i(t)} 1/q_j(t)\right)$, considering the inverse of $q_i(t)$ and the total one among the

[1]Here, for the SREF sorting method, $r_M$ and $\tau_M$ are computed so that they are bigger than the (known) $r_i > 0$ and $\tau_i > 0$ of the elements in the queue, to avoid clipping. For the FIFO case, the inverse of the time waited so far is considered as $q_i(t)$. $q_i(t)$ could also be redefined for the other sorting methods.

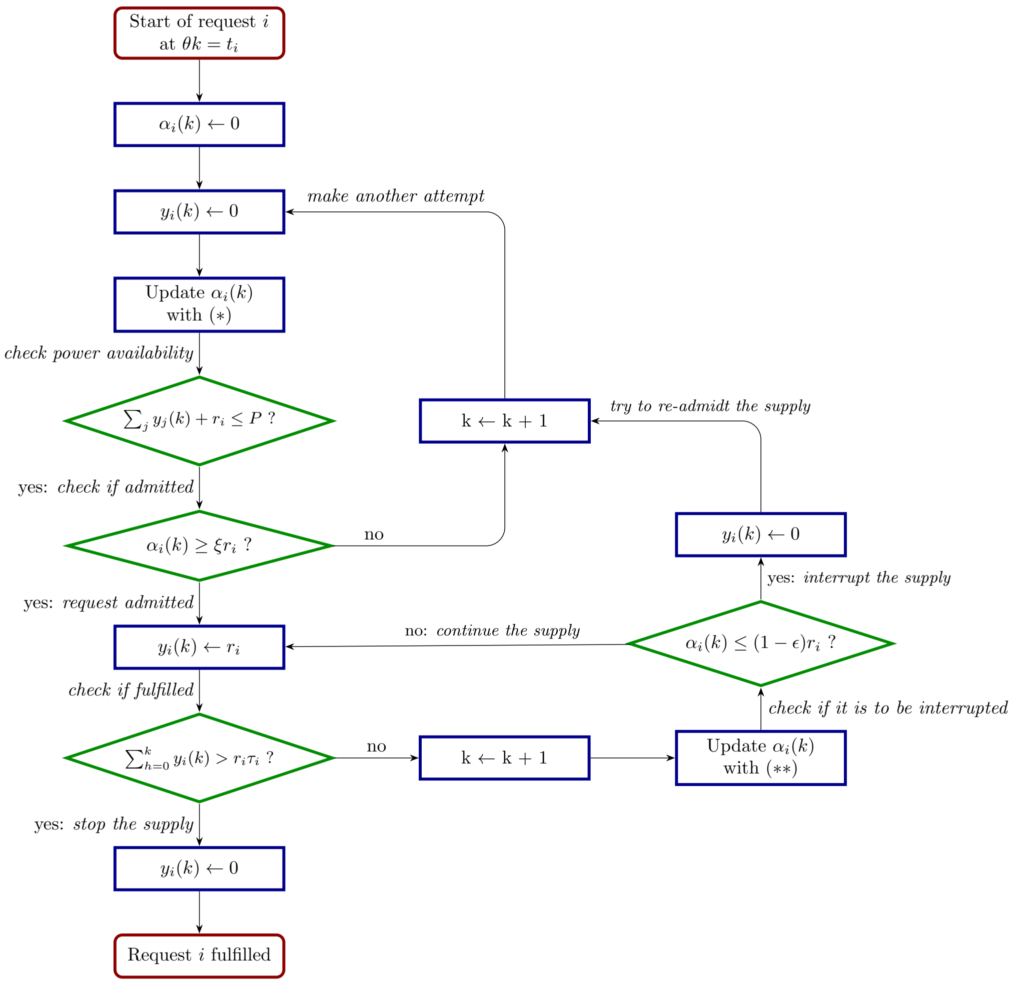

Start of request $i$ at $\theta k = t_i$
$\alpha_i(k) \leftarrow 0$
$y_i(k) \leftarrow 0$
make another attempt
Update $\alpha_i(k)$ with $(*)$
check power availability
$\sum_j y_j(k) + r_i \le P$ ?
k ← k + 1
try to re-admidt the supply
yes: check if admitted
$\alpha_i(k) \ge \xi r_i$ ?
no
$y_i(k) \leftarrow 0$
yes: interrupt the supply
yes: request admitted
no: continue the supply
$y_i(k) \leftarrow r_i$
$\alpha_i(k) \le (1-\epsilon) r_i$ ?
check if fulfilled
check if it is to be interrupted
$\sum_{h=0}^{k} y_i(k) > r_i \tau_i$ ?
no
k ← k + 1
Update $\alpha_i(k)$ with $(**)$
yes: stop the supply
$y_i(k) \leftarrow 0$
Request $i$ fulfilled

Figure 4.10: Flow-chart of the reservation-variable-based strategy for the interruptible $IT$ case. $(*)$ refers to control Eq. (4.5), and $(**)$ refers to control Eq. (4.7).

others yet to be processed. The assigned resource is finally clipped to $r_i$. The process is possibly reiterated until all the power is assigned, or all requests are fulfilled.

### 4.3.4 The p-persistent-based heuristic (*VR* strategy)

The p-persistent-based heuristic is not considered for the $VR$ supply strategy.

### 4.3.5 The reservation-variable-based heuristic (*VR* strategy)

The decentralized reservation-variable-based strategy for the $VR$ supply strategy sets the power supply $y_i(t)$ of request $i$ as a function of some $\alpha_i(t)$. In particular, as the power supply cannot exceed $r_i$, and the total power supply $y_{tot}(t)$ cannot exceed $P$, it is imposed that

$$y_i(t) = \min(P - y_{tot}(t), r_i, \alpha_i(t)).$$

For all active request $i \in \mathcal{S}(t) \cup \mathcal{R}(t)$, $\alpha_i(t)$ varies according to Eq. (4.5) whenever there is enough resource for the supplying ($\alpha_i(t) \le P - y_{tot}(t)$), and switches to Eq. (4.7) otherwise ($\alpha_i(t) > P - y_{tot}(t)$), to reduce $y_i(t)$ and the saturation of the system. In particular,

$$\dot{\alpha}_i(t) = \begin{cases} \mu_i(t) \cdot (r_i - \alpha_i(t)) \cdot (1-\sigma(t))^{\rho}, & \text{if } \alpha_i(t) \le P - y_{tot}(t), \\ -\nu_i(t) \cdot \alpha_i(t) \cdot (\sigma(t))^{\rho}, & \text{otherwise}, \end{cases}$$

with $\alpha(t_i) = 0$.

# Chapter 5
# Application to EV charging scheduling

In this Chapter, the proposed optimal control framework from Chapter 3 and the presented online heuristics from Chapter 4 are evaluated (a-posteriori) and compared.

These have been implemented in Matlab. The Opti Toolbox interface [63] and the SCIP solver [64] have been used for solving the LP and MILP problems. All the simulations were performed on a dual-core Intel Core i3 at 2.3 GHz with 8 GB of RAM.

The specific case of electric vehicles (EV) battery charging scheduling scenario in a power supply facility has been considered. For the charging requests, realistic data have been obtained from the ACN-Data dataset, which collects data recorded at the existing charging site at the Caltech university campus [25]. Since not all the required data can be obtained from this dataset, some of them have been generated stochastically.

## 5.1 Scenario and data

In the presented framework, each generic request $i$ is characterized by three quantities:

- $t_i$, the release time;
- $r_i$, the maximum supply rate;
- $\tau_i$, the minimum processing time, i.e., the length of the supply, if this occurs at the maximum rate $r_i$ without interruptions. $E_i = r_i\tau_i$ is the requested amount of resource that must be supplied.

In the specific case of EV charging scheduling, the considered charging system is composed by a set of charging stations and a set of $n$ EVs, each of which makes a charging request for their batteries. Many existing standards regulate the charging process and the required infrastructure [18]. Traditionally, uncoordinated charging is performed at charging stations (i.e., batteries are charged at as soon as the requests are made at maximum rate possible). However, this increases the load at peak hours, which leads to the risk of overheating and damaging of the infrastructure, and to the need of a grid that supports a higher capacity. To overcome this, many models of coordinated charging (also called smart charging) have been proposed, both centralized and distributed, in which the timing and the charging power are optimized.

The following quantities are also specified:

- $R_{CS}$, the maximum output power of each charging station, which can be delivered to the battery of the connected EV;
- $P$, the system capacity, that is the maximum overall power that can be supplied by all these charging stations at any time;
- $R_{EV,i}$, the maximum input power that the battery of the EV making the $i$th request can accept (also known as *acceptance rate*);
- $C_i$, the capacity of the battery of the EV making the $i$th request, which is the maximum amount of energy that can be stored in there;
- $E_i$, the requested energy for request $i$, which is smaller or equal to $C_i$ (assuming no discharging). Here, it is assumed that the value of $E_i$ is provided by the user before the charging starts.

Several models for the actual EV charging power profile have been proposed [65, 66, 57]. To simplify the system for the proposed approach, a basic approximated model of EVs charging has been considered for the charging requests, by which the ideal supply is performed at maximum rate $r_i$, without interruptions and for a duration of $\tau_i$. Non-idealities are taken into account by defining the average charging efficiency $\mu_{eff_i} \in [0, 1]$, see [67]. Then, $r_i$ and $\tau_i$ can be approximated as

$$r_i = min(R_{CS}, R_{EV_i}), \tag{5.1}$$

$$\tau_i = \frac{1}{\mu_{eff_i}} \cdot \frac{E_i}{r_i}. \tag{5.2}$$

The first relation means that the maximum supply rate is limited by the minimum between the acceptance rate of the battery of the EV and the maximum output power of the charging station. The second relation means that the actual charging time is increased compared to the ideal one $E_i/r_i$, due to possible losses during the charging process.

The data that will be used in the simulations are obtained from the ACN-Data dataset [25, 68], which collects information that was recorded at the existing charging site at the Caltech university campus, which has 54 charging stations (with maximum output power $R_{CS} = 7\ kW$) and is open to the public, despite most usage being by faculty, staff, and students.

For each power request $i$, the following quantities are reported, among others:

- $t_i^*$, the time at which the request has been made;
- $E_i^*$, the actual delivered energy during the charging session;
- $r_i^*$, the average value of the supplied power rate at which the charging was performed.

The analysis of the data shows that the total number of charging requests per day does not depend on the time of the year. The periodicity is weekly: weekdays (Monday to Friday) are similar one to each other, while on weekend days there is a smaller number of charging requests. Most of the requests (and the peak of the demanded power) occurs in the weekdays morning, and partially in the afternoon, while there are very few in the evening, by night and during weekends, which results in not having overlapping requests between different days: generally, the requests from one day can be served before the ones from the next day start. This behavior is due to the fact that the site is mostly used during work hours by faculty, staff, and students.

Then, it is reasonable to consider the requests from one day only. The requests of Tuesday, 19 February 2019 have been considered, which are $n = 32$ (see Table 5.1). Assuming a sampling time of $\theta = 5$ minutes, the day is divided into $N = 288$ time-slots.

Table 5.1: Input data (raw) for the 32 charging requests $i$ for the day Tuesday, 19 February 2019, from the ACN-Data dataset [25].

| $i$ | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $t_i/\theta$ | 88 | 92 | 99 | 102 | 103 | 104 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 114 |
| $r_i^*$ $[kW]$ | 2.9307 | 2.6386 | 1.9342 | 3.1777 | 2.4992 | 2.2219 | 4.8917 | 2.7312 | 1.4199 | 3.0242 | 6.5855 | 2.8137 | 2.7181 | 0.98436 | 6.4608 | 3.4964 |
| $E_i^*$ $[kWh]$ | 32.189 | 15.568 | 13.314 | 2.754 | 2.166 | 17.479 | 46.308 | 24.535 | 7.904 | 8.014 | 13.61 | 4.924 | 24.916 | 4.971 | 30.689 | 12.587 |
| $i$ | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** | **26** | **27** | **28** | **29** | **30** | **31** | **32** |
| $t_i/\theta$ | 116 | 117 | 120 | 123 | 126 | 126 | 130 | 136 | 165 | 213 | 221 | 239 | 241 | 241 | 246 | 256 |
| $r_i^*$ $[kW]$ | 0.20808 | 3.364 | 1.9951 | 1.4888 | 1.8197 | 4.1547 | 2.5433 | 1.6915 | 2.3192 | 2.5278 | 3.3416 | 3.5009 | 5.6494 | 5.985 | 6.1348 | 1.7167 |
| $E_i^*$ $[kWh]$ | 0.919 | 7.625 | 16.127 | 3.201 | 15.164 | 13.572 | 2.077 | 8.373 | 4.677 | 0.969 | 4.901 | 16.279 | 6.591 | 9.975 | 7.873 | 0.515 |

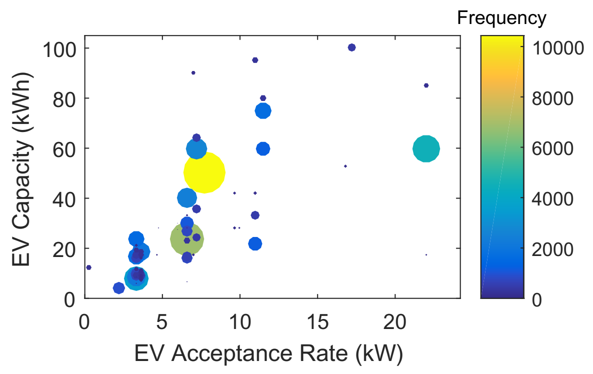
However, since in the charging site at the Caltech university campus an adaptive algorithm that varies the power is actually used to optimize the charging [21, 22], the $r_i^*$ and $\tau_i^* = (E_i^*/r_i^*)/\mu_{eff_i}$ cannot be used directly as the required $r_i$ and $\tau_i$, since they are already "optimized", in some way. Also, the acceptance rate $R_{EV_i}$ and the capacity $C_i$ of each EV's battery is not reported in the dataset, nor is the corresponding car model from which this information can be retrieved. Then, to get both $r_i$ and $\tau_i$ using Eqs. (5.1) and (5.2), data have been integrated stochastically assigning an acceptance rate $\widetilde{R}_{EV_i}$ and a capacity $\widetilde{C}_i$ and to each request, taking into account the popularity of the most common EVs models [69] and the acceptance rate and capacity of their batteries [70, 71], as described in the next subsection.

### 5.1.1 Data preprocessing

Consider the data of the charging sessions from [25]. These have been first preprocessed by removing the recorded requests with missing values or with inconsistent data (e.g., negative charging duration or charging duration greater than one day). Denote the resulting dataset as the *"EV requests" dataset.*

The distributions of the capacity $C_i$ and the acceptance rate $R_{EV,i}$ of the most common EV models have also been considered. In particular, different EV models and their popularity (in Washington State) were obtained from [69]; these have been integrated with data (acceptance rate and battery capacity) from [70, 71]. Denote the resulting dataset as the *"EV models" dataset.*

In Fig. 5.1, left, the relations between the capacity $C_i$ and the acceptance rate $R_{EV,i}$ of the most popular EV models are reported as a scatter plot: the size and color of each point are based on the popularity of each model. There is a weak positive correlation between $C_i$ and $R_{EV,i}$, in general: batteries with larger capacity tend to support a larger acceptance rate. As an example, assuming to have charging stations with $R_{CS} = 7\ kW$, the relations between the minimum charging times $\tau_i$ to get a full battery charging from 0% to 100% (i.e., $E_i = C_i$) and the maximum rate $r_i$, obtained using Eqs. (5.1) and (5.2), are reported in Fig. 5.1, right.



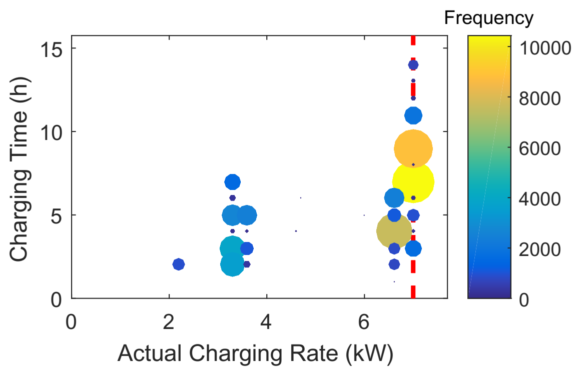


Figure 5.1: Left: relations between the acceptance rate and the capacity of the batteries of the most common models of EV according to the "EV models" dataset. Right: relations between the actual charging rate of the same models of EV and the charging duration to get a full battery charging, when the maximum output power of the charging stations is 7 $kW$ (see red dashed line). In both graphs, each point represents a model of EV: its size and color are representative of the popularity of that particular model.

The two "EV requests" and "EV models" datasets have been integrated by stochastically assigning a random car model (and its acceptance rate $\widetilde{R}_{EV_i}$) from the latter dataset to each request (characterized by $r_i^*$ and $E_i^*$) in the former dataset, choosing among the models which have a battery with an acceptance rate greater or equal to $r_i^*$ and a capacity greater or equal to $E_i^*$ (indeed, it must hold that $E_i^* \leq \widetilde{C}_i$ and $r_i^* \leq \widetilde{R}_{EV,i}$); the probability is higher for the models which have a capacity and acceptance rate closer to $E_i^*$ and $r_i^*$, respectively.

In particular, consider the generic request $i$ of the "EV requests" dataset. Let $\mathcal{M}$ be the set of the capacity/acceptance rate pairs $(R_{EV}, C)$ from the "EV models" dataset. Then, consider the subset

$$\mathcal{M}_i = \{(R_{EV}, C) \in \mathcal{M} : r_i^* \leq R_{EV} \text{ and } E_i^* \leq C\}.$$

The joint 2D probability distribution $p_i(R_{EV}, C)$ of the selected pairs $(R_{EV}, C) \in \mathcal{M}_i$ is computed using the bivariate histogram bin counts and normalizing the total sum to 1.

Also, a 2D weighting function $w_i(R_{EV}, C)$ is defined to make the pairs $(R_{EV}, C) \in \mathcal{M}_i$ closer to the pair $(r_i^*, E_i^*)$ more probable, as

$$w_i(R_{EV}, C) = w_{R,i}(R_{EV}) \cdot w_{C,i}(C),$$

in which

$$w_{R,i}(\xi) = \left(1 - \frac{\xi - r_i^*}{M_R - r_i^*}\right)^2, \qquad M_R = \max_{(R_{EV},\cdot)\in\mathcal{M}_i} \{R_{EV}\} + \delta R,$$

$$w_{C,i}(\xi) = \left(1 - \frac{\xi - E_i^*}{M_C - E_i^*}\right)^2, \qquad M_C = \max_{(\cdot,C)\in\mathcal{M}_i} \{C\} + \delta C,$$

where $\delta R$ and $\delta C$ are two positive scalars with small values, introduced to avoid divisions by 0. By definition, $w_i(r_i^*, E_i^*) = 1$, and $w_i(R_{EV}, C) \to 0$ as $R_{EV}, C$ increase.

The final weighted joint 2D probability distribution $p_{w,i}(R_{EV}, C)$ is eventually obtained as:

$$p_{w,i}(R_{EV}, C) = \frac{w_i(R_{EV}, C) \cdot p_i(R_{EV}, C)}{\int_{r_i^*}^{M_R} \int_{E_i^*}^{M_C} w_i(r, c) \cdot p_i(r, c)\ dc\ dr}.$$

For each real request $i$, a pair $(\widetilde{R}_{EV,i}, \widetilde{C}_i)$ is chosen randomly based on this weighted distribution $p_{w,i}(R_{EV}, C)$. Then, the required input data are finally obtained as

- the release time $t_i$ is the real one, taken from the dataset, $t_i = t_i^*$;
- the maximum rate is obtained from Eq. (5.1), using $\widetilde{R}_{EV_i}$, as $r_i = \min\left(R_{CS}, \widetilde{R}_{EV_i}\right)$;
- the request duration is obtained from Eq. (5.2), as $\tau_i = \frac{1}{\mu_{eff_i}} \cdot \frac{E_i^*}{r_i}$.

For instance, consider the 22th request with $r_i^* = 4.1547\ kW$ and $E_i^* = 13.572\ kWh$. The 2D weighting function $w_i(R_{EV,C})$, the joint 2D probability distribution $p_i(R_{EV}, C)$ and the weighted joint 2D probability distribution $p_{w,i}(R_{EV}, C)$ are represented in Fig. 5.2. The pair $(\widetilde{R}_{EV,i}, \widetilde{C}_i)$ is chosen randomly according to $p_{w,i}(R_{EV}, C)$, resulting in $\widetilde{R}_{EV_i} = 6.6\ kW$ and $\widetilde{C}_i = 40\ kWh$. Then, setting $R_{CS} = 7\ kW$ and $\mu_{eff,i} = 0.9$, it holds that $r_i = \min(7, 6.6) = 6.6\ kW$ and $\tau_i = 13.572/6.6/0.9 = 2.2848\ h = 137.1\ min$, or $\text{ceil}(\tau_i/\theta) = 28$ time slots.

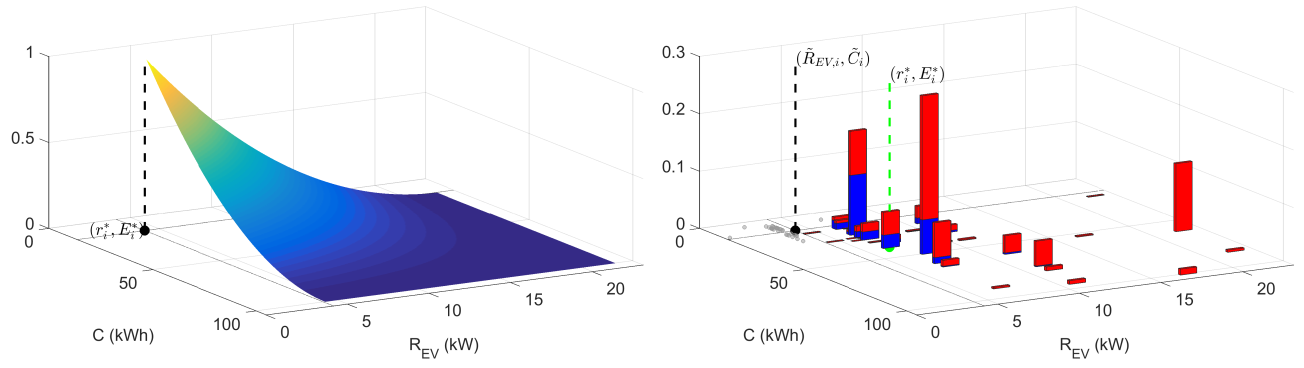


Figure 5.2: Left: the 2D weighting function $w_i(R_{EV}, C)$ for the request $i = 22$. It is one at $(r_i^*, E_i^*)$ (see the black point highlighted), and it decreases to zero as $R_{EV} > r_i^*$ and $C > E_i^*$ increase. Right: the joint 2D probability distribution $p_i(R_{EV}, C)$ (red bars) and $w_i(R_{EV}, C) \cdot p_i(R_{EV}, C)$, which is a scaled version of the weighted joint 2D probability distribution $p_{w,i}(R_{EV}, C)$ (blue bars). The points $(R_{EV}, C)$ with $R_{EV} < r_i^*$ and $C < E_i^*$) are excluded from the computations (see gray points). Note that the multiplication by the weighing function makes the pairs $(R_{EV}, C)$ which are far from $(r_i^*, E_i^*)$ less probable to be selected. A random point, here highlighted in green, is selected according to $p_{w,i}(R_{EV}, C)$ and gives $(\widetilde{R}_{EV,i}, \widetilde{C}_i)$, from which the values $r_i$ and $\tau_i$ to be used in the simulations are computed.

### 5.1.2 The input data for the requests

Assume that each charging station in the considered site has a maximum output power of $R_{CS} = 7\ kW$ and that the charging efficiency of each battery is $\mu_{eff,i} = 0.9$. The data obtained using the procedure reported above for the selected day, Tuesday 19 February 2019, and adopted in the simulations, are reported in Table 5.2. No timing constraints (i.e., deadlines) have been considered.

Table 5.2: Preprocessed input data for the 32 charging requests $i$ for the day Tuesday, 19 February 2019, that are used in the simulations.

| $i$ | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $t_i/\theta$ | 88 | 92 | 99 | 102 | 103 | 104 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 114 |
| $r_i\ [kW]$ | 7.0 | 6.6 | 6.6 | 3.3 | 7.0 | 6.6 | 7.0 | 7.0 | 6.6 | 3.3 | 7.0 | 3.3 | 6.6 | 3.3 | 7.0 | 6.6 |
| $\tau_i/\theta$ | 62 | 32 | 27 | 12 | 5 | 36 | 89 | 47 | 16 | 33 | 26 | 20 | 51 | 20 | 59 | 26 |
| $i$ | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** | **26** | **27** | **28** | **29** | **30** | **31** | **32** |
| $t_i/\theta$ | 116 | 117 | 120 | 123 | 126 | 126 | 130 | 136 | 165 | 213 | 221 | 239 | 241 | 241 | 246 | 256 |
| $r_i\ [kW]$ | 6.6 | 6.6 | 3.6 | 7.0 | 3.6 | 6.6 | 3.3 | 3.3 | 3.6 | 3.3 | 3.6 | 7.0 | 6.6 | 6.6 | 6.6 | 3.3 |
| $\tau_i/\theta$ | 2 | 16 | 60 | 6 | 57 | 28 | 9 | 34 | 18 | 4 | 19 | 31 | 14 | 21 | 16 | 2 |

The distribution of the requests over the day is reported in Fig. 5.3, top. Assuming an uncoordinated charging policy, a system capacity of about 100 $kW$ would be needed to support the supply of all the requested power, see Fig. 5.3, bottom. In this Chapter it is assumed that the system capacity is limited to $P = 56\ kW$, hence the requests made in the peak hours (between 9:00 and 12:00) need to be delayed, interrupted and/or their rate reduced to avoid overloading.

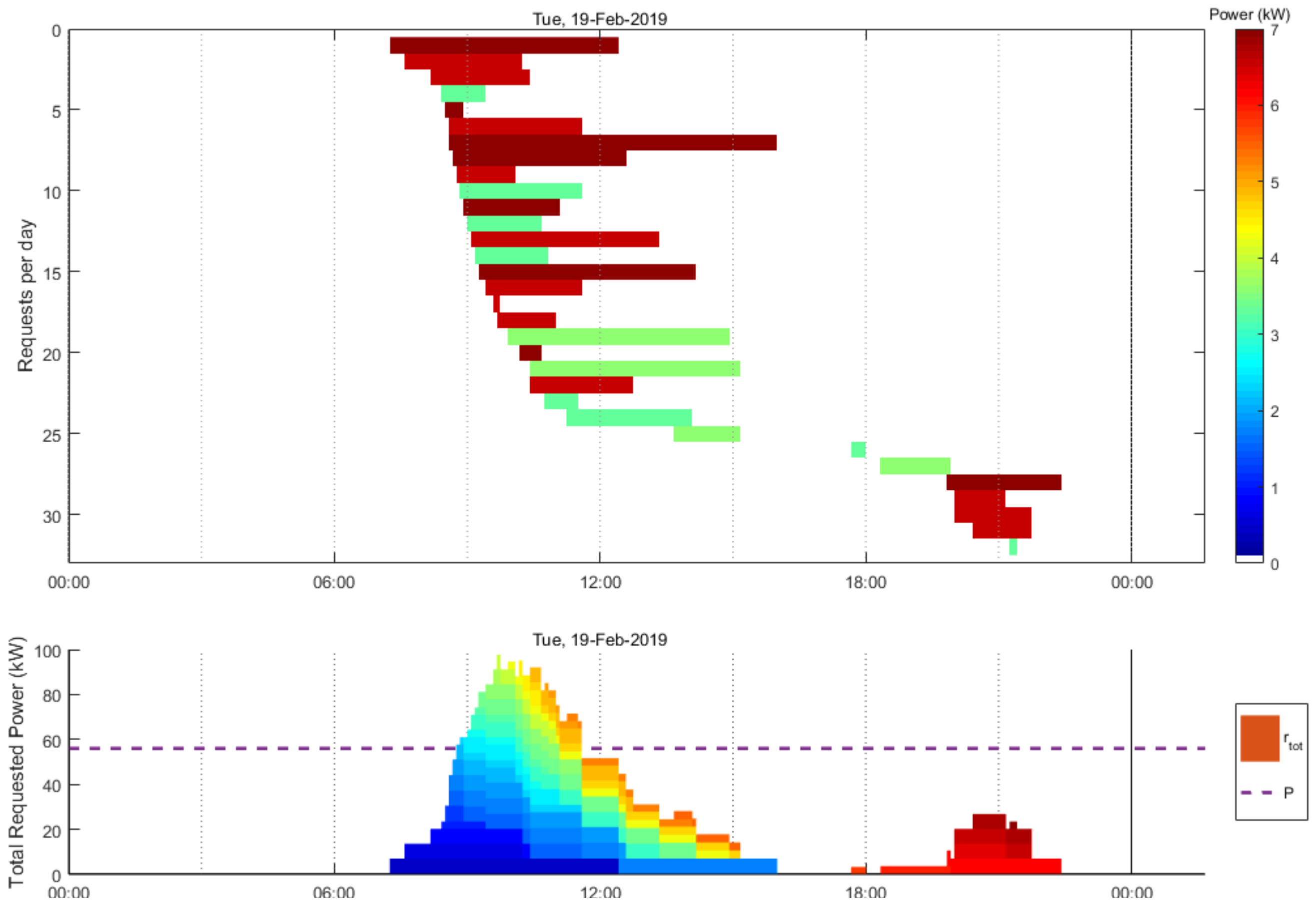


Figure 5.3: Distribution of the charging requests. Top: each row refers to a different request $i$, which is indicated by a rectangle located at $t_i$ and of length $\tau_i$. Its color refers to the value of $r_i$. Bottom: total requested power (colored area) and system capacity (dashed violet line), assuming uncoordinated charging. Here, the color is used to distinguish the power assigned to each request: blue refers to the older requests, and red to the ones occurred at the end.

### 5.1.3 Parameters, simulation settings, and metrics

The parameters of the heuristics have been tuned experimentally through trial and error, also based on the realistic input data. In particular, the following parameters are set:

- for the enhanced versions based on the impact Eq. (4.1) and the admission and interruption priorities, $r_M = 7\ kW$, $\tau_M = 300\ min$, $\rho_A = 0.5$ and $\rho_I = 20$;
- for the *IT-packets* case, the energy of the packets is fixed as $E_{pkt} = 5\ kWh$; the duration $\tau_{pkt}$ of a packet depends on $r_i$: if $r_i = 3.3\ kW$, $\tau_{pkt} = 60 \cdot E_{pkt}/r_i = 90.90$ minutes; if $r_i = 3.6\ kW$, $\tau_{pkt} = 83.33$ minutes; if $r_i = 6.6\ kW$, $\tau_{pkt} = 45.45$ minutes; if $r_i = 7\ kW$, $\tau_{pkt} = 42.86$ minutes (recall that these $\tau_i$ are discretized as floor$[(\tau_i - 1)/\theta] + 1$);
- for the predictive control heuristic, $T_{pc} = 6$ hours and $\Theta = \theta$. A timeout of 30 seconds to solve the optimization problems at each step is also imposed;
- for the reservation variable-based heuristic, $\mu_0 = 8$, $\nu_0 = 0.15$, $\rho = 0.5$, and $\epsilon = 0.9999$.

The variability due to non–determinism in the implementation of real-time strategies, is accounted for by repeating each simulation $K = 50$ times and averaging the results. In particular, for each instance $h$ of the same heuristic, denote the corresponding variables by superscript "$(h)$"; for instance, for request $i$, $\omega_i^{(h)}$ is the waiting time, $\Delta_i^{(h)}$ is the admission delay, and $x_i^{(h)}$ is the integral variable accounting for the delay.

Then, the following metrics are considered to evaluate (a-posteriori) the strategies and compare them:

- $\omega_{av}$, the average waiting time per request, $\omega_{av} = \frac{1}{K}\left(\sum_{s=1}^{K} \bar{\omega}^{(h)}\right) = \frac{1}{Kn}\left(\sum_{s=1}^{K}\sum_{i=1}^{n} \omega_i^{(h)}\right)$; that is to be minimized (recall that minimizing the average $\omega_i$ is equivalent to minimizing the average overall delay $\delta_i$);
- $X_{av}$, the average integral of $x_i(t)$ per request, $X_{av} = \frac{1}{K}\left(\sum_{s=1}^{K} \bar{X}^{(h)}\right) = \frac{1}{Kn}\left(\sum_{s=1}^{K}\sum_{i=1}^{n} x_i^{(h)}\right)$;
- $\Delta_{av}$, the average admission delay per request, $\Delta_{av} = \frac{1}{K}\left(\sum_{s=1}^{K} \bar{\Delta}^{(h)}\right) = \frac{1}{Kn}\left(\sum_{s=1}^{K}\sum_{i=1}^{n} \Delta_i^{(h)}\right)$.

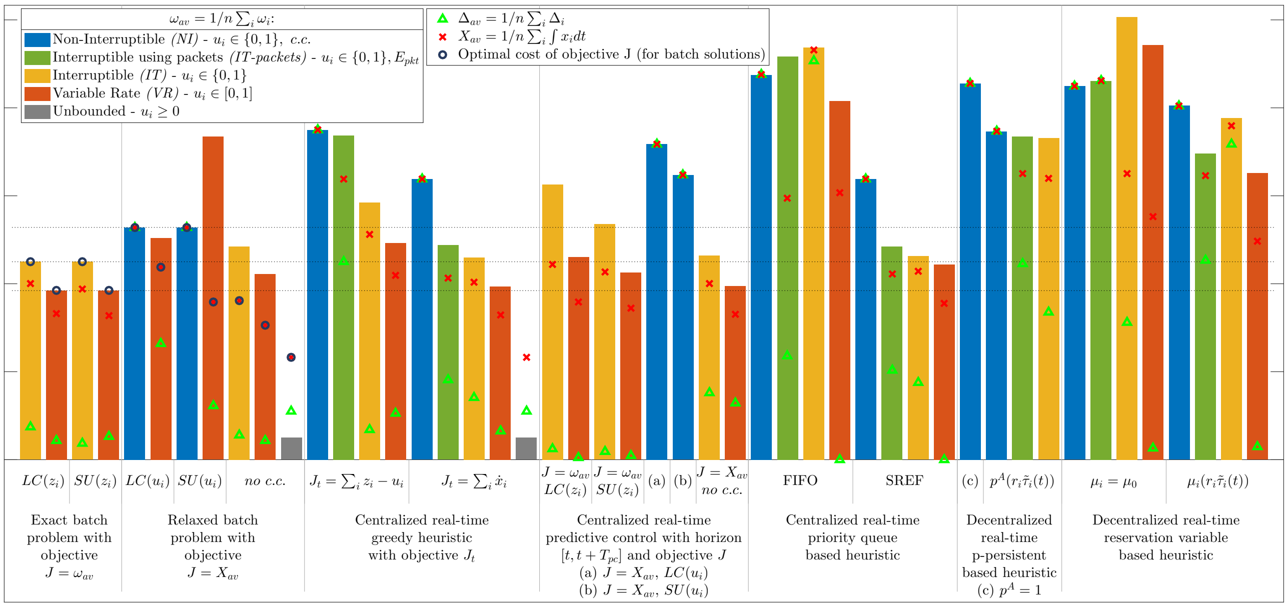


Figure 5.4: Comparison between the average waiting time per user $\omega_{av}$ (indicated by vertical bars) of the presented batch solutions and real-time heuristics for the *NI*, *IT*, and *VR* cases, obtained by simulating one day of real input data with a resolution of 5 minutes (see Table 5.3). The colors of the bars represent the classes of supplying: *NI*, *IT*, *IT*-packets or *VR*. For each solution, the average admission delay per user ($\Delta_{av}$, indicated by green triangles $\triangle$) and the average integral of $x_i(t)$ ($X_{av}$, indicated by red crosses $\times$) are also reported, as well as the optimal cost of the batch solutions for the specific objective $J$ considered (in blue circles $\circ$). All these quantities are expressed in minutes. *c.c.*: contiguity constraint Eq. (3.4i); $LC(u_i)$ (or $SU(u_i)$): the *Linear Combination* (or *Start-Ups*) alternative form is used to implement contiguity constraints Eq. (3.4i); $LC(z_i)$ (or $SU(z_i)$): the *Linear Combination* (or *Start-Ups*) alternative form is used to implement contiguity constraints Eq. (2.11). (*source:* [1], © 2022 IEEE)

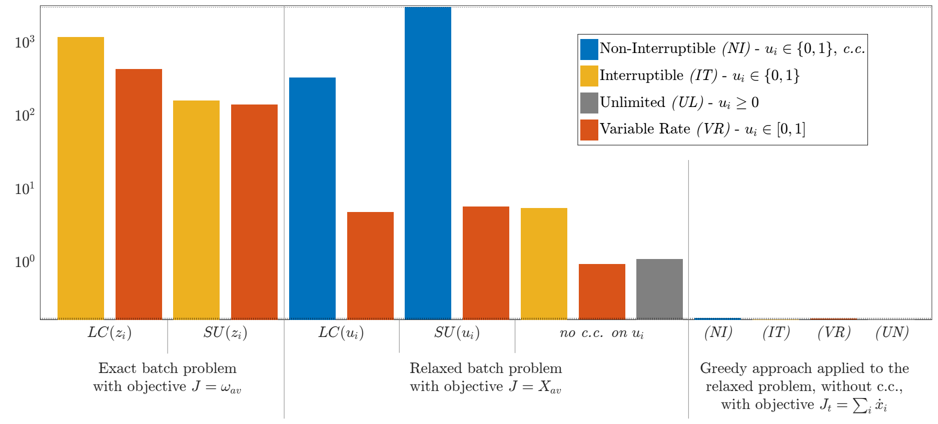


Figure 5.5: Comparison between the solving times, expressed in seconds, of the presented batch problems from Fig. 5.4. The average solving times of each iteration of the greedy strategy applied to the relaxed problem without *LC* or *SU* contiguity constraints on $u_i(t)$ are also reported and are clearly negligible. The $y$ axis is in log scale.

## 5.2 Results

The performances of the considered batch and online strategies are presented in Table 5.3, where the numeric values of $X_{av} = (\sum_i \int_0^\infty x_i dt)/n$ and $\omega_{av} = (\sum_i \omega_i)/n$ are reported for each of them. Recall that the batch relaxed problem minimizing $X_{av}$ with $0 \leq u_i \leq 1$ provides a lower bound for the optimal delays $\omega_{av}$ (computed a-posteriori for the online strategies).

To better compare the results, a visual representation is depicted in the bar chart in Fig. 5.4, where the vertical bars represent the achieved average waiting time $\omega_{av} = (\sum_i \omega_i)/n$. The color of the bars represents the specific supply strategy. The following values are also reported: the optimal cost $J$ of the (batch) optimization problems (denoted by $\circ$), $\Delta_{av} = (\sum_i \Delta_i)/n$ (denoted by $\triangle$) and $X_{av} = (\sum_i \int_0^\infty x_i dt)/n$ (denoted by $\times$).

In Fig. 5.5, the time needed to solve the problem is also represented as a bar chart. It is evident that the problem implemented by MILP takes way more time compared to the ones implemented by LP. The average solving time for each iteration of some versions of the greedy strategy is also reported, which is clearly negligible.

The scheduling of the exact optimal *NI*, *IT* and *VR* solutions are reported in Figs. 5.6 to 5.8 (at Pages 59 to 60), respectively. Recall that both the LC and SU constraints can be equivalently used to implement contiguity constraints (either for $u_i(t)$ or $z_i(t)$): for each of the three supply strategies, while the optimal cost is reached in

Table 5.3: Numerical values of the (a-posteriori) simulation results $X_{av}$ and $\omega_{av}$ presented in Fig. 5.4. All quantities are expressed in minutes. The optimal $X_{av}$ or $\omega_{av}$ for each strategy of supplying are highlighted in bold. (*source:* [1], © 2022 IEEE)

| | **Problem** | | **NI case** | | **IT case** | | **VR case** | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | $(u_i \in \{0,1\}, c.c.)$ | | $(u_i \in \{0,1\})$ | | $(u_i \in [0,1])$ | | $(u_i \geq 0)$ | |
| | | | $X_{av}$ | $\omega_{av}$ | $X_{av}$ | $\omega_{av}$ | $X_{av}$ | $\omega_{av}$ | $X_{av}$ | $\omega_{av}$ |
| **BATCH** | Exact optimal with objective $J = \min \omega_{av}$ | $LC(z_i)$ | - | - | 20.025 | **22.500** | 16.605 | **19.219** | - | - |
| | | $SU(z_i)$ | - | - | 19.437 | **22.500** | 16.409 | **19.219** | - | - |
| | Relaxed optimal with objective $J = X_{av}$) | $LC(u_i)$ | **26.406** | **26.406** | - | - | 21.850 | 25.208 | - | - |
| | | $SU(u_i)$ | **26.406** | **26.406** | - | - | 17.957 | 36.719 | - | - |
| | | no c.c. | - | - | **18.077** | 24.219 | **15.263** | 21.094 | **11.603** | 2.500 |
| **CENTRALIZED REAL-TIME** | Greedy heuristic with objective $J_t$ | $J_t = \sum_i z_i - u_i$ | 37.500 | 37.500 | 25.614 | 29.219 | 20.953 | 24.650 | - | - |
| | | $J_t = \sum_i z_i - u_i$, pkt | - | - | 31.887 | 36.875 | - | - | - | - |
| | | $J_t = \sum_i \dot{x}_i$, | 31.875 | 31.875 | 20.160 | 22.969 | 16.438 | 19.688 | **11.603** | 2.500 |
| | | $J_t = \sum_i \dot{x}_i$, pkt | - | - | 20.637 | 24.375 | - | - | - | - |
| | Predictive control heuristic with objective $J$ | $J = \omega_{av}$, $LC(z_i)$ | - | - | 22.208 | 31.302 | 17.904 | 23.021 | - | - |
| | | $J = \omega_{av}$, $SU(z_i)$ | - | - | 21.374 | 26.771 | 17.253 | 21.250 | - | - |
| | | $J = X_{av}$, $LC(u_i)$ | 35.885 | 35.885 | - | - | - | - | - | - |
| | | $J = X_{av}$, $SU(u_i)$ | 32.344 | 32.344 | - | - | | - | - | - |
| | | $J = X_{av}$, no c.c. | - | - | 20.055 | 23.229 | 16.531 | 19.740 | - | - |
| | Priority queue based heuristic | FIFO | 43.750 | 43.750 | 46.604 | 46.875 | 30.337 | 40.781 | - | - |
| | | SREF | 31.875 | 31.875 | 21.446 | 23.125 | 17.748 | 22.188 | - | - |
| | | FIFO, pkt | - | - | 29.763 | 45.828 | - | - | - | - |
| | | SREF, pkt | - | - | 21.085 | 24.219 | - | - | - | - |
| **DECENT. REAL-TIME** | P-persistent based heuristic | $p_i^A = 1$ | 42.756 | 42.756 | - | - | - | | - | - |
| | | $p_i^A(r_i, \widetilde{\tau}_i)$ | 37.316 | 37.316 | 32.007 | 36.541 | - | | - | - |
| | | $p_i^A(r_i, \widetilde{\tau}_i)$, pkt | - | - | 32.488 | 36.747 | - | - | - | - |
| | Reservation variable based heuristic | $\mu_i(t) = \mu_0$ | 42.456 | 42.456 | 32.496 | 50.316 | 27.605 | 47.131 | - | - |
| | | $\mu_i(r_i, \widetilde{\tau}_i)$ | 40.238 | 40.238 | 37.945 | 38.859 | 24.853 | 32.566 | - | - |
| | | $\mu_i(t) = \mu_0$, pkt | - | - | 43.056 | 43.056 | - | - | - | - |
| | | $\mu_i(r_i, \widetilde{\tau}_i)$, pkt | - | - | 32.316 | 34.819 | - | - | - | - |

*c.c.*: contiguity constraint Eq. (3.4i) on $u_i(t)$;
$LC(u_i)$: the *Linear Combination* alternative form is used to implement contiguity constraints Eq. (3.4i);
$SU(u_i)$: the *Start-Ups* alternative form is used to implement contiguity constraints Eq. (3.4i);
$LC(z_i)$: the *Linear Combination* alternative form is used to implement contiguity constraints Eq. (2.11);
$SU(z_i)$: the *Start-Ups* alternative form is used to implement contiguity constraints Eq. (2.11);
"pkt": the *IT-packets* strategy of supplying is adopted;
"-": the cases that were not considered.

any case, the distribution can be slightly different, because the optimal solution is generally not unique.

The scheduling of the relaxed problem with continuous $0 \leq u_i(t) \leq 1$ is reported in Fig. 5.9a (at Page 61). The result of applying a greedy approach to it is shown in Fig. 5.9b: this is generally not optimal. The corresponding results when the constraints on $u_i(t)$ are relaxed to $u_i(t) \geq 0$ are reported in Figs. 5.9c and 5.9d: recall that in this case, both distributions are optimal.

Finally, in Figs. 5.10 to 5.13 (at Pages 61 to 67) examples of distribution of the scheduled supply when applying the considered online heuristics are reported.

A discussion of these results is presented in the next Section 5.3.

## 5.3 Discussion

Firstly, from Figs. 5.6 to 5.13, each tested strategy makes all the requests fulfilled before the end of the day, without exceeding the overall capacity limit $P$ of the system. Moreover, every strategy admits immediately the requests when the congestion is low, e.g., the first and the last requests of the day.

In general, it can be seen that the less the supply $u_i(t)$ is constrained, the better the available power tends to be exploited, although this is not a general rule. For instance, compare the total supplied power of the exact *NI* case in Fig. 5.6 and the total supplied power of the exact *VR* case in Fig. 5.8: with the *NI* supply, around 9:00 some of the available power is not supplied to anyone and is therefore wasted. On the contrary, with the *VR* supply the full capacity is occupied (as long as the total requested power exceeds $P$: the maximum output rate of supply is still limited), and this results in lower delays experienced by the users. This is even more evident in the limit case in which $u_i(t) \geq 0$ is unbounded.

Consider now the average waiting times $\omega_{av}$ visualized in Fig. 5.4. It can be easily verified that

- the predictions of Theorem 2.2 are always satisfied for each strategy (except for the unbounded case $u_i \geq 0$, which is not covered by the Theorem). Namely, given a solution $u_1, \dots, u_n$, the corresponding delays fulfill $\Delta_{av} \leq X_{av} \leq \omega_{av}$;
- for the specific *NI* case, as expected from Theorem 2.2, given a solution $u_1, \dots, u_n$, the corresponding delays fulfill $\Delta_{av} = X_{av} = \omega_{av}$;
- the relations between the batch costs summarized in Section 3.4 are satisfied;
- in particular, the optimal cost of the relaxed problem Eq. (3.18) with $0 \leq u_i \leq 1$ is smaller or equal to the average waiting times of all the other strategies (again, except for the one allowing $u_i(t) \geq 0$), as it is a lower bound.

Moreover, in support of the choice of adopting the functional Eq. (3.3) in the proposed optimal control technique, note that $X_{av}$ is generally close to the corresponding $\omega_{av}$ (with just few exceptions).

Regarding the unbounded case, Fig. 5.9 shows that actually the supply of each request $i$ is never fulfilled before $t_i + \tau_i$, even if more power could be assigned to it to anticipate the completion: this occurs because of the constraint $x_i(t) \geq 0$. For instance, consider the request 7: its delay could certainly be reduced by allowing a higher rate at around 15:00, making the completion time $t_i + \delta_i$ even smaller than $t_i + \tau_i$.

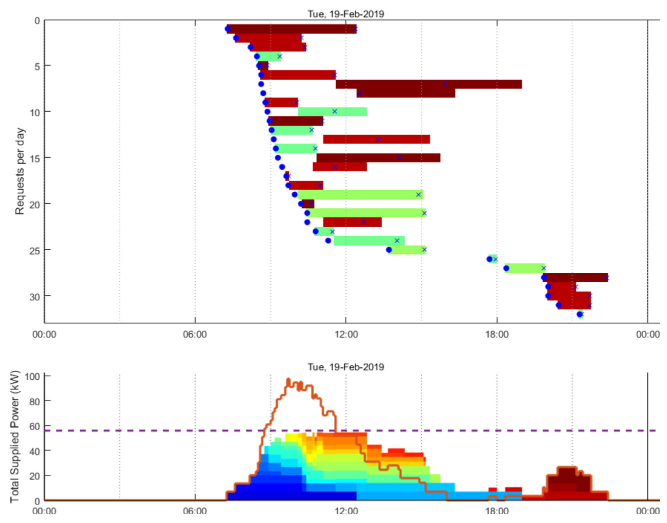
The batch solutions provide the optimal average waiting times $\omega_{av}$ of the *NI*, *IT* and *VR* cases, which are 26.406, 22.500 and 19.219 minutes, respectively. Some lower bounds (in terms of $X_{av}$) for such optimal costs are obtained from the relaxed batch versions, which are reduced between 20% and 50%: the relaxed problem Eq. (3.18) provides a lower bound (for the *VR* case) of 15.263 minutes, while the unbounded problem imposing $u_i(t) \geq 0$ (as well as the corresponding greedy strategy) provides a lower bound of 11.603 minutes.

Regarding the exact problems, note that both the implementations of the contiguity constraints LC and SU (either for $u_i(t)$ or $z_i(t)$) produce the same results when applied to the same problem and strategy, except when the relaxed versions are considered. Considering the exact problems, Figs. 5.6 to 5.8 show that the scheduling is quite similar for the same supply strategy, too. As already stated, the solution is in general not unique, so there can be few minimal differences. The performance in solving the problem, however, can be quite different, see Fig. 5.5: from these simulations, it seems that the SU constraints are best applied to variables $z_i(t)$, while the LC ones to variables $u_i(t)$, due to the resulting lower computation times (a further investigation is however required).

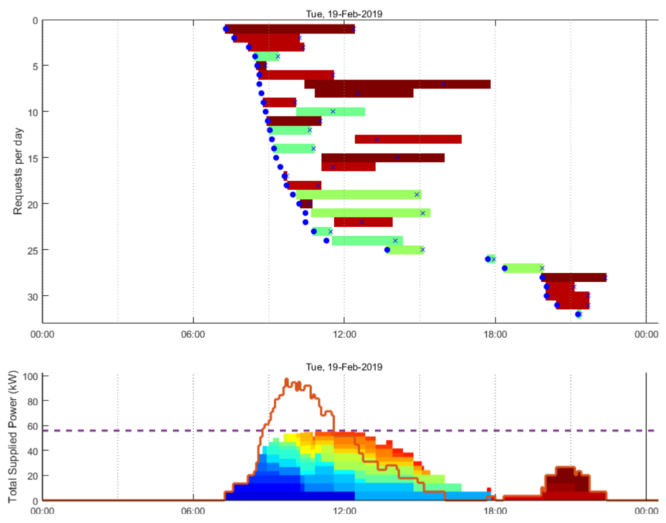
To confirm the findings reported above, generally, as expected, the *VR* versions of the batch and centralized online heuristics perform better than the corresponding *IT* ones, which in turn outperform the *NI* ones. Even the a-posteriori performance loss compared to the optimum follows a similar trend. However, this does not hold for the decentralized and the FIFO strategies: for the former, better tuning of the parameters would possibly be needed; for the latter, this is expected, as scheduling early requests with large impact potentially prevents multiple users to be admitted, increasing their waiting times.

Predictably, the real-time centralized strategies outperform the decentralized ones, which use only local data to optimize, except for the FIFO priority-queue-based one, for the reasons expressed above.

Generally, the greedy and model-predictive approaches do not reach the optimal cost (when evaluated a-posteriori), as expected. The only exception is the greedy strategy applied to the unbounded relaxed case with $u_i(t) \geq 0$, which is proven optimal. However, in some of the performed simulations, sometimes the a-posteriori optimality has been reached anyway. A surprising result is that the simpler greedy strategy, which only considers the current time for optimization, and not a finite horizon, performs similarly, and sometimes even better, in terms of $\omega_{av}$, than the model-predictive one, and the relaxed batch problem, too.



(a) Using the $LC$ constraints for $u_i(t)$.



(b) Using the $SU$ constraints for $u_i(t)$.

Figure 5.6: Distribution of the scheduled supply for the exact optimal non-interruptible ($NI$) supply strategy problem, minimizing $X_{av} = \sum_i x_i/n (\equiv \omega_{av})$. Top: each row refers to the supply of a different request $i$, and the colored area is located between $t_i + \Delta_i$ and $t_i + \delta_i$. Its color refers to the value of the supply $y_i(t) = r_i u_i(t)$, see the color bar in Fig. 5.3. The blue circle $\circ$ and cross $\times$ indicate $t_i$ and $t_i + \tau_i$, respectively. Bottom: total supplied power (colored area), total requested power (red line, see Fig. 5.3), and system capacity (dashed violet line). Here, the color is used to distinguish the power assigned to each request: blue refers to the older requests, and red to the ones occurred at the end, as in Fig. 5.3.

The enhanced priority-queue-based strategy prioritizing the requests with smaller remaining energy to supply provides (a-posteriori) results close to optimum for the $IT$, $IT$-*packets*, and $VR$ cases (again, sometimes the optimum has been also reached in some other performed simulations), while it is apparent that a FIFO strategy does not provide good performance, with results also worse than some decentralized methods with limited knowledge. The enhanced version of the p-persistent and reservation-variable heuristic prioritizing the requests with smaller impact $q_i(r_i, \widetilde{\tau}_i(t))$, instead of the one made earlier, provides the best (a-posteriori) results among the decentralized online strategies, getting a $\omega_{av}$ at most twice the optimum. Such average waiting times can be considered good, as they are achieved despite not knowing any data in advance, nor any data about the other requests.

Consider now the computational times of the batch problems, visualized in Fig. 5.5 in log scale. From the first four bars, all the exact problems minimizing $J = \omega_{av}$ in the $IT$ and $VR$ supply strategies, using either the $LC(z_i)$ or $SU(z_i)$ constraints, require a large amount of time to be solved (from several minutes up to about an hour) despite the problem considering only $n = 32$ requests. Indeed, a MILP problem is to be solved in those cases, with the possible integer variables being $u_i(k)$ (for the $IT$ case), $b_{ih}$ (for the $LC(z_i)$ constraints), and $z_i(k)$. As expected, the $IT$ problems take longer to be solved, as well as the ones corresponding to the case where the $LC(z_i)$ constraints are used, as there are more integer variables.

Consider now the exact $NI$ case, minimizing $J = X_{av}$ (recall that, in this case, $X_{av} \equiv \omega_{av}$), using either the $LC(x_i)$ or $SU(x_i)$ constraints (see the fifth and seventh bars of Fig. 5.5): the solving times are large, too, as, again, a MILP problem is to be solved, with the possible integer variables being $u_i(k)$ and $b_{ih}$ (for the $LC(x_i)$ constraints). A formulation minimizing $J = \omega_{av}$ is also possible, however the complexity would be even larger, as either the $LC(z_i)$ or $SU(z_i)$ constraints would be needed, other than the $LC(x_i)$ or $SU(x_i)$ constraints.

The solving times are reduced when relaxing all of the above-mentioned integer variables (see the sixth and eighth bars of Fig. 5.5), resulting in just few seconds, as the problems reduce to LP.

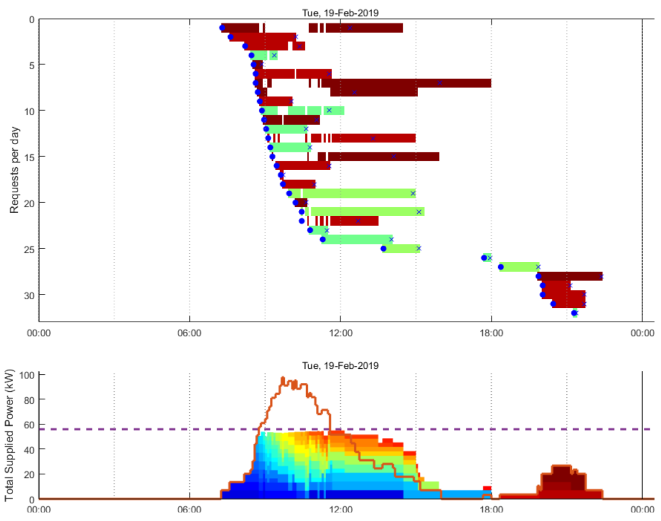
A similar solving time is achieved when considering the relaxed problem minimizing $J = X_{av}$ for the $IT$ case, when no contiguity constrained is imposed (see the ninth bar of Fig. 5.5): there are less constraints variables, despite $u_i(k)$ are still integer.

In comparison, the relaxed (non-exact) batch problem with $0 \leq u_i(t) \leq 1$ and $u \geq 0$ (see the tenth and eleventh bar of Fig. 5.5) only require about 1 second; the computation time is now lower as the problem is an LP one, with no contiguity constraints imposed.

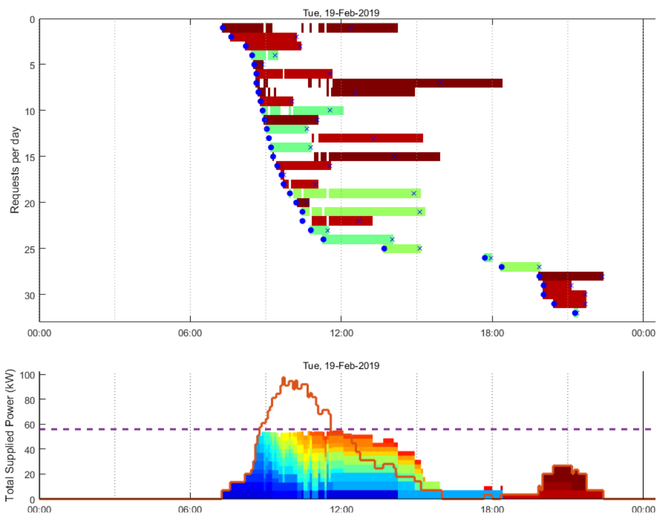
The average solving time for each iteration of the greedy strategy (see the last four bars in Fig. 5.5) is less than 0.1 seconds, which is a negligible amount compared to the times reported above. A trade-off between the computation time and the optimality of the result exists. Since the performance loss of the greedy heuristic with respect to the exact optimal solution for the three supply strategies is not that large, especially for the $NI$ and $VR$ cases, and in any case it is within 20%, the advantages of adopting it are evident when just a reference value for the optimal waiting time is required, since the computation times are reduced by at least the 1000%, which is remarkable, since in the batch problem all data are available for optimization, while in the greedy one no information about future requests is provided.

A similar argument can be made for the relaxed problem: in this case, however, the optimal cost is a lower bound for the optimal minimum average waiting time. Then, for a preliminary analysis, the relaxed (possibly greedy) solutions can give an approximate idea of the minimal delay to expect, especially for larger problems, where the exact solutions might be more difficult to find in a reasonable time.

The computation times of the priority queue-based, p-persistent-based, and reservation variable-based real-time heuristics are less than 1 second at each iteration, thus negligible compared to the duration of the time-slots (5 minutes). Finally, for the predictive control heuristics, the timeout of 30 seconds was only occasionally reached.

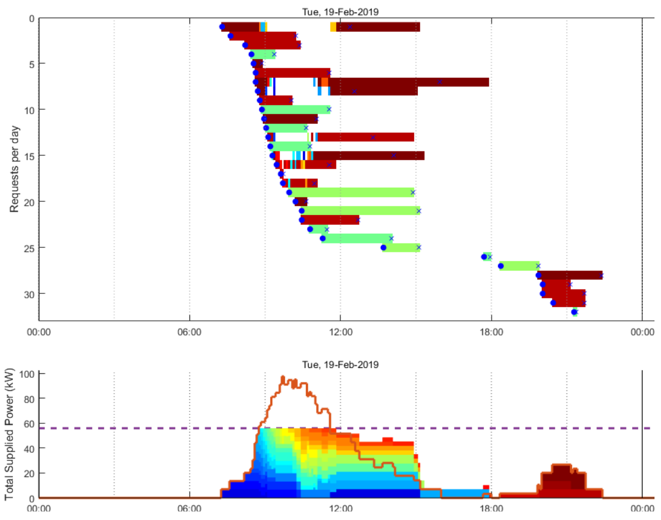


(a) Using the $LC$ constraints for $z_i(t)$.

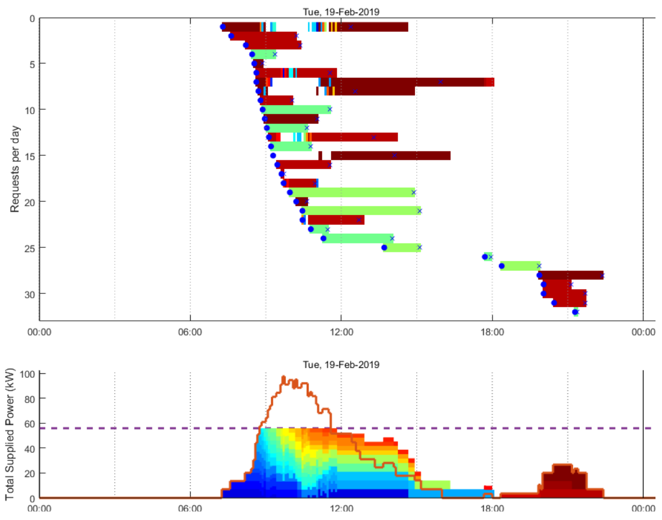


(b) Using the $SU$ constraints for $z_i(t)$.

Figure 5.7: Distribution of the scheduled supply for the exact optimal interruptible ($IT$) supply strategy problem, minimizing $\omega_{av} = \sum_i \omega_i/n$. See Fig. 5.6 for a description.

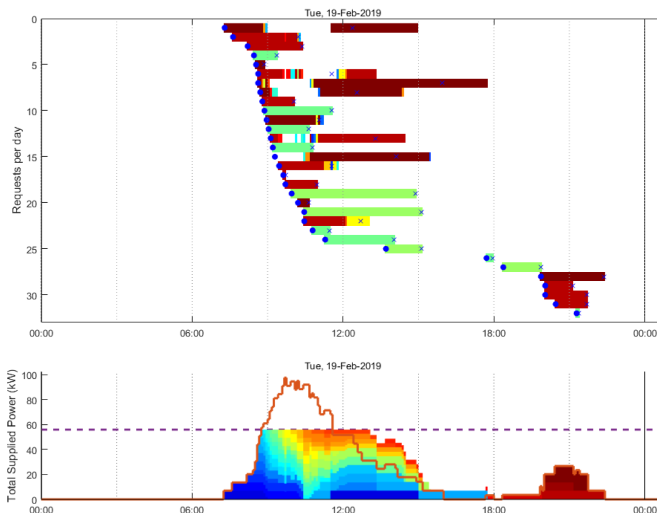


(a) Using the $LC$ constraints for $z_i(t)$.

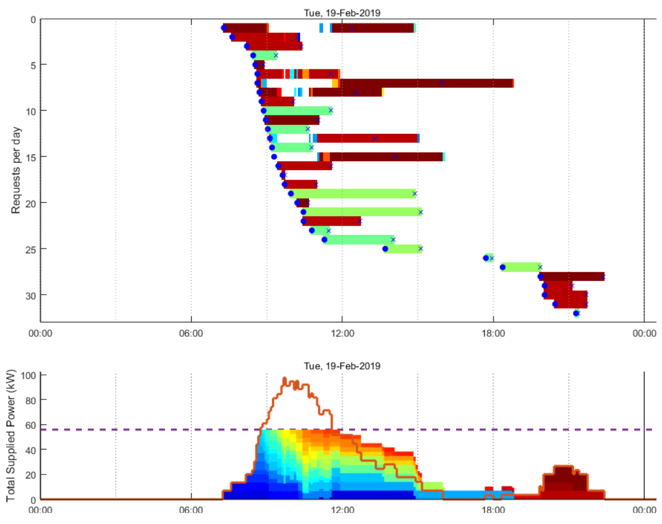


(b) Using the $SU$ constraints for $z_i(t)$.

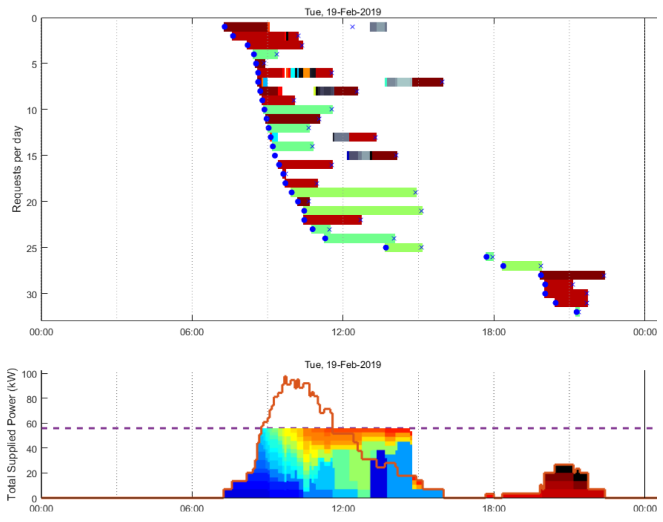
Figure 5.8: Distribution of the scheduled supply for the exact optimal variable rate ($VR$) supply strategy problem, minimizing $\omega_{av} = \sum_i \omega_i/n$. See Fig. 5.6 for a description.



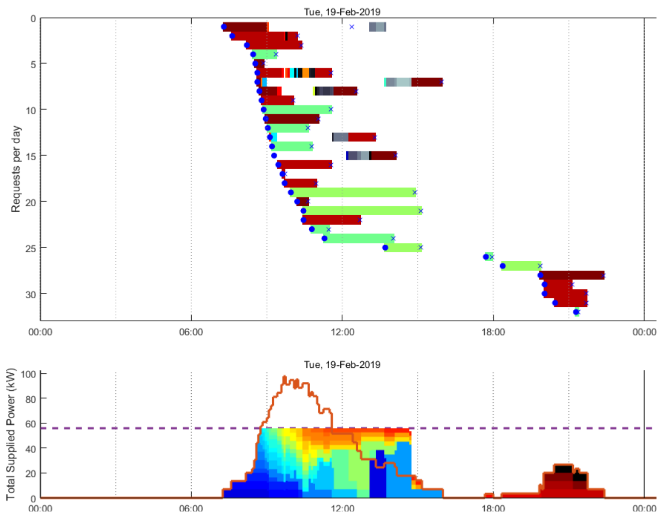
(a) Batch solution, with $0 \leq u_i(t) \leq 1$.



(b) Greedy solution, with $0 \leq u_i(t) \leq 1$.

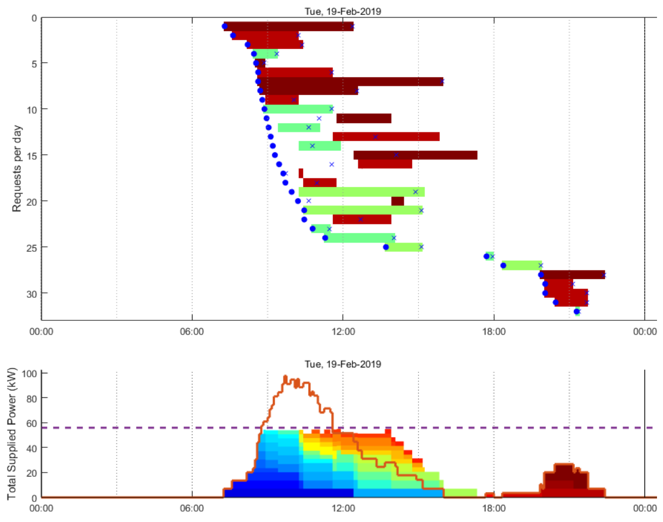


(c) Batch solution, with $u_i(t) \geq 1$.

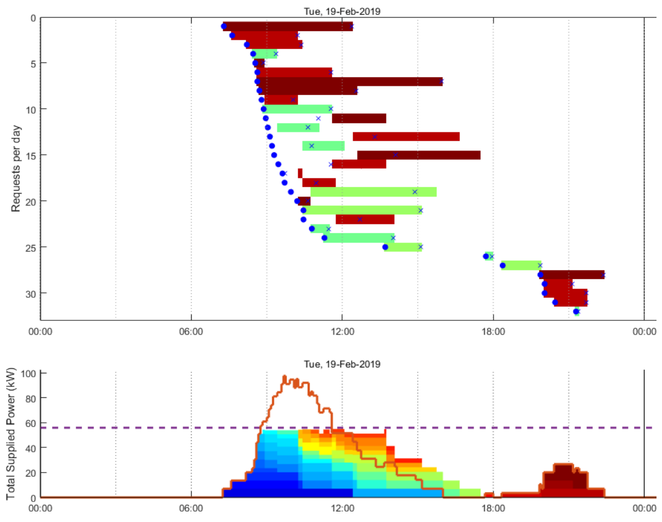


(d) Greedy solution, with $u_i(t) \geq 1$.

Figure 5.9: Distribution of the scheduled supply when the relaxed problem minimizing $J = X_{av} = \sum_i \int_0^\infty [x_i(t)/n]$ is considered (without contiguity constraints on $u_i(t)$). See Fig. 5.6 for a description. In subfigures Figs. 5.9c and 5.9d, greyscale colors indicate that the assigned power is above $r_i$: from just above $r_i = 7\ kW$ (black) up to $45\ kW$ (light gray).

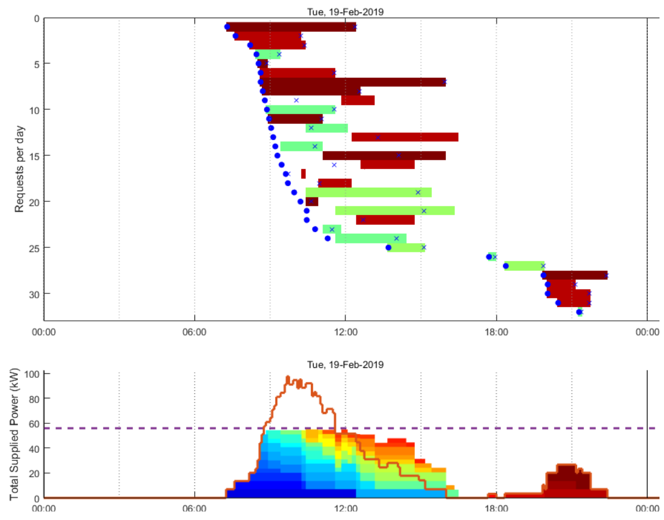


(a) Greedy heuristic with objective $J_t = \sum_i (z_i - u_i)$.

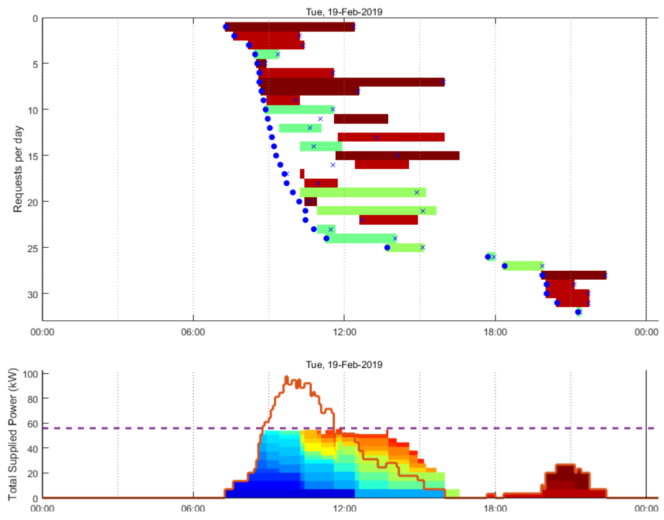


(b) Greedy heuristic with objective $J_t = \sum_i (\dot{x}_i)$.

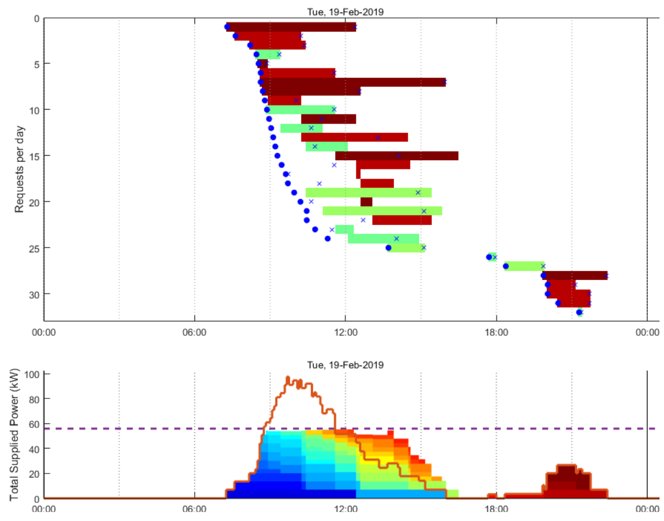
Figure 5.10: Distribution of the scheduled supply when the *NI* online heuristics are adopted. See Fig. 5.6 for a description. *(continued on the next page).*



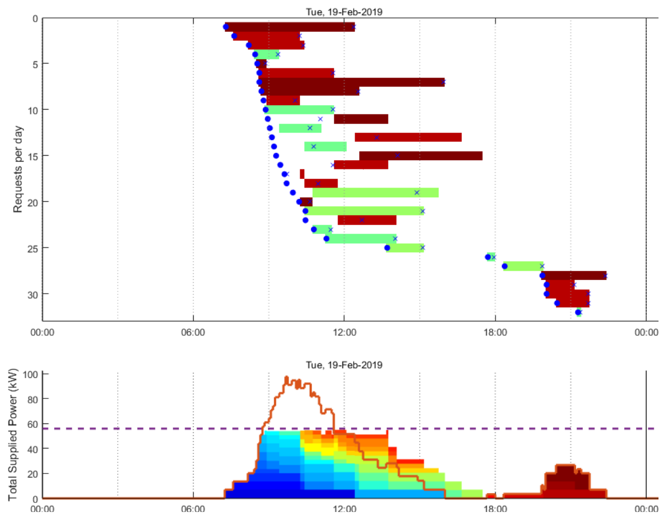
(c) Predictive control heuristic with objective $J = X_{av}$, using LC constraints on $u_i(t)$.



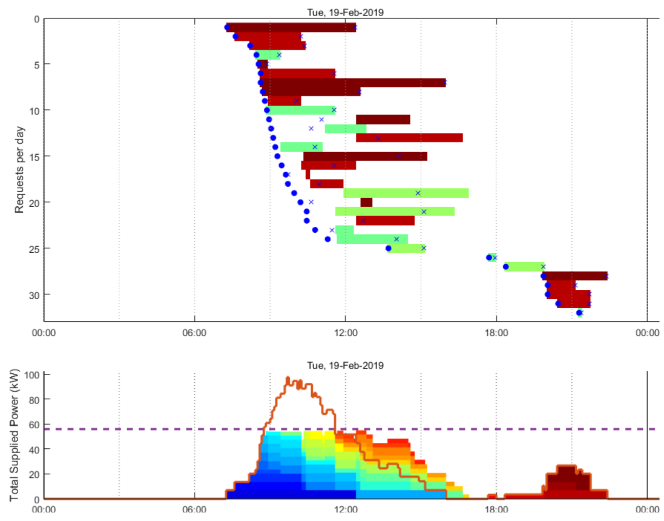
(d) Predictive control heuristic with objective $J = X_{av}$, using SU constraints on $u_i(t)$.



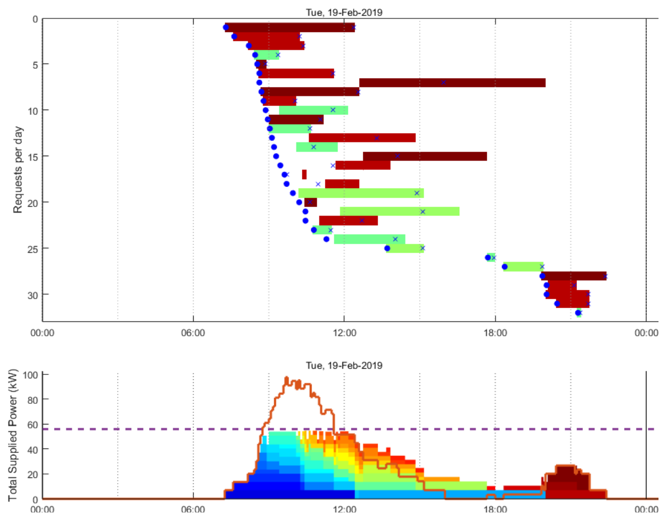
(e) Priority queue-based heuristic, using FIFO ordering.



(f) Priority queue based heuristic, using SREF ordering.

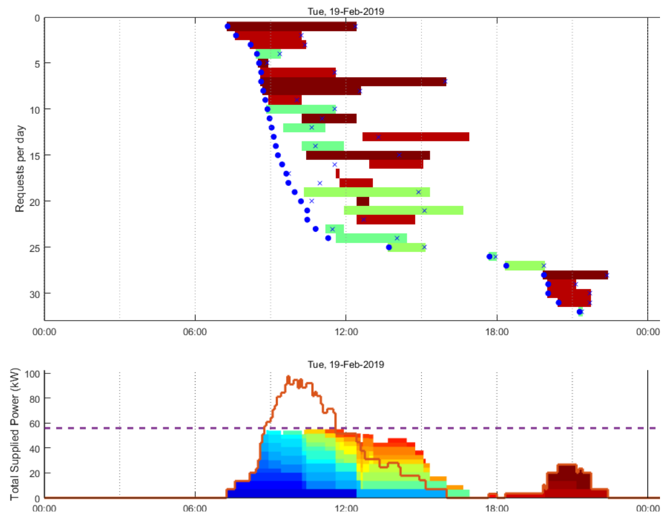


(g) $p$-persistent based heuristic with $p_i^A = 1$.

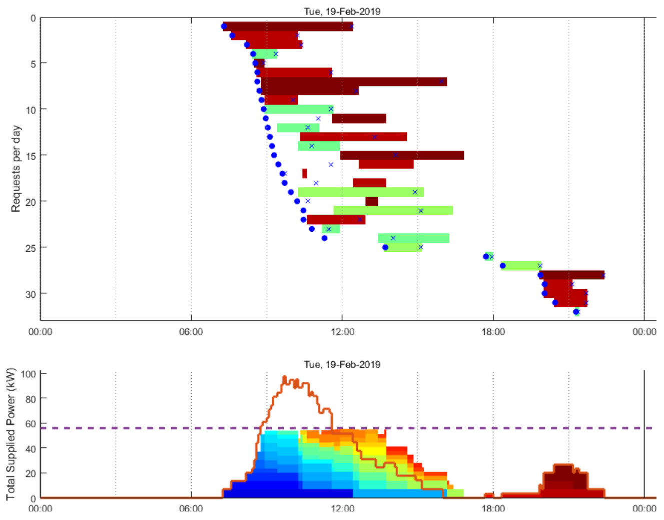


(h) $p$-persistent based heuristic with $p_i^A(r_i, \widetilde{\tau}_i)$.

Figure 5.10: *(continued)* Distribution of the scheduled supply when the *NI* online heuristics are adopted. See Fig. 5.6 for a description. *(continued on the next page)*.

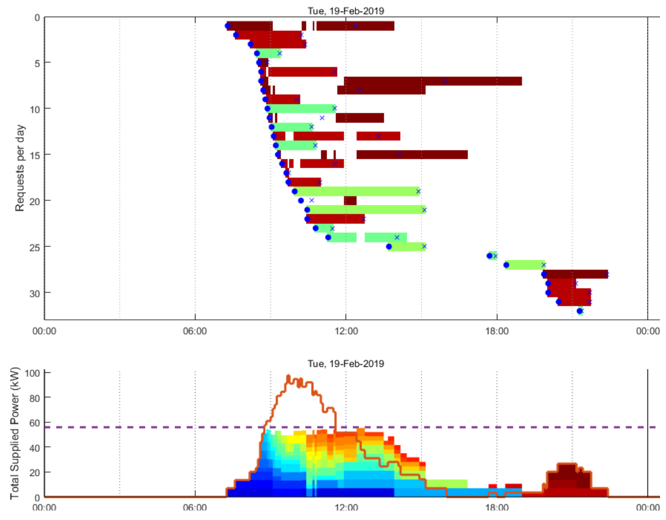


(i) Reservation variable based heuristic with $\mu_i(t) = \mu_0$.

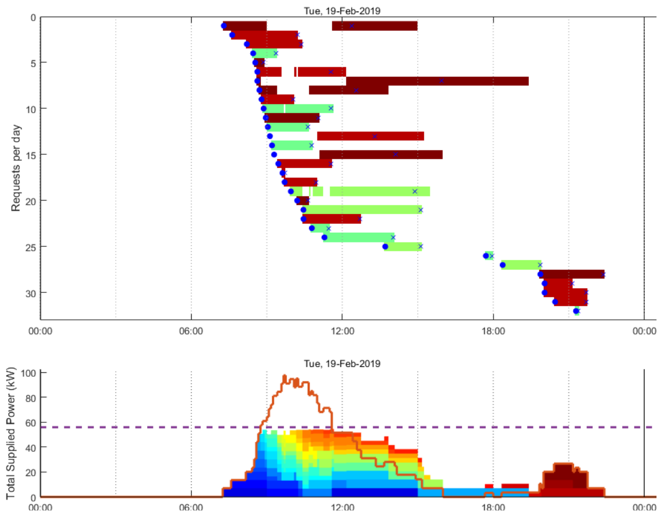


(j) Reservation variable based heuristic with $\mu_i(r_i, \widetilde{\tau}_i(t))$.

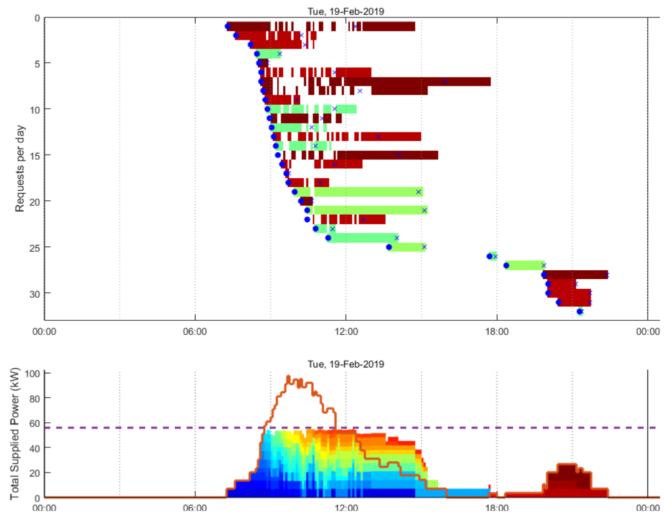
Figure 5.10: *(continued)* Distribution of the scheduled supply when the *NI* online heuristics are adopted. See Fig. 5.6 for a description.



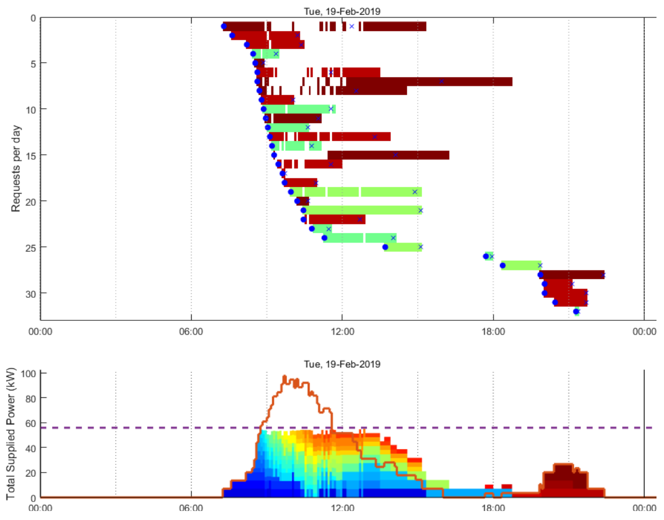
(a) Greedy heuristic with objective $J_t = \sum_i (z_i - u_i)$.



(b) Greedy heuristic with objective $J_t = \sum_i (\dot{x}_i)$.

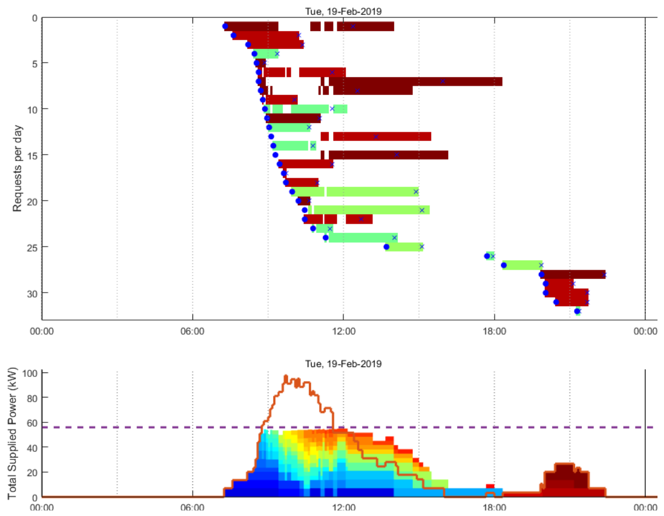


(c) Predictive control heuristic with objective $J = \omega_{av}$, using LC constraints on $z_i(t)$.

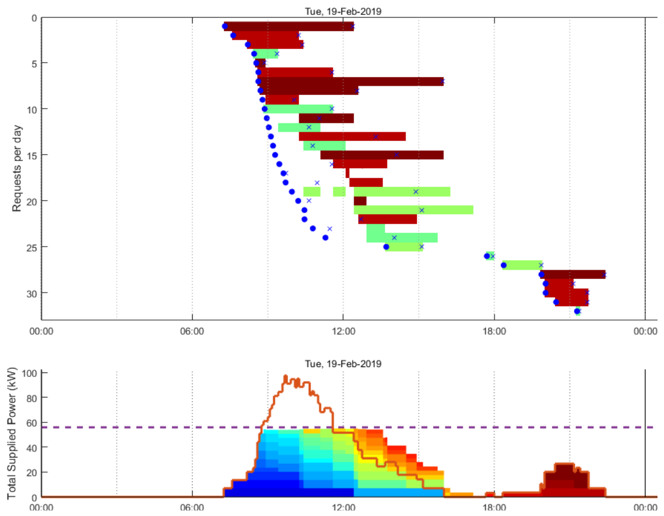


(d) Predictive control heuristic with objective $J = \omega_{av}$, using SU constraints on $z_i(t)$.

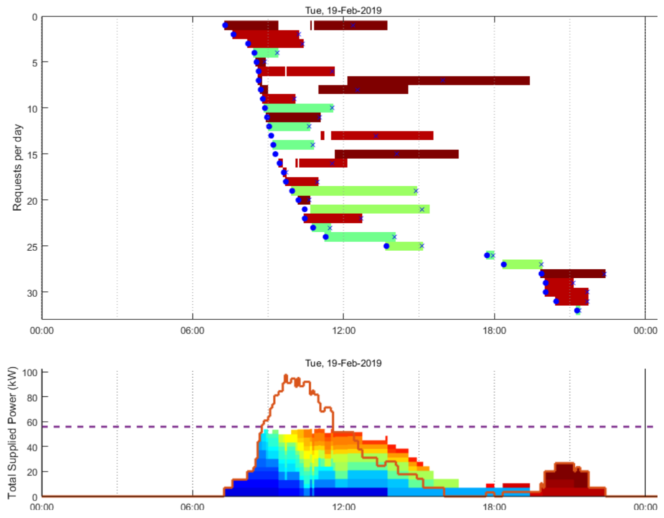
Figure 5.11: Distribution of the scheduled supply when the *IT* online heuristics are adopted. See Fig. 5.6 for a description. *(continued on the next page)*.



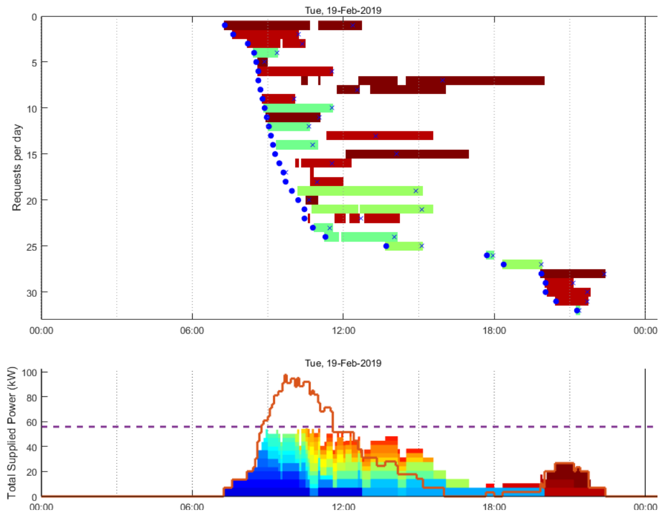
(e) Predictive control heuristic with objective $J = X_{av}$, without contiguity constraints on $u_i(t)$.



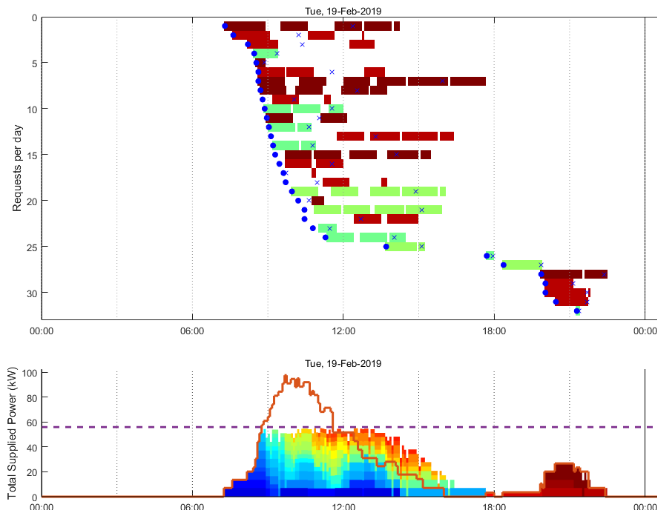
(f) Priority queue-based heuristic, using FIFO ordering.



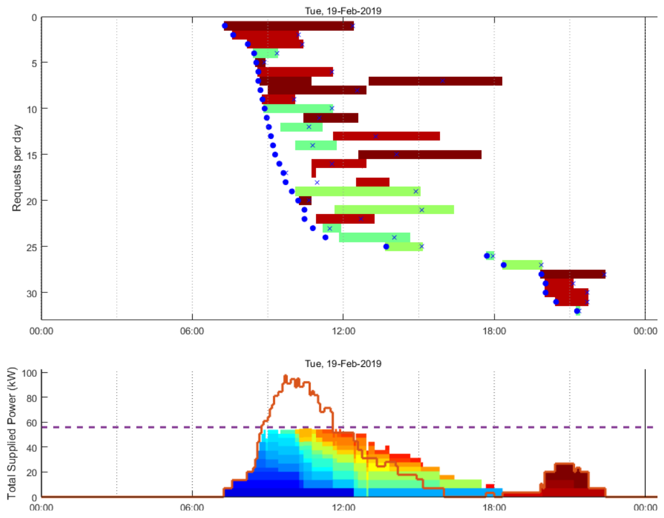
(g) Priority queue based heuristic, using SREF ordering.



(h) $p$-persistent based heuristic with $p_i^A(r_i, \widetilde{\tau}_i)$, and $p_i^I(r_i, \widetilde{\tau}_i)$.

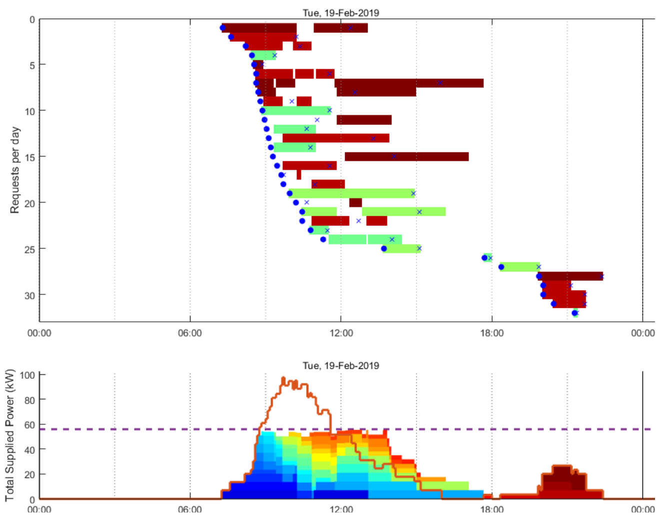


(i) Reservation variable based heuristic with $\mu_i(t) = \mu_0$,$\nu_i(t) = \nu_0$.

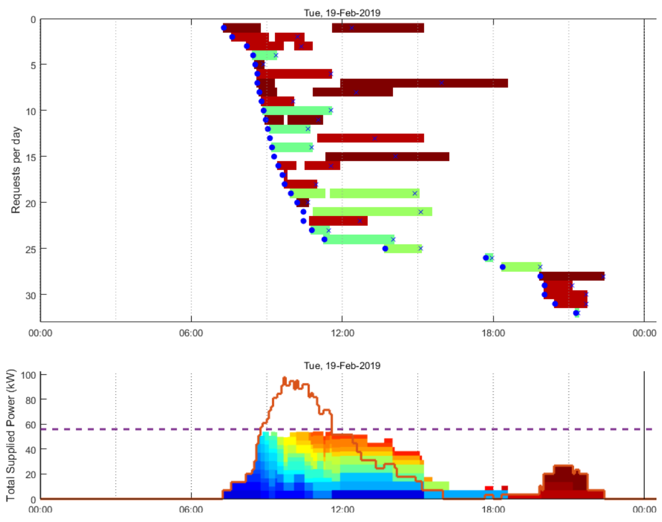


(j) Reservation variable based heuristic with $\mu_i(r_i, \widetilde{\tau}_i(t))$, and $\nu_i(r_i, \widetilde{\tau}_i(t))$.

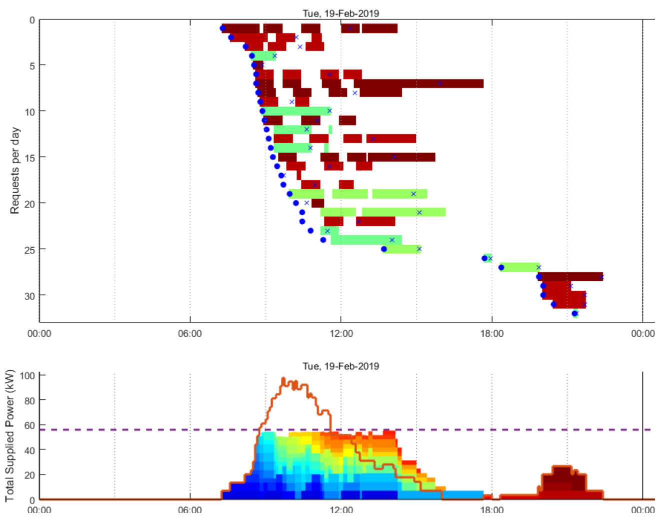
Figure 5.11: *(continued)* Distribution of the scheduled supply when the $IT$ online heuristics are adopted. See Fig. 5.6 for a description.



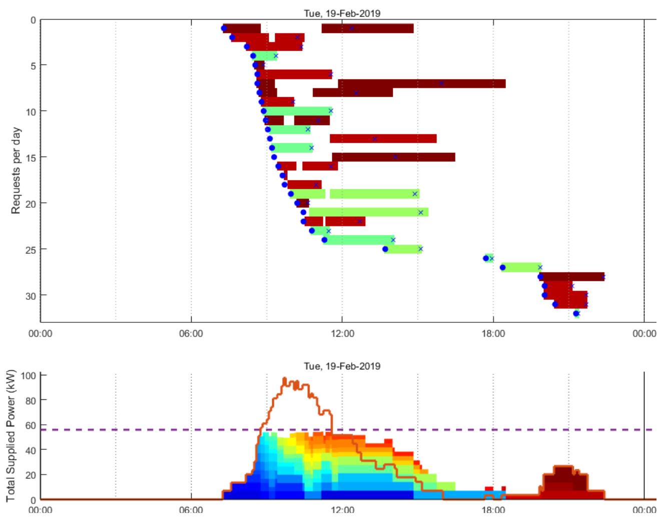
(a) Greedy heuristic with objective $J_t = \sum_i (z_i - u_i)$.



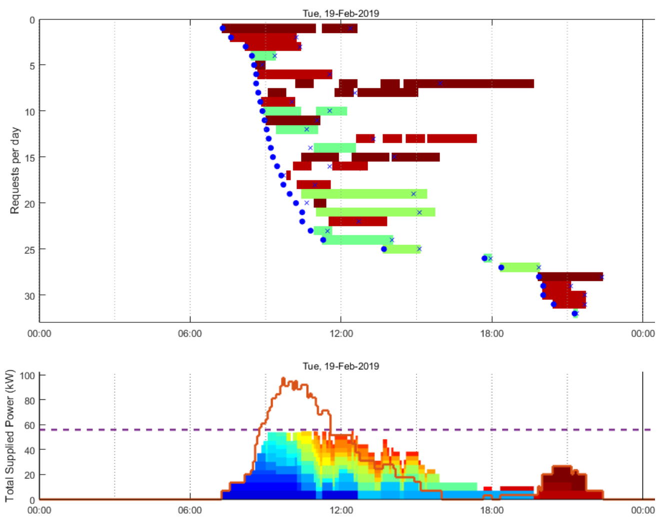
(b) Greedy heuristic with objective $J_t = \sum_i (\dot{x}_i)$.



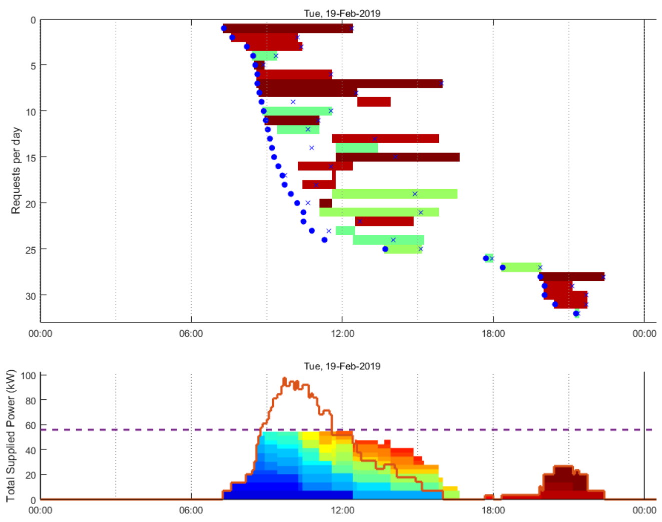
(c) Priority queue-based heuristic, using FIFO ordering.



(d) Priority queue based heuristic, using SREF ordering.

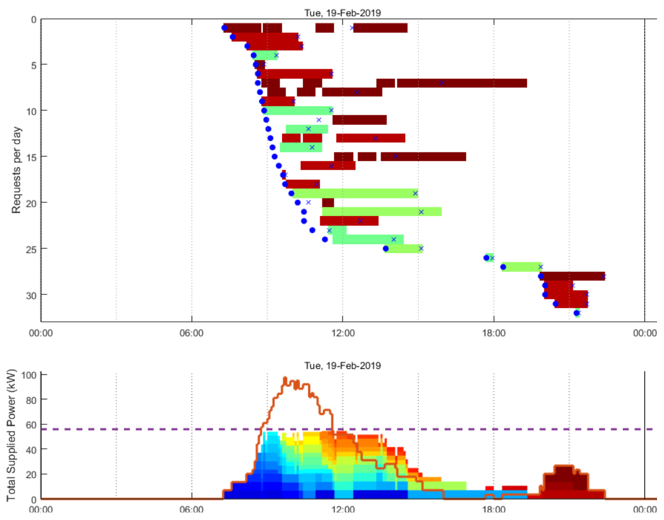


(e) $p$-persistent based heuristic with $p_i^A(r_i, \widetilde{\tau}_i)$, and $p_i^I(r_i, \widetilde{\tau}_i)$.



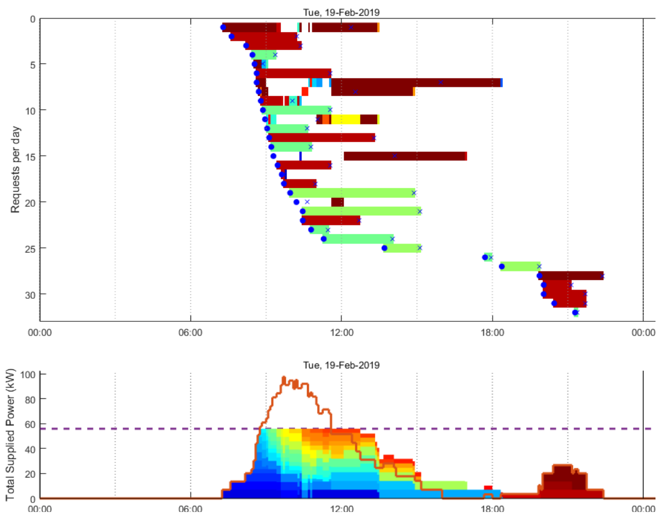
(f) Reservation variable based heuristic with $\mu_i(t) = \mu_0$, and $\nu_i(t) = \nu_0$.

Figure 5.12: Distribution of the scheduled supply when the $IT$-packets online heuristics are adopted. See Fig. 5.6 for a description. *(continued on the next page)*.

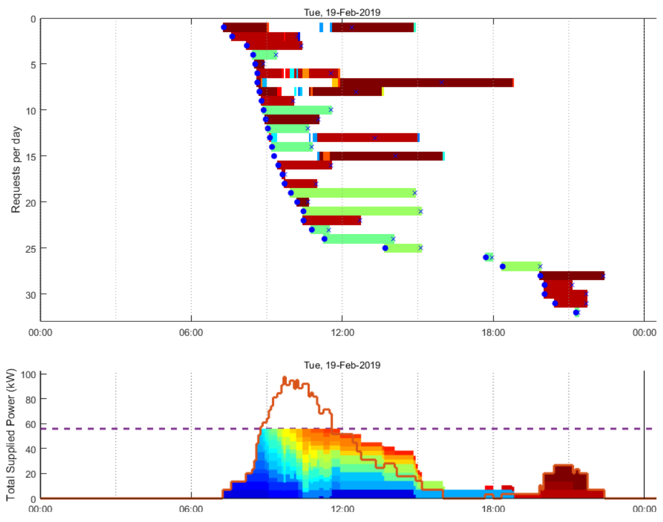


(g) Reservation variable based heuristic with $\mu_i(r_i, \widetilde{\tau}_i(t))$, and $\nu_i(r_i, \widetilde{\tau}_i(t))$.

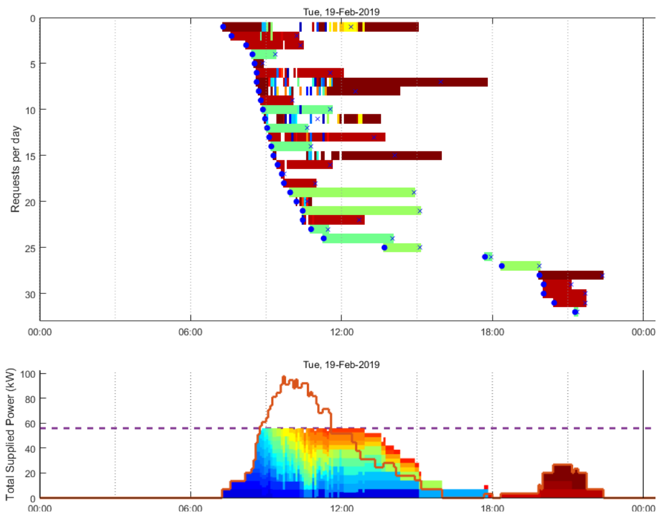
Figure 5.12: *(continued)* Distribution of the scheduled supply when the $IT$-packets online heuristics are adopted. See Fig. 5.6 for a description.



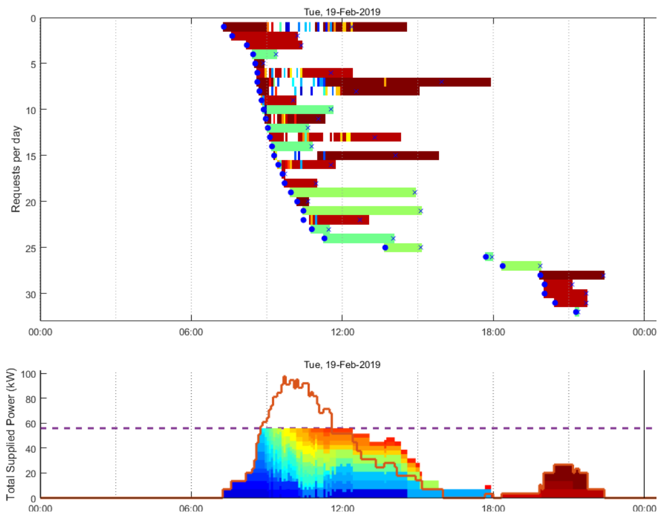
(a) Greedy heuristic with objective $J_t = \sum_i (z_i - u_i)$.



(b) Greedy heuristic with objective $J_t = \sum_i (\dot{x}_i)$.

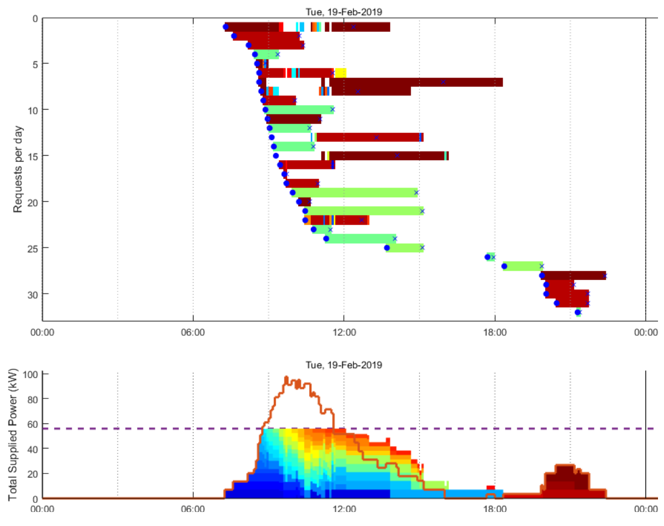


(c) Predictive control heuristic with objective $J = \omega_{av}$, using LC constraints on $z_i(t)$.

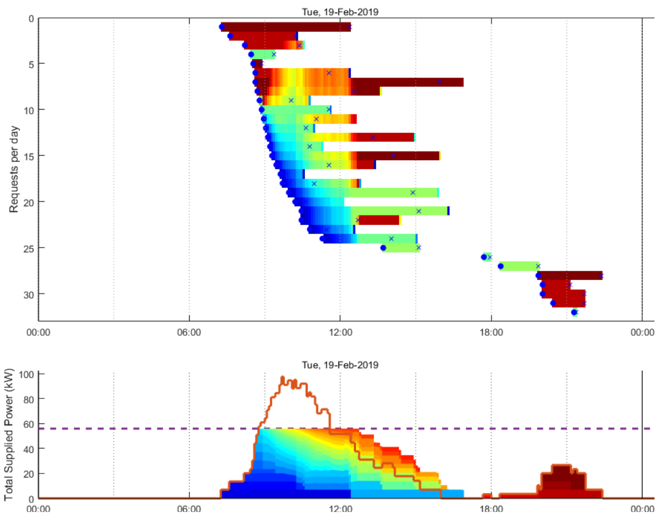


(d) Predictive control heuristic with objective $J = \omega_{av}$, using SU constraints on $z_i(t)$.

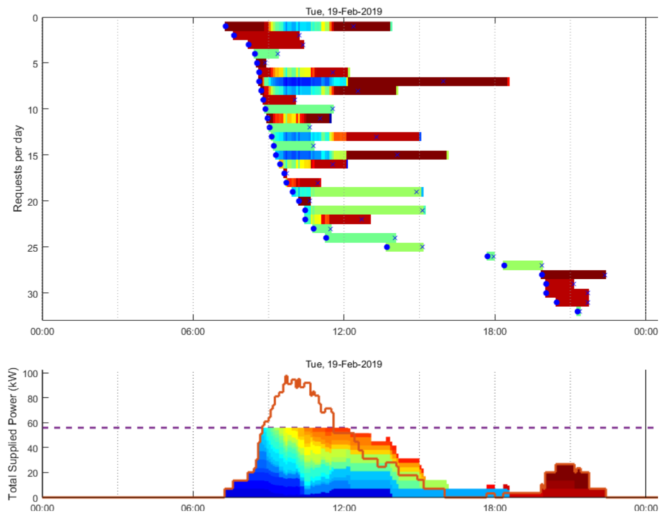
Figure 5.13: Distribution of the scheduled supply when the $VR$ online heuristics are adopted. See Fig. 5.6 for a description. *(continued on the next page).*



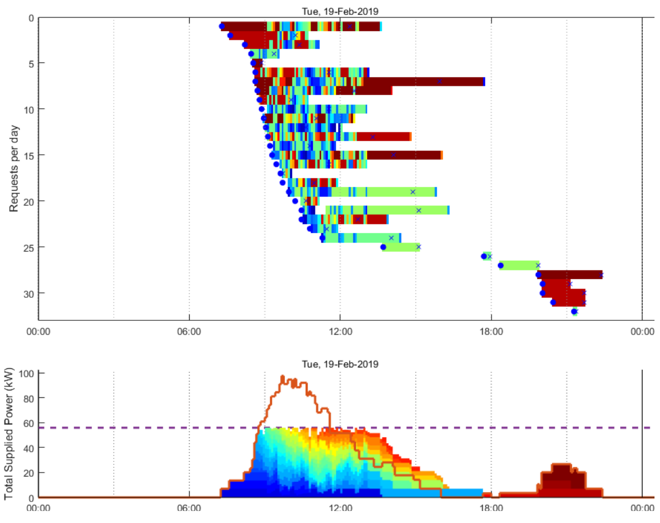
(e) Predictive control heuristic with objective $J = X_{av}$, without contiguity constraints on $u_i(t)$.



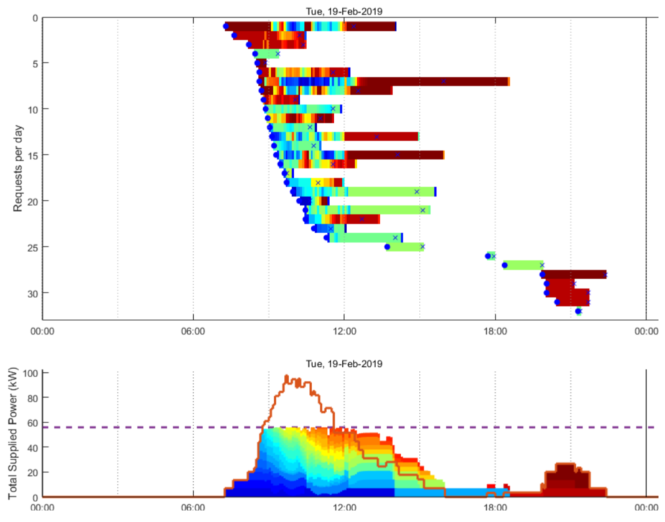
(f) Priority queue-based heuristic, using FIFO ordering.



(g) Priority queue based heuristic, using SREF ordering.



(h) Reservation variable based heuristic with $\mu_i(t) = \mu_0$,$\nu_i(t) = \nu_0$.



(i) Reservation variable based heuristic with $\mu_i(r_i, \widetilde{\tau}_i(t))$, and $\nu_i(r_i, \widetilde{\tau}_i(t))$.

Figure 5.13: *(continued)* Distribution of the scheduled supply when the *VR* online heuristics are adopted. See Fig. 5.6 for a description.

# Part II
# Decentralized agent-based policies for path problems

Based on [2], under review: Francesca Rosset, Franco Blanchini, and Raffaele Pesenti. “An agent–based decentralized threshold policy finding the constrained shortest paths”. In: (2022). Submitted

# Chapter 6

# Introduction to Part II

This Part presents the work introduced in [2], which is the result of the collaboration between Franco Blanchini, Raffaele Pesenti and me.

Consider an environment described by a directed network that presents some *source* nodes and some *sink* nodes that connect it to the external environment. Some traveling agents (tokens) are injected into the source nodes and enter the network from the external environment. Such tokens explore the network, which is completely unknown to them, by traveling from node to node along the existing arcs trying to reach a sink node to leave the network. The decisions of each token on whether to leave a given node or in which node to move next are governed by a policy. When a sink node is actually reached, tokens are expelled from the network and return to the external environment. However, not all tokens may be able to do that: some of them might be forced to stop in a node along their way, and this prevents them to reach a sink. This is unavoidable, since tokens have to explore the network before reaching the exit. It will be discussed how one can recover trapped tokens at the end.

Now suppose that a given finite integer cost is associated with each arc and that the tokens have to pay the corresponding cost at each arc traversal. It is assumed that the traveling tokens not only want to find a route to exit the network, but also that they would like to do that preferably paying the minimum cost. For instance, the arc cost might represent the physical arc length, the arc traversal time, or the consumed/recovered energy to traverse the arc; then, the minimum cost route that the token would like to follow is the shortest route, the fastest route, or the route that makes agents consume the least amount of energy, respectively.

A decentralized threshold policy is introduced, by which each token independently decides whether to stop in the currently occupied node or to move to some adjacent node along the connecting arc. This decision depends only on local information regarding the currently occupied node, the adjacent neighboring node and the connecting arc cost (the *threshold*), exploiting the information about the number of tokens stopped in the two nodes (which represent the *state* of the two nodes). It does not require the knowledge of the network, its topology, or the location of the sinks, nor the information about the nodes traversed so far. Hence, tokens can be assumed *memoryless* and the network can be considered *unknown* to them. Note that the arc costs can be determined locally by the arc endnodes and remain unknown to the tokens until they reach the arc tail node. The main idea behind this rule is anticipated in the next Section 6.1.

It turns out that applying a very simple local rule is eventually effective not only in routing the injected tokens toward the sinks, but also in routing them to the *closer* sinks, through the *minimum cost (shortest) paths*, provided that the network does not present non-positive cost cycles.

Note that the optimality is achieved *in the long run*: initially, there is a transient phase in which tokens might not be able to reach a sink to exit the network and are forced to stop and accumulate in some nodes of the network, instead. However, this makes the states of the nodes vary initially, influencing the routing decisions of the successive tokens injected into the network. It will be shown that at some point the state of the network stabilizes, reaching a *steady-state* in terms of tokens accumulated in the nodes, meaning that indeed all the newly injected tokens are able to reach a sink, in particular through the shortest path. The possibility that the tokens travel indefinitely in the network or that there is an accumulation of infinitely many tokens in a node is excluded, and the number of tokens that have to be injected to reach the steady-state is bounded.

It will be also shown that a simple variation of the proposed policy ensures that, under the assumption of a strongly connected network, all the tokens injected in the sources are collected in the sinks. While in the initial transient phase tokens do not necessarily follow the shortest paths, the optimality is eventually achieved. In this enhanced version, the performance is also improved, meaning that far fewer tokens are required to be injected into the network to reach the steady-state.

In a similar scenario, an additional given finite integer secondary cost might also be associated with each arc, which is independent from the one introduced above and which can induce a constraint on the tokens' movements. In particular, the tokens have also a secondary cost at each arc traversal and, to continue moving, their total cumulative secondary cost paid so far cannot exceed a given maximum value. For instance, the secondary arc cost might represent, again, the arc's physical length, the arc traversal time, or the consumed/recovered energy to traverse the arc. Such secondary cost has a different meaning for the token: the token must exclude all the routes that are too long, take too long to be traversed, or consume too much energy, respectively. As an example, the token with limited traveling autonomy might want to take the fastest route possible that minimizes the total traversal time without exceeding a given energy consumption. As another example, tokens carrying some information might want to reach the closer physical sink without exceeding a given traversal time, after which such information would become outdated.

The decentralized threshold policy introduced for the unconstrained case can be easily adapted to deal with these secondary costs, remaining a decentralized one, and it turns out that eventually tokens are routed through the shortest *feasible* path to the closest sink.

Remark that although the routes found by tokens turn out to be the (possibly constrained) *shortest* paths to the sinks, the proposed policy is not presented as a new method for solving the well-known shortest path problem. Indeed, the main result from this Part is that applying a very simple local threshold rule, using the least amount of information, results in an optimal self-organizing emerging global behavior, that is the discovery of the shortest paths in a network in the long run [72, 73], even in the case in which some constraints are applied to them, which, as it will be seen, makes the shortest path problem hard to solve. As it will be discussed later in Sections 6.1 and 6.2, the model presented here is indeed inspired by the decentralized threshold-based flow control introduced in [72] for single-source-single-sink networks: despite the different nature of the problem, the similar effect of concentrating the flow along the (unconstrained) shortest path is achieved in the long run. In the proposed agent-based model, this holds even if path constraints, multiple sources and sinks, or negative arc costs are considered. Moreover, the reaching of a steady-state with the above-mentioned properties does not depend on the initial state of the nodes, i.e., on the fact that some tokens are already deposited in the nodes. As a result, the proposed policy also turns out to be adaptive: if the network is dynamic and some modifications are applied to it (e.g., there is a failure of a node, or the insertion of a new link), if the new configuration is kept for a sufficiently long time, the optimal paths followed by the tokens are eventually updated.

Remark that having a decentralized policy provides many advantages, as it is fault-tolerant, it easily supports large, unknown and dynamic networks (tokens do not need to be notified about the state of the network, nor if possible modifications occur) and it supports privacy.

The proposed policy could be employed in packet transmission through an unknown network, where packets not only carry information, but are also routing agents. A more specific example of application in the context of smart grids and IoT is the following, in the so-called *sensors networks* [74, 75]. Consider some sensors (the sources) placed in an environment which measure some data. This data is transmitted in a network of transceivers (the nodes) and is to be collected in a set of nodes (the sinks). It is not important where data are collected, but the only request is that they leave the network, reaching a data storage facility (sink) among the available ones. The application of the proposed decentralized policy ensures that in the long run data flow in the network and are collected in the closest sinks through the shortest paths with respect to a given cost, for instance, the physical distance. Constraints on the paths could be enforced to exclude paths that take too much time to be traversed, making the information in the data "outdated". Despite some data might be initially lost, at steady-state all valid data are collected in the sinks. The support for possibly failing of some nodes or links is also guaranteed.

In the rest of this Chapter, first, the intuitive idea behind the proposed policy is presented in Section 6.1, and after that, the literature review is reported in Section 6.2 and the main contributions summarized in Section 6.3. Then, in Chapter 7 the setup is described, and the considered problem is stated. In Chapter 8 and Chapter 9 the unconstrained problem and constrained problem, respectively, are formally studied, and the main results are reported. Finally, in Chapters 10 to 12, the results of some simulations are reported to validate the proposed approach, considering a simple network, a large dynamic network and a class of small-world networks, respectively. The proofs are reported in Appendix B.

## 6.1 The main idea

Here, the main idea behind the proposed policy is introduced.

Consider a network composed of some nodes and some oriented arcs. It is assumed that in each node a token buffer is present. Then, the *state* $x_i$ *of a node* $i$ is defined as the number of tokens deposited in it with respect to an initial zero level of the node, see Fig. 6.1. In [72], there is the same concept of state, although it is continuous.

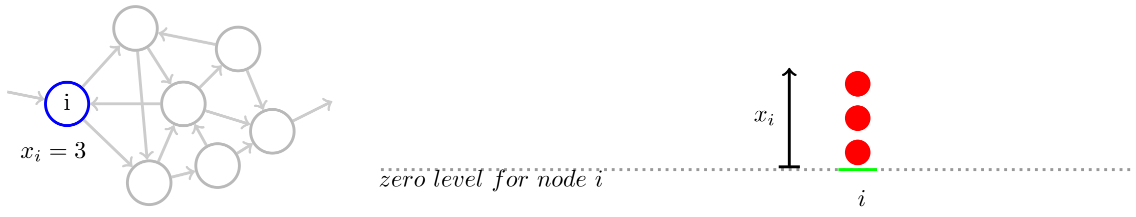


Figure 6.1: Graphical representation of the state $x_i$ of a node $i$: assuming a zero reference level, the state can be seen as the number of tokens deposited in the node.

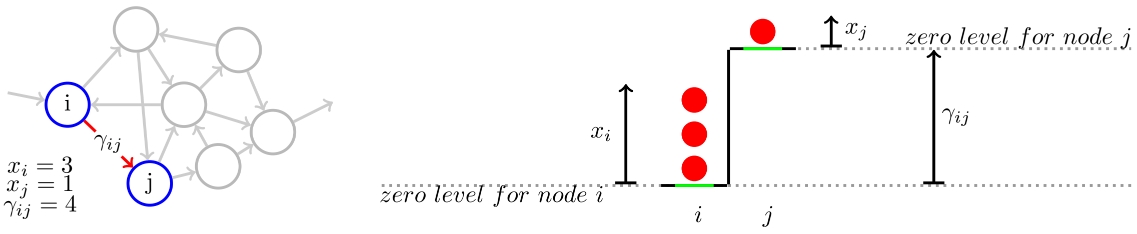


Figure 6.2: Graphical representation of a directed weighted arc $(i, j)$ connecting node $i$ to node $j$, with arc cost $\gamma_{ij}$, as a *step*. The height of this step is equal to the arc costs $\gamma_{ij}$ and represents the displacement between the zero reference levels of the two nodes.

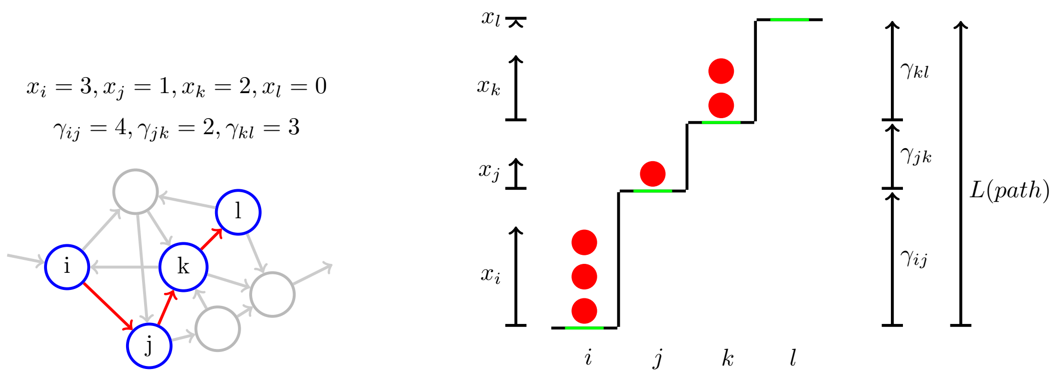

$x_i = 3, x_j = 1, x_k = 2, x_l = 0$
$\gamma_{ij} = 4, \gamma_{jk} = 2, \gamma_{kl} = 3$
i
j
k
l
$x_l$
$x_k$
$x_j$
$x_i$
$\gamma_{kl}$
$\gamma_{jk}$
$\gamma_{ij}$
$L(path)$
$i$
$j$
$k$
$l$

Figure 6.3: Graphical representation of a path $p = \{i, j, k, l\}$ as a *stair*. The height of each step is equal to the corresponding arc cost. The total elevation of the stair is equal to the path length $L(p) = \gamma_{ij} + \gamma_{jk} + \gamma_{kl}$.

Then, each directed weighted arc $(i, j)$ connecting two nodes $i$ and $j$ can be represented as a *step* whose height is equal to the (integer) arc cost $\gamma_{ij}$, see Fig. 6.2. Accordingly, each possible route $p$ in the network can be represented by a *stair* whose steps have heights equal to the route's arcs' costs: the elevation of this stair is equal to the corresponding total route cost $L(p)$, see Fig. 6.3. Remark that in a network multiple routes are present (possibly, there are infinite routes if there are cycles in the network and nodes can be traversed multiple times), each of which is to be represented by a different stair.

### 6.1.1 Transition rule

Consider a generic arc $(i, j)$. To define the tokens' transition rule between nodes, a *threshold* is introduced, equal to the arc cost $\gamma_{ij}$. In [72], the flow is controlled based on this same threshold and the difference of the states $x_i - x_j$: when the latter is greater than the threshold, the greater it is, the greater is the (possibly saturated) flow; otherwise, the flow is set to 0. Here, only two choices are possible: permit or deny the transition of the single tokens along the arc. Depending on the (integer) value of $x_i - x_j$, three conditions can emerge, see Fig. 6.4:

- *below threshold*: $x_i - x_j < \gamma_{ij}$, i.e., the difference between the states of the two nodes is smaller than the threshold;
- *at threshold*: $x_i - x_j = \gamma_{ij}$, i.e., the difference between the states of the two nodes is equal to the threshold;
- *above threshold*: $x_i - x_j > \gamma_{ij}$, i.e., the difference between the states of the two nodes is greater than the threshold.

When an arc $(i, j)$ is *below threshold* or *at threshold*, the top token in node $i$ cannot proceed moving to node $j$, as a barrier blocks it and upward displacements are not allowed, see Fig. 6.4. Conversely, when the arc is *above*

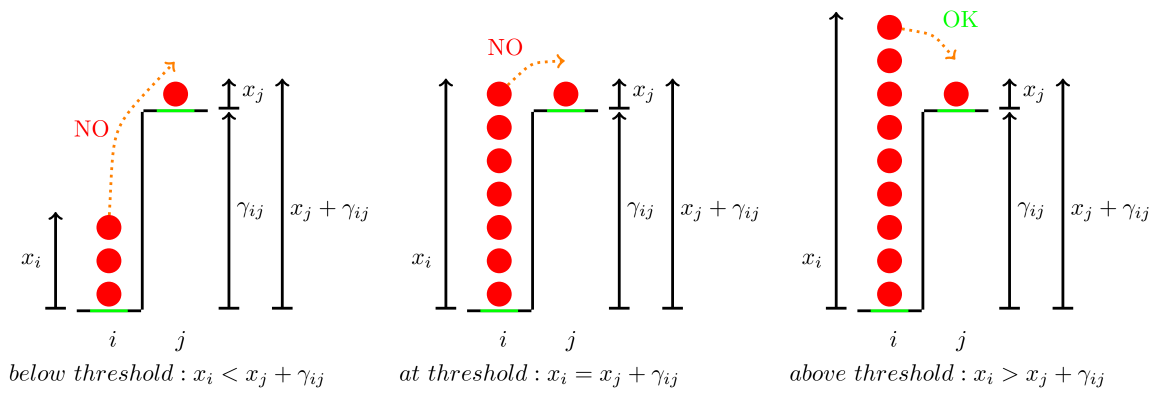

NO
$x_i$
$x_j$
$\gamma_{ij}$
$x_j + \gamma_{ij}$
$i$
$j$
*below threshold* : $x_i < x_j + \gamma_{ij}$
NO
*at threshold* : $x_i = x_j + \gamma_{ij}$
OK
*above threshold* : $x_i > x_j + \gamma_{ij}$

Figure 6.4: Examples of *below threshold* (left), *at threshold* (center), and *above threshold* (right) conditions for an arc $(i, j)$ joining node $i$ to node $j$. Only horizontal or downward token displacements between nodes are allowed. Hence, a token can move from node $i$ to node $j$ only if the above threshold condition holds.

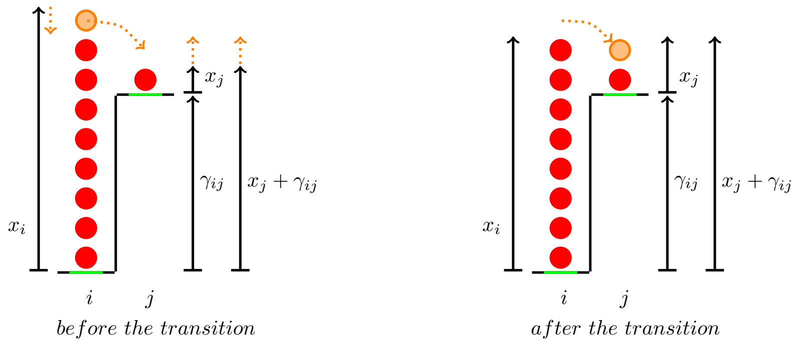


Figure 6.5: Examples of movement of an actual token when the "above threshold" condition holds for two consecutive nodes $i$ and $j$ connected by an arc.

*threshold*, displacements are allowed: this changes the states of the two nodes as $x_i \leftarrow x_i - 1$ and $x_j \leftarrow x_j + 1$, see Fig. 6.5. Then, a transition of a token from node $i$ to node $j$ occurs only if

$$x_i - x_j > \gamma_{ij}, \tag{6.1}$$

and states are updated as $x_i \leftarrow x_i - 1$ and $x_j \leftarrow x_j + 1$. Otherwise, the transition is *denied* (i.e., the token does not leave $i$). Note that in the above expression, it is assumed that the token to be moved is initially counted in $x_i$, even if it has just been injected from the outside. A token is *above-threshold* in node $i$ if condition Eq. (6.1) holds for some node $j$ adjacent to node $i$, and it is *under-threshold* otherwise. The routing decision depends only on local information about the current node $i$, the adjacent node $j$ and the connecting arc.

As it will be proven later, the essential result obtained by applying this policy is that if tokens are introduced in the network from some source nodes and have to reach some sinks (destinations) and if such introduction is persistent, then in the long run all tokens will travel in the network along the shortest path.

### 6.1.2 An example in a simple network

Consider the network in Fig. 6.6 with 5 nodes and 5 arcs. Tokens are injected regularly in the source node 1. Tokens reaching sink node 4 are immediately expelled from the network. As this network is very simple, only two paths with no repeated nodes are possible, namely: $p_1 = \{1, 2, 3, 4\}$, which is the shortest path, and $p_2 = \{1, 2, 5, 3, 4\}$. Hence, only two *stairs* are to be analyzed and it is easy to display what happens as tokens are injected into the network.

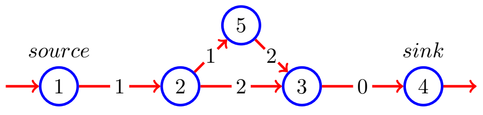


Figure 6.6: A simple network with 5 nodes and 5 arcs. Tokens are injected regularly in the source node 1. Tokens reaching sink node 4 are immediately expelled from the network.

Consider an initial zero state $x_1 = x_2 = x_3 = x_4 = x_5 = 0$ at time $k = 0$. Then, at every time $k$, a new token is injected in the source node 1, which starts moving according to the transition rule reported above. It is assumed that when a token in node 2 can move both to node 3 and 5, it moves to node 5 preferentially, i.e., it tries to take the non-shortest path. Fig. 6.7 shows the evolution of the states as new tokens are injected into the network. The first tokens stop in some node $1, 2, 3, 5$ without reaching the sink 4, as they cannot proceed further. As more tokens are injected, the stairs are *filled* and brought to the zero level of the sink node; this level is first reached by the stair associated with the shortest path, as fewer tokens are needed to do so. At some point, this allows all newly injected tokens to reach the sink and be collected, without modifying the state any further. The tokens stopped in the nodes are lost; however, once the steady-state is reached, all the newly injected tokens can reach the sink through the shortest path.

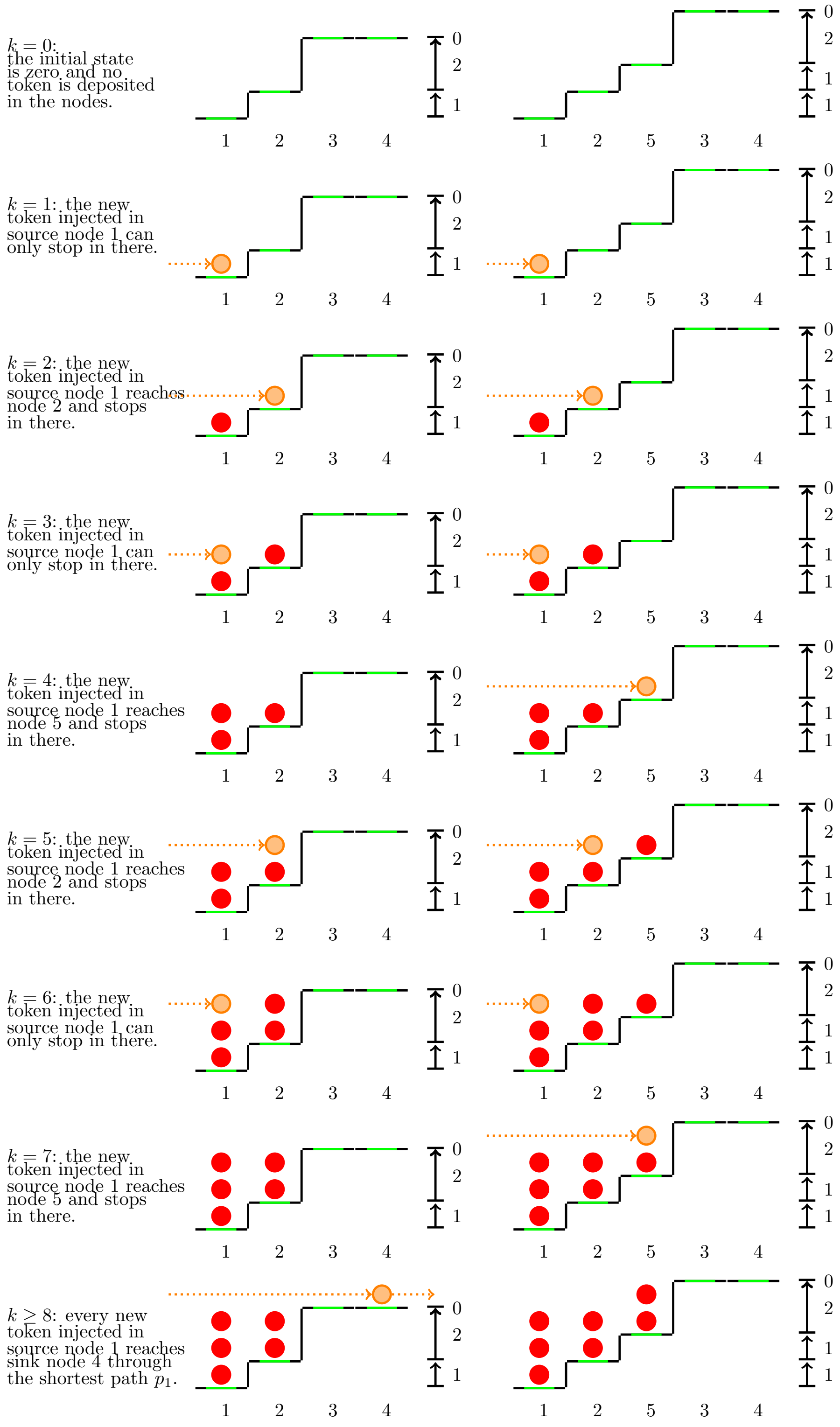

$k = 0$: the initial state is zero and no token is deposited in the nodes.
$k = 1$: the new token injected in source node 1 can only stop in there.
$k = 2$: the new token injected in source node 1 reaches node 2 and stops in there.
$k = 3$: the new token injected in source node 1 can only stop in there.
$k = 4$: the new token injected in source node 1 reaches node 5 and stops in there.
$k = 5$: the new token injected in source node 1 reaches node 2 and stops in there.
$k = 6$: the new token injected in source node 1 can only stop in there.
$k = 7$: the new token injected in source node 1 reaches node 5 and stops in there.
$k \geq 8$: every new token injected in source node 1 reaches sink node 4 through the shortest path $p_1$.
1 2 3 4
1 2 5 3 4
0
2
1

Figure 6.7: Graphical representation of the proposed policy applied to the network in Fig. 6.6, for the first time instants $k$. The stairs on the left refer to the (shortest) path $p_1 = \{1, 2, 3, 4\}$; the stairs on the right refer to the path $p_2 = \{1, 2, 5, 3, 4\}$.

### 6.1.3 Negative costs and states

So far, to simplify the exposition of the proposed policy, the state of the nodes and the arc costs have been assumed non-negative. Having negative arc costs simply means having a downward step: the same definition for the below, at, and above-threshold condition continues to hold, see Fig. 6.8, and the transition rule remains the same.

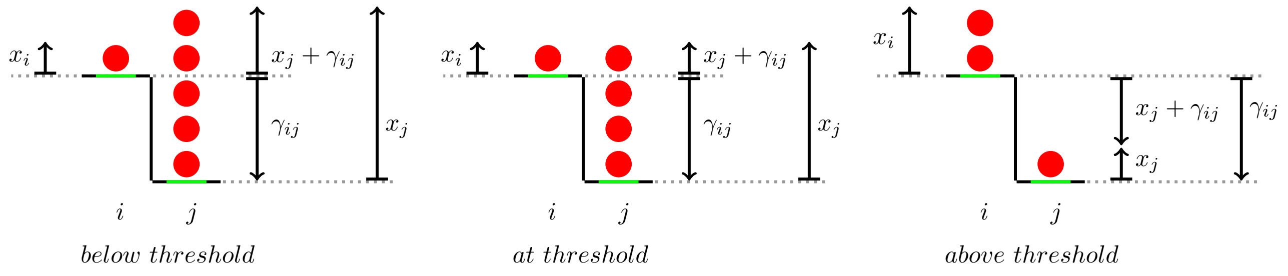


Figure 6.8: Arcs with negative cost are represented by downward steps. The meaning of above, at, below-threshold conditions are well-defined.

The *state of a node* is to be generalized when negative states are allowed. Denote the tokens considered so far as *actual tokens*, which might carry some information. The simpler way to interpret negative states is assuming that there are some non-informative *virtual* tokens that are accounted in each node and can be virtually moved from node to node when Eq. (6.1) holds but no actual token is present: the zero level of the node is set at some predefined level of such virtual tokens. The state of a node considers both the virtual and the actual tokens as a whole, without distinction, and counts them with respect to the zero level of the node, allowing for negative values (see Fig. 6.9). The idea behind the rule is the same, as differences of states are considered, and not their absolute values.

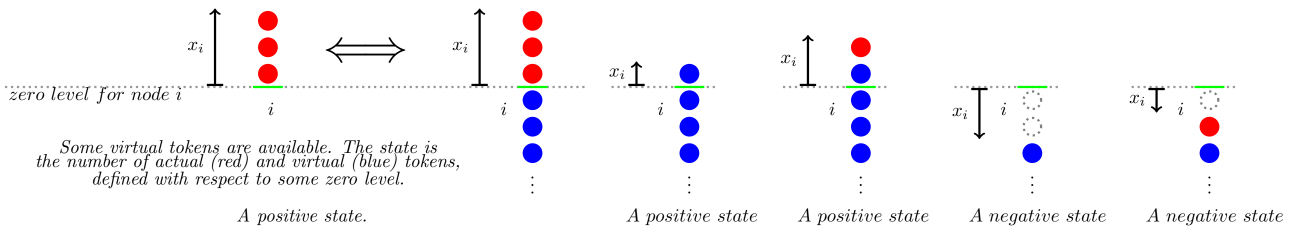


Figure 6.9: Some virtual tokens are assumed to be present in the network, other than the actual ones. The state is the number of tokens (both actual and virtual) expressed with respect to a predefined zero-level. Such virtual tokens might move from node to node, just like the actual tokens do. The state of the node does not distinguish between them and just considers the number of tokens below, at, or above the zero level. Red: actual (informative) tokens; blue: virtual (non-informative) tokens.

For instance, consider arc $(i,j)$ with $\gamma_{ij} = -2$ and $x_i = x_j = 0$ (i.e., there are no actual tokens). Such arc is above-threshold, as $x_i - x_j = 0 > \gamma_{ij} = -2$. A virtual token is moved from $i$ to $j$, making the state become $x_i = -1$ and $x_j = 1$: the new configuration is at threshold, see Fig. 6.10.

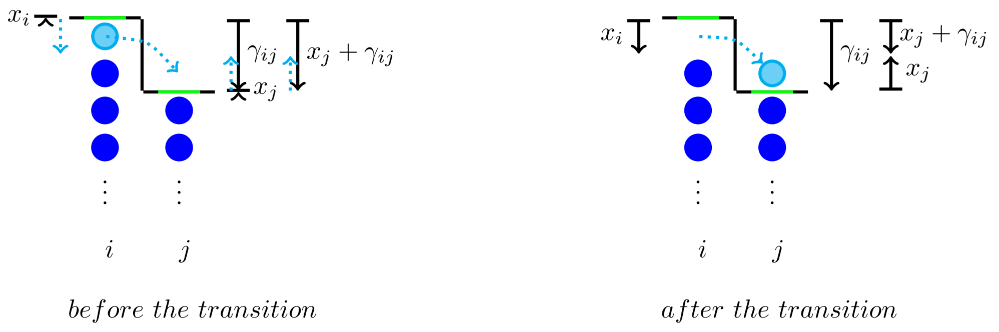
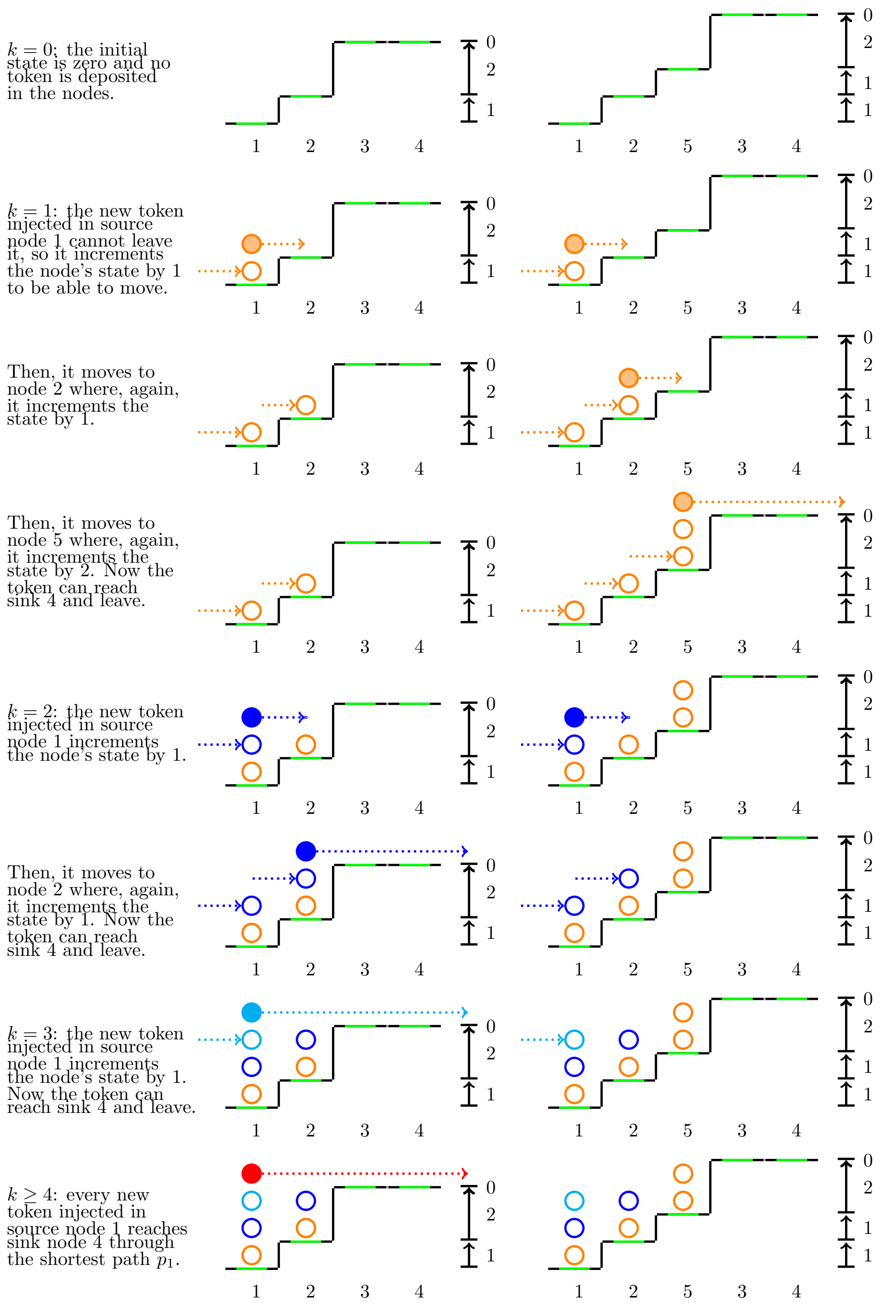


Figure 6.10: Examples of movement of a non-informative token when the "above threshold" condition holds for two consecutive nodes $i$ and $j$ connected by an arc.

Another interpretation is that the state is just a counter of the value of the tokens deposited in there. Actual tokens have value +1. When no actual (positive) token is present in a node $i$ and the above threshold condition holds for some arc $(i,j)$, a pair of virtual tokens is generated in $i$, one positive (with value +1) and one negative (with value $-1$), which does not change the state. Then, the negative token does not leave the node $i$, while the positive one moves to $j$: the state of $i$ becomes $x_i - 1$ and the state of $j$ becomes $x_j + 1$. Note that if in node $i$ there are both a negative and a positive token, the net value of these two tokens is 0 for the state: if such positive token is indeed virtual, the pair of positive/negative tokens is destroyed (no information is lost).

**Remark 6.1: An enhanced rule**

As it will be seen later, virtual tokens can also be artificially introduced to avoid an accumulation of actual tokens in the nodes. The main difference from the original rule is that now, when a token is in a node where there are no outgoing arcs for which the above-threshold condition holds, instead of stopping in there, it increases the state of the node it occupies until a transition to some adjacent node is possible. This improves the performance, and, if a strongly connected network is considered, ensures that all injected tokens are received. The application of this enhanced rule for the network in Fig. 6.6 is explained in Fig. 6.11. All the injected tokens find a way to leave the network. In the beginning, some tokens may take the non-optimal path, but, eventually, all the newly injected tokens can reach the sink through the shortest path.



$k = 0$: the initial state is zero and no token is deposited in the nodes.

$k = 1$: the new token injected in source node 1 cannot leave it, so it increments the node's state by 1 to be able to move.

Then, it moves to node 2 where, again, it increments the state by 1.

Then, it moves to node 5 where, again, it increments the state by 2. Now the token can reach sink 4 and leave.

$k = 2$: the new token injected in source node 1 increments the node's state by 1.

Then, it moves to node 2 where, again, it increments the state by 1. Now the token can reach sink 4 and leave.

$k = 3$: the new token injected in source node 1 increments the node's state by 1. Now the token can reach sink 4 and leave.

$k \geq 4$: every new token injected in source node 1 reaches sink node 4 through the shortest path $p_1$.

Figure 6.11: Graphical representation of the enhanced rule applied to the network in Fig. 6.6, for the first time instants $k$. The stairs on the left refer to the (shortest) path $p_1 = \{1, 2, 3, 4\}$; the stairs on the right refer to the path $p_2 = \{1, 2, 5, 3, 4\}$. Tokens filled with white are virtual tokens.

### 6.1.4 Constrained paths

The proposed policy can be adapted to support constraints on the paths. These constraints are expressed based on some secondary costs $\sigma_{ij}$ assigned to each arc. In particular, each token is characterized by a constrained cost $c$ which is the cumulative cost it has paid at each arc traversal since it has first started moving: $c$, however, cannot exceed a given value $C_{max}$. For instance, assume that $\sigma_{ij} = 1$ for all arcs: this means that the secondary cost of a path is the number of traversed arcs, so that a limitation is set to the maximum number of traversed arcs.

As moving tokens proceeds along their way, their constrained cost is updated as follows: if a token moves from node $i$ to node $j$ along an arc $(i, j)$ with secondary cost $\sigma_{ij}$, then, $c \leftarrow c + \sigma_{ij}$.

The rule proposed for the unconstrained case can be simply adapted to the constrained by assuming that:

- each token keeps tracks of the cumulative constrained cost $c$ of its traveled route up to the current occupied node;
- each node $i$ buffers the tokens deposited in there according to their constrained cost $c$; the state becomes a multi-component state, where each generic component $x_i^c$ denotes the number of tokens deposited in node $i$ with constrained cost $c$, see Fig. 6.12;
- a token currently in node $i$ with constrained cost $c$ makes its routing decision along the generic arc $(i, j)$ considering only the agents stopped in nodes $i$ and $j$ that have paid its same secondary costs $c$ and $c + \sigma_{ij}$, respectively, i.e., $x_i^c$ and $x_j^{c+\sigma_{ij}}$;
- denying a transition along arc $(i, j)$ if $c + \sigma_{ij} > C_{max}$, because such path is not allowed.

Then, the adapted policy can be summarized as follows; a transition of a token in node $i$ with constrained cost $c$ along arc $(i, j)$ occurs only if

$$x_i^c - x_j^{c+\sigma_{ij}} > \gamma_{ij} \quad \text{and} \quad c + \sigma_{ij} \leq C_{max}.$$

It will be shown that the adapted policy is optimal, i.e., in the long run, all the newly injected tokens leave the network through the constrained shortest path.

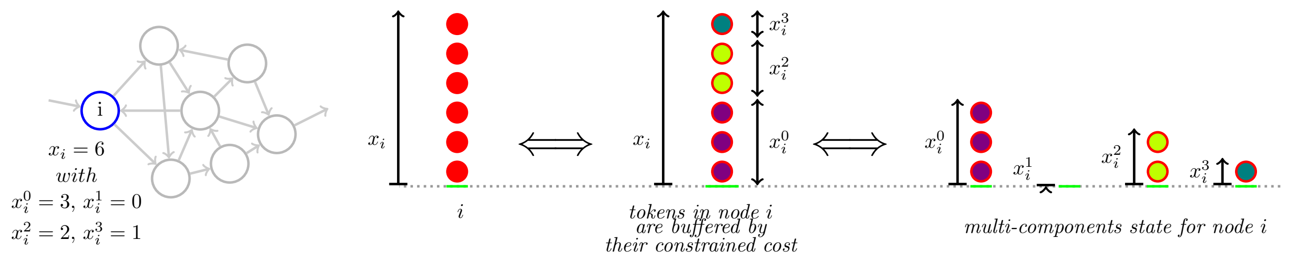


Figure 6.12: Graphical representation of the state $x_i$ of a node $i$ and its components in a constrained system: assuming a zero reference level, each component of the state can be seen as the number of tokens with a given constrained cost deposited in the node, which collects all the tokens with the same constrained costs.

## 6.2 Literature review

As already pointed out, the main focus of this Part is not presenting a new solution for the *shortest path problem* (SPP). Yet, some works that have been proposed to solve the SPP have some characteristics in common with the proposed approach (e.g., being decentralized or adaptive), hence they are reviewed next.

The *shortest path problem* (SPP) under known data is one of the most studied network problems [76], and many variants and solution algorithms have been proposed over the years [77]. Two of the most famous algorithms that are currently used to solve the SPP efficiently are Dijkstra's algorithm [78], which solves the problem in polynomial time for non-negative arc cost networks, and the Bellman-Ford algorithm [79, 80], which adds the support for negative cost arcs, at the expense of a slightly higher computational burden. In the latter case, it is usually assumed that no negative cycle is present in the considered network. Removing this assumption makes the SPP NP-hard and it turns out that the standard polynomial time algorithms are not able to find a solution, as indefinite loops emerge in the negative cycles (although some of them, including the Bellman-Ford algorithm, are able to detect this condition).

Moreover, in general, arc costs are usually assumed to be rational values. The more specific case in which costs are integer provides advantages from an algorithmic perspective in terms of space efficiency, speed of arithmetic operations, and stability [81, 82].

The above-mentioned algorithms, despite they could solve the SPP very efficiently, assume that the network is static and fully known. Conversely, in this Part, the network is assumed unknown and possibly dynamic: indeed, the moving tokens discover locally the nodes of the network as they proceed traveling from node to node, without remembering their traversed route; tokens are unaware of possible modifications occurring to the network, too. Several solutions have been proposed for supporting unknown [83, 84] and dynamic networks [85, 86]. Hereinafter, the works proposing decentralized agent–oriented methods are considered, using only local information and easily adaptable to possible modifications of the network. Note that algorithms like Dijkstra's and Bellman-Ford's require that the optimization is to be repeated from the beginning if a modification has occurred on the network.

In [87] and [88], some optimal decentralized methods based on consensus are presented. A model based on reinforced random walks is proposed in [73], where it is shown that shortest paths emerge. Also, many heuristics have been proposed, which are adaptive, but cannot guarantee finding the optimal shortest path, by the very definition of heuristics. Meta-heuristics inspired by nature are based on ant colony [89, 90, 91], river formation dynamics [92, 93], amoebas (slime molds) [94, 95], genetic algorithms [96], particle swarm optimization [97]. Moreover, learning automata have been employed in [98]; methods exploiting learning automata can also find stochastic shortest paths in stochastic networks, whose arc costs are random variables to be sampled [99, 100]. Most of the above-mentioned works, e.g., [89, 90, 91, 92], support only non-negative costs.

Many adaptive shortest-path routing protocols exist, for packet-switched networks [101, 102], ad-hoc and mesh networks [103, 104], and wireless sensor networks (WSN) [74, 75], supporting multipath routing, scalable performance, self-organizing behavior, the locality of interaction and network failure detection and backup [75]. The network architecture considered in this Part is indeed the one assumed in WSN; however, here some specific aspects such as link congestion and throughput are neglected.

Regarding the specific threshold policy presented in this Part, as already mentioned, this is inspired by the decentralized threshold-based control presented in [72] for (continuous) flow networks, where it is shown that such control is effective in concentrating the flow along the shortest path in the single–source–single–sink problem in the long run. Remark that here the problem is different, as a flow control policy is considered, instead of a navigation agent–oriented policy. A mechanism of this kind also governs natural phenomena, like lightnings, in the choice of their path [105]. Interestingly, it will be shown that a similar phenomenon observed in lightnings emerges also when applying the proposed strategy: when a shortest path is discovered, initially some non-optimal paths are still taken by some injected tokens, but become shorter and shorter, until they "vanish" and eventually all the newly injected tokens take one of the discovered shortest outgoing paths.

It is also interesting to mention that the SPP can be studied by modeling the networks as electronic analog circuits composed of Zener diodes (directed arcs) or non-linear resistors (undirected arcs), or as strings with knots, adopting a threshold mechanism [106, 107, 108, 109, 110].

Remark that the proposed strategy is decentralized and the routing decisions are individual and depend only on local information regarding the number of tokens stopped in the neighbouring nodes, but not directly on the other tokens routing decisions. This is different, for instance, from pedestrian flow models [111] and mean-field game routing problem formulations [112], where the interactions between agents influence those decisions.

When some secondary costs are also associated with each arc of the network and constraints are imposed on the possible routes, making those with a total secondary cost exceeding a given value infeasible, the *(resource) constrained shortest path problem* (CSPP) is to be considered [113], which is an extension of the SPP. Differently from the SPP, which could be solved in polynomial time by some centralized algorithms under the assumption of networks with no negative circuits, this constrained problem is known to be NP-hard [114]; it has been widely studied, too [115, 116].

State-of-the-art methods for solving the CSPP still consider static known networks, and are based on dynamic programming labeling methods [113, 117, 118, 119], the k-th shortest path algorithm [120], Lagrangian relaxation [117, 120] and pulses propagation [121].

Different methods handle the infeasible paths differently. For instance, in [118, 119] a preprocessing phase first reduces the initial network by deleting infeasible nodes and arcs, and then the resulting network is transformed into an extended one, where the SPP is solved. Instead, in [121] infeasible partial paths are pruned during the exploration of the network. In this Part, the concept of the extended network will be exploited to demonstrate the optimality of the presented policy in the constrained problem.

Some heuristics have been proposed, too, for the CSPP. An adaptive amoeba algorithm is combined with the Lagrangian relaxation algorithm in [122]. [123] introduces a bio-inspired rule. [124] presents a hybrid particle swarm optimization-variable neighborhood search algorithm. A multi-constrained routing strategy based on ant-colony optimization is introduced in [125]. Finally, an improved random walk search heuristic policy is studied in [126].

## 6.3 Contributions

To summarize, the main contribution of this Part (and [2]) is the definition of a local threshold rule that provides a global optimal result for both the SPP and the CSPP problems. In particular:

- the specification of a decentralized agent-based policy using a very simple local threshold rule to route tokens in an unknown network, requiring only the knowledge of local information (no information about the rest of the network, nor the past traveled route is required); this policy supports arcs with negative costs (assuming no non-positive circuits in the network) and can also be adapted to support constraints on the traveled routes;

- if tokens are persistently injected in some source nodes of the network, the proposed policy is able to make them reach some sink nodes, in the long run: after an initial transient phase in which tokens may be even forced to stop in some nodes of the network, a steady-state is eventually reached where no token stops in the nodes anymore;

- differently from the cited heuristics, the proposed policy is proven to be optimal in the long run: at steady-state the optimal (possibly constrained) shortest paths emerge, which are the paths mentioned in the previous point that are followed by the tokens at steady-state;

- the proposed policy is adaptive, as new optimal paths are eventually formed when dynamic modifications occur on the network; there is no need to restart from the beginning, as the past optimization results are exploited;

- the specification of an enhanced version of the proposed decentralized policy for strongly connected networks that improves the performance and, in the unconstrained case, ensures that all tokens reach a sink.

# Chapter 7

# Problem setup

In this Chapter, the considered system composed of a directed weighted network and a certain number of agents (tokens) moving in there is first introduced. Then, the main assumptions and the main problems to be solved are stated.

## 7.1 Weighted directed networks with moving tokens

A weighted directed network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ is considered, with $n$ nodes and $m$ arcs. The set of the nodes of the network is denoted by $\mathcal{N} = \{1, 2, \dots, n\}$ and the set of the arcs of the network is denoted by $\mathcal{A} = \{1, 2, \dots, m\}$.

Each arc $(i, j) \in \mathcal{A}$ of the network connecting node $i \in \mathcal{N}$ to node $j \in \mathcal{N}$ is characterized by two finite weights: a cost $\gamma_{ij} \in \mathbb{Z}$ and a secondary cost $\sigma_{ij} \in \mathbb{Z}$, which will be referred to by *constrained cost*. The system will be called *unconstrained system* when such constrained costs $\sigma_{ij}$ are not considered, and *constrained system* otherwise.

The considered network is connected to the external environment through some special nodes, called *source* and *sink* nodes, which are a subset of the nodes of the network. The set of the *source nodes* is denoted by $\mathcal{S} \subseteq \mathcal{N}$ and the set of the *sink nodes* is denoted by $\mathcal{T} \subseteq \mathcal{N}$, with $\mathcal{T} \cap \mathcal{S} = \emptyset$.

Some *tokens* are regularly injected from the external environment to the network, specifically in the source nodes in $\mathcal{S}$. Such tokens travel from node to node along the existing arcs, according to their direction and based on a given policy, trying to reach a sink node to leave the network. Such policy allows or denies the transitions along the arcs; if a transition is denied, the token must stop in the node it currently occupies, preventing it to reach a sink. Otherwise, if the token is allowed to continue moving, once it arrives in a sink node in $\mathcal{T}$, it is immediately expelled from the network to the external environment.

Then, each injected token will either stop in a node of the network or reach a sink, where it is expelled from the network. The last node occupied by the moving token (possibly a sink node) will be referred to as *destination node*.

Moreover, for each node $i \in \mathcal{N}$, the subset of the neighboring nodes reachable by traversing at most one arc is also introduced, as

$$\mathcal{N}_i = \{j \in \mathcal{N} : (i, j) \in \mathcal{A}\} \cup \{i\}, \quad \text{for all} \quad i \in \mathcal{N}.$$

### 7.1.1 Possible traveled routes: paths, walks and circuits

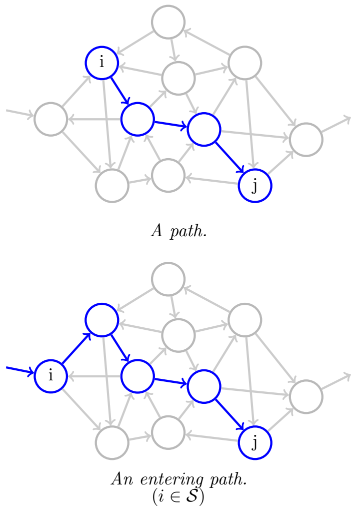
Tokens traverse the network, moving from node to node along the existing arcs. A classification of the types of routes that they may travel is made, see Fig. 7.1.

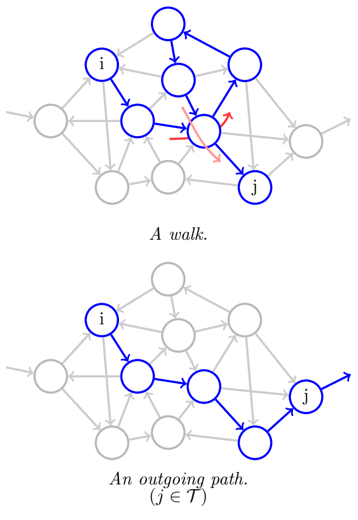
**Definition 7.1:**
A *path* $p = \{h_s \in \mathcal{N}, s = 1, 2 \dots, r\}$ joining a *starting node* $i$ and a *terminal node* $j$ on $\mathcal{G}$ is an ordered sequence of $r$ non-repeated nodes $i = h_1, h_2 \dots, h_r = j$ joined by an arc, i.e., such that $(h_s, h_{s+1}) \in \mathcal{A}$ and $h_s \neq h_t$, for all $s, t = 1, 2, \dots, r$, with $s \neq t$.

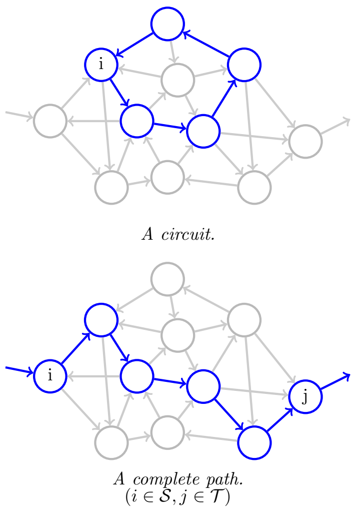
A *walk* $w = \{h_s \in \mathcal{N}, s = 1, 2 \dots r\}$ joining a *starting node* $i$ and a *terminal node* $j$ on $\mathcal{G}$ is an ordered sequence of $r$ possibly repeated nodes $i = h_1, h_2 \dots h_r = j$ joined by an arc, i.e., such that $(h_s, h_{s+1}) \in \mathcal{A}$.

A *circuit* $c = \{h_s \in \mathcal{N}, s = 1, 2 \dots r\}$ on $\mathcal{G}$ is a closed walk, where only the first and the last node coincide, i.e., an ordered sequence of $r$ nodes $i = h_1, h_2 \dots h_r = i$ such that $(h_s, h_{s+1}) \in \mathcal{A}$, $h_1 = h_r$, and $h_s \neq h_t$, for all $s, t = 1, 2, \dots, r - 1$, with $s \neq t$.

The *traveled route of a token* at time $t$ is the ordered sequence of nodes reached by the moving token since the moment it has *first* started moving (or, equivalently, since time 0) up to time $t$. It might be a path, a walk, or a circuit depending on such nodes.







*A path.* *A walk.* *A circuit.*

*An entering path.* $(i \in \mathcal{S})$ *An outgoing path.* $(j \in \mathcal{T})$ *A complete path.* $(i \in \mathcal{S}, j \in \mathcal{T})$

Figure 7.1: The possible routes in a network.

Each token might first start moving from a source node (after being injected in there) or from any node (if, for some reason, it was already in the network); in the former case, $h_1 = i \in \mathcal{S}$ and the resulting traveled path/walk is *entering* the network. Moreover, a traveling token might stop moving in a certain node, and possibly resume moving later on; if the token is able to reach a sink and leave the network, then, $h_r = j \in \mathcal{T}$ and the corresponding traveled path/walk is said *outgoing*. A token injected in a source travels a *complete* path/walk if it is able to reach a sink and exit. Note that the traveled route of a token *is not reset* if the token stops in a node and resumes moving after a while.

A sequence of traversed arcs is associated with each path/walk/circuit. Then, each route can be characterized based on the costs and the constrained costs of such traversed arcs, by introducing the following concepts.

**Definition 7.2:**
The *length* of a path $p$ (or a walk, or a circuit) is the sum of the arc costs of all the traversed arcs:

$$L(p) \doteq \sum_{h_k, h_{k+1} \in p} \gamma_{h_k h_{k+1}}. \tag{7.1}$$

In constrained systems, the *constrained cost* of a path $p$ (or a walk, or a circuit) is the sum of all its constrained arc costs:

$$C(p) \doteq \sum_{h_k, h_{k+1} \in p} \sigma_{h_k h_{k+1}}. \tag{7.2}$$

Depending on the sign of $L(p)$ the path/walk/circuit is said *positive* (for the costs $\gamma_{ij}$) if its length is positive, $L(p) > 0$, *non-positive* if $L(p) \leq 0$, and *negative* if $L(p) < 0$. Similar definitions hold with respect to the costs $\sigma_{ij}$, based on the sign of $C(p)$.

In constrained systems, the constrained cost $C(p)$ is used to specify which routes a token can take when moving in the network; a token can continue moving as long as its traveled route is feasible.

**Definition 7.3:**
In a constrained system, a path $p = \{h_1, h_2, \ldots, h_r\}$ (or a walk, or a circuit) is said *feasible* if and only if

$$C(p) \leq C_{max}, \tag{7.3}$$

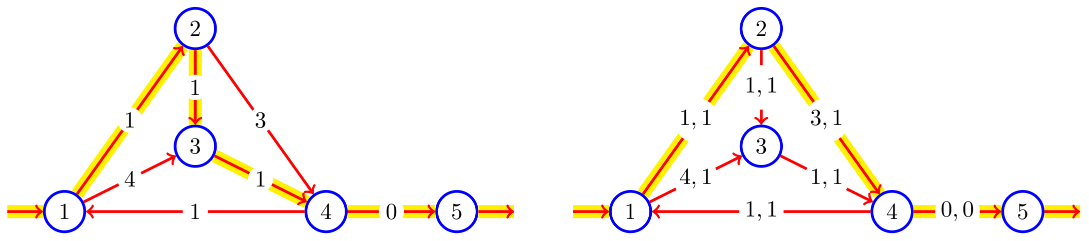
where $C_{max} \in \mathbb{N}$ is the maximum constrained cost allowed for a path, and the partial subpaths $p_i = \{h_1, \ldots, h_i\}$ from $h_1$ to the intermediate node $h_i, i < r$ are feasible, too.

The constrained cost of the traveled path of a token takes into account all the arcs traversed since the token has first started moving (i.e., since time 0). Note that feasibility is defined recursively and takes into account all the intermediate step nodes. When constrained costs are non-negative, $\sigma_{ij} \geq 0$, the feasibility of the subpaths is automatically satisfied by Eq. (7.3). Otherwise, Eq. (7.3) is not sufficient. For instance, the path $p = \{A, B, C\}$ with $\sigma_{AB} = 4$ and $\sigma_{BC} = -2$ is not feasible for $C_{max} = 3$: despite $C(p) = 4 - 2 = 2 < C_{max} = 3$, the subpath $p_B = \{A, B\}$ is not feasible, as $C(p_B) = 4 > C_{max} = 3$. If a token reaches a node from which it cannot proceed to any other node without making the traveled route infeasible, it is forced to stop in that node.

**Remark 7.1:**
The length $L(p)$ and the constrained costs $C(p)$ of the traveled route $p$ have different meanings for a traveling token. Such token would like to take the shortest path with respect to the length $L(p)$. Moreover, in a constrained system, the token cannot take any route with a constrained cost greater than $C_{max}$. Then, $L(p)$ is a quantity to be minimized, while $C(p)$ defines a constraint.

For instance, if the arc costs $\gamma_{ij}$ are physical lengths and the secondary arc costs $\sigma_{ij}$ are traversal times, the tokens might want to take a route that minimizes the total traveled physical length, while not exceeding a given traversal time.

**Example 7.1:**
Consider the network $\mathcal{G}$ in Fig. 7.2. The outgoing path $p = \{1, 2, 4, 5\}$ connecting source node 1 to sink node 5 has length $L(p) = \gamma_{1,2} + \gamma_{2,4} + \gamma_{4,5} = 1 + 3 + 0 = 4$, and constrained cost $C(p) = \sigma_{1,2} + \sigma_{2,4} + \sigma_{4,5} = 1 + 1 + 0 = 2$. This is the shortest one in a constrained system with $C_{max} = 2$. Instead, the shortest unconstrained path is $p = \{1, 2, 3, 4, 5\}$ with $L(p) = 3$ and $C(p) = 3$.



*Unconstrained system.* *Constrained system with $C_{max} = 2$.*

Figure 7.2: A simple network $\mathcal{G}$, both considering an unconstrained system (left) and a constrained system (right). The values $\gamma_{ij}$ and $\sigma_{ij}$ are indicated for each arc $(i, j)$. The shortest paths are highlighted.

### 7.1.2 Buffers in the nodes

A buffer is present in each node, which collects the tokens that have stopped in there.

It is assumed that such tokens are stored in a last-in-first-out (LIFO) manner: if a new token stops in the node, it is deposited at the top of the buffer; it is also the first one to possibly leave the node to move to another one. The FIFO case works as well.

A *state* is associated with each node, based on the tokens deposited in its buffer.

The *state* $x_i(t_k) \in \mathbb{Z}$ *of a node* $i \in \mathcal{N}$ at time $t_k \geq 0$ is simply the number of tokens present in node $i$ at that time, defined with respect to the "zero level of the node". A negative state is possible, and means that the number of tokens is below this zero level; in this regard, it can be assumed that initially, in the network, there is already a given amount of non-informative/virtual tokens, and that the zero-level is indeed their initial level. Specifically, let $r_i(t_k)$ be the possibly negative total number of tokens in a node (both coming from the external environment or already present in the network) at time $t_k$ and let $R_i = r_i(0)$ be the "zero level of the node". Then, the concept of state is reinterpreted as

$$x_i(t_k) = r_i(t_k) - R_i.$$

**Remark 7.2:**
The interpretation of negative states could be revised by avoiding assuming that there are already some tokens present in the network at time 0, e.g., by taking $x_i(t_k) = l_i(t_k) + c_i(t_k)$, where $l_i$ is the (possibly negative) *zero level of the actual tokens* at time $t_k$, i.e., of the tokens that were injected in the sources, defined with respect to the zero level of the node, and $c_i$ the (non-negative) *number of actual tokens* present in the node, defined with respect to the level $l_i$ of the node. The state $x_i$ is the (possibly negative) *level of actual tokens* present in the node, defined with respect to the zero level of the node. When an actual token is present in the network and is to be moved, $c_i$ changes and $l_i$ remains unchanged. Otherwise, if, for some reason, a transition needs to

be performed when there are no actual tokens present in the node, $l_i$ changes and $c_i$ remains unchanged: this latter possibility could be seen as the generation of a virtual token at time $t_k$.

When constraints on the path length are present, the buffers' state needs to be redefined. Each token keeps the information of its own partial constrained cost, namely the secondary cost already accumulated in its previous transitions.

In each node, tokens are assumed to be buffered according to their *current constrained cost* $c = C(p)$, where $p$ is the traveled route of each token, i.e., according to the sum of the constrained costs $\sigma_{ij}$ that they paid along the route they used to reach $i$. Assume, for simplicity, that $\sigma_{ij} \geq 0$ for all arcs $(i, j) \in \mathcal{A}$, so that $C(p) \geq 0$ for all possible routes. Then, the *state* $x_i(t_k) \in \mathbb{Z}^{C_{max}+1}$ *of a node* $i \in \mathcal{N}$ at time $t_k \geq 0$ in now a vector whose components $x_i^c(t_k)$, for $c = 0, \dots, C_{max}$, are the number of tokens with current constrained cost $c$ present in the node $i$ at that time,

$$x_i(t_k) = \begin{bmatrix} x_i^0(t_k) \\ \vdots \\ x_i^{C_{max}}(t_k) \end{bmatrix}.$$

In general, if $\sigma_{ij} \in \mathbb{Z}$, there is a component for all the possible values that $C(p)$ can assume. The information about the constrained costs in the state components will be used to take into account the feasibility of the traveled routes. Note that, for an unconstrained system, all paths are actually feasible, thus this information is not needed.

Finally, the *state of the network at time* $t_k$ is the vector $x(t_k) \in \mathbb{Z}^n$ of the nodes' states at that time,

$$x(t_k) = \begin{bmatrix} x_1(t_k) \\ \vdots \\ x_n(t_k) \end{bmatrix}.$$

The total number of tokens in the network at generic time $t_k$ (with respect to the zero reference level) is given by counting the tokens present in each node: in an unconstrained system, this is denoted by

$$V(x(t_k)) = \sum_{i \in \mathcal{N}} x_i(t_k),$$

while in a constrained system, letting $\mathcal{C}$ be the set of all the possible values assumed by $c = C(p)$, this becomes

$$V(x(t_k)) = \sum_{i \in \mathcal{N}} \sum_{c \in \mathcal{C}} x_i^c(t_k).$$

Under particular conditions, function $V$ will be shown to be monotonically increasing and will have an essential role in proving the optimality results.

**Remark 7.3:**
By definition of sink node, $x_i(t_k) = 0$ for all sink nodes $i \in \mathcal{T}$ for all $t_k \geq 0$, since all tokens arriving in there are expelled.
For a constrained system, it also holds that $x_i^c(t_k) = 0$ for all sink nodes $i \in \mathcal{T}$, for all possible constrained costs of the traveled paths $0 \leq c \leq C_{max}$, for all $t_k \geq 0$.

## 7.2 Decentralized policies

Tokens decide in which node to move in and if/where to stop according to a decentralized policy, see Fig. 7.3.

**Definition 7.4:**
A token routing policy is *decentralized* if tokens make their routing decisions on the basis of local information only. Specifically, for a token in node $i$, only the information about the nodes $j \in \mathcal{N}_i$, the arcs $(i, j)$ with $j \in \mathcal{N}_i$, and the token itself is available.

Note that by the definition of decentralized policy, all the information about non-neighboring nodes is not available. The rest of the network might even be unknown to the moving agent, and it is discovered as it proceeds moving from node to node.

In the specific case considered here, it is assumed that this policy uses the least amount of information possible and that the tokens have no memory of the nodes/arcs of the traveled route $p$ so far, except for its current constrained cost $c = C(p)$. Then, just the following quantities will be considered as available for formulating a decision for the token:

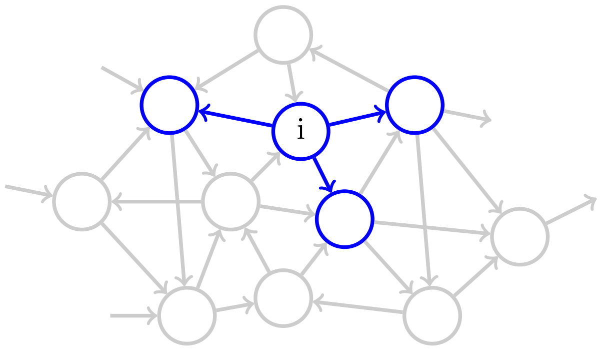


Figure 7.3: Decentralized policy: for a token in node $i$, only the information on the neighboring nodes $j \in \mathcal{N}_i$ and the corresponding arcs $(i, j)$, which are highlighted in blue, are available.

- the state of the current node $x_i$, and, if a constrained system is considered, the components $x_i^c$;
- the state of the neighboring nodes $x_j$, for all $j \in \mathcal{N}_i \setminus \{i\}$, and, if a constrained system is considered, the corresponding components $x_j^c$;
- the cost $\gamma_{ij}$ of the incident arcs $(i, j)$, for all $j \in \mathcal{N}_i \setminus \{i\}$, and, if a constrained system is considered, the corresponding constrained cost $\sigma_{ij}$;
- if a constrained system is considered, the current constrained cost $c = C(p)$ of the traveled route $p$ so far.

**Remark 7.4:**
The routing decision can be performed either by the *token agent*, or equivalently by the *node agent*, or equivalently by the *arc agent*.

## 7.3 Timing of the system dynamics

The system is modeled as a discrete-time, multiple-time-scale dynamical system. In particular, two time-scales are considered to describe the motion of the injected tokens:

- a coarser *slow dynamic* timescale $t_0 < t_1 < ... < t_k < ...$ marks the times at which tokens are injected into the source nodes. The $k$th token entering the network is the one injected at $t_k$.
- a finer *fast dynamic* timescale $t_k = t_k^0 < t_k^1 < ... < t_k^{N_k}$, marks the times of the $N_k$ *elementary transitions* that a generic token $k$ performs from $t_k$ until it stops in a node or reaches a sink.

**Definition 7.5:**
An *elementary transition* is the movement of a token from a node $i \in \mathcal{N}$ to a node $j \in \mathcal{N}$ along the arc $(i, j) \in \mathcal{A}$ connecting the two nodes.

A *network transition* is the overall set of all the elementary transitions occurring between two consecutive time instants $t_k$ and $t_{k+1}$ of the slow dynamics.

Sometimes, however, under certain conditions, some of the tokens already present in the network might be able to move, other than the injected token. Some intermediate time instants $t_{k,r}$ of the slow dynamic in-between $t_k$ and $t_{k+1}$ are introduced, such that $t_k < t_{k,1} < t_{k,2} < ... < t_{k,r} < ... < t_{k+1}$, in which no token is injected, but the token that starts moving is one already present in the network. For the generic token that starts moving at $t_{k,r}$, the corresponding finer *fast dynamic* timescale $t_{k,r} = t_{k,r}^0 < t_{k,r}^1 < ... < t_{k,r}^{N_{k,r}}$ marks the times of the $N_{k,r}$ *elementary transitions* that such token performs from $t_{k,r}$ until it stops in a node or reaches a sink.

To formally study the system, it will be assumed that *only a token at a time* can move in the network. Then, a priority criterion will be adopted, by which the tokens already present in the network start moving first, one at a time, until no other one can move further. A new token can be injected into the network only after the one injected just before it is not able to move anymore and there are no other tokens that can move, too. Hence, the following must hold for the instants of the slow dynamics:

$$t_0 < t_{0,1} < t_{0,2} < ... < ... < t_1 < \ldots < t_k < t_{k,1} < t_{k,2} < ... < t_{k+1} < ....$$

Moreover, for any two consecutive time instant of the slow dynamics $t_k < t_k^1 < \cdots < t_k^{N_k} < t_{k+1}$ (a similar relation is imposed if the times $t_{k,r}$ are involved). This will be better specified in the Assumption 7.4 reported in the next Section, where all the assumptions made for the considered system are stated.

Hereinafter, for simple notation, $t_k$ will indicate the generic time instant of the slow dynamic in which either a token already present in the network or a newly injected one starts moving, where the former possibility takes priority. Also, considering a generic time instant of the slow dynamic $t_k$, the following notation will be adopted:

$$x(k) \doteq x(t_k), \quad \text{for } k = 0, 1, \ldots.$$

A similar notation will be used for $V(k)$) and the other quantities that will be introduced later on.

## 7.4 Assumptions

Several assumptions are introduced for the network, the traveling tokens, and the adopted policy.

**Assumption 7.1:**
Network $\mathcal{G}$ is

(a) weakly connected,

and has

(b) at least one path between each source node $i \in \mathcal{S}$ and (at least) one sink node $j \in \mathcal{T}$. If the system is constrained, such path must be feasible, i.e., it must have a constrained cost not greater than $C_{max}$;

(c) no non-positive circuits with respect to the costs $\gamma_{ij}$ and no negative circuits with respect to the costs $\sigma_{ij}$; namely, $L(c) > 0$ and $C(c) \geq 0$ for all possible circuit $c$;

(d) the costs $\gamma_{ij}$ and $\sigma_{ij}$ of all its arcs $(i, j) \in \mathcal{A}$ which are upper bounded by some $\bar{\gamma} \in \mathbb{N}$ and $\bar{\sigma} \in \mathbb{N}$, respectively. Namely, $\gamma_{ij} \leq \bar{\gamma}$ and $\sigma_{ij} \leq \bar{\sigma}$;

(e) no negative paths between any pair of sink nodes in $M$.

In this Assumption 7.1, condition (a) simply means that isolated/disconnected network components are not considered.

Condition (b) ensures that for any token injected in the network (in any source node) there exists at least one outgoing path it could follow to leave the network, if no limiting rule is applied to its displacements.

Condition (c) prevents the possibility of the tokens looping indefinitely in the network and ensures that the shortest route in a network is necessarily a path (and not a walk), both for the constrained and unconstrained system.

Condition (d) implies that path lengths and costs are finite.

Finally, condition (e) implies that once a token reaches any sink node $i \in \mathcal{T}$ it can leave the network immediately, so that the state of sinks nodes is kept zero, $x_i \equiv 0$. Indeed, suppose that a token moves trying to discover the shortest path in a network where there are two sink nodes $i, j \in \mathcal{T}$ connected by a negative path $p$, $L(p) < 0$, and, for simplicity, only one source $h \in \mathcal{S}$. If the shortest possible path connects the source $h$ to the sink $j$ and passes through node $i$, it turns out that tokens would actually leave the network earlier, as soon as they reach node $i$, through a "suboptimal" shortest path: if they continued moving, the length of the traveled path might be reduced even further by leaving the network in node $j$. This assumption is specific to the proposed policy, but it is not limiting: each sink node $i \in \mathcal{T}$ can be converted into a non-sink node and connected to some new sink node $j \in \mathcal{T}$ through an arc $(i, j)$ with $\gamma_{ij} = 0$ (and possibly $\sigma_{ij} = 0$), to maintain the same shortest paths, see Fig. 7.4.

Conditions in Assumption 7.1 are not restrictive, as they are typically made in shortest path problems to ensure the existence of a solution: under them, the unconstrained problem can be solved in polynomial time by some centralized algorithms.

**Assumption 7.2:**
Tokens are:

(a) memoryless of the traveled path;

(b) only in case of a constrained system, able to update (by integrating) the cumulative constrained cost $c$ of their traveled paths.

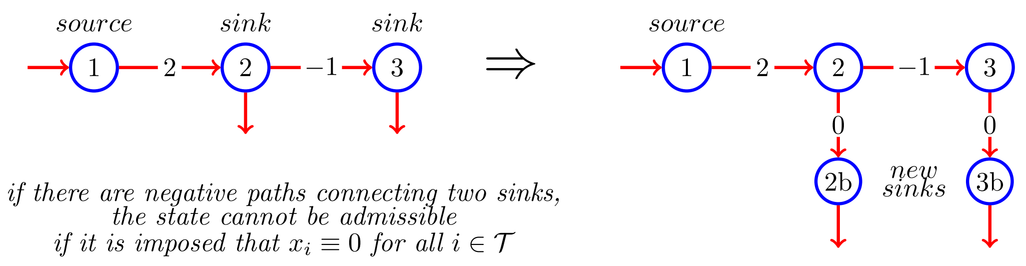

source
sink
sink
1
2
2
−1
3
⇒
source
1
2
2
−1
3
0
0
2b
3b
new sinks
if there are negative paths connecting two sinks, the state cannot be admissible if it is imposed that $x_i \equiv 0$ for all $i \in \mathcal{T}$

Figure 7.4: Assumption 7.1.e is not limiting: an equivalent network without negative paths connecting two sink nodes can always be constructed.

Assumption 7.2 means that, along with the requirement for a decentralized policy, the least amount of information is required to make decisions about the tokens' movements.

**Assumption 7.3:**
Token movements are:

(a) asynchronous;

(b) fast: once a token is injected in the network, or in general it first starts moving from a node, it reaches the destination node before any other token is inserted in the network.

In Assumption 7.3 condition (a) means that each token moves autonomously, without coordination with the other tokens, coherently with the assumption of a decentralized policy. Condition (b) means that the time taken by tokens to move in the network is of orders of magnitude smaller than the time intervals between successive injections of tokens in the network. In the practice, this latter assumption can be relaxed by requiring that there cannot be multiple tokens traveling simultaneously in adjacent nodes.

**Assumption 7.4: Slow-fast dynamic system**
Network $\mathcal{G}$ has two kinds of time dynamics:

1. *Fast dynamics.* Each token moving in the network $\mathcal{G}$ takes a negligible time to complete its path, as long as elementary transitions are possible.

2. *Slow dynamics.* After a token is injected at time $t_k$, a new token can be injected at time $t_{k+1}$ with the interval $t_{k+1} - t_k$ long enough to ensure that the token that started moving at $t_k$ has reached its destination node (possibly a sink).

Assumption 7.4 implies that at each time at most one token can move in the network. At each $t_k$, the other tokens that are not said token are assumed "frozen". Tokens that were injected prior to $t_k$ have already reached a destination node, and, if this is a sink node, they have also already left the network. Then, to study the variation of the state of the network and the behavior of the tokens, it is sufficient to consider only the final network state reached after the token that had started moving at $t_k$ stops moving, which will be the network state at $t_{k+1}$.

Remark that assuming that only one token at a time can move in the entire network is needed just to easily study the system and its properties. In a realistic scenario, several tokens move simultaneously in the network. The point is that the tokens are studied considering them one at the time. Clearly, this is absolutely legitimate when tokens do not interact: having simultaneous movements involving different nodes produces the same effects on the state of the network as imposing that only one token at a time can move. When different tokens travel simultaneously from/to the same node, it is reasonable to assume that the node can process only one token at a time, locally, with an arbitrary priority. Then, it is not difficult to see that the overall effect corresponds to the case in which the tokens are considered separately and reach an admissible state. Then, as it will be also shown in the examples from Chapters 10 to 12, dropping this assumption does not change the properties of the system. Moreover, it will be shown that under a certain condition of the state ("admissible state"), which will always be eventually reached, if only one token is injected, this will automatically be the only one to move in the network, until it either stops or exit the network: if the frequency of injection is not too fast, the above assumption is satisfied without knowing the state of the entire network.

**Remark 7.5:**
Assumption 7.4 introduces some determinism in the mechanism in a somewhat arbitrary way.
Relaxing Assumption 7.4 means: allowing simultaneous movements of tokens involving different nodes, assuming a negligible time just for the elementary arc transitions, and allowing new token injections before a moving

token reaches its destination. Then, all the elementary transitions should be taken into account to study the state evolution. If two tokens have to move from the same node, giving precedence to one or to the other might makes a difference in the resulting state.
The results that will be achieved in this Part II will still hold, however, the handling of the problem would become much more complicated.

## 7.5 Problem statement and its variations

The main problem that will be considered in this Part deals with traveling agents injected in a constrained system.

**Problem 7.1: Constrained problem**
Define a decentralized policy that, in the long run, makes each token injected in the source nodes $\mathcal{S}$ reach sink nodes in $\mathcal{T}$, along a feasible path of minimum length and of constrained cost not greater than $C_{max}$.

A special (relaxed) case of this problem is obtained by disregarding the secondary costs $\sigma_{ij}$, allowing all the paths regardless of their constrained cost.

**Problem 7.2: Unconstrained problem**
Define a decentralized policy that, in the long run, makes each token injected in the source nodes $\mathcal{S}$ reach sink nodes in $\mathcal{T}$, along a path of minimum length.

This unconstrained version of the problem is much simpler than the constrained version. Indeed, recall that the problem of finding the constrained shortest path is an NP-hard problem, while the (unconstrained) shortest path can be discovered in polynomial time by some well-known centralized algorithm like the Dijkstra's algorithm [78] and the Bellman-Ford algorithm [79, 80]. Then, the unconstrained problem will be first addressed in the next Chapter 8. The solution to the constrained problem will be built upon the solution to the unconstrained case in Chapter 9.

Note that, in practice, the constrained Problem 7.1 reduces to the unconstrained Problem 7.2 either when $\sigma_{ij} \equiv 0$ (for any $C_{max} \in \mathbb{N}$) or when $C_{max} = \infty$: the outcome is the same. However, as will be discussed in Chapter 9, the adopted policy has different performance depending on whether the secondary costs $\sigma_{ij}$ are disregarded, or they are assumed zero, or it is assumed that $C_{max} = \infty$.

CHAPTER 8

# A decentralized agent-based policy finding the shortest paths

In this Chapter an unconstrained system is assumed and Problem 7.2 is addressed. A decentralized threshold policy is introduced, by which tokens are routed from node to node. In the long run, this policy makes all the newly injected tokens reach the closest sink along the shortest paths.

As the main result, it will be shown, by analyzing the slow dynamics, that the system may reach only a subset of all possible states, named the set of admissible states, and it will eventually reach an (admissible) steady-state. Such a steady-state is associated with the shortest path routing for all tokens.

## 8.1 Admissibility of the state

The concept of the admissibility of the state of the network is first introduced to define the proposed policy and its properties.

**Definition 8.1: Admissibility in an unconstrained system.**
The state $x$ of a network $\mathcal{G}$ is admissible if and only if $x_i - x_j \leq \gamma_{ij}$ for all $(i,j) \in \mathcal{A}$.

The following definitions are also introduced based on the definition of admissible state:

**Definition 8.2:**
An arc $(i,j) \in \mathcal{A}$ is said:

- *above-threshold* if $x_i - x_j > \gamma_{ij}$;
- *at-threshold* if $x_i - x_j = \gamma_{ij}$;
- *below-threshold* if $x_i - x_j < \gamma_{ij}$;
- *under-threshold* if $x_i - x_j \leq \gamma_{ij}$, i.e., if it is either below or at-threshold.

A token in node $i \in \mathcal{N}$ is said:

- *above-threshold* if there exists an above-threshold arc $(i,j), j \in \mathcal{N}_i$ joining node $i$ and an adjacent node $j$;
- *at-threshold* if there exists an at-threshold-arc $(i,j), j \in \mathcal{N}_i$, but no above-threshold arc $(i,j), j \in \mathcal{N}_i$;
- *below-threshold* if there exists no above-threshold or at-threshold arc $(i,j), j \in \mathcal{N}_i$.
- *under-threshold* if there exists no above-threshold arc $(i,j), j \in \mathcal{N}_i$.

By the definition of admissible state, the following property is derived: a state is admissible if for any two nodes $i, j \in \mathcal{N}$ there is no direct path between them, from $i$ to $j$, with cost smaller than the difference of the states of the two nodes, $x_i - x_j$.

**Lemma 8.1:**
The state $x$ of a network $\mathcal{G}$ is admissible if and only if

$$x_i - x_j \leq L(p), \tag{8.1}$$

for all $i, j \in \mathcal{N}$, for all paths $p$ connecting $i$ to $j$.

From the above Lemma, the set of admissible states is empty if

- there is a negative circuit $p = i = h_1, h_2, \dots, h_p = i$ because Eq. (8.1) becomes $x_i - x_j = 0 \leq L(p)$, which does not hold, as $L(p) < 0$. Then, Assumption 7.1.c is needed.
- there is a negative path $p$ that joins two sink nodes $i, j \in \mathcal{T}$. Indeed, Eq. (8.1) becomes $x_i - x_j = 0 \leq L(p)$, which does not hold, because $L(p) < 0$. Then, Assumption 7.1.e is needed.

There exist infinite configurations of the system in which the state of the network is admissible. Indeed, recall that states might be negative, too, and the definition of admissibility considers differences of values, instead of absolute values. The set of all the admissible states of the network is denoted by $\mathcal{O} \subset \mathbb{Z}^n$.

## 8.2 Dynamics of the state of the network

Tokens movements in a network can be described by the corresponding sequence of elementary transitions, from the fast dynamic perspective. The movements of tokens, however, change the state of the network.

Consider a network whose state is $x$ and assume that a token is just injected in source node $i \in \mathcal{S}$ at time $t_k = t_k^0$: the immediate effect of this operation is to increase (possibly temporarily) the state $x_i$ of node $i$ by 1, as a new token is present in there, i.e., $x_i \leftarrow x_i + 1$. To describe this behavior, the input vector $\nu = [\nu_1, \nu_2, \dots, \nu_n]^\top \in \mathbb{N}^n$ is introduced, whose generic component $\nu_h$, associated with node $h \in \mathcal{N}$, is defined as

$$\nu_h = \begin{cases} 1, & \text{if the injection occurs at node } h, \\ 0, & \text{otherwise.} \end{cases}$$

Recalling the assumption that only one token at a time can enter the network and move in-between two time instants of the slow dynamic, it must be $\sum_h \nu_h \in \{0, 1\}$ and, in particular, $\sum_h \nu_h = 1$ if and only if there is a token injection, and $\sum_h \nu_h = 0$ otherwise. Let $e_i \in \mathbb{Z}^n$ be the $i$th canonical basis vector, i.e., the vector whose components are 1 at the $i$th position, and 0 otherwise. Then, if a token is injected in node $i$, it must be $\nu = e_i$ (otherwise, it is $\nu = [0, \dots, 0]^\top$). Consequently, $x_h \leftarrow x_h + \nu_h$ for any node $h \in \mathcal{N}$, that is $x \leftarrow x + \nu = x + e_i$.

Now, consider a network whose state is $x$ and assume that a token performs an elementary transition from node $i \in \mathcal{N}$ to non-sink node $j \setminus \mathcal{T} \in \mathcal{N}$ at a generic time $t_k^r$ of the fast dynamic: the immediate effect of this operation is to decrease the state $x_i$ of node $i$ by 1 (the token has just left this node) and increase (possibly temporarily) the state $x_j$ of node $j$ by 1, as a new token is present in there, i.e., $x_i \leftarrow x_i - 1$ and $x_j \leftarrow x_j + 1$. Again, the token is necessarily the only one to move in the network at that time. To describe this behavior, the control vector $u \in \mathbb{N}^m = [u_1, u_2, \dots, u_m]^\top$ is introduced, whose generic component $h$, associated with arc $h = (i, j) \in \mathcal{A}$, is defined as

$$u_h = u_{ij} = \begin{cases} 1, & \text{if the elementary transition occurs along arc } h = (i, j), \\ 0, & \text{otherwise.} \end{cases}$$

Again, $\sum_h u_h = 1$ if and only if there is an elementary transition, and $\sum_h u_h = 0$ otherwise. Then, if an elementary transition occurs from node $i$ to non-sink node $j$ along arc $h = (i, j)$, it must be $u = e_h$ (otherwise, it is $u = [0, \dots, 0]^\top$). Consequently, $x_i \leftarrow x_i - u_{ij}$ and $x_j \leftarrow x_j + u_{ij}$ for any node arc $h \in \mathcal{A}$.

If $j \in \mathcal{T}$ is a sink node, it must have a zero state anytime by definition, i.e., $x_j \equiv 0$, even after an elementary transition moves a token in this sink node $j$: the token is simply instantaneously expelled from the network to have admissibility, i.e., $x_j \leftarrow (x_j + 1) - 1 = x_j = 0$. So, an elementary transition from node $i \in \mathcal{N}$ to sink node $j \in \mathcal{T}$ results in just $x_i \leftarrow x_i - 1$.

**Example 8.1:**
Consider the network in Fig. 7.2. Let $x(t_k)$ be the current state at time $t_k$. Assume that a token is injected at time $t_k$ in node 1 and travels a path $p = \{1, 2, 3\}$. The following elementary operations are performed:

- time $t_k = t_k^0$ (fast dynamic): the token is injected in node 1, hence $x_1 \leftarrow x_1 + 1$;

Table 8.1: Overall effects on the network state $x$ of all the network transition of a token moving from a source node $i$ up to a destination node $j$, along a given traveled path, in the unconstrained case. $e_i$ is the $i$th canonical basis vector.

| | DESTINATION | |
|---|---|---|
| ORIGIN | External environment ($j \in \mathcal{T}$) | A node in the network ($j \in \mathcal{N} \setminus \mathcal{T}$) |
| External environment ($i \in \mathcal{S}$) | $x \leftarrow x$ | $x \leftarrow x + e_j$ |
| Already in the network ($i \in \mathcal{N}$) | $x \leftarrow x - e_i$ | $x \leftarrow x - e_i + e_j$ |

- time $t_k^1$ (fast dynamic): the token moves to node 2, hence $x_2 \leftarrow x_2{+}1$ and $x_1 \leftarrow x_1{-}1$; this latter operation cancels the state variation in node 1, which returns to the value it had before the token injection;
- time $t_k^2$ (fast dynamic): the token moves to node 3, hence $x_3 \leftarrow x_3{+}1$ and $x_2 \leftarrow x_2{-}1$; this latter operation cancels the state variation in node 2, which returns to the value it had before the token injection; the node has reached its destination.

Considering the complete chain of elementary transitions, only the state of node 3 has been modified: $x(t_{k+1}) = x(t_k) + e_3$.

So far, only single elementary transitions or single token injections have been considered, from the perspective of the fast dynamic. The whole chain of operations can be modeled from the perspective of the slow dynamic. Overall, the state of the network obtained after the token has reached its destination and stopped moving is well-defined:

- if a token is injected (from the external environment) in source node $i \in \mathcal{S}$ and moves up to a non-sink destination node $j \in \mathcal{N} \setminus \mathcal{T}$ where it stops moving and is deposited, only the state of node $j$ is affected, so that the network state is updated as $x \leftarrow x + e_j$;
- if a token is injected (from the external environment) in source node $i \in \mathcal{S}$ and moves up to sink destination node $j \in \mathcal{T}$ where it is expelled (to the external environment), the network state does not change: $x \leftarrow x$;
- if a token already present in the network in any node $i \in \mathcal{N}$ moves up to a non-sink destination node $j \in \mathcal{N} \setminus \mathcal{T}$ where it stops moving and is deposited, both the states of nodes $i$ and $j$ are affected (the token is just transferred), so that the network state is updated as $x \leftarrow x - e_i + e_j$;
- if a token already present in the network in any node $i \in \mathcal{N}$ moves up to sink destination node $j \in \mathcal{T}$ where it is expelled (to the external environment), only the state of node $i$ is affected, so that the network state is updated as $x \leftarrow x - e_i$.

Note that the overall state variation due to the traveling of a token does not depend on the traveled path, but only on the origin of the token (i.e., whether it was injected from the external environment or it was already present in the network) and its destination (i.e., whether it stops in a non-sink node where it is deposited, or it reaches a sink node where it is expelled from the network). The above-mentioned results are summarized in Table 8.1.

Consequently, it is not necessary to temporize the fast dynamic explicitly to study the system: only the slow dynamics is therefore temporized, by assuming a change at time $t_k$, $k = 0, 1, 2, \dots$ whenever a round of fast dynamics (a network transition) completes and possibly a new token is injected. Indeed, whenever a new token is to be moved at time $t_k$ of the slow dynamic, after a network transition the state will be updated as detailed in Table 8.1. Then, the next token transition at $t_{k+1}$ can be considered. Still, a policy must be specified to determine controls $u_{ij}$ and the actual destination of the moving token, i.e., the sequence of its elementary transitions.

The input and the control can be redefined from the slow dynamic perspective, by taking into account all the elementary transitions that occurred during a network transition. Hereinafter, they will be specified as

$$u(k) = \sum_r u(t_k^r) \quad \text{and} \quad \nu(k) = \nu(t_k^0),$$

where the relations hold component-wise. Note that the second relation follows from the assumption that a token injection may occur only at $t_k = t_k^0$. Then, the slow dynamics of network $\mathcal{G}$ is specified for each node by the following state equation

$$x_i(k+1) = \begin{cases} x_i(k) - \sum_{j \in \mathcal{N}_i} u_{ij}(k) + \sum_{j : i \in \mathcal{N}_j} u_{ji}(k) + \nu_i(k), & \text{if } i \notin \mathcal{T} \text{ (non-sink node)}, \\ 0, & \text{if } i \in \mathcal{T} \text{ (sink node)}. \end{cases} \tag{8.2}$$

This can be compactly written as

$$x(k+1) = x(k) + Bu(k) + v(k), \tag{8.3}$$

where $B$ is the generalized incidence matrix[1] of network $\mathcal{G}$ whose rows associated with the sink nodes are multiplied by 0 (i.e., sink nodes are treated as external environment). Equations of this type have been used in the context of flow network and manufacturing [127, 24].

## 8.3 Decentralized transition rule

The control $u_{ij}$ introduced in the previous Section specifies whether an elementary transition along arc $(i, j)$ is permitted and occurs, or not. The following simple *transition rule* is introduced, which specifies the network transition of a moving token: a token injected in a source node $i$ of a network in state $x$ undergoes a sequence of elementary transition from node to node until it remains above-threshold in the nodes that it reaches; eventually, either it reaches a node $j$ where it is under-threshold or it reaches a sink node where it leaves the network.

First, recall that the elementary transitions in the fast dynamics are not temporized. Then, under Assumption 7.4, the *decentralized threshold policy* can be formalized as follows.

**Policy 8.1: Decentralized threshold policy (unconstrained system).**
Consider the following conditions for the control $u_{ij}$ specifying the policy for each arc $(i, j) \in \mathcal{A}$:

a) at most a token can enter a node, i.e.,

$$\sum_{j:i\in\mathcal{N}_j} u_{ji} + \nu_i \leq 1, \ \forall i \in \mathcal{N}; \tag{8.4a}$$

b) at most a token can leave a node, i.e.,

$$\sum_{j\in\mathcal{N}_i} u_{ij} \leq 1, \ \forall i \in \mathcal{N}; \tag{8.4b}$$

c) a transition along an arc $(i, j) \in \mathcal{A}$ may occur only if the arrival of a token in $i$ makes the difference between the number of tokens present in $i$ and in $j$ exceed the value of the arc cost $\gamma_{ij}$, i.e., only if

$$x_i + \sum_{l:i\in\mathcal{N}_l} u_{li} + \nu_i - x_j > \gamma_{ij}. \tag{8.4c}$$

Then, the control is

$$u_{ij} = \begin{cases} 1, & \text{if Eq. (8.4a) and Eq. (8.4b) and Eq. (8.4c) hold,} \\ 0, & \text{otherwise.} \end{cases} \tag{8.5}$$

Conditions a) and b) in Policy 8.1 simply means that each node can process only one token at a time: it can either receive a single token and/or move it out, or do nothing. In other words, the same node can process multiple tokens between two time instants of the slow dynamic, but not simultaneously. However, it will be shown that the proposed policy ensures that no token can return to the same node (assuming the network does not change).

In condition c), recall that at each instant of the slow dynamic $t_k$, initially (at $t_k^0$) the arrival of a token is possible only if this is injected from the external environment, so that $\sum_{l:i\in\mathcal{N}_l} u_{li} = 0$ in Eq. (8.4c) at the beginning of each network transition. Later on, it can be non-zero: assuming, for instance, that an elementary transition occurs from node $l$ to node $i$, i.e., $u_{li} = 1$, to test the condition for the successive elementary transition condition for the arc $(i, j)$, $\sum_{l:i\in\mathcal{N}_l} u_{li}$ will be non-zero in Eq. (8.4c).

**Example 8.2:**
Reconsider the network in Fig. 7.2. Let $x(k) = [2, 1, 0, 0, 0]^\top$ be the current state at time $t_k$. Recall that $\gamma_{1,2} = \gamma_{2,3} = \gamma_{3,4} = 1$, $\gamma_{1,3} = 4$ and $\gamma_{2,4} = 3$ and assume that a token is injected at time $t_k$ in node 1. The following elementary operations are performed:

- initially, the token is injected in node 1, hence $\nu_1 = 1$ and $\sum_{l:1\in\mathcal{N}_l} u_{l1} = u_{4,1} = 0$; then, considering arc

[1]The incidence matrix is a $n \times m$ matrix whose columns describe the arcs of the network, and the rows the nodes; the generic column $B_h$ associated with arc $h = (i, j)$ has only two non-zero elements: $B_{h,i} = -1$ and $B_{h,j} = 1$; arcs connecting the nodes of the network with the external environment have only one non-zero entry (see, e.g., Example 8.2; refer to subsection 14.1.3 at Page 140 for more details).

$(1,2)$, condition Eq. (8.4c) becomes $x_1+u_{4,1}+\nu_1-x_2=2+0+1-1=2>\gamma_{1,2}=1$, so that $u_{1,2}=1$ and the transition is allowed: the token moves to node 2. Note that the transition along arc $(1,3)$ was not possible, because condition Eq. (8.4c) did not hold, as $x_1+u_{4,1}+\nu_1-x_3=2+0+1-0=3<\gamma_{1,3}=4$;

- then, considering arc $(2,3)$, condition Eq. (8.4c) becomes $x_2+u_{1,2}+\nu_2-x_3=1+1+0-0=2>\gamma_{2,3}=1$, so that $u_{2,3}=1$ and the transition is allowed: the token moves to node 3. Note that the transition along arc $(2,4)$ was not possible, because condition Eq. (8.4c) did not hold, as $x_2+u_{1,2}+\nu_2-x_4=1+1+0-0=2<\gamma_{2,4}=3$;
- the network transition is completed, because a transition along arc $(3,4)$ is not possible, since condition Eq. (8.4c) does not hold, as $x_3+(u_{2,3}+u_{1,3})+\nu_3-x_4=0+(1+0)+0-0=1=\gamma_{3,4}=1$.

Letting the arcs $1=(1,2),\ 2=(1,3),\ 3=(2,3),\ 4=(2,4),\ 5=(3,4),\ 6=(4,1),\ 7=(4,5)$, for this network transition it holds that

$$u(k)=\begin{bmatrix}1 & 0 & 1 & 0 & 0 & 0 & 0\end{bmatrix}^\top,$$

and

$$\nu(k)=\begin{bmatrix}1 & 0 & 0 & 0 & 0\end{bmatrix}^\top.$$

Also, the generalized incidence matrix $B$ of the network where the sink node is treated as external is

$$B=\begin{bmatrix}-1 & -1 & 0 & 0 & 0 & 1 & 0\\ 1 & 0 & -1 & -1 & 0 & 0 & 0\\ 0 & 1 & 1 & 0 & -1 & 0 & 0\\ 0 & 0 & 0 & 1 & 1 & -1 & -1\\ 0 & 0 & 0 & 1 & 0 & 0 & 0\end{bmatrix}.$$

Considering the complete chain of elementary transitions, only the state of node 3 has been modified. Indeed,

$$x(k+1)=x(k)+Bu(k)+\nu(k)=\begin{bmatrix}2\\1\\0\\0\\0\end{bmatrix}+\begin{bmatrix}-1\\0\\1\\0\\0\end{bmatrix}+\begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix}=\begin{bmatrix}2\\1\\1\\0\\0\end{bmatrix}=x(k)+e_3.$$

Policy 8.1 has the following properties:

- it is decentralized, as each transition is decided on the basis of local information, i.e., $u_{ij}$ for arc $(i,j)$ depends only on $\nu_i,x_i,x_j,\gamma_{ij}$, for all $j\in\mathcal{N}_i$;
- it makes all the elementary transitions that may occur happen: a token continues moving as long as Eqs. (8.4a) to (8.4c) continue holding; if the token reaches a node where it is under-threshold, the network transition is complete for that token.
- it imposes that all the routes followed by the moving tokens are paths (and not walks). More formally, the subnetwork of $\mathcal{G}$ induced by the arcs $(i,j)$ such that $u_{ij}=1$:
  - includes no circuits, due to Eqs. (8.4a), (8.4c) and (8.5) and Assumption 7.1.c on the absence of non-positive circuits in $\mathcal{G}$ (see proof of the next Theorem 8.1 in Appendix B);
  - is composed of not-intersecting paths, due to Eqs. (8.4a) and (8.4b).
- in presence of arcs with negative $\gamma_{ij}<0$, negative states may be reached. All the results provided so far, as well as those reported next, hold without changes. Indeed, in Policy 8.1, condition Eq. (8.5) does depend on difference of states, rather than their absolute values.

**Remark 8.1:**
If the state $x$ of the network is currently admissible at time $t_k$ and no new token is injected in the network, by Policy 8.1, no token moves. Indeed, at time $t_k$, for any node $i\in\mathcal{N}$, $\nu_i=0$ and there is no token arrival from other nodes initially, so that $\sum_{l:i\in\mathcal{N}_l}u_{li}=0$. Then, condition Eq. (8.4c) reduces to $x_i-x_j>\gamma_{ij}$, which is false for all $j\in\mathcal{N}_i$, by definition of admissible state. Hence, $u_{ij}=0$ for all arcs $(i,j)\in\mathcal{A}$.

If the state $x$ of the network is currently admissible at time $t_k$ and a new token is injected in source node $i\in\mathcal{S}$, by Policy 8.1, this is the only one to move and "instantaneously" reaches as destination either a node from which no transition is possible or a sink node (see the next Theorem 8.1). Indeed, the same findings of the previous case apply; the only difference is that now, just for the source node $i$ in which the token is injected, $\nu_i = 1$. Then, condition Eq. (8.4c) reduces to $x_i + 1 - x_j > \gamma_{ij}$, which may be true or false:

- if the token is below-threshold in $i$ (i.e., $x_i - x_j < \gamma_{ij}$ for all $j \in \mathcal{N}_i$), Eq. (8.4c) is false and the token stops.
- if the token is at-threshold in $i$ (i.e., $x_i - x_j = \gamma_{ij}$ for some $j \in \mathcal{N}_i$), Eq. (8.4c) is true and the token moves to node $j$. Once in node $j$, the process is repeated, recalling that now $\nu_j = 0$, but $u_{ij} = 1$.

Formally, it holds that

$$x(k+1) = \begin{cases} x(k), & \text{if the injected token reach a sink,} \\ x(k) + e_j, & \text{if the injected token reaches node } j \notin \mathcal{T}. \end{cases}$$

Finally, if the state $x$ of the network is currently not admissible at time $t_k$ and no new token is injected in the network, by Policy 8.1, a token always starts moving, despite $\nu_i = 0$ for all $i \in \mathcal{N}$ and $\sum_{l:i\in\mathcal{N}_l} u_{li} = 0$. Indeed, there exists at least one arc $(i, j)$ which is above-threshold ($x_i - x_j > \gamma_{ij}$) and condition Eq. (8.4c), which reduces to $x_i - x_j > \gamma_{ij}$, is automatically satisfied. So a network transition is triggered, which results in

$$x(k+1) = \begin{cases} x(k) - e_i, & \text{if the moving token reach a sink,} \\ x(k) - e_i + e_h, & \text{if the moving token reaches node } h \notin \mathcal{T}. \end{cases}$$

Recall that by assumption, if a token already present in the network can move, it takes priority over the injected ones.

## 8.4 Properties of the state of the system

Hereinafter, a system composed by a network $\mathcal{G}$ governed by dynamics described by Policy 8.1 and state Eq. (8.2) will be denoted by $(\mathcal{G}, f)$. The properties of the state of this system are reported next.

### 8.4.1 Special admissible states

A special subset of the set of admissible states $\mathcal{O}$ is the *rest set* $\partial\mathcal{O} \subseteq \mathcal{O}$, which contains all the fixed points of Eq. (8.2), i.e., all the states $x$ which are not modified: $x(k+1) = x(k)$. This might happen only for admissible states: if $x$ were non-admissible, a token already in the network would trigger a network transition which inevitably modifies the state, as the traveled path cannot be a circuit, see Table 8.1. Then, such network transitions are necessarily triggered by a token injection, and result in the token reaching a sink node and leaving the network.

**Definition 8.3:**
The admissible state $x \in \mathcal{O}$ is a *rest state* if it belongs to the rest set $x \in \partial\mathcal{O}$, i.e., if there there exists *at least one source node* $i \in \mathcal{S}$ such that injecting a token in there ($v(k) = e_i$) leaves the state unchanged: if $x(k) = x$ then $x(k+1) = x$.

The rest state $\bar{x} \in \partial\mathcal{O}$ is said a *global rest state* if injecting a token *in any source node in* $\mathcal{S}$, no matter which, leaves the state unchanged.

A *maximal rest state* $\bar{\bar{x}}$ is a global rest state with the property that injecting a token *in any node in* $\mathcal{N}$ (roughly assuming that any node can become a source) leaves the state unchanged.

In general, there are multiple rest states and global rest states $\bar{x}$. Also, each global rest state is a steady-state, since with it the number of tokens stopped in the network nodes cannot change:

$$V(\bar{x}) = \bar{V}.$$

If the system reaches a global rest state, it is guaranteed that all the tokens injected in the sources reach a sink node and leave the network. To solve Problem 7.1, Policy 8.1 must drive the state Eq. (8.2) of $(\mathcal{G}, f)$ to a global rest state.

Instead, it will be shown later in subsection 8.5.1 that the *maximal rest state* $\bar{\bar{x}}$ is unique. Several properties of this special global rest state will also be pointed out.

### 8.4.2 Networks in an admissible state

Here, the case in which the initial state $x$ of the network is admissible is considered.

The first result is that any admissible state is bounded, as stated by the next Lemma.

**Lemma 8.2: Boundedness of the admissibility states**
In a system $(\mathcal{G}, f)$ with $x(k) \in \mathcal{O}$, it holds that

$$V(x(k)) \leq \bar{\gamma}\frac{n(n-1)}{2}, \forall k > 0.$$

Another important result is that for any admissible state $x(k) \in \mathcal{O}$, a new well-defined admissible state $x(k+1) \in \mathcal{O}$ is reached when applying Policy 8.1 (well-posedness and positive invariance), as stated by the next Theorem. In other words, once the state is admissible, it remains admissible, unless the network is modified.

**Theorem 8.1: Well posedness and positive invariance**
A system $(\mathcal{G}, f)$ initialized with $x(0) \in \mathcal{O}$ is well posed and the set $\mathcal{O}$ of the admissible states is positively invariant and finite. In particular, for each time $k \geq 0$, either $x(k+1) = x(k)$ or $x(k+1) = x(k) + e_j$ for some $j \in \mathcal{N}$.

Note that, in general, no bound can be expressed for the number $\bar{k}$ of tokens needed to reach a global rest state, unless further assumptions are made. Indeed, assume that there are multiple source nodes: if the tokens are all injected in only one source, a global rest state might never be reached. Moreover, any new token injected in the sources whose state has already converged (as well as the state of the nodes of the outgoing paths from these sources) contributes to increasing the value of $\bar{k}$, but not the value of $V(x)$. Indeed, $\bar{k}$ depends on when, how often and how regularly new tokens are inserted in each source.

A persistent token injection must be assumed to claim some results.

**Definition 8.4:**
The injection of tokens in a source is said *persistent* if the injected tokens number is unbounded.

Then, for any initial state $x(0) \in \mathcal{O}$, a global rest state is reached after the insertion of a finite number of tokens, if this occurs in all the source nodes. A bound can be provided assuming a single-source network.

**Theorem 8.2:**
A system $(\mathcal{G}, f)$ initialized with $x(0) \in \mathcal{O}$ always reaches a rest state in $\partial\mathcal{O}$ in finite time $\bar{k}$ if at each time $k = 0, \dots, \bar{k}-1$ a new token is injected into the network (under persistent injection). In particular, the system reaches a global rest state if a sufficiently high number of tokens is inserted in *all* the source nodes. For a single-source network, the maximum number of new tokens that need to be inserted to reach a global rest state is

$$\bar{K} = \bar{\gamma}\frac{n(n-1)}{2} - V(x(0)),$$

where $V(x(0))$ is the initial number of tokens present in the buffers of the network nodes, i.e., $\bar{k} \leq \bar{K}$.

#### Some considerations about the complexity

The time to reach a global rest state, measured in number of injected tokens, is exponential in the size of the problem input [2].

### 8.4.3 Networks in a non-admissible state

Here, the case in which the initial state $x$ of the network is not admissible is considered. The same Policy 8.1 under Assumption 7.4 is applied. However, due to the definition of non-admissible state, one or more arcs satisfy the above-threshold condition Eq. (8.4c) (even if there is no token injection): hence, multiple elementary transitions involving tokens already present in the buffer of the nodes are possible. Recall that to ensure that only one token at a time can move in the network, as required by Assumption 7.4, a priority criterion is adopted, prioritizing the traveling of the tokens already present in the network (see Section 7.3).

The non-admissible state condition is, however, only temporary if no new token is injected: the system $(\mathcal{G}, f)$ will eventually reach a state in $\mathcal{O}$ in finite time. Then, the results from subsection 8.4.2 can be applied.

**Theorem 8.3: Attractiveness of $\mathcal{O}$**
A system $(\mathcal{G}, f)$ initialized with $x(0) \notin \mathcal{O}$ always reaches a state in $\mathcal{O}$ in finite time if no new token is injected into the network.

## 8.5 Optimality of the traveled paths

In this Section a system $(\mathcal{G}, f)$ in an admissible state is considered; this is not a limitation because, by Theorem 8.3, an admissible state in $\mathcal{O}$ is reached anyway in finite time if the state is not admissible.

Then, an (injected) token moving from node $i \in \mathcal{N}$ to some other node $j \in \mathcal{N}$ (not necessarily a sink), always reaches such node moving along the shortest path (or one of the shortest paths in case of multiple possibilities) with respect to the costs $\gamma_{ij}$.

**Theorem 8.4: Shortest path**
A system $(\mathcal{G}, f)$ in an admissible state $x(k) \in \mathcal{O}$ at time $k$ is given. If $v(k) = e_i$ for some $i \in \mathcal{N}$ the injected token reaches its destination node $j$ following the unconstrained shortest path from $i$ to $j$.

In the special case in which the reached node is a sink, i.e., $j \in \mathcal{T}$, the optimality of the outgoing path followed by the injected token is ensured.

**Corollary 8.1: Shortest outgoing path**
A system $(\mathcal{G}, f)$ in an admissible state $x(k) \in \mathcal{O}$ at time $k$ is given. If a token injected in a source node $i \in \mathcal{S}$ reaches a sink node $j \in \mathcal{T}$ along the (shortest unconstrained) path $p$, then

- $j$ is among the closest sink nodes to $i$, i.e., there exists no $j' \in \mathcal{T}$ such that $L(p') < L(p)$, where $p'$ is the shortest path from $i$ to $j'$;
- $x_i(k) = L(p)$;
- in general, $x_h(k) = L(p_h)$, where $p_h$ is the sub-path of $p$ starting from node $h$ and reaching $j$, for all the nodes $h \in p$ of the path.

This is the most important result of this Part: despite a very simple decentralized policy is adopted to route the memoryless agents using only local information, in the long run, the shortest paths emerge, and each new injected token can eventually leave the network traveling along them.

### 8.5.1 Maximal rest state

The concept of *maximal rest state* can now be better specified as the (unique) state in which the amount of tokens in each node is equal to the minimum length to the closest sink.

**Definition 8.5: Alternative equivalent definition**
The *maximal rest state* $\bar{\bar{x}}$ of a system $(\mathcal{G}, f)$ is the unique global rest state defined as

$$\bar{\bar{x}} = \{\bar{\bar{x}}_i = L(p_i) : i \in \mathcal{N}\},$$

where $L(p_i)$ is the length of the shortest path from node $i$ to its closest sink node. If $i \in \mathcal{T}$, its "closest" sink node is the very $i$, resulting in $\bar{\bar{x}}_i = L(p_i) = 0$, in accordance with the definition of sink node.

If the network is in the maximal rest state, a new token injected in *any node* $i \in \mathcal{N}$ *of the network* immediately reaches a sink node following the shortest path of length $L(p_i)$. Recall that if the network were in a global rest state, this statement would be true only for new tokens injected in *any source node* $i \in \mathcal{S} \subseteq \mathcal{N}$ *of the network*.

The maximal rest state $\bar{\bar{x}}$ has the following important properties, which follows from Corollary 8.1:

- $\bar{\bar{x}}$ depends only on the network structure, as the shortest paths between nodes are properties of the network;
- $x \leq \bar{\bar{x}}$ component-wise, for all admissible states $x \in \mathcal{O}$;
- $\bar{\bar{x}}$ is a global rest state for any source set $\mathcal{S}$;
- for any global rest state $\bar{x}$ of a network, $\bar{x} \leq \bar{\bar{x}}$ component-wise and $\bar{x}_i = \bar{\bar{x}}_i$ for all nodes along at least one shortest path connecting each source to the closest sink.

### 8.5.2 Multiple source nodes, sink nodes, and shortest outgoing paths

The results presented so far do not make any particular assumption on the number of source nodes, sink nodes, and possible shortest paths between them (even for a single-source-single sink network). Here, the effects of having these conditions are summarized.

Having multiple source nodes simply means that tokens are injected in more than one node. A global rest $\bar{x}$ will be eventually reached under persistent injection in all the sources, and, since any global rest state is upper bounded by the maximal rest state $\bar{\bar{x}}$, generally the number of nodes $i \in \mathcal{N}$ for which $\bar{x}_i = \bar{\bar{x}}_i$ increases as the number of sources increases. The main exception is the case in which these new sources are placed along the shortest paths between an existing source and the closest sink.

As already discussed, having multiple sink nodes simply means that tokens have multiple points to leave the network. Eventually, each token injected in a given source will reach the closest sink to that source, i.e., it will leave the network along the shortest path possible, once a global rest state is reached.

Note that a multiple-sinks network can be transformed into a single-sing network, by transforming all the sink nodes $i \in \mathcal{T}$ into non-sink nodes, and connecting them to a new single external sink node $O$ by some arcs $(i, O)$, with $\gamma_{iO} = 0$ (and possibly $\sigma_{iO}$), so that the shortest paths of this new network have not changed; then, all the injected tokens exit through sink node $O$ along the shortest path from $i$ to $O$, which automatically passes through the original sink node that was closer to $i$.

**Remark 8.2:**
A special network is the one where the nodes $i \in \mathcal{N}$ are partitioned into sources and sinks, i.e., such that $\mathcal{S} \cup \mathcal{T} = \mathcal{N}$ with $\mathcal{S} \cap \mathcal{T} = \varnothing$; there is only one unique global rest state, which coincides with the maximal rest state $\bar{\bar{x}}$. If the assumption $\mathcal{S} \cap \mathcal{T} = \varnothing$ is dropped, the same holds: in this case, a token injected in a sink node is immediately expelled from the network.

Finally, when there are multiple shortest outgoing paths from node $i$ (even to different sinks), these have all the same (minimal) length, so the maximal rest state $\bar{\bar{x}}_i$ is still unique. If the system is in the maximal rest state, all these multiple shortest paths are active and potentially available for new tokens; conversely, if the system is in a generic global rest state, at least one is active, but not necessarily all of them. Note that if multiple outgoing shortest paths exist and are traversable by the tokens, it means that for some nodes there are necessarily multiple outgoing arcs which are all above threshold when a token arrives in there. Depending on the policy of choice for the next node to visit in case of multiple arcs above threshold, some of these shortest paths may never be traversed.

**Example 8.3:**
Consider the network in Fig. 8.1 with three sources $s_1, s_2, s_3$ and four sinks $d_1, d_2, d_3, d_4$, and $\gamma_{i,j} = 1$ for all arcs $(i, j)$, and assume the state of the network is a global rest state. All the tokens injected in source node $s_1$ reach the sink $d_2$ through the independent unique shortest path highlighted in red. From source node $s_2$ three possible outgoing shortest paths exist to nodes $d_3, d_4$, which are highlighted in blue. Assume that $x_i = \bar{\bar{x}}_i$ for all the nodes along these shortest paths. A policy must be specified to determine which path to take, at nodes $s_2$ and $i$. All the tokens injected in source node $s_3$ join one of the possible shortest paths from node $s_2$, reaching $d_3$ to leave the network. No token is collected at sink node $d_1$.

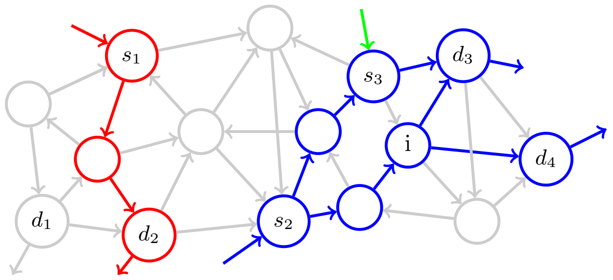


Figure 8.1: A simple network $\mathcal{G}$ with three sources $s_1, s_2, s_3$ and four sinks $d_1, d_2, d_3, d_4$. All the arc costs are assumed unitary, $\gamma_{ij} = 1$. The shortest paths from each source are highlighted. The highlighted nodes have a state equal to the corresponding maximal rest state components $x_i = \bar{\bar{x}}_i$.

## 8.6 Dynamic networks

In this Section, the case in which Policy 8.1 is adopted in a scenario in which the network configuration is dynamic is considered. In particular, the following are assumed time-varying: the topology of network $\mathcal{G}(k) = (\mathcal{N}(k), \mathcal{A}(k))$, the cost of each arc $\gamma_{ij}(k)$ for all $(i, j) \in \mathcal{A}(k)$, the set of the source nodes $\mathcal{S}(k)$ and the one of the sink nodes $\mathcal{T}(k)$. It turns out that the proposed policy is adaptive.

Under the assumption that Assumption 7.1 continue to hold for all time $k \geq 0$, once the configuration of the network changes, the "initial state" of the new configuration of the system will be non-zero, in general. Still, the results presented so far are independent from the initial state. Indeed, by Theorems 8.3 and 8.4, whenever a network changes its configuration, if the new initial state is not admissible, first an admissible state will be reached, and then a new global rest state will be reached too, achieving the optimality in the new scenario. An advantage of Policy 8.1 is that the transient among different configurations is, in general, fast compared to the case in which the optimization restarts from zero, since the results from previous optimization phases are exploited.

A network can change from $k$ to $k+1$, due to a connection (arc) which becomes active or inactive, or by a change of cost or by a node, possibly a source or a sink, which becomes active/inactive. Because of the modifications on the network, the admissibility of the state $x(k)$, the set $\mathcal{O}$, the rest set $\partial\mathcal{O}$ and maximum rest state $\bar{\bar{x}}$ can change, too. In any case, it is assumed that Assumption 7.1 continue holding.

To simplify, the following notation is used next:

- At time $k$, before the modification: $\mathcal{G} = \mathcal{G}(k)$, $\mathcal{N} = \mathcal{N}(k)$, $\mathcal{A} = \mathcal{A}(k)$, $\gamma_h = \gamma_h(k)$, $\mathcal{S} = \mathcal{S}(k)$, $\mathcal{T} = \mathcal{T}(k)$, $x = x(k)$, $\bar{\bar{x}} = \bar{\bar{x}}(k)$, and so on;

- At time $k+1$, after the modification: $\mathcal{G}' = \mathcal{G}(k+1)$, $\mathcal{N}' = \mathcal{N}(k+1)$, $\mathcal{A}' = \mathcal{A}(k+1)$, $\gamma'_h = \gamma_h(k+1)$, $\mathcal{S}' = \mathcal{S}(k+1)$, $\mathcal{T}' = \mathcal{T}(k+1)$, $x' = x(k+1)$, $\bar{\bar{x}}' = \bar{\bar{x}}(k)$, and so on.

Below, the above-mentioned simple modifications on the network involving a single node $\hat{\imath} \in \mathcal{N}$ or arc $\hat{h} = (\hat{\imath}, \hat{\jmath}) \in \mathcal{A}$ are briefly analyzed in more detail. Then, it is easy to generalize to simultaneous modifications, as the effects are overlapped.

1. **Failure of an arc**: arc $\hat{h} = (\hat{\imath}, \hat{\jmath}) \in \mathcal{A}$ is removed (disabled) from $\mathcal{G}$, thus $\mathcal{A}' = \mathcal{A} \setminus \{\hat{h}\}$.

2. **Increasing a cost**: the cost of arc $\hat{h} = (\hat{\imath}, \hat{\jmath}) \in \mathcal{A}$ is increased by $\Delta_{\hat{h}} > 0$, thus $\gamma'_{\hat{h}} = \gamma_{\hat{h}} + \Delta_{\hat{h}}$.

   Firstly, note that Item 1 can be seen as a special case of Item 2, assuming $\Delta_{\hat{h}} = +\infty$: the cost of $\hat{h}$ becomes so large that it is like the arc does not exist anymore.

   In both Items 1 and 2, the new admissible state set $\mathcal{O}'$ is a superset of $\mathcal{O}$: $\mathcal{O}' \supseteq \mathcal{O}$, so any admissible state $x$ remains such. This holds because $x_i - x_j \leq \gamma_{ij} \leq \gamma'_{ij}$ for all $(i, j) \in \mathcal{A}$.

   A non-admissible state $x$ might become admissible if the only arc that makes it such is $\bar{h}$. Conversely, $\partial\mathcal{O}$ may change.

   The maximum rest state $\bar{\bar{x}}'$ can only be greater or equal to $\bar{\bar{x}}$ component-wise, because for the paths (from any node in $\mathcal{N}$ to the closest sink) passing through $\hat{h}$, whose cost is increased, there might be new alternative shorter paths not passing through $\hat{h}$.

   In both cases, if $x \in \partial\mathcal{O}$, the newly injected tokens might need to find a new outgoing shortest path if $\hat{h}$ was in one of the paths followed by them prior to the modification and no alternative path is available yet, either because such path is no more available (Item 1) or its cost has increased too much (Item 2).

3. **Insertion of an arc** : arc $\hat{h} = (\hat{\imath}, \hat{\jmath}) \notin \mathcal{A}$ is added (re-enabled) to $\mathcal{G}$, thus $\mathcal{A}' = \mathcal{A} \cup \{\hat{h}\}$.

4. **Decreasing a cost**: the cost of arc $\hat{h} = (\hat{\imath}, \hat{\jmath}) \in \mathcal{A}$ is decreased by $\Delta_{\hat{h}} > 0$, thus $\gamma'_{\hat{h}} = \gamma_{\hat{h}} - \Delta_{\hat{h}}$.

   In both Items 3 and 4, the new admissible state set $\mathcal{O}'$ is a subset of $\mathcal{O}$: $\mathcal{O}' \subseteq \mathcal{O}$, so an admissible state may become not admissible if the involved arc $\bar{h}$ becomes not admissible. In this latter case, a transition time interval is needed to reach an admissible state in $\mathcal{O}'$.

   A non-admissible state $x$ cannot become admissible, as the arcs that are non-admissible remains and $x_i - x_j > \gamma_{ij} \geq \gamma'_{ij}$ for some $(i, j) \in \mathcal{A}$.

   The maximum rest state $\bar{\bar{x}}'$ can only be smaller or equal to $\bar{\bar{x}}$ component-wise, since there can be new alternative shortest paths (from any node in $\mathcal{N}$ to the closest sink) passing through $\hat{h}$.

5. **Adding a source**: node $\hat{\imath} \in \mathcal{N} \setminus \mathcal{S}$ becomes a source, thus $\mathcal{S}' = \mathcal{S} \cup \{\hat{\imath}\}$.

6. **Removing a source** : node $\hat{\imath} \in \mathcal{S}$ stops being a source, thus $\mathcal{S}' = \mathcal{S} \setminus \{\hat{\imath}\}$.

   For both Items 5 and 6, the admissible state set $\mathcal{O}'$ does not change, because the admissibility property does not depend on the location of the source nodes. Hence, if $x$ is admissible, $x'$ remains admissible, while if $x$ is non-admissible, $x'$ remains non-admissible.

   For the same reason, the maximum rest state $\bar{\bar{x}}'$ remains the same as $\bar{\bar{x}}$, too.

   Instead, the global rest set might change. In Item 5, $\partial\mathcal{O}' \subseteq \partial\mathcal{O}$: tokens injected in $\hat{\imath}$ might need to discover the new shortest outgoing paths, even if $x \in \partial\mathcal{O}$. This condition might be avoided if the new source $\hat{\imath}$ is located in one of the shortest outgoing paths followed by the tokens injected in $\mathcal{S}$. Conversely, in Item 6, $\partial\mathcal{O}' \supseteq \partial\mathcal{O}$: if $x \in \partial\mathcal{O}$, it also holds that if $x \in \partial\mathcal{O}'$ and the optimal steady-state is already reached.

7. **Adding a sink**: node $\hat{\imath} \in \mathcal{N} \setminus \mathcal{T}$ becomes a sink, thus $\mathcal{T}' = \mathcal{T} \cup \{\hat{\imath}\}$. The state $x_{\hat{\imath}}$, if positive, is depleted of the token stored in there and is forced to be $x'_{\hat{\imath}} = 0$, so $V(x) \geq V(x')$ (assuming $x_{\hat{\imath}} \geq 0$).

   An admissible state $x$ of $\mathcal{G}$ remains admissible if and only if, after setting $x_{\hat{\imath}} = 0$, all the arcs connected to $\hat{\imath}$ are admissible.

   The maximum rest state $\bar{\bar{x}}'$ can only be smaller or equal to $\bar{\bar{x}}$ component-wise, since any alternative shortest path (from any node in $\mathcal{N}$ to the closest sink) could reach the new sink $\hat{\imath}$, which is possibly closer.

   If $x \in \partial\mathcal{O}$, the newly injected tokens might need to find new shortest paths to $\hat{\imath}$, if these are shorter than the current optimal ones.

8. **Removing a sink**: node $\hat{\imath} \in \mathcal{T}$ stops being a sink, thus $\mathcal{T}' = \mathcal{T} \setminus \{\hat{\imath}\}$. Also, the state of $\hat{\imath}$ stops being forced to zero.

   This operation does not modify the admissibility property of state $x$ of $\mathcal{G}$.

   The maximum rest state $\bar{\bar{x}}'$ can only be greater or equal to $\bar{\bar{x}}$ component-wise, since any alternative shortest path (from any node in $\mathcal{N}$ to the closest sink) cannot reach sink $\hat{\imath}$ anymore.

   If $x \in \partial\mathcal{O}$, new injected tokens might need to find new shortest paths if $\hat{\imath}$ was in the closest sink they were reaching and no alternative optimal paths are available, yet.

9. **Failure of a node**: one node $\hat{\imath}$ is removed (disabled) from $\mathcal{G}$, which means that all the (ingoing or outgoing) arcs connected to that node $\hat{\imath}$ are disabled, too (see the effects of Item 1). Moreover, if that node $\hat{\imath}$ is a source or a sink node, such source or sink is also removed (see Items 6 and 8). Thus $\mathcal{N}' = \mathcal{N} \setminus \{\hat{\imath}\}$, $\mathcal{A}' \subseteq \mathcal{A}$, $\mathcal{S}' \subseteq \mathcal{S}$ and $\mathcal{T}' \subseteq \mathcal{T}$. Also, all the token stored in $\hat{\imath}$ are (possibly temporarily) lost, $V(x) \geq V(x')$ (assuming $x_{\hat{\imath}} \geq 0$).

10. **Insertion of a node**: node $\hat{\imath}$ is added (re-enabled) to $\mathcal{G}$, which means that all the (ingoing or outgoing) arcs connected to that node $\hat{\imath}$ are re-enabled, too (see the effects of Item 3). Moreover, if that node $\hat{\imath}$ was a source or a sink (before it was previously disabled), such source or sink is also re-enabled (see Items 5 and 7). Thus, $\mathcal{N}' = \mathcal{N} \cup \{\hat{\imath}\}$, $\mathcal{A} \subseteq \mathcal{A}'$, $\mathcal{S} \subseteq \mathcal{S}'$ and $\mathcal{T} \subseteq \mathcal{T}'$. Also, if $\hat{\imath}$ is re-enabled, all the tokens which were stored in $\hat{\imath}$ could possibly be restored.

    In both Items 9 and 10, the state vector $x$ changes space, losing/acquiring one component. The admissibility of a state is preserved after removal (Item 9), meaning that the new state (in the new space) is admissible; if $x \in \partial\mathcal{O}$, new injected tokens might need to find new shortest paths, if $\hat{\imath}$ was in one of the paths followed by them prior to the modification and no alternative path is available yet. The admissibility is not preserved in general after activating a node with new connections (Item 10): if $x \in \partial\mathcal{O}$, new injected tokens might find new shortest paths passing through $\hat{\imath}$.

A final point to be investigated is whether a persistently changing network can become indefinitely congested, i.e., if the number of tokens in the network can grow up to infinity. The following sufficient Assumption is introduced.

**Assumption 8.1:**

- At time $k$ an event can be either an injection of one token or a network reconfiguration. In the latter case, sufficient time is given before inserting new tokens so that the system can reach an admissible state.
- The costs of the arcs are non-negative (and upper bounded by $\bar{\gamma}$): $0 \leq \gamma_{ij} \leq \bar{\gamma}$, for all arcs $(i, j)$.
- Assumption 7.1 holds for any network configuration.
- The number of nodes of the network is upper bounded by a positive number $\bar{n}$.

Under the above Assumption, the network cannot become indefinitely congested, because, under the assumption of non-negative arc costs, the number of tokens in the network coincides with the value of $V(x)$ and can never exceed the value $\bar{\gamma}\bar{n}(\bar{n}-1)/2$.
Indeed, consider a change of configuration at a time $k'$. By assumption, $x(k')$ was admissible before the modification, and hence $V(x(k')) \leq \bar{\gamma}|\mathcal{N}(k')|(|\mathcal{N}(k')|-1)/2 \leq \bar{\gamma}\bar{n}(\bar{n}-1)/2$ (see Lemma 8.2). After a finite number of time instants, at some time $k'' \geq k'$, a new admissible state is reached, for which the same inequality reported above holds. This continues holding for any admissible state of any possible configuration. Note that, by Theorem 8.2, if $x(k')$ become not admissible after the change of configuration at a time $k'$, $V(x(k))$ cannot increase between $k'$ and $k''$. This means that the total number of tokens in the network is bounded.

## 8.7 Non-integer arc costs

So far it has been assumed that the arcs' costs $\gamma_{ij}$ have integer values. From an algorithmic perspective, this has advantages in terms of space efficiency, speed of arithmetic operations, and stability [81, 82]. Indeed, when working with non-integer arithmetic, numerical errors might be critical when applying the proposed threshold mechanism. However, in a more realistic scenario, non-integer arc costs $\widetilde{\gamma}_{ij}$ might be necessary. To continue using the proposed policy, these non-integer costs $\widetilde{\gamma}_{ij}$ have to be transformed to some integer costs $\gamma_{ij}$, so that the proposed policy is applied to the same network whose arcs have these integer costs.

A first trivial possibility is just rounding these costs, possibly after scaling them by a factor $\alpha \in \mathbb{Q}$, as

$$\gamma_{ij} = \text{round}(\alpha\widetilde{\gamma}_{ij}).$$

This rounding inevitably introduces some approximation errors that may change the distribution of the shortest paths [81]. The same applies when using the ceiling or the flooring functions [81]. Computing an integer approximation that does not topologically change the shortest paths is an NP-hard problem [82].

**Example 8.4: Approximating the non-integer arc costs as integers: rounding, flooring and ceiling**

Consider the network in Fig. 8.2 with non-integer arc costs $\widetilde{\gamma}_{ij}$. The *actual* (unique) shortest path is $p_3 = \{1, 2, 5, 4\}$, with $L(p_3) = 5.7$.

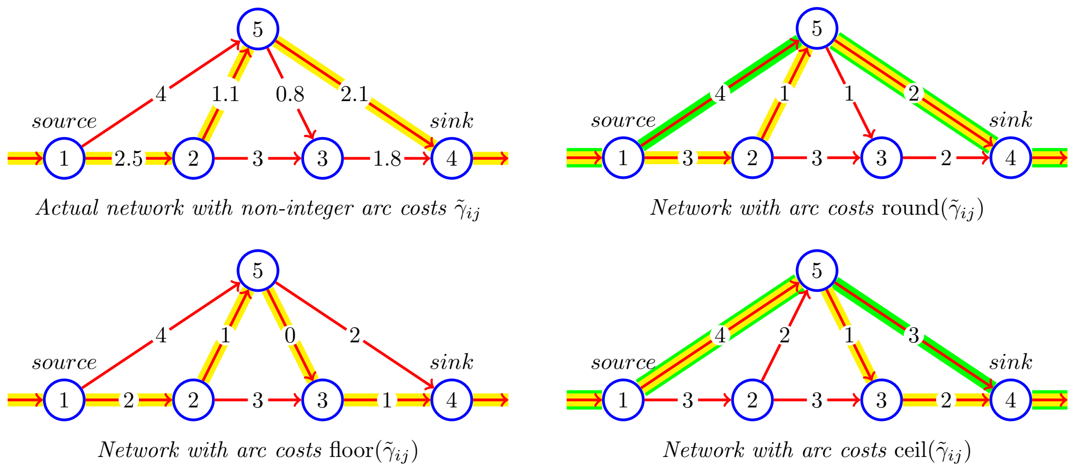

source
sink
1
2
3
4
5
4
1.1
0.8
2.1
2.5
3
1.8
Actual network with non-integer arc costs $\widetilde{\gamma}_{ij}$
4
1
1
2
3
3
2
Network with arc costs round($\widetilde{\gamma}_{ij}$)
4
1
0
2
2
3
1
Network with arc costs floor($\widetilde{\gamma}_{ij}$)
4
2
1
3
3
3
2
Network with arc costs ceil($\widetilde{\gamma}_{ij}$)

Figure 8.2: A network with non-integer arc costs $\widetilde{\gamma}_{ij}$. Different networks are obtained by approximating the arc costs to integers using the rounding, flooring, and ceiling functions. The shortest paths are highlighted.

When the network costs are rounded, a new shortest path $p_5 = \{1, 5, 4\}$ emerges, other than $p_3$, whose length increases to $L(p_3) = L(p_5) = 6$. When the network costs are approximated using the flooring function, the only shortest path is $p_2 = \{1, 2, 5, 3, 4\}$, with length $L(p_2) = 4$: $p_3$ is no longer an optimal path. Finally, when the network costs are approximated using the ceiling function, two new shortest paths $p_4 = \{1, 5, 3, 4\}$ and $p_5 = \{1, 5, 4\}$ emerge, with length $L(p_4) = L(p_5) = 7$: again, $p_3$ is no longer an optimal path. These results are summarized in Table 8.2. In all the three considered cases approximating the non-integer costs by simply rounding or taking the flooring or the ceiling, the distribution of the shortest paths changes, in terms of traversed arcs, number of paths, and path costs. While some shortest paths are eventually discovered, these are not the same as the ones of the original actual network.

Table 8.2: Length $L(p)$ for each possible path $p_1, p_2, p_3, p_4, p_5$ of the actual network in Fig. 8.2 with actual costs $\widetilde{\gamma}_{ij}$, and the networks where the costs are approximated to integers. Values in bold refer to the shortest (unconstrained) paths in each network.

| **Costs of the arcs** | **Length of** $p_1 = \{1,2,3,4\}$ | **Length of** $p_2 = \{1,2,5,3,4\}$ | **Length of** $p_3 = \{1,2,5,4\}$ | **Length of** $p_4 = \{1,5,3,4\}$ | **Length of** $p_5 = \{1,5,4\}$ |
|---|---|---|---|---|---|
| $\widetilde{\gamma}_{ij}$ | 7.3 | 6.2 | **5.7** | 6.6 | 6.1 |
| **round**($\widetilde{\gamma}_{ij}$) | 8 | 7 | **6** | 7 | **6** |
| **floor**($\widetilde{\gamma}_{ij}$) | 6 | **4** | 5 | 5 | 6 |
| **ceil**($\widetilde{\gamma}_{ij}$) | 8 | 8 | 8 | **7** | **7** |

Note that applying the proposed policy without rounding the non-integer costs, i.e., using the non-integer arc costs as thresholds, is equivalent to taking the flooring of the costs, hence this does not ensure convergence to the actual shortest paths. Indeed, for the above-threshold condition, recalling that $\mathrm{floor}(\widetilde{\gamma}_{ij}) \leq \widetilde{\gamma}_{ij}$, it follows that:

$$x_i - x_j > \widetilde{\gamma}_{ij} \geq \mathrm{floor}(\widetilde{\gamma}_{ij}),$$

hence, regardless of using the original arc cost as threshold or its flooring, a transition occurs and a token moves from node $i$ to node $j$. For the below or at threshold condition, recalling that a real number can be written as the sum of its flooring and its fractional part $0 \leq \mathrm{frac}(\widetilde{\gamma}_{ij}) < 1$, and that $x_i - x_j$ is integer, it follows that:

$$x_i - x_j \leq \widetilde{\gamma}_{ij} = \mathrm{floor}(\widetilde{\gamma}_{ij}) + \mathrm{frac}(\widetilde{\gamma}_{ij}) < \mathrm{floor}(\widetilde{\gamma}_{ij}) + 1, \quad \text{that is} \quad x_i - x_j \leq \mathrm{floor}(\widetilde{\gamma}_{ij}),$$

hence, in both cases, no transition occurs from node $i$ to node $j$.

**Example 8.5: Using non-integer arc costs as thresholds**

Consider an arc $(i, j)$ with actual cost $\widetilde{\gamma}_{ij} = 2.6$. The states of its end nodes are $x_i$ and $x_j$. Let $x_j = 1$ and consider the following three cases:

- if $x_i = 3$, the below threshold condition holds when considering the actual cost as threshold, and the at threshold condition holds when taking its flooring, making no token move from node $i$ to node $j$:

$$x_i - x_j = 2 < \widetilde{\gamma}_{ij} = 2.6 \quad \text{and} \quad x_i - x_j = 2 = \mathrm{floor}(\widetilde{\gamma}_{ij}) = 2;$$

- if $x_i = 4$, the above threshold condition holds both when considering the actual cost as threshold and when taking its flooring, making a token move from node $i$ to node $j$:

$$x_i - x_j = 3 > \widetilde{\gamma}_{ij} = 2.6 \quad \text{and} \quad x_i - x_j = 3 > \mathrm{floor}(\widetilde{\gamma}_{ij}) = 2;$$

- if $x_i = 5$, the above threshold condition holds both when considering the actual cost as threshold and when taking its flooring, making a token move from node $i$ to node $j$:

$$x_i - x_j = 4 > \widetilde{\gamma}_{ij} = 2.6 \quad \text{and} \quad x_i - x_j = 4 > \mathrm{floor}(\widetilde{\gamma}_{ij}) = 2.$$

These facts can also be visualized in Fig. 8.3, recalling the *step* representation of an arc. Note that when $x_i = 4$, one can argue that the step *blocks* the transition. This condition can be modeled by taking the ceiling of the arc costs, but still, the shortest path distribution might change.

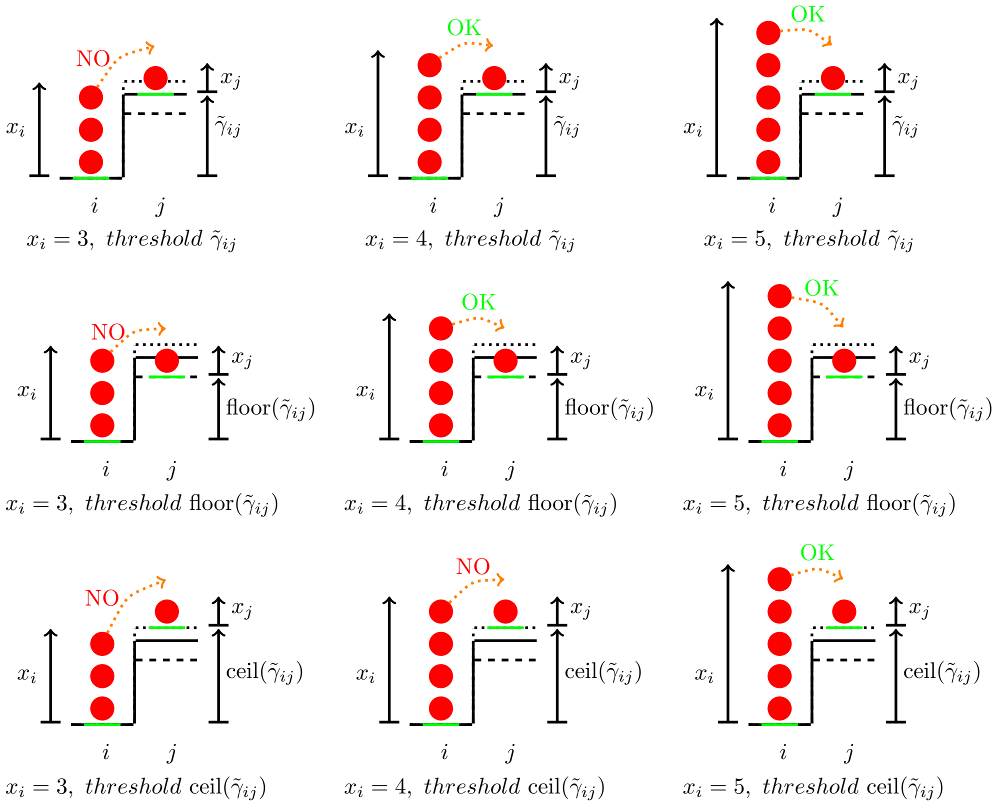

NO
OK
OK
NO
OK
OK
NO
NO
OK
$x_i$
$x_j$
$\tilde{\gamma}_{ij}$
floor($\tilde{\gamma}_{ij}$)
ceil($\tilde{\gamma}_{ij}$)
$i$
$j$
$x_i = 3$, *threshold* $\tilde{\gamma}_{ij}$
$x_i = 4$, *threshold* $\tilde{\gamma}_{ij}$
$x_i = 5$, *threshold* $\tilde{\gamma}_{ij}$
$x_i = 3$, *threshold* floor($\tilde{\gamma}_{ij}$)
$x_i = 4$, *threshold* floor($\tilde{\gamma}_{ij}$)
$x_i = 5$, *threshold* floor($\tilde{\gamma}_{ij}$)
$x_i = 3$, *threshold* ceil($\tilde{\gamma}_{ij}$)
$x_i = 4$, *threshold* ceil($\tilde{\gamma}_{ij}$)
$x_i = 5$, *threshold* ceil($\tilde{\gamma}_{ij}$)

Figure 8.3: Examples of token transition for a non-integer cost arc. Using the non-integer arc cost as threshold is equivalent to using its flooring.

Hereinafter, it is assumed that the arc costs have rational values, which is a natural assumption in path problems. Let $\widetilde{\gamma}_{ij} \in \mathbb{Q}$ be the (actual) non-integer cost of the generic arc $(i,j) \in \mathcal{A}$ of the network, which can be written as the ratio between an integer numerator $n_{ij} \in \mathbb{Z}$ and a positive integer denominator $d_{ij} \in \mathbb{N} \setminus \{0\}$,

$$\widetilde{\gamma}_{ij} = \frac{n_{ij}}{d_{ij}}.$$

Then, the least common multiple (lcm) of the denominators $d_{ij}$ of the arcs' costs is computed as

$$\mu = \operatorname{lcm}(\{|d_{ij}|\colon (i,j) \in \mathcal{A}\}).$$

Notice that, by the definition of $\mu$, $\mu\widetilde{\gamma}_{ij}$ is integer. Then, the network with costs $\gamma_{ij} = \mu\widetilde{\gamma}_{ij}$ is an integer cost network, with the same shortest paths as the one with costs $\widetilde{\gamma}$, as the path lengths are just scaled by $\mu$. Then, the proposed policy can be successfully applied.

When the arc costs are scaled to get integer costs $\gamma_{ij} = \mu\widetilde{\gamma}_{ij}$, it is no longer true that, for each node $i \in \mathcal{N}$, the value of the maximal rest state $\bar{\bar{x}}_i$ is equal to the length $L_{min,i}$ of the *actual* shortest paths (i.e., with respect to costs $\widetilde{\gamma}$) from $i$ to the closest sink; it is rather scaled by $\mu$, that is

$$\bar{\bar{x}}_i = \mu L_{min,i}.$$

As an alternative approach, scaling the arc costs can be avoided by introducing the "value" $\rho \in \mathbb{Q}$ assigned to each token and changing the interpretation of the "state" $x_i$ of a node $i \in \mathcal{N}$. Let $X_i$ be the number of tokens present in the buffer of node $i$. So far, a value of $\rho = 1$ was essentially given to each token, and the state $x_i$ was interpreted as the number of tokens stored in there:

$$x_i = X_i.$$

Then, whenever a token moves from $i$ to $j$, it holds that:

$$x_i \leftarrow x_i - 1 \quad \text{and} \quad x_j \leftarrow x_j + 1.$$

Now, the state $x_i$ is interpreted as the total "value" of the tokens present in the buffer of the node, which is the number of tokens in there multiplied by the generic value $\rho$:

$$x_i = \rho X_i.$$

Then, whenever a token moves from $i$ to $j$, its value is just "transferred":

$$x_i \leftarrow x_i - \rho \quad \text{and} \quad x_j = x_j + \rho.$$

As a result,

$$\bar{\bar{x}}_i = L_{min,i}.$$

**Remark 8.3:**
Applying the proposed policy with $\rho = 1$ and (integer) costs $\mu\widetilde{\gamma}_{ij}$ as thresholds is equivalent to applying it with $\rho = 1/\mu$ and (non-integer) costs $\widetilde{\gamma}_{ij}$ as thresholds. Indeed, the transition conditions are, respectively,

$$x_i - x_j = X_i - X_j > \mu\widetilde{\gamma}_{ij},$$

and

$$x_i - x_j = \rho X_i - \rho X_j = \frac{X_i - X_j}{\mu} > \widetilde{\gamma}_{ij},$$

which are equivalent.

For both the equivalent approaches proposed to deal with non-integer costs, the main limitations are:

- the scaling factor $\mu$ (and $\rho$) must be the same for all arcs, otherwise, the shortest paths' distribution might change. In other words, $\mu$ is a fixed global parameter that must be known in advance;
- the larger is the precision of the costs, the larger is the number of tokens required to converge, even if costs are in the same range;
- when using $\rho = 1/\mu$ and the non-integer costs $\widetilde{\gamma}_{ij}$ as threshold, non-integer arithmetic is involved.

**Example 8.6:**
Consider again the network from Example 8.4, reported in again on the left of Fig. 8.4, with actual costs $\widetilde{\gamma} = [2.5, 4, 3, 1.1, 1.8, 0.8, 2.1] = [5/2, 4/1, 3/1, 11/10, 9/5, 4/5, 21/10]$. From the denominators of these costs, it results that $\mu = 10$. Then, the proposed threshold policy is applied to this network, assigning the value $\rho = 1/\mu = 0.1$ to the tokens. Equivalently, the network on the right of Fig. 8.4 with integer costs $\gamma = \mu\widetilde{\gamma} =$ (using $\rho = 1$) can be considered, which has the same shortest paths, just with costs scaled by $\mu$.

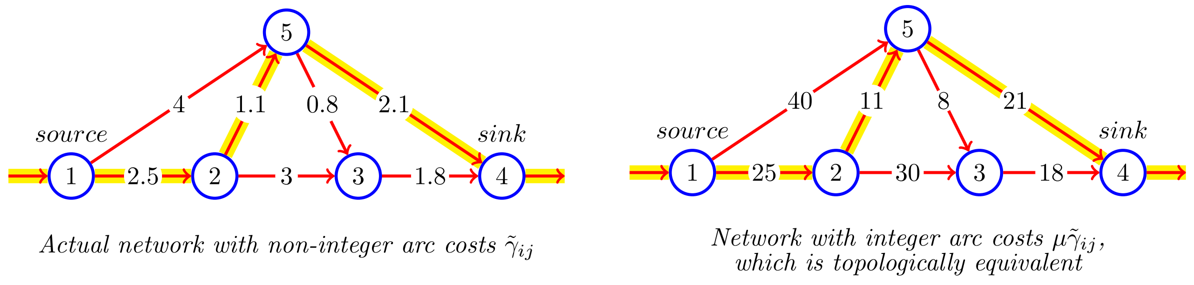

5
4
1.1
0.8
2.1
source
sink
1
2.5
2
3
3
1.8
4
Actual network with non-integer arc costs $\tilde{\gamma}_{ij}$
5
40
11
8
21
source
sink
1
25
2
30
3
18
4
Network with integer arc costs $\mu\tilde{\gamma}_{ij}$, which is topologically equivalent

Figure 8.4: A network with non-integer arc costs $\widetilde{\gamma}_{ij}$. On the left, a network that has integer costs $\gamma_{ij}$ and is topologically equivalent to it in terms of shortest paths is represented.

Consider now the network in Fig. 8.5, with costs $\widetilde{\gamma} = [2.505, 4.053, 3.068, 1.107, 1.824, 0.815, 2.112] = [501/200, 4053/1000, 767/250, 1107/1000, 228/125, 163/200, 264/125]$. By taking $\mu = 1000$, the network with costs $\gamma = \mu\widetilde{\gamma}$ is obtained. Despite having the same topology of the network in Fig. 8.4, with the costs being in the same range, just with higher precision, applying the proposed policy here requires far more tokens to converge.

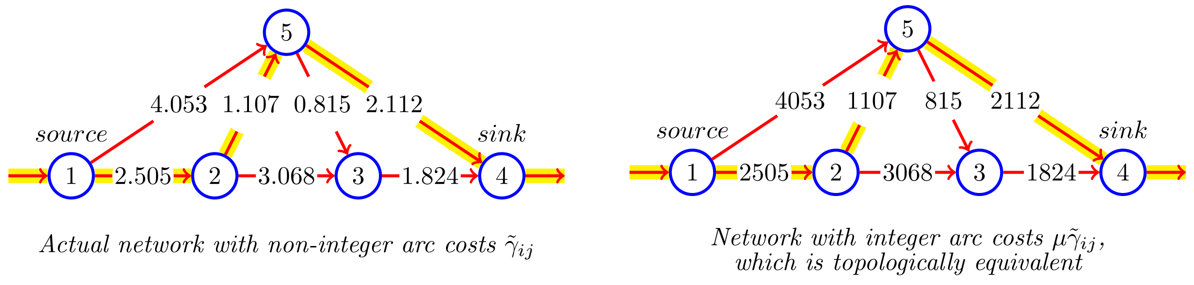

5
4.053
1.107
0.815
2.112
source
sink
1
2.505
2
3.068
3
1.824
4
Actual network with non-integer arc costs $\tilde{\gamma}_{ij}$
5
4053
1107
815
2112
source
sink
1
2505
2
3068
3
1824
4
Network with integer arc costs $\mu\tilde{\gamma}_{ij}$, which is topologically equivalent

Figure 8.5: The same of Fig. 8.4, with the only difference being that the rational costs $\widetilde{\gamma}_{ij}$ are in the same range, but have an higher precision.

## 8.8 An enhanced policy considering virtual tokens

It has been shown that Policy 8.1 is optimal, in the sense that, under certain assumptions, it drives the state of a network to a global rest state, which guarantees that eventually, all newly injected tokens in the sink nodes of the network are able to reach the closest sink node, through the shortest paths, and do not get lost. However, one of the main drawbacks of Policy 8.1 is that most of the tokens injected during the transitory phase cannot reach the sinks, as they are deposited in the node buffers to "fill" their states and possibly allow the successive tokens to proceed moving.

Trivially, this might be mitigated by sending non-informative tokens during the initial transitory, and informative tokens when a global rest state is reached, so that no information is lost. However, the problem of detecting locally the time at which the convergence occurs remains.

A simple variation of the threshold Policy 8.1 ensures that *all* the injected tokens, even those arriving during the initial transient, reach a sink node, hence can be informative, in the case in which paths are not constrained and the following Assumption is introduced.

**Assumption 8.2:**
Network $\mathcal{G}$ is strongly connected.

The concept of virtual (non-informative) tokens is reconsidered, which are generated locally in a node to make elementary transitions always possible whenever an informative token arrives in there.

**Policy 8.2: Enhanced decentralized threshold policy (unconstrained system).**
When an (informative) token reaches a node $i \in \mathcal{N}$ in which the above-threshold condition does not hold for any outgoing arc, i.e.,

$$x_i(k) + 1 - x_j(k) \leq \gamma_{ij}, \quad \forall j \in \mathcal{N}_i,$$

instead of stopping the token, generate a block of $\gamma_{i\hat{j}} + x_{\hat{j}}(k) - x_i(k)$ *virtual tokens* in $i$, where $\hat{j} = \arg\min_{j \in \mathcal{N}_i} \{\gamma_{ij} + x_j(k) - x_i(k)\}$, so that the value of state $x_i$ immediately becomes: $x_i(k) \to x_{\hat{j}}(k) + \gamma_{i\hat{j}}$ and the arc $(i, \hat{g})$ becomes above-threshold. Then, the informative token moves to the node $\hat{j}$.

This enhanced policy ensures that the more the number of virtual tokens are generated by the traveling agents along their way, the less the number of (actual, informative) tokens needs to be injected in the source nodes of the network to converge to a global rest state. Essentially, the generation of a block of virtual tokens at a given time is equivalent to having injected the same amount of (actual) tokens in the network at different earlier times.

Then, this policy becomes independent from the arc costs values $\gamma_{ij}$ and results in way better performance compared to the original Policy 8.1. The number of injected tokens needed to reach a global rest state depends on the number of arcs of the shortest paths and, in general, the enhancements become more evident as $\bar{\gamma}$ grows.

**Example 8.7:**
Consider an arc $(i, j)$ with states $x_i = x_j = 0$ and cost $\gamma_{ij} = 1000$. An arrival of 1000 injected tokens in $i$ is required to allow the transition to node $j$ using Policy 8.1; with the enhanced Policy 8.2, as soon as the first token arrives in $i$, the generation of virtual tokens “fills” the state $x_i$ and allows immediate transition to $j$. The immediate next token arriving in $i$ can reach $j$ without further operations.

The assumption of a strongly connected network ensures that tokens cannot travel indefinitely or be stuck in nodes with no outgoing arc. As the injected tokens continue moving from node to node, and the costs (thresholds) are finite, it is guaranteed that such informative tokens always reach sink nodes, as the state approaches the maximal rest state.

Still, the fact that all the injected tokens can reach a sink node does not mean that all the tokens reach it through the shortest paths. Indeed, the tokens injected initially, before the global rest state is reached, might take longer non-optimal routes. A sufficient number of virtual tokens are to be generated to reach a global rest state. Differently from the case considered so far, those routes *might* also be walks: tokens might return to the same node, because generating a block of virtual tokens at a single time is equivalent to injecting the same amount of tokens at different times, and tokens injected at different times might traverse the same nodes.

Finally, remark that while the number of injected tokens needed to converge to a global rest state is reduced, the number of “elementary transitions” (i.e., generation of virtual tokens), and hence the time necessary to reach that global rest state, remains pseudo-polynomial in the size of the problem input.

Chapter 9

# A decentralized agent-based policy finding the constrained shortest paths

In this Chapter a constrained system is assumed and Problem 7.1 is addressed. A decentralized threshold policy is obtained, which is built on the decentralized policy presented in the previous Chapter 8. Indeed, it will be shown that this new problem can be equivalently seen as an (unconstrained) shortest path problem in the so-called *expanded network*, which is derived from the actual considered network. In the long run, this policy makes all the newly injected tokens reach the closest sink along the constrained shortest paths.

For the simplicity of explanation, the constrained costs will be assumed non-negative for all the arcs of the network ($\sigma_{ij} \geq 0$); the extension to negative $\sigma_{ij}$ is straightforward and will be briefly discussed, too.

## 9.1 Constrained costs and feasible routes

Just like in the unconstrained case, tokens travel in the network, from node to node, according to a given policy, which determines whether they can proceed moving or they must stop at some node. Whenever a token is forced to stop in a non-sink node, it is deposited in the corresponding buffer.

Recall that in a constrained system, each token keeps track of the constrained cost $c = C(p)$ of its traveled route $p$: this information is needed to avoid the token taking unfeasible paths, whose constrained cost is greater than a given value $C_{max} \in \mathbb{N}$. This is different from an unconstrained system, where basically all the paths are feasible, so this information is not required.

Specifically, at time 0 is assumed that $c = 0$ for all the tokens. A token that is injected into the network later on also has $c = 0$. Then, whenever a generic token moves from node $i \in \mathcal{N}$ to an adjacent node $j \in \mathcal{N}_i$ through an arc $(i, j) \in \mathcal{A}$ with constrained cost $\sigma_{ij}$, it updates its constrained cost as

$$c \leftarrow c + \sigma_{ij}.$$

A token currently in node $i \in \mathcal{N}$ is said to *fall asleep* in $i$ if any tentative to leave the node would result in a violation of constraint Eq. (7.3), making its traveled path infeasible. In particular, if:

$$c + \sigma_{ij} > C_{max}, \quad \forall (i, j) \in \mathcal{A}, j \in \mathcal{N}_i. \tag{9.1}$$

An asleep token cannot continue moving; thus, it stops definitively in node $i$, independently from the adopted policy.

Let $\mathcal{C}$ be the set of all the possible values assumed by $c \leq 0$ considering all the possible routes in the network, and $C = |\mathcal{C}|$ the size of this set. Also, denote $N = nC$ and $M = mC$. Finally, let $\mathcal{C}_{ij}$ be the set of all the possible values assumed by $c \in \mathcal{C}$ to get a feasible transition from arc $i$ to arc $j$: $\mathcal{C}_{ij} = \{c : c \in \mathcal{C} \text{ and } c + \sigma_{ij} \in \mathcal{C}\}$. Note that, for the generic arc $(i, j) \in \mathcal{A}$, for any $c \in \mathcal{C}_{ij}$, both $c \leq C_{max}$ and $c + \sigma_{ij} \leq C_{max}$ must hold, i.e., $c \leq \min\{C_{max}, C_{max} - \sigma_{ij}\}$.

In the specific case considered here, in which non-negative $\sigma_{ij} \geq 0$ are considered and $c$ is initialized to 0, the range of values that can be assumed by $c$ is indeed $[0, C_{max}]$, i.e., $\mathcal{C} = \{0, 1, \ldots, C_{max}\}$, $C = C_{max} + 1$, $N = n(C_{max} + 1)$, $M = m(C_{max} + 1)$, $\mathcal{C}_{ij} = \{0, 1, ..., C_{max} - \sigma_{ij}\}$.

Recall that to account for possible constraints on the routes, the tokens deposited in each node $i \in \mathcal{N}$ are buffered according to their constrained cost $c$, i.e., the total constrained cost paid by such tokens to reach such node. Then, now a multi-component state is considered for each node:

$$x_i = [x_i^0, x_i^1, \dots, x_i^{C_{max}}]^\top \in \mathbb{Z}^C,$$

where $x_i^c$ is the number of tokens with constrained cost $c$ stopped in node $i$ (including asleep tokens), defined with respect to the "zero level" of the node. There is a component for each possible value assumed by the cost $c \in \mathcal{C}$, so that there are $C = C_{max} + 1$ components in $x_i$.

Then, the state of the network has $N = nC = n(C_{max} + 1)$ components:

$$x = [x_1, x_2, \dots, x_n]^\top = [x_1^0, x_1^1, \dots, x_1^{C_{max}}, x_2^0, x_2^1, \dots, x_2^{C_{max}}, \dots, x_n^0, x_n^1, \dots, x_n^{C_{max}}]^\top \in \mathbb{Z}^N.$$

## 9.2 Admissibility of the state

The definition of admissible state is slightly different from that of the unconstrained system from Section 8.1, because of the constraints given by the route feasibility and the multi-component states $x_i^c$ that are introduced in each node. Hence, it is revised as follows.

**Definition 9.1: Admissibility in a constrained system**

The state $x$ of a network $\mathcal{G}$ is admissible if and only if $x_i^c - x_j^{c+\sigma_{ij}} \leq \gamma_{ij}$ for all $(i, j) \in \mathcal{A}$, for all $c \in \mathcal{C}_{ij}$.

A token in generic node $i \in \mathcal{N}$ with constrained cost $c$ assumes a constrained cost $c + \sigma_{ij}$ after moving to some adjacent node $j \in \mathcal{N}_i$: only "compatible" tokens buffered in the two nodes are compared when evaluating the admissibility of the state of the network, which is measured by the corresponding state components $x_i^c$ and $x_j^{c+\sigma_{ij}}$. This must hold for all the arcs $(i, j)$, and for all the possible values that $c$ can assume to get a feasible transition. In the simplified case assumed here, in which $\sigma_{ij} \geq 0$, $\mathcal{C}_{ij} = \{0, 1, ..., C_{max} - \sigma_{ij}\}$.

The definitions of below, at, above, under-threshold from Definition 8.2 can be easily adapted to the constrained case, by considering "$x_i^c - x_j^{c+\sigma_{ij}}$" instead of "$x_i - x_j$".

## 9.3 Dynamics of the state of the network

Everything from Section 9.3 could be reformulated, taking into account the state components $x_i^c$ instead of the states $x_i$.

Again, from the fast dynamic perspective, tokens' movements in the network can be described by the corresponding sequence of elementary transitions, which change the state of the network based on the constrained cost of the moving token.

In particular, injecting a token in source node $i \in \mathcal{S}$ at time $t_k = t_k^0$ results in the increasing of the state component $x_i^0$ of node $i$ by 1, i.e., $x_i^0 \leftarrow x_i^0 + 1$, since all the newly injected tokens have $c = 0$. This behavior is now described by the input vector

$$\nu = [\nu_1^0, \nu_1^1, ..., \nu_1^{C_{max}}, \nu_2^0, \nu_2^1, ..., \nu_2^{C_{max}}, \dots, \nu_n^0, \nu_n^1, ..., \nu_n^{C_{max}}]^\top \in \mathbb{N}^N,$$

whose generic component $\nu_h^c$, associated with node $h \in \mathcal{N}$ (and corresponding to state component $x_h^c$) is defined as

$$\nu_h^c = \begin{cases} 1, & \text{if the injection occurs at node } h \text{ and c=0,} \\ 0, & \text{otherwise.} \end{cases}$$

Note that $\nu_h^c \equiv 0$ for all possible non-zero values assumed by $c$. Then, $x_h^c \leftarrow x_h^c + \nu_h^c$ for any node $h \in \mathcal{N}$ and for any possible $c \in \mathcal{C}$, that is $x \leftarrow x + \nu = x + e_a$, with $a = C(i - 1) + 1 + c$ defining the only non-zero element of vector $e_a \in \mathbb{N}^N$, that is the $a$th canonical basis vector.

Instead, if a token with constrained cost $c$ performs an elementary transition from node $i \in \mathcal{N}$ to non-sink node $j \in \mathcal{N} \setminus \mathcal{T}$ at a generic time $t_k^r$ of the fast dynamic, the state component $x_i^c$ of node $i$ is decreased by 1 and the state component $x_j^{c+\sigma_{ij}}$ of node $j$ is increased by 1, i.e., $x_i^c \leftarrow x_i^c - 1$ and $x_j^{c+\sigma_{ij}} \leftarrow x_j^{c+\sigma_{ij}} + 1$. Now, the control describing the possible movement of a token along an arc $(i, j)$ depends on the constrained cost $c = C(p)$ of its traveled route $p$ up to $i$. Then, this behavior is described by the control vector

$$u = [u_1^0, u_1^1, ..., u_1^{C_{max}}, u_2^0, u_2^1, \dots, u_2^{C_{max}}, \dots, u_m^0, u_m^1, \dots, u_m^{C_{max}}]^\top \in \mathbb{N}^M,$$

whose generic component $u_h^c$, associated with arc $h = (i, j) \in \mathcal{A}$, is defined as

$$u_h^c = u_{ij}^c = \begin{cases} 1, & \text{if an elementary transition of a token with constrained cost } c \\ & \text{(of its traveled route up to node } i\text{) occurs along arc } h = (i, j), \\ 0, & \text{otherwise.} \end{cases}$$

Consequently, $x_i^c \leftarrow x_i^c - u_{ij}^c$ and $x_j^{c+\sigma_{ij}} \leftarrow x_j^{c+\sigma_{ij}} + u_{ij}^c$ for any node arc $h \in \mathcal{A}$ and for any possible $c \in \mathcal{C}$.

If $j \in \mathcal{T}$ is a sink node, it must have an empty state anytime by definition, for any possible value of $c \in \mathcal{C}$, i.e., $x_j^c \equiv 0$, as it is assumed that any token arriving in a sink is instantaneously expelled from the network. An elementary transition of a token with constraint cost $c$ from node $i \in \mathcal{N}$ to sink node $j \in \mathcal{T}$ results in just $x_j^{c+\sigma_{ij}} \leftarrow x_j^{c+\sigma_{ij}} + u_{ij}^c$.

The overall state variation due to the traveling of a token depends on the origin of the token, its destination, and the traveled route (but just on its constrained cost, and not on the specific traversed nodes). Even in this case, it is therefore possible to temporize the slow dynamic only. The slow dynamics of network $\mathcal{G}$ is specified for each node $i \in \mathcal{N}$ and for each possible value of the constrained cost $c \in \mathcal{C}$ by the following state equation

$$x_i^c(k+1) = \begin{cases} x_i^c(k) - \sum\limits_{j \in \mathcal{N}_i} u_{ij}^c(k) + \sum\limits_{j: i \in \mathcal{N}_j} u_{ji}^{c-\sigma_{ji}}(k) + \nu_i^c(k), & \text{if } i \notin \mathcal{T} \text{ (non-sink node)}, \\ 0, & \text{if } i \in \mathcal{T} \text{ (sink node)}. \end{cases} \tag{9.2}$$

## 9.4 Decentralized transition rule

The control $u_{ij}^c$ introduced in the previous Section specifies whether an elementary transition along arc $(i, j)$ is permitted and occurs, or not, for a token which has constrained cost $c$ when in node $i$. The idea of the transition rule is the same as the one presented in Section 8.3: a token injected in a source node $i$ of a network in state $x$ continues moving from node to node until it remains above-threshold in the nodes that it reaches; eventually, either it reaches a node $j$ where it is under-threshold or it reaches a sink node where it leaves the network.

Then, the *decentralized threshold* Policy 8.1 can be easily adapted to this case as follows, taking into account that tokens might fall asleep if their route would become infeasible by continuing moving.

**Policy 9.1: Decentralized threshold policy (constrained system).**
Consider the following conditions for the control $u_{ij}^c$ specifying the policy for each arc $(i, j) \in \mathcal{A}$ and constrained costs $c \in \mathcal{C}$:

a) at most a token can enter a node, regardless of its constrained cost $c$, i.e.,

$$\sum_{j: i \in \mathcal{N}_j} \sum_{c-\sigma_{ji} \in \mathcal{C}} u_{ji}^{c-\sigma_{ji}} + \nu_i^0 \leq 1, \ \forall i \in \mathcal{N}; \tag{9.3a}$$

b) at most a token can leave a node, i.e.,

$$\sum_{j \in \mathcal{N}_i} \sum_{c \in \mathcal{C}} u_{ij}^c \leq 1, \ \forall i \in \mathcal{N}; \tag{9.3b}$$

c) a token in node $i$ with constrained cost $c$ can move to node $j$ through arc $(i, j)$ only if

$$x_i^c + \sum_{l: i \in \mathcal{N}_l} u_{li}^{c-\sigma_{li}} + \nu_i^0 - x_j^{c+\sigma_{ij}} > \gamma_{ij}; \tag{9.3c}$$

d) a necessary condition to allow the transition of a token with constrained cost $c$ in node $i$ along an arc $(i, j)$ is

$$c + \sigma_{ij} \leq C_{max}, \quad \text{for some } j \in \mathcal{N}_i. \tag{9.3d}$$

Then, the control is

$$u_{ij}^c = \begin{cases} 1, & \text{if Eq. (9.3a) and Eq. (9.3b) and Eq. (9.3c) and Eq. (9.3d) hold,} \\ 0, & \text{otherwise.} \end{cases} \tag{9.4}$$

Conditions a) and b) in Policy 9.1 simply mean that each node can process only one token at a time. Recall that $\nu_i^c = 0$ for all $c \neq 0$. Condition c) adapts the transition rule to the case in which constrained costs are to be taken into account. Condition d) is meant to avoid paths becoming infeasible, possibly making tokens fall asleep.

Policy 9.1 remains decentralized but, generally, as it will be shown, it does not guarantee that all the routes followed by the moving tokens are paths (and not walks).

**Remark 9.1:**
If the state $x$ of the network is currently admissible at time $t_k$ and no new token is injected in the network, no token moves.

If the state $x$ of the network is currently admissible at time $t_k$ and a new token is injected in source node $i \in \mathcal{S}$, this is the only one possibly moving in the network.

Finally, if the state $x$ of the network is currently not admissible at time $t_k$ and no new token is injected in the network, a token already present in the network moves, which takes priority over the injected ones.

## 9.5 Expanded network model

The problem of finding the constrained shortest path in a given network $\mathcal{G}$ can be reduced to finding the (unconstrained) shortest path in the so-called *expanded network* [118, 119]. This property can be exploited to prove that the decentralized Policy 9.1 solves the considered Problem 7.1.

The expanded network of $\mathcal{G}(\mathcal{N}, \mathcal{A})$, which will be denoted by $\mathcal{G}_E(\mathcal{N}_E, \mathcal{A}_E)$, has the following characteristics.

- It is composed by $n$ nodes replicated for $C$ steps (each one corresponding to a specific constrained cost $c \in \mathcal{C}$), so that $|\mathcal{N}_E| = |\mathcal{N}||\mathcal{C}| = nC$. Let $i^c$ be the $i$th node of $\mathcal{G}$ replicated at step $c$ of $\mathcal{G}_E$, for $c \in \mathcal{C}$.
- Any arc $(i, j) \in \mathcal{A}$ is replicated, at most, $C - 1$ times in $\mathcal{G}_E$, becoming $(i^c, j^{c+\sigma_{ij}})$ for $c \in \mathcal{C}_{ij}$, so that $|\mathcal{A}| \leq |\mathcal{A}_E| \leq |\mathcal{A}|(C-1) = m(C-1)$.

  If $\sigma_{ij} = 1$ for all $(i, j) \in \mathcal{A}$, i.e., if the number of arcs that a token can traverse is upper bounded, $|\mathcal{A}_E| = m(C-1)$.
- For each source $i \in \mathcal{S}$ of $\mathcal{G}$, a single source is set in node $i^0$ of $\mathcal{G}_E$.
- For each sink $i \in \mathcal{T}$ of $\mathcal{G}$, $C$ sinks are set in $\mathcal{G}_E$, in nodes $i^c$, for $c \in \mathcal{C}$.

Recall that if $\sigma_{ij} \geq 0$, $C = C_{max} + 1$, $\mathcal{C} = \{0, 1, ..., C_{max}\}$, and $\mathcal{C}_{ij} = \{0, 1, ..., C_{max} - \sigma_{ij}\}$.

In the next Examples 9.1 and 9.2, two examples of expanded networks are reported.

The information about the constrained costs of the arcs of $\mathcal{G}$ is now modeled directly in the topology of $\mathcal{G}_E$, hence an unconstrained system can be considered: no constraint is imposed on the paths of the expanded network $\mathcal{G}_E$, and, consequently, each node $i^c \in \mathcal{N}_E$ has a single-component (scalar) state $x_{i^c}$.

There is a one-to-one correspondence between the (scalar) state $x_{i^c}$ of any node $i^c \in \mathcal{N}_E$ and $x_i^c$, the $c$-th component of the (vector) state $x_i$ of node $i \in \mathcal{N}$. By construction, there is also a one-to-one correspondence between any unique feasible path or walk $p$ between any pair of nodes in $\mathcal{G}$, traveled by a token taking into account its constrained cost $C(p) \leq C_{max}$, and a unique path in $\mathcal{G}_E$, with the same length $L(p)$. Finally, if a token falls asleep in node $i \in \mathcal{N}$ of $\mathcal{G}$, this corresponds to reaching a non-sink node $i^c \in \mathcal{N}_E$ with no outgoing arcs.

**Theorem 9.1:**
Given a system $(\mathcal{G}, f)$ under Assumption 7.1, its expanded network $\mathcal{G}_E$:

- is acyclic if and only if there exists no circuit $\varphi$ in $\mathcal{G}$ such that $C(\varphi) = 0$ (zero constrained cost) and, in particular, any walk in $\mathcal{G}$ becomes a valid path in $\mathcal{G}_E$;
- presents a positive length $L(\phi) > 0$ for any circuit $\varphi$ in $\mathcal{G}$ with $C(\varphi) = 0$.

The above Theorem ensures that there are no non-positive length circuits in $\mathcal{G}_E$: then, $\mathcal{G}_E$ is an unconstrained network satisfying Assumption 7.1.

Note that some nodes of $\mathcal{G}_E$ might not be reachable by the tokens injected in the sources: in the case in which the initial state $x$ is admissible, the corresponding state components in $\mathcal{G}$ will always remain 0, if the initial state is zero and Policy 9.1 is applied. If the initial state is not admissible, there might be some transitions of tokens between any pair of nodes, even those not reachable by the sources. The transitions of tokens with $c < \min\{\mathcal{C}\}$ should be denied by default.

**Example 9.1:**
Consider again the network $\mathcal{G}$ from Fig. 7.2, reported here in Fig. 9.1, left. The corresponding expanded network $\mathcal{G}_E$ is depicted on the right, in the case in which $C_{max} = 2$.
Note that $\mathcal{G}_E$ is acyclic. Arcs derived from arc $(4,5)$ of $\mathcal{G}$ are directed "downward", since $\sigma_{4,5} = 0$; all the other arcs are directed "rightward", since in this example $\sigma_{ij} > 0$ for all the others arcs $(i,j) \neq (4,5)$. Multiple sinks are derived in node $\mathcal{G}_E$ from node 5 of $\mathcal{G}$.
Note that a token traveling path $1 \to 2 \to 3$ in $\mathcal{G}$ would fall asleep in node 3. In $\mathcal{G}_E$ this corresponds to reaching node $3^2$, which has no outgoing arcs, through path $1^0 \to 2^1 \to 3^2$.
Also, in $\mathcal{G}_E$ nodes $2^0, 3^0, 4^0, 5^0, 1^1, 4^1, 5^1, 1^2, 2^2$ are not reachable by the source node $1^0$, meaning that, in $\mathcal{G}$, $x_2^0(k) = x_3^0(k) = x_4^0(k) = x_5^0(k) = x_1^1(k) = x_4^1(k) = x_5^1(k) = x_1^2(k) = x_2^2(k) = 0, \forall k \geq 0$ if $x(0) \in \mathcal{O}$ is admissible and initialized to 0.

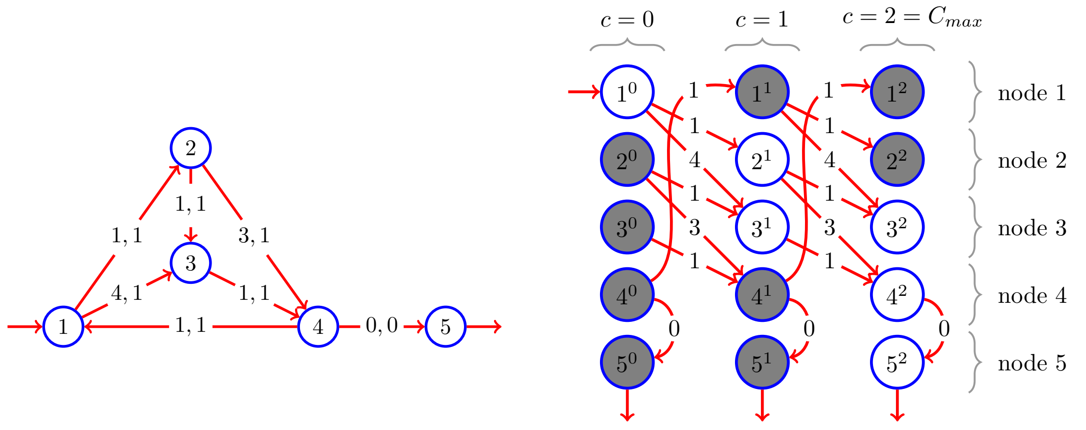


Figure 9.1: The simple network $\mathcal{G}$ from Fig. 7.2 (left) and the corresponding expanded network $\mathcal{G}_E$, when $C_{max} = 2$ (right). The values $\gamma_{ij}$ and $\sigma_{ij}$ are indicated for each arc $(i,j)$ of $\mathcal{G}$; the value $\gamma_{ij}$ is indicated for each arc $(i,j)$ of $\mathcal{G}_E$. In $\mathcal{G}_E$, nodes highlighted in grey are not reachable by tokens injected in the sources.

**Example 9.2:**
Consider the network $\mathcal{G}$ in Fig. 9.2, left. The corresponding expanded network $\mathcal{G}_E$ is depicted on the right, in the case in which $C_{max} = 4$.
The possible feasible routes in $\mathcal{G}$ are paths $p_1 = \{1,4\}$, $p_2 = \{1,2,3,4\}$ and walk $w_1 = \{1,2,3,1,4\}$. An *unique unconstrained path* is associated with each of them in $\mathcal{G}_E$: $p_{E,1} = \{1^0, 4^1\}$, $p_{E,2} = \{1^0, 2^1, 3^2, 4^3\}$ and walk $p_{E,3} = \{1^0, 2^1, 3^2, 1^3, 4^4\}$, respectively.

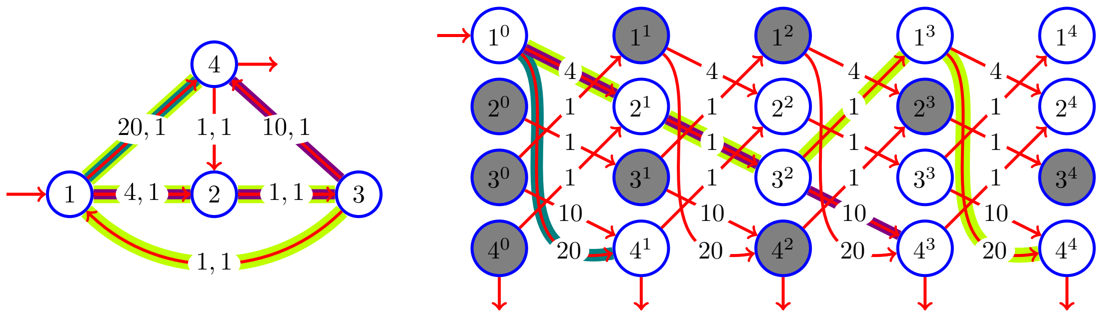


Figure 9.2: A simple network $\mathcal{G}$ (left) and the corresponding expanded network $\mathcal{G}_E$ when $C_{max} = 4$ (right). The values $\gamma_{ij}$ and $\sigma_{ij}$ are indicated for each arc $(i,j)$ of $\mathcal{G}$; $\gamma_{ij}$ is indicated for each arc $(i,j)$ of $\mathcal{G}_E$. In $\mathcal{G}_E$, nodes highlighted in grey are not reachable by tokens injected in the sources. The possible paths/walks in $\mathcal{G}$ are highlighted, as well as the corresponding paths in $\mathcal{G}_E$.

## 9.6 Properties of the system

All the properties that are valid for Policy 8.1 introduced for the unconstrained system are valid for Policy 9.1 introduced for the constrained system. In particular, in the long run, tokens injected in the network $\mathcal{G}$, which are routed by applying Policy 9.1 and whose routes $p$ are constrained by $C(p) \leq C_{max}$, eventually are all able to reach the closest sink though the shortest constrained paths. If the network is dynamic, the constrained system is able to adapt to dynamic conditions, too. The reason is explained next.

Firstly, by applying the results proven in Section 8.4 to network $\mathcal{G}_E$, which satisfy Assumption 7.1, all the tokens injected in any source node $(i^0, i \in \mathcal{S})$, which are routed applying Policy 8.1, will eventually be routed to the closest sink $(j^c, j \in \mathcal{T}$, for some $0 \leq c \leq C_{max})$ through the shortest paths, in the long run. This solves Problem 7.2 for the network $\mathcal{G}_E$. Then, the next Theorem proves that a solution for Problem 7.2 in $\mathcal{G}_E$ actually solves Problem 7.1 in $\mathcal{G}$. As there is a one-to-one correspondence between applying Policy 8.1 in $\mathcal{G}_E$ and applying Policy 9.1 in $\mathcal{G}$, this proves the statements reported above.

**Theorem 9.2:**
A system $(\mathcal{G}, f)$ is given that satisfies Assumptions 7.1 to 7.4. Then a shortest path from a source $i^0$ (with $i \in \mathcal{S}$) to a sink $j^c$ (with $j \in \mathcal{S}$ and for some $0 \leq c \leq C_{max}$) in its expanded network $\mathcal{G}_E$ corresponds to a shortest feasible path in $\mathcal{G}$.

Note that from the proof of the above Theorem (see Appendix B), the condition $C(\varphi) \geq 0$ for any circuit $\varphi$ in $\mathcal{G}$ from Assumption 7.1 is necessary. Otherwise, the shortest feasible route from a source to a sink in $\mathcal{G}$ might turn out to be a walk.

**Remark 9.2:**
Applying Policy 9.1 to a constrained system $(\mathcal{G}, f)$ is equivalent to consider an unconstrained (and uniquely defined) system $(\mathcal{G}_E, f)$ where Policy 8.1 is applied.

This is the second most important result of this Part: despite a very simple decentralized policy is adopted to route the memoryless agents using only local information, in the long run, the constrained shortest paths emerge, and each new injected token can eventually leave the network traveling along them. The constrained shortest path problem can be traced back to an unconstrained shortest path problem in the corresponding expanded network; it should be noted, however, that the latter is larger by some factor $C$.

### Some considerations about the complexity

The time to reach a global rest state, measured in number of injected tokens, is exponential in the size of the problem input [2].

**Remark 9.3:**
When $C_{max} = \infty$, Problem 7.1 reduces to Problem 7.2. However, applying the policy derived for the constrained system is inefficient, as all the (possibly infinite) state components $x_i^c$ are still accounted for (despite considering just the scalar states $x_i$ of the unconstrained system would be sufficient). While, most of these components will be 0 (as the resulting route is a path and not a walk), it is like the problem is actually solved in the more complex expanded network.
When $\sigma_{ij} = 0$ for all arcs $(i, j) \in \mathcal{A}$, Problem 7.1 reduces, again, to Problem 7.2. Now, the set of the nodes of the portion of the expanded network that is reachable is equal to $\mathcal{N}$. Hence, now the problem is solved in the same network.

## 9.7 Negative arc costs

Given Remark 9.2, the proposed policy continues to work even in the presence of arcs $(i, j) \in \mathcal{A}$ of network $\mathcal{G}$ with negative costs $\gamma_{ij}$ (under Assumption 7.1). The initial state is not admissible if it is initialized to zero; then, initially, there are some elementary transitions of tokens already in the network; only transitions of tokens with constrained cost $c \geq \min\{\mathcal{C}\}$ can be considered, since this is the minimum value that $c$ can assume.

In particular, if $\sigma_{ij} \geq 0$ for all arcs $(i, j) \in \mathcal{A}$, the cost paid by each injected token is necessarily $c \geq 0$. Instead, if there are some arcs $(i, j) \in \mathcal{A}$ of network $\mathcal{G}$ with negative constrained costs $\sigma_{ij}$ (under Assumption 7.1), the possible values that the constrained cost $c$ paid by each traveling token along its traveled path include negative values. This means that some components $x_i^c$ of the state of some node $i \in \mathcal{N}$ with $c < 0$ can be reachable. However, thanks to the assumption of the absence of circuits $p$ with negative constrained costs $C(p)$ in $\mathcal{G}$ and the fact that the constrained cost $c$ of each token is initialized with $c = 0$, it can be easily seen that $c$ is lower bounded by some finite non-positive value $L \in \mathbb{Z}$ such that all the components $x_i^c, c < L$ are never reached, so they could be set to 0 and be neglected. This can be easily verified by visualizing the expanded network.

All the discussion presented so far, including Theorem 9.2, is still valid by considering $\mathcal{C} = \{L, ..., 0, ..., C_{max}\}$ instead of $\mathcal{C} = \{0, ..., C_{max}\}$, i.e., considering $C = C_{max} + 1 - L$ components $x_i^c$ of the states $x_i$ for each node, $N = nC = n(C_{max} + 1 - L)$ components $x_i^c$ and $\nu_i^c$ of the state $x$ and input $\nu$, respectively, and, finally, $M = mC = m(C_{max} + 1 - L)$ components $u_h^c$ of the control $u$.

**Example 9.3:**

Consider the network $\mathcal{G}$ in Fig. 9.3, left, which has arc $(1, 3)$ with negative constrained cost $\sigma_{1,3} = -2$ and all positive circuits with respect to the constrained costs $\sigma_{ij}$. The corresponding expanded network $\mathcal{G}_E$ is reported on the right. Note that $\mathcal{G}_E$ is acyclic, despite some arcs deriving from arc $(1, 3)$ being directed "leftward". Note that no reachable node exists for $c < -2$, so the reachable portion of the expanded network is finite; hence, here $L = -2$.

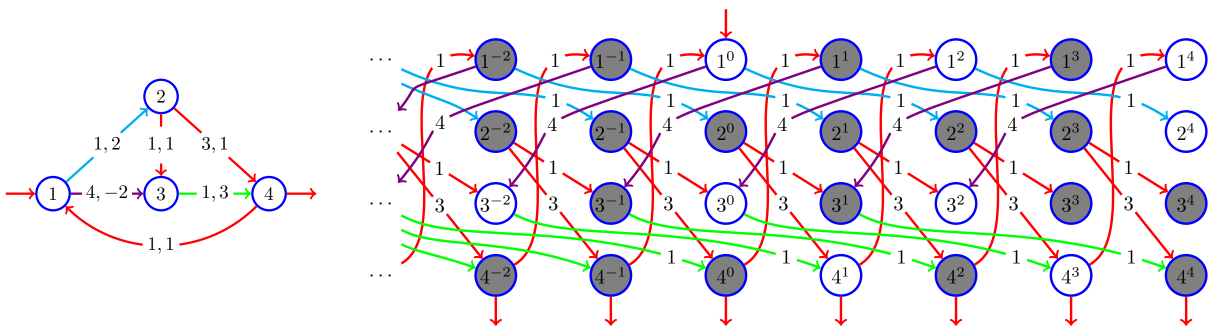


Figure 9.3: A simple network $\mathcal{G}$ (left) with an arc $(1, 3)$ with negative constrained cost $\sigma_{1,3} = -2$ and the corresponding expanded network $\mathcal{G}_E$ (right), assuming $C_{max} = 4$. In $\mathcal{G}_E$, nodes highlighted in grey are not reachable by tokens injected in the sources. $\mathcal{G}$ has only positive circuits with respect to the constrained costs $\sigma_{ij}$.

Consider now the network $\mathcal{G}$ in Fig. 9.4, left, which has arc $(1, 3)$ with negative constrained cost $\sigma_{1,3} = -2$ and circuit $p = \{1, 3, 4, 1\}$ with $C(p) = 0$. The corresponding expanded network $\mathcal{G}_E$ is reported on the right. Note that $\mathcal{G}_E$ is no more acyclic, since the (positive) cycles $p = \{1^0, 3^{-2}, 4^{-1}, 1^0\}$ and $p = \{1^4, 3^2, 4^3, 1^4\}$ are present (see Theorem 9.1). Note that no reachable node exists for $c < -2$, so the reachable portion of the expanded network is finite; hence, here $L = -2$.

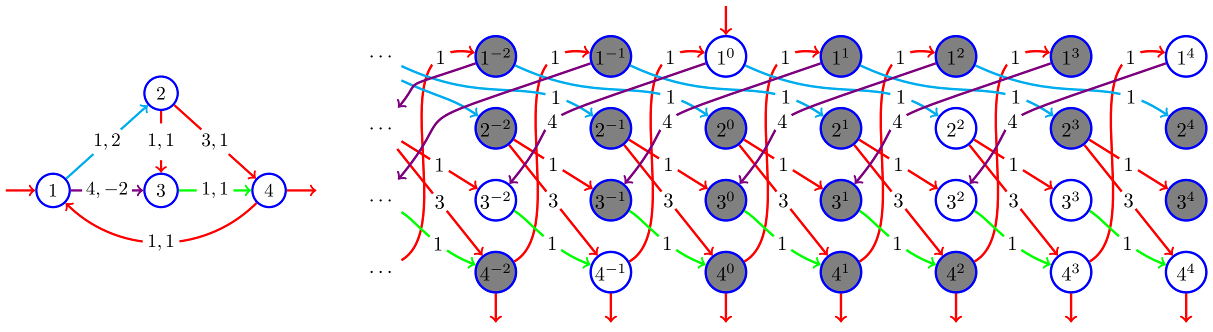


Figure 9.4: A simple network $\mathcal{G}$ (left) with an arc $(1, 3)$ with negative constrained cost $\sigma_{1,3} = -2$ and the corresponding expanded network $\mathcal{G}_E$ (right), assuming $C_{max} = 4$. In $\mathcal{G}_E$, nodes highlighted in grey are not reachable by tokens injected in the sources. Circuit $p = \{1, 3, 4, 1\}$ in $\mathcal{G}$ has $C(p) = 0$.

Finally, consider the network $\mathcal{G}$ in Fig. 9.5, left, which has arc $(1, 3)$ with negative constrained cost $\sigma_{1,3} = -2$ and circuit $p = \{1, 3, 4, 1\}$ with $C(p) = -1$. The corresponding expanded network $\mathcal{G}_E$ is reported on the right. Note that $\mathcal{G}_E$ has now infinite possible routes, e.g., $p = \{1^0, 2^2, 4^3, 1^4, 3^2, 4^2, 1^3, 3^1, 4^1, 1^2, ..., 1^c, 3^{c-2}, 4^{c-2}, 1^{c-1}, ...\}$. All the states $x_i^c$ with $c \leq C_{max}$ are reachable, and $L$ is unbounded. This is the reason for which Assumption 7.1 is required, stating that there cannot be any negative circuit with respect to the costs $\sigma_{ij}$.

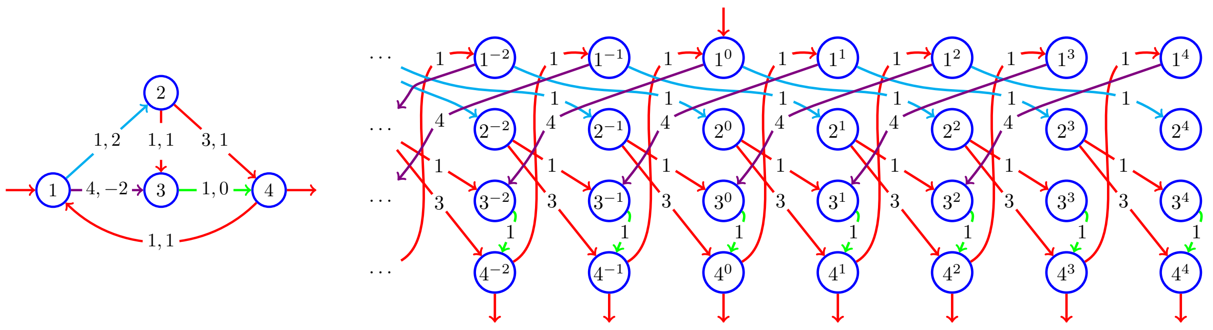


Figure 9.5: A simple network $\mathcal{G}$ (left) with an arc $(1, 3)$ with negative constrained cost $\sigma_{1,3} = -2$ and the corresponding expanded network $\mathcal{G}_E$ (right), assuming $C_{max} = 4$. In $\mathcal{G}_E$, all the nodes are reachable. Circuit $p = \{1, 3, 4, 1\}$ in $\mathcal{G}$ has $C(p) = -1$.

## 9.8 Non-integer arc costs

The same considerations from Section 8.7 regarding the possibility of using non-integer arc costs $\gamma_{ij}$ apply here, because of Remark 9.2. In particular, Remark 8.3 is simply adapted in Remark 9.4.

**Remark 9.4:**
Let $X_i^c$ be the number of tokens in node $i$ with constrained cost $c$, and $x_i^c$ the corresponding state/value. Applying the proposed policy with $\rho = 1$ and (integer) costs $\mu\widetilde{\gamma}_{ij}$ as thresholds is equivalent to applying it with $\rho = 1/\mu$ and (non-integer) costs $\widetilde{\gamma}_{ij}$ as thresholds. Indeed, the transition conditions are, respectively,

$$x_i^c - x_j^{c+\sigma_{ij}} = X_i^c - X_j^{c+\sigma_{ij}} > \mu\widetilde{\gamma}_{ij}, \quad \text{and} \quad c + \sigma_{ij} \le C_{max},$$

and

$$x_i^c - x_j^{c+\sigma_{ij}} = \rho X_i^c - \rho X_j^{c+\sigma_{ij}} = \frac{X_i^c - X_j^{c+\sigma_{ij}}}{\mu} > \widetilde{\gamma}_{ij}, \quad \text{and} \quad c + \sigma_{ij} \le C_{max},$$

which are equivalent.

Regarding the secondary costs $\sigma_{ij}$, let $\widetilde{\sigma}_{ij} \in \mathbb{Q}$ be the generic non-integer secondary costs, again, assumed rational. Then, as these secondary costs are used in the constraints of the paths, the policy continues to work. The main drawback of having rational costs is that the size of $\mathcal{C}$ increases, and so does the number of states $x_i^c$ and convergence time, too. Again, a value $\nu \in \mathbb{N}$ can be defined such that all the constrained costs $\nu\widetilde{\sigma}_{ij}$ are integer; as of Eq. (7.2) the secondary cost of a path $p$ becomes $\nu C(p)$, and constraint Eq. (7.3) is indeed equivalent to $\nu C(p) \le \nu C_{max}$.

## 9.9 An enhanced policy considering virtual tokens

As a result of Remark 9.2, the enhancement policy from Section 8.8 can be easily adapted to the constrained case, too, by using the state components $x_i^c$ instead of the scalar $x_i$.

However, now tokens that fall asleep are still lost, thus it cannot be guaranteed that initially all the injected tokens are collected in the sinks, even if the network is strongly connected.

**Policy 9.2: Enhanced decentralized threshold policy (constrained system).**
When an (informative) token with constrained cost $c \in \mathcal{C}$ reaches a node $i \in \mathcal{N}$ in which it falls asleep, i.e., if

$$c + \sigma_{ij} > C_{max} \quad \text{for all } (i,j) \in \mathcal{A}, j \in \mathcal{N}_i,$$

stop in node $i$. Otherwise, if the above-threshold condition does not hold for any outgoing arc of node $i$ that keeps the followed route feasible, i.e.,

$$x_i^c(k) + 1 - x_j^{c+\sigma_{ij}}(k) \le \gamma_{ij}, \quad \forall j \in \mathcal{N}_i \text{ such that } c + \sigma_{ij} \le C_{max},$$

instead of stopping the token, generate a block of $\gamma_{i\hat{j}} + x_{\hat{j}}^{c+\sigma_{i\hat{j}}}(k) - x_i^c(k)$ *virtual tokens* in $i$, where

$$\hat{j} = \underset{\substack{j \in \mathcal{N}_i,\ \text{s.t.} \\ c+\sigma_{ij} \le C_{max}}}{\arg\min} \{\gamma_{ij} + x_j^{c+\sigma_{ij}}(k) - x_i^c(k)\},$$

so that the value of state $x_i$ immediately becomes: $x_i^c(k) \to x_{\hat{j}}^{c+\sigma_{i\hat{j}}}(k) + \gamma_{i\hat{j}}$ and the arc $(i, \hat{j})$ becomes above-threshold. Then, the informative token moves to the node $\hat{j}$.

CHAPTER 10

# Illustrative example in a small network

In this Chapter, the proposed policy is applied to the tokens injected in the simple network of Fig. 7.2, both for the constrained and unconstrained system. Since the network is very small, the evolution of the states of the nodes and the routes traveled by the tokens can be easily analyzed.

These proposed policies have been implemented in C and compiled as Matlab MEX functions; simulations were performed in Matlab. All the simulations were performed on a dual-core Intel Core i3 at 2.3 GHz with 8 GB of RAM.

## 10.1 Scenario and data

The simple network of Fig. 7.2 is considered, which is composed of 5 nodes and 7 arcs. Tokens are injected periodically in the source node 1 and, as soon as they reach the sink node 5, they leave the network.

The initial state is zero for all the nodes. As all the arc costs are non-negative, such initial state is admissible.

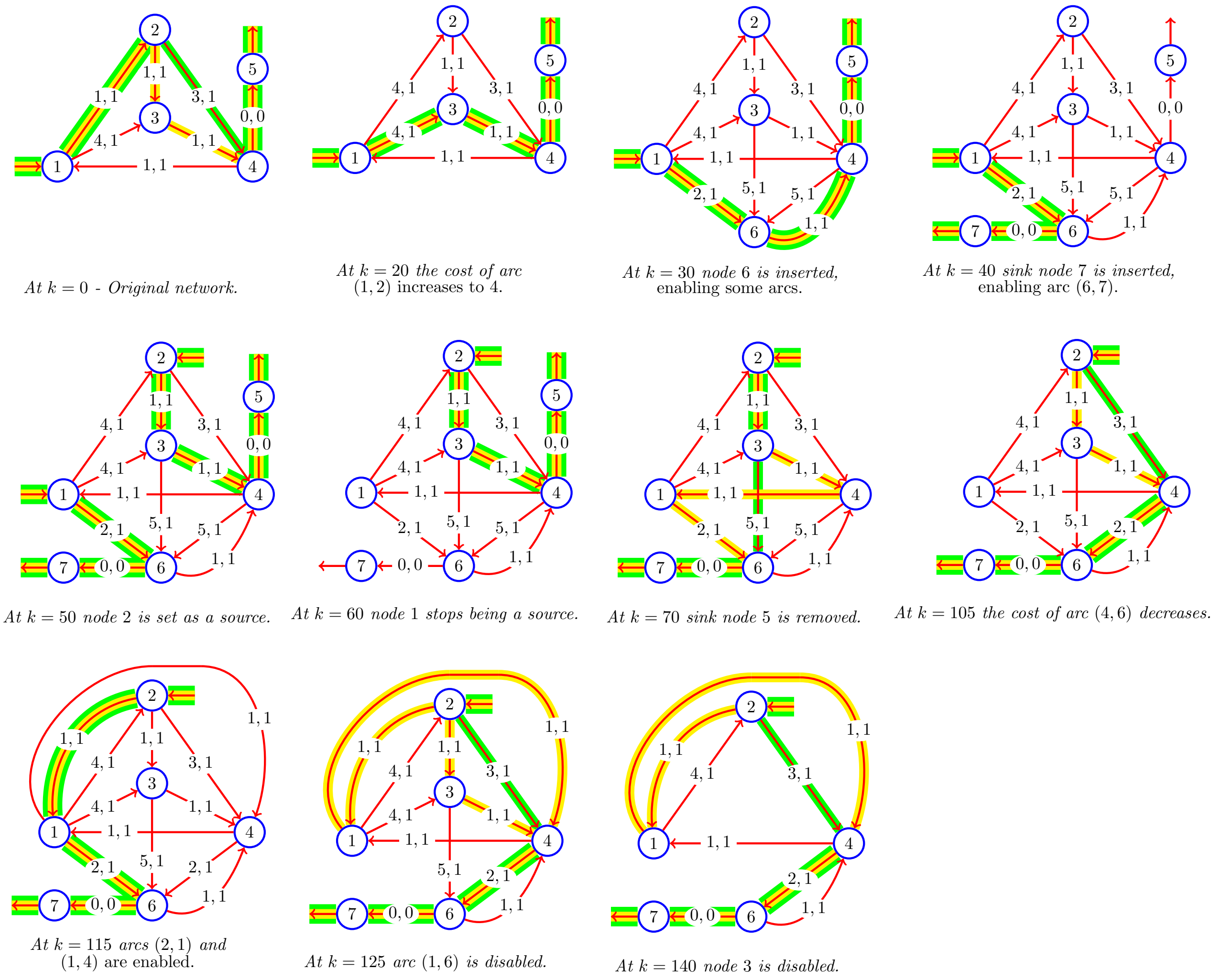
Initially, the network is assumed static, and the transitory is studied. Then, after the system stabilizes, the following modifications are applied (see Fig. 10.1):

- At $k = 20$, the cost of arc $(1, 2)$ is increased to $\gamma_{1,2} = 4$.
- At $k = 30$, node 6 is added, enabling arcs $(1, 6)$, $(3, 6)$, $(4, 6)$ and $(6, 4)$, with costs $\gamma_{1,6} = 2$, $\gamma_{3,6} = 5$, $\gamma_{4,6} = 5$, $\gamma_{6,4} = 1$, and $\sigma_{1,6} = \sigma_{3,6} = \sigma_{4,6} = \sigma_{6,4} = 1$.
- At $k = 40$, sink node 7 is added, which is connected to node 6 by an arc $(6, 7)$ with $\gamma_{6,7} = 0$ and $\sigma_{6,7} = 0$.
- At $k = 50$, node 2 is set as a source.
- At $k = 60$, node 1 stops being a source.
- At $k = 70$, sink node 5 is removed.
- At $k = 105$, the cost of arc $(4, 6)$ is decreased to $\gamma_{4,6} = 2$.
- At $k = 115$, arcs $(2, 1)$ and $(1, 4)$ are enabled, with $\gamma_{2,1} = \gamma_{1,4} = 1$ and $\sigma_{2,1} = \sigma_{1,4} = 1$.
- At $k = 125$, arc $(1, 6)$ is disabled.
- At $k = 140$, node 3 is disabled.

### 10.1.1 Simulation settings

To simulate real environments, the fast dynamic needs to be temporized. This means that all the elementary node-to-node transitions are to be accounted for explicitly.

The interval between two time instants $k$ and $k + 1$ of the slow dynamics corresponds to $\mathcal{N} + 1$ instants of the fast dynamics. A new token is injected at each time $k$ of the slow dynamic. Then, if the network is in an admissible state, the injected token is the only one in the network that is allowed to move; at each instant of the fast dynamic the injected token traverses an arc, until either it is not allowed to move anymore, or it has reached a sink: this occurs before the next token is injected. Conversely, if the network is in a non-admissible state, at each instant of the fast dynamic, simultaneous transactions involving different nodes are allowed.



*At $k = 0$ - Original network.*

*At $k = 20$ the cost of arc* $(1, 2)$ increases to 4.

*At $k = 30$ node 6 is inserted,* enabling some arcs.

*At $k = 40$ sink node 7 is inserted,* enabling arc $(6, 7)$.

*At $k = 50$ node 2 is set as a source.*

*At $k = 60$ node 1 stops being a source.*

*At $k = 70$ sink node 5 is removed.*

*At $k = 105$ the cost of arc $(4, 6)$ decreases.*

*At $k = 115$ arcs $(2, 1)$ and* $(1, 4)$ are enabled.

*At $k = 125$ arc $(1, 6)$ is disabled.*

*At $k = 140$ node 3 is disabled.*

Figure 10.1: (*A simple network*). The initial network at time $k = 0$ and the successive modifications occurred at the indicated time $k$. The existing unconstrained shortest paths are hightailed in yellow, and the existing constrained shortest paths in green.

Moreover, whenever the agent is in a node that has more than one outgoing arc satisfying the above-threshold condition, a choice model must be specified for determining in which node to move in. Two different types of choices are considered for the next node to move in:

- *deterministic choice model*: the agent scans the outgoing arcs in a predefined order and chooses to traverse the first one that is above-threshold;
- *stochastic choice model*: the agent randomly chooses the outgoing above-threshold arc to traverse.

## 10.2 Metrics

The following metrics are introduced to evaluate the system, which are computed once the global rest state (steady-state) is reached, for each network configuration:

- $L_{ss}$: average length $L(p)$ of the paths travelled by the tokens injected at steady-state;
- $C_{ss}$: average constrained cost $C(p)$ of the paths travelled by the tokens injected at steady-state;
- $E_{ss}$: average number of arcs of the paths traveled by the tokens injected at steady-state;
- $T_{ss}$: time (expressed in number of injected tokens) needed to reach a global rest state (steady-state), where tokens injected in any source will always leave the network, without accumulating in the nodes, i.e., $k$ such that $V(x(k)) = \bar{V}$;
- $V_{ss}$: total number of tokens deposited in the nodes when the global rest state (steady-state) is reached, i.e., $V(x(k)) = \bar{V}$;

- $l_{ss}$: number of injected tokens that have been lost during the transitory, when the global rest state (steady-state) is reached, because either they are stopped in a node, or their path is not feasible, or the node they are deposited has been removed. This is computed as the difference between the tokens injected in the network over a considered time interval, minus the tokens which are expelled through the sink nodes, in the same time interval. A negative value of $l_{ss}$ is possible when at the beginning of the considered time interval the state is not admissible, because in that case, some tokens already in the network might be able to move and leave the network.
- $N_{na}$: time (expressed in number of injected tokens) needed for the state of the network to become admissible, if the initial state is non-admissible. This may happen when there are negative-weighted arcs and zero initial state, or when the initial state is non-zero, or when the network changes. 0 means the initial state is already admissible.

## 10.3 Results and discussion

The results reported below consider the deterministic choice model. Indeed, for this simple network, the results adopting the stochastic choice model are almost the same as the ones adopting the deterministic choice model, with little to no differences, hence they are omitted. The reason is that tokens rarely have more than one possibility for choosing the next node to move in.

At first, the network has been considered static (see the original network at $k = 0$ in Fig. 10.1). The paths followed by the first injected tokens at each instant of the slow dynamics $k = 1, 2, ...$ until a global rest state is reached are displayed in Fig. 10.2, for both the unconstrained and constrained cases, adopting both the original policy and the enhanced one. The effects on the states of the nodes can be retrieved from Table 10.1, where the values of such states after each transition are reported. It can be seen that a global rest state is reached very fast, in very few time instants; as expected, the enhanced policy is faster than the original one.

Regarding the behavior of the tokens injected during the initial transitory, consider the original policy. From Fig. 10.2 and Table 10.1, the first few injected tokens cannot reach the sink nodes, as they are stopped in the internal nodes of the network, increasing the corresponding states. In particular, when a token, at time $k$, reaches a non-sink node $j$ along a path traversing $P > 1$ nodes, the next injected tokens will travel paths that are shorter and shorter in terms of traversed arcs. Once the traveled path is actually composed only by the source node, a longer traversed path composed by $P$ or more tokens can be traversed (a simple explanation for this fact follows from subsection 6.1.2 at Page 74). In general, a behavior of this kind emerges also in more complex networks and paths, although it must be taken into account that generally many alternative paths may exist, so this process will take longer: depending on the adopted policy, tokens might continue taking longer paths as long as they are available. When the initial state is not zero, the behavior of the tokens is similar, however, it is faster, since the buffers of the nodes might be already partially filled. After the transient, all the injected tokens are routed through the shortest paths.

For the constrained case, similar considerations can be made, however now tokens stop also because their traveled paths would become infeasible if they continued moving. Then, all the traveled paths are feasible, i.e., their constrained cost is at most $C_{max} = 2$. After the transient, all the tokens are routed through the constrained shortest paths. Compared to the unconstrained case, more tokens are required to reach a steady-state. This is expected, since the constrained problem has a larger complexity and can be equivalently seen as an unconstrained problem in the network in Fig. 9.1, which is about $C_{max}$ times larger than the original one in Fig. 7.2.

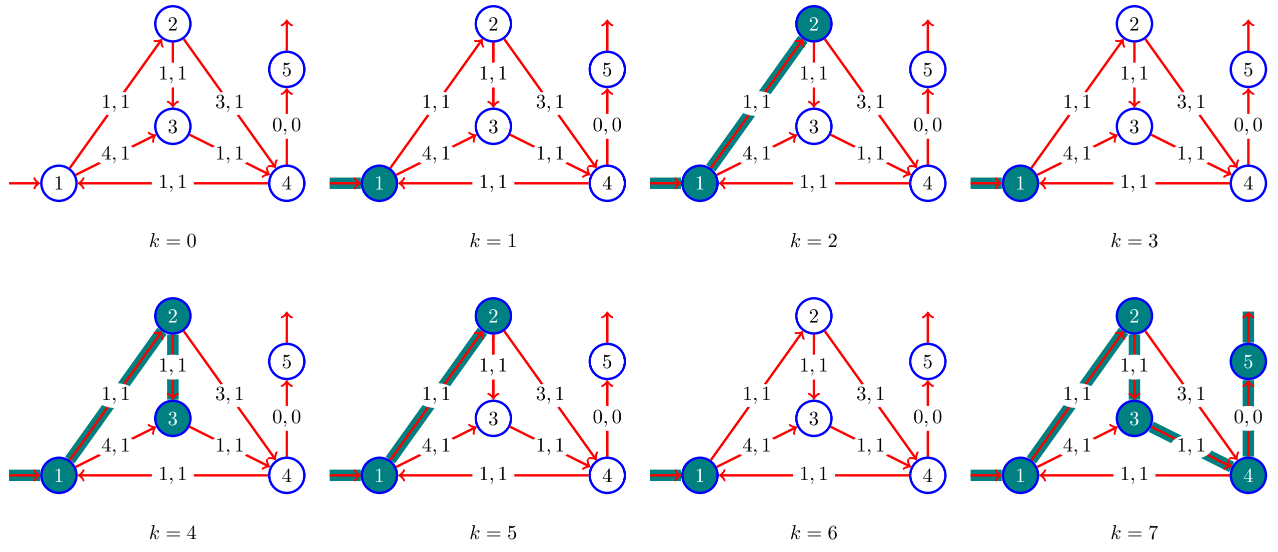
Note that, in both cases, the number of tokens $V(x(k))$ in the system increases by 1 at each token insertion, until a global rest state is reached.

Regarding the enhanced policy, it can be seen that the global rest state is reached faster: the improvements coming from the enhanced policy are, however, not that much evident here, due to the simplicity of the network.

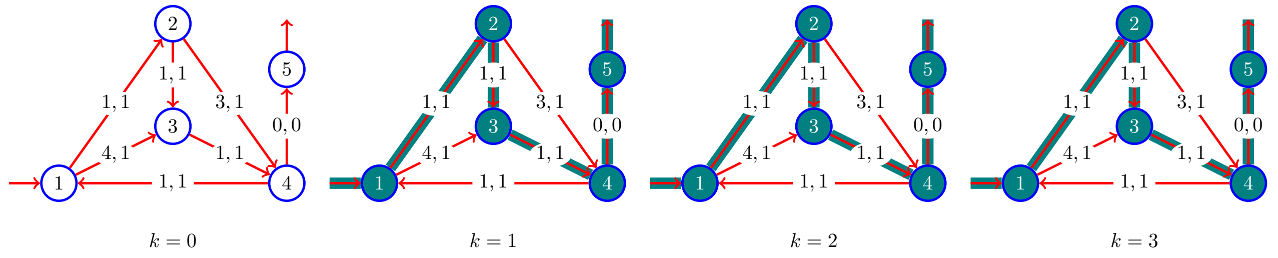
For the unconstrained scenario, all injected tokens actually leave the network through the shortest path (see Fig. 10.2). While it is expected that all the injected tokens can leave the network, the fact that this occurs along the shortest outgoing path is specific to this particular simple example. Still, from Table 10.1, the steady-state is reached only after few time instants of the slow dynamic; indeed, the first few injected tokens modify the state of the nodes along their traveled paths, to proceed.

For the constrained scenario, even in this simple example, the enhanced policy does not prevent the first injected tokens to take infeasible paths. Still, it takes less time for these paths to be blocked, due to the first injected tokens that modify the state of the nodes along their traveled paths to continue moving until such paths become infeasible. Eventually, all the injected tokens are routed through the constrained shortest paths.

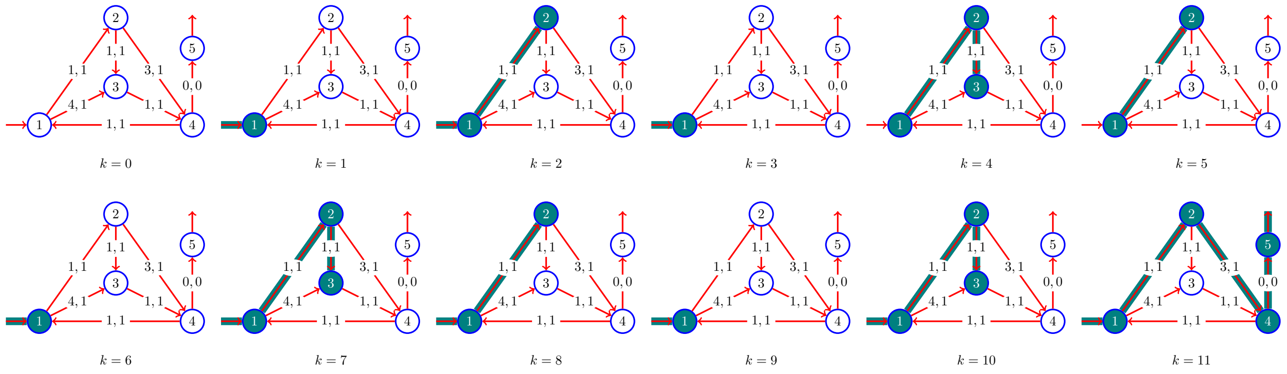
Note that now, in both cases, the number of tokens $V(x(k))$ in the system can increase by more than 1 at each token insertion, until a global rest state is reached.



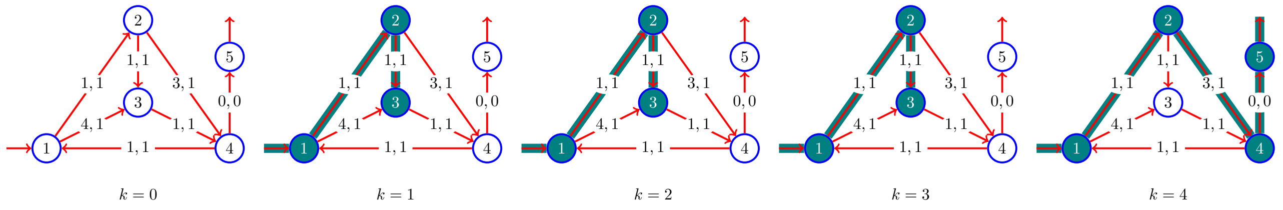
(a) Unconstrained system, original policy.



(b) Unconstrained system, enhanced policy.



(c) Constrained system, original policy.



(d) Constrained system, enhanced policy.

Figure 10.2: (*A simple network*). The path travelled by the first tokens injected at times $t_k$, i.e., the $k$th instants of the slow dynamic, during the initial transitory, until the steady-state is reached. If a path stops in a node, it means that the token is deposited in that node. For the enhanced policy, states are also modified along the way. The effects on the states of the nodes can be retrieved from Table 10.1, where the values of such states are reported.

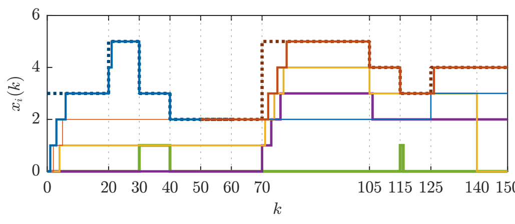
Table 10.1: (*A simple network*). The state $x_i(k)$ of the nodes reached after the transitions of the token injected at time $t_k$, i.e., the $k$th instant of the slow dynamic, during the initial transitory (see the paths in Fig. 10.2). If a constrained system is considered, the state components $x_i^c(k)$ are also indicated. Dots indicate that the steady-state is reached. Bold numbers indicate the states that have just been modified, due to the token stopping in a node, or, for the enhanced policy, to virtual tokens.

| | | **Original policy** | | | | | | | | | | | | | **Enhanced policy** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ***k*:** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **0** | **1** | **2** | **3** | **4** | **5** | **6** |
| **Unconstrained system** | $x_1$ | 0 | **1** | 1 | **2** | 2 | 2 | **3** | 3 | $\cdots$ | | | | | 0 | **1** | **2** | **3** | 3 | $\cdots$ | |
| | $x_2$ | 0 | 0 | **1** | 1 | 1 | **2** | 2 | 2 | $\cdots$ | | | | | 0 | **1** | **2** | 2 | 2 | $\cdots$ | |
| | $x_3$ | 0 | 0 | 0 | 0 | **1** | 1 | 1 | 1 | $\cdots$ | | | | | 0 | **1** | 1 | 1 | 1 | $\cdots$ | |
| | $x_4$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $\cdots$ | | | | | 0 | 0 | 0 | 0 | 0 | $\cdots$ | |
| | $x_5$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $\cdots$ | | | | | 0 | 0 | 0 | 0 | 0 | $\cdots$ | |
| | $V(x)$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 6 | $\cdots$ | | | | | 0 | 3 | 5 | 6 | 6 | $\cdots$ | |
| **Constrained system** | $x_1^0$ | 0 | **1** | 1 | **2** | 2 | 2 | **3** | 3 | 3 | **4** | 4 | 4 | $\cdots$ | 0 | **1** | **2** | **3** | **4** | 4 | $\cdots$ |
| | $x_1^1,\ x_1^2$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $\cdots$ | 0 | 0 | 0 | 0 | 0 | 0 | $\cdots$ |
| | $x_2^0,\ x_2^2$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $\cdots$ | 0 | 0 | 0 | 0 | 0 | 0 | $\cdots$ |
| | $x_2^1$ | 0 | 0 | **1** | 1 | 1 | **2** | 2 | 2 | **3** | 3 | 3 | 3 | $\cdots$ | 0 | **1** | **2** | **3** | 3 | 3 | $\cdots$ |
| | $x_3^0,\ x_3^1$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $\cdots$ | 0 | 0 | 0 | 0 | 0 | 0 | $\cdots$ |
| | $x_3^2$ | 0 | 0 | 0 | 0 | **1** | 1 | 1 | **2** | 2 | 2 | **3** | 3 | $\cdots$ | 0 | **1** | **2** | **3** | 3 | 3 | $\cdots$ |
| | $x_4^0,\ x_4^1,\ x_4^2$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $\cdots$ | 0 | 0 | 0 | 0 | 0 | 0 | $\cdots$ |
| | $x_5^0,\ x_5^1,\ x_5^2$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $\cdots$ | 0 | 0 | 0 | 0 | 0 | 0 | $\cdots$ |
| | $x_1$ | 0 | **1** | 1 | **2** | 2 | 2 | **3** | 3 | 3 | **4** | 4 | 4 | $\cdots$ | 0 | **1** | **2** | **3** | **4** | 4 | $\cdots$ |
| | $x_2$ | 0 | 0 | **1** | 1 | 1 | **2** | 2 | 2 | **3** | 3 | 3 | 3 | $\cdots$ | 0 | **1** | **2** | **3** | 3 | 3 | $\cdots$ |
| | $x_3$ | 0 | 0 | 0 | 0 | **1** | 1 | 1 | **2** | 2 | 2 | **3** | 3 | $\cdots$ | 0 | **1** | **2** | **3** | 3 | 3 | $\cdots$ |
| | $x_4$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $\cdots$ | 0 | 0 | 0 | 0 | 0 | 0 | $\cdots$ |
| | $x_5$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $\cdots$ | 0 | 0 | 0 | 0 | 0 | 0 | $\cdots$ |
| | $V(x)$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 10 | $\cdots$ | 0 | 3 | 6 | 9 | 10 | 10 | $\cdots$ |

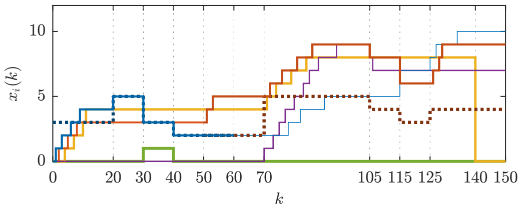
Consider now the simulation over a larger time period, where modifications are applied over time to the network, producing different configurations (see Fig. 10.1). Since the adaption time is generally almost immediate even when adopting the original policy, the results obtained when adopting the enhanced policy are omitted.

The behavior of the states of the nodes of the network is reported in Fig. 10.3. In particular, in Fig. 10.3a the time evolution of the states $x_i$ of each node $i$ is depicted for the unconstrained system. Initially, the system reaches a global rest state $\bar{x}$ starting from a zero state, and, after each modification, the system adapts way faster, without resetting the state. Some modifications have longer adaption times (e.g., at $k = 70$, when a sink is removed); still, a new global rest state $\bar{x}$ is reached faster than starting from an empty state.

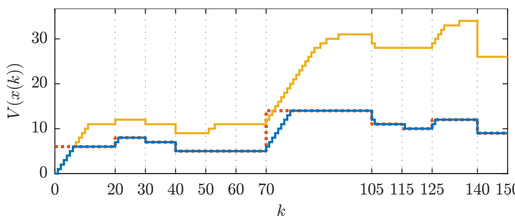
The time evolution of the states $x_i^c(k)$ and $x_i(k) = \sum_{c=0}^{C_{max}} x_i^c(k)$ when paths are constrained by $C_{max} = 2$ is reported in Figs. 10.3b and 10.3d. The state variables converge to higher values than the corresponding ones in the unconstrained case, because there are in general more than one non-zero components of the state in each



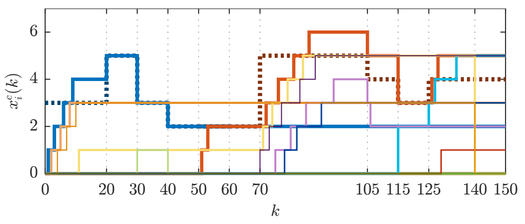
(a) $x_i(k)$ for each node of the unconstrained system. Dotted line: $\bar{\bar{x}}_i(k)$ for the sources.



(b) $x_i(k) = \sum_{c=0}^{C_{max}} x_i^c(k)$ for each node of the constrained system, $C_{max} = 2$. Dotted: $\bar{\bar{x}}_i(k)$ for the sources of the unconstrained system.



(c) Blue: $V(x(k))$ for the unconstrained system; dotted red: $V(\bar{\bar{x}}(k))$ for the unconstrained system; yellow: $V(x(k))$ for the constrained system with $C_{max} = 2$.



(d) $x_i^c(k)$, $c \in \{0, \ldots, C_{max}\}$, for each node of the constrained system, $C_{max} = 2$. Dotted line: $\bar{\bar{x}}_i(k)$ for the sources of the unconstrained system.

Figure 10.3: (*A simple network*). Time evolution of the states of the nodes, considering the modifications from Fig. 10.1 on the network.

node, and the length of the shortest feasible paths might be larger than that of the unconstrained shortest paths.

In Fig. 10.3c, the $V(x(k))$ of both systems are compared. It is confirmed that the unconstrained system always converges faster than the constrained one, i.e., overall fewer tokens are required. When modifying the network, the maximal rest state $\bar{\bar{x}}$ might change, and so does $V(\bar{\bar{x}})$; as expected, $V(x(k)) \leq V(\bar{x}(k))$ for the unconstrained system and, as the network is very simple, it *saturates* at global rest state, namely $V(\bar{x}(k)) = V(\bar{\bar{x}}(k))$.

Some animations of these simulations are available at `https://users.dimi.uniud.it/~franco.blanchini/examples_mkv.html`, showing the evolution over time of the nodes' states and the paths followed by the tokens.

The metrics introduced in Section 10.2 are reported in Table 10.2, for each possible network configuration; each interval starts at the time $k$ in which a change occurs in the network and stops when the next change occurs.

Analyzing the metrics $L_{ss}, C_{ss}, E_{ss}$ and comparing them with the paths depicted in Fig. 10.1 for each network configuration, it can be verified that the shortest paths are discovered, both constrained and unconstrained. For some configurations of the network, the shortest paths on the unconstrained and constrained system coincide (see Fig. 10.1); the time to reach a steady-state and the total number of tokens in the network is not necessarily the same, however, because of the slightly different policy that uses multi-component states in the constrained system, and the fact that during the transitory some paths with constrained costs greater than $C_{max}$, which are not viable in the constrained case, might be traversed by the tokens in the unconstrained case.

Removing arcs/nodes/sinks or increasing arc costs might increase the accumulation of tokens in the network, as the length of the shortest paths might increase (see metric $V_{ss}$). Conversely, restoring nodes/arcs or inserting sinks, or decreasing the arc costs might decrease the number of tokens in some nodes, as the length of the shortest paths might decrease. In these four latter cases, at $k = 30, 40, 105, 115$, the network briefly becomes non-admissible (see metric $T_{na}$); a new admissible state is reached very quickly. Removing sources does not have any effect, while inserting some of them makes the corresponding states increase, reaching the maximal values.

When the network is admissible, some injected tokens (the only ones moving) cannot reach the sinks, because they are accumulated in the nodes instead, and they get lost, see the metric $l_{ss}$. Many tokens are lost during the initial transitory, while very few during the successive ones, after each modification on the network. The main exception is the modification at $k = 70$, where a sink node is removed: the reason is that outgoing paths to this node are no more available, and new ones are to be discovered. Instead, when the network is not admissible, some tokens that are already present in the nodes can move, possibly leaving the network: this is the reason for which the metric $l_{ss}$ assumes negative values (more tokens leave the network than the ones injected in a given time interval). Eventually, in the unconstrained system, 9 of the 150 injected tokens stop in the nodes, without reaching the sinks, and 3 are lost due to the failing of node 3. In the constrained case, 26 of the 150 injected tokens stop in the nodes, and 8 are lost due to the failing of node 3.

Finally, as already stated, the network is so simple that applying a deterministic or stochastic choice model produces very few differences, as the number of alternative paths is reduced. The main difference can be seen when $k \in [125, 139]$, where two shortest (unconstrained) outgoing paths exists (see Fig. 10.1), which have the same minimum length. When adopting the deterministic choice model, only one of them is chosen, by all the injected tokens; when adopting a stochastic model, each token takes one of the two randomly.

Table 10.2: (*A simple network*). Comparison of the simulation results at each configuration of the network, considering the modifications from Fig. 10.1 on the network. A deterministic choice model is assumed.

| Time interval | Unconstrained, original | | | | | | | Constrained, original | | | | | | | $C_{max}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $L_{ss}$ | $C_{ss}$ | $E_{ss}$ | $T_{ss}$ | $V_{ss}$ | $l_{ss}$ | $T_{na}$ | $L_{ss}$ | $C_{ss}$ | $E_{ss}$ | $T_{ss}$ | $V_{ss}$ | $l_{ss}$ | $T_{na}$ | |
| [ 0, 19] | 3 | 3 | 4 | 6 | 6 | 6 | 0 | 4 | 2 | 3 | 10 | 10 | 10 | 0 | 2 |
| [ 20, 29] | 5 | 2 | 3 | 1 | 8 | 2 | 0 | 5 | 2 | 3 | 1 | 12 | 2 | 0 | 2 |
| [ 30, 39] | 3 | 2 | 3 | 0 | 7 | -1 | 1 | 3 | 2 | 3 | 0 | 11 | -1 | 1 | 2 |
| [ 40, 49] | 2 | 1 | 2 | 0 | 5 | -2 | 1 | 2 | 1 | 2 | 0 | 9 | -2 | 1 | 2 |
| [ 50, 59] | $\{2,2\}$ | $\{1,2\}$ | $\{2,3\}$ | 0 | 5 | 0 | 0 | $\{2,2\}$ | $\{1,2\}$ | $\{2,3\}$ | 3 | 11 | 2 | 0 | 2 |
| [ 60, 69] | 2 | 2 | 3 | 0 | 5 | 0 | 0 | 2 | 2 | 3 | 0 | 11 | 0 | 0 | 2 |
| [ 70, 99] | 5 | 4 | 5 | 8 | 14 | 9 | 0 | 6 | 2 | 3 | 17 | 29 | 18 | 0 | 2 |
| [105, 114] | 4 | 3 | 4 | 1 | 11 | -3 | 2 | 5 | 2 | 3 | 0 | 27 | -2 | 1 | 2 |
| [115, 124] | 3 | 2 | 3 | 1 | 10 | -1 | 2 | 3 | 2 | 3 | 0 | 27 | 0 | 1 | 2 |
| [125, 139] | 4 | 3 | 4 | 1 | 12 | 2 | 0 | 5 | 2 | 3 | 3 | 31 | 4 | 0 | 2 |
| [140, 150] | 4 | 3 | 4 | 0 | 9 | 0 | 0 | 5 | 2 | 3 | 0 | 23 | 0 | 0 | 2 |

*Note:* Each row refers to a different network configuration (see Fig. 10.1). $C_{max}$ is given for the constrained system. **Metrics.** $L_{ss}$: average length $L(p)$ of the paths travelled by the tokens injected at steady-state; $C_{ss}$: the corresponding constrained cost $C(p)$; $E_{ss}$: the corresponding number of traversed arcs; $T_{ss}$: time (number of injected tokens) needed to reach a global rest state, i.e., $k$ such that $V(x(k)) = \bar{V}$; $V_{ss}$: the corresponding value of $V(x(k)) = \bar{V}$ at global rest state; $l_{ss}$: total number of tokens lost during the transitory, because either they are stopped in a node, or their path is not feasible, or the node they are deposited has been removed; $T_{na}$: time (number of injected tokens) needed for the state of the network to become admissible, if the initial state in non-admissible. For the time interval [50, 59], $L_{ss}, C_{ss}, E_{ss}$ are inficated for both the sources, separately.

# Chapter 11

# Illustrative example in a large grid network

In this Chapter the proposed policies Policies 8.1 and 9.1 are applied to a large grid network of 2500 nodes and 19404 arcs with integer (possibly negative) arcs' costs. The choice of the grid network is due to being able to visualize the results easily, as well as some emergent behaviours, i.e., macroscopic phenomenons that can be observed only in larger systems, and that are not evident at microscopic/local level. First, a static scenario is considered, and the effects of applying a deterministic or stochastic choice model are evaluated. Then, some modifications on the network are applied, to show that the policies are adaptive in dynamic environments.

These proposed policies have been implemented in C and compiled as Matlab MEX functions; simulations were performed in Matlab. All the simulations were performed on a dual-core Intel Core i3 at 2.3 GHz with 8 GB of RAM.

## 11.1 Scenario and data

Consider the map of size $50 \times 50$ pixels represented in Fig. 11.1a, where each pixel is associated with a node $i$ and is characterized by an integer value representing its "altitude" $h_i$. A network is created, connecting each node to all the existing neighbor nodes in its 8-neighborhood through an arc. To assign a (possibly negative) cost to each arc $(i, j)$ ensuring Assumption 7.1, the values $h_i$ and $h_j$ of the pixels associated with $i$ and $j$, respectively, are considered and, given the difference $dh = h_j - h_i$, such cost is computed as:

$$\gamma_{ij} = \begin{cases} ceil(m^-(dh - h_0)), & \text{if } dh \leq h_0, \\ ceil(m^+(dh - h_0)), & \text{otherwise,} \end{cases}$$

where $h_0 \in \mathbb{Z}, h_0 < 0$, represents the difference $dh$ associated with a zero cost, and $m^-, m^+ \in \mathbb{R}$, with $0 < m^- < m^+$, ensure positive circuits. Negative gradients of $h_i$ such that $dh < h_0$ correspond to negative cost arcs. Intuitively, when traveling uphill some energy needs to be spent (positive costs), while when traveling downhill some energy can be recovered (negative costs). Indeed, this expression is a simplification of the energy conservation rule for energy recuperation systems (e.g., regenerative braking for electric vehicles): physically, due to the conservation of energy, traveling in a circuit will never result in a negative consumption (or, equivalently, a positive regeneration).

The following parameters have been used: $h_0 = -30, m^- = 0.4$ and $m^+ = 0.9$. Also, the constrained cost is set to 1 for all arcs $\sigma_{ij} = 1$ for each arc, i.e., the number of traversed arcs is limited.

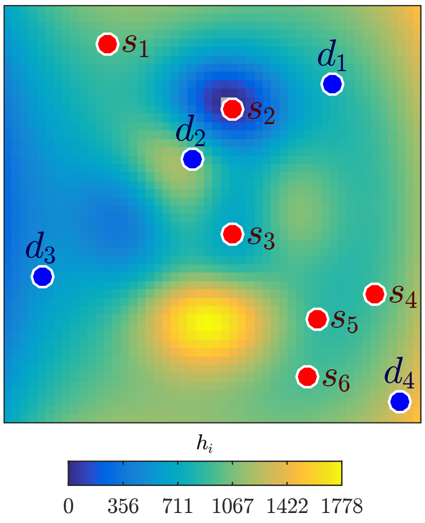
The resulting network has 2500 nodes and 19404 arcs, with $\gamma_{ij} \in [-97, 273]$. Six sources $s_1, \ldots, s_6$ and four sinks $d_1, \ldots, d_4$ are set (see red and blue dots in Fig. 11.1a, respectively).

For evaluation purposes, the length of the shortest path from each node to each sink can be easily computed offline, using the Bellman-Ford algorithm. Similarly, to determine the (minimum) value of the total constrained cost $C_{max}$ to be imposed, the minimum constrained cost from each source to each sink can be computed, too. For the given source nodes, these results are summarized in Table 11.1. $C_{max} = 25$ is considered.

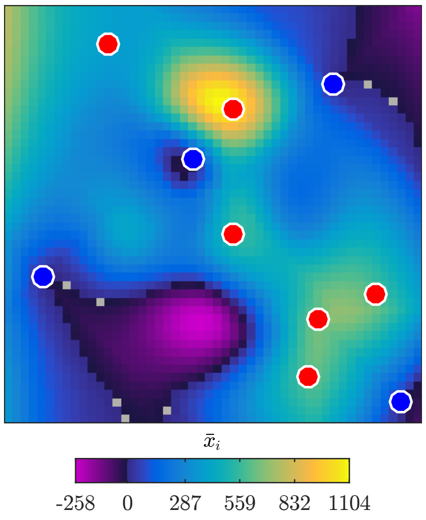
Also, the maximal rest state $\bar{\bar{x}}$ for the unconstrained system is reported in Fig. 11.1b, where each pixel, associated with a node $i$, shows the value of $\bar{\bar{x}}_i$, i.e., the minimum distance from $i$ to the closest sink. Recall that the maximal rest state is independent from where the sources are located. Obviously, nodes around the sinks have

Table 11.1: (*A large grid network*). Minimum length $L_{min}$ and minimum constrained cost $C_{min}$ of the paths between each source $s_i$ and each sink $d_i$. Here, $C_{min}$ is also the constrained cost of the shortest paths (except for $s_2 \to d_3$, where the shortest path has $C = 31$). Values in bold refer to the shortest (unconstrained) path from each source. Values in italics refer to the shortest feasible path from each source when $C_{max} = 25$.

| | $d_1$ | | $d_2$ | | $d_3$ | | $d_4$ | |
|---|---|---|---|---|---|---|---|---|
| | $L_{min}$ | $C_{min}$ | $L_{min}$ | $C_{min}$ | $L_{min}$ | $C_{min}$ | $L_{min}$ | $C_{min}$ |
| $s_1$ | 608 | 27 | *496* | *14* | **383** | **28** | 1527 | 43 |
| $s_2$ | ***1058*** | ***12*** | 1141 | 6 | 1367 | 23 | 2133 | 35 |
| $s_3$ | 637 | 18 | 637 | 9 | ***529*** | ***23*** | 1145 | 20 |
| $s_4$ | ***597*** | ***25*** | 817 | 22 | 776 | 40 | 720 | 13 |
| $s_5$ | 698 | 28 | 750 | 19 | **635** | **33** | *660* | *10* |
| $s_6$ | 844 | 35 | 896 | 26 | **608** | **32** | *640* | *11* |



(a) Map of $h_i$. There are hills in the yellow areas, and the altitude decreases as the color becomes bluer: arc costs $\gamma_{ij}$ are negative for the arcs directed downhill, and non-negative otherwise. Note that tokens injected in $s_2$ have to travel uphill to reach a sink, along a positive length path. Conversely, possible tokens present in the yellow areas tend to travel downhill, along negative paths.



(b) Map of the maximal rest states $\bar{\bar{x}}_i$ for the unconstrained system, which are equal to the length of the shortest path from each node $i$ to the closest sink. There is a larger accumulation of tokens in the yellow areas, e.g., around source $s_2$, as the tokens injected in there have to travel uphill to reach a sink, paying a positive cost. The number of tokens is below the zero reference level (negative states) in the violet areas, e.g., in the areas with the larger altitude, as tokens tend to go downhill to reach a sink, paying a negative cost.

Figure 11.1: (*A large grid network*). The map of the network. Red circles: source nodes; blue circles: sink nodes, colored pixels: nodes of the network. The color of each pixel represents the corresponding value of $h_i$ and $\bar{\bar{x}}_i$, respectively, with gray representing 0. Each node is connected to its existing 8-neighborhood nodes.

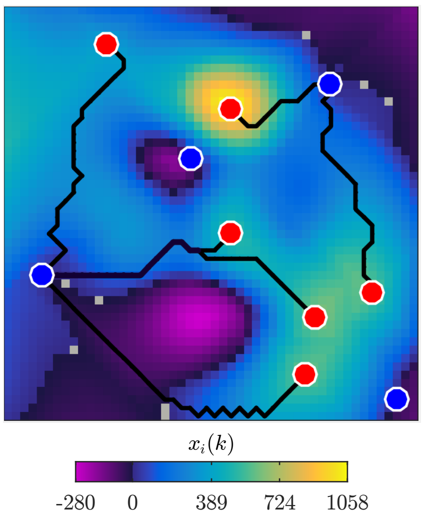
$\bar{\bar{x}}_i$ close to zero. Nodes from which the shortest path to the closest sink is directed downhill tend to have negative $\bar{\bar{x}}_i$, while if such path is directed uphill, the corresponding $\bar{\bar{x}}_i$ is positive.

Initially, this network, initialized with a zero state, is assumed static and the initial transitory that leads the state of the network to a global rest state is analyzed. Recall that since there are some negative cost arcs, initially the state is non-admissible, and some transitions of virtual/non-informative tokens occur, but eventually, an admissible state is reached and, later on, the system reaches a global rest state.

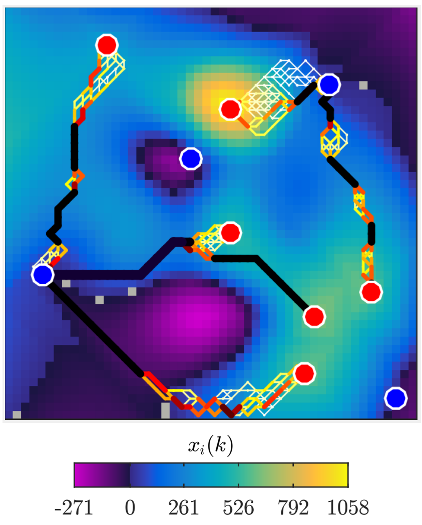
Then a dynamic scenario is considered, evolving from the previous case; successive modifications on the network are performed, each one occurring after the system stabilizes (see Fig. 11.7 at Page 125, first columns).

- At $k = 18000001$, some nodes are removed.
- At $k = 21000001$, some of these are re-enabled.
- At $k = 24000001$, some nodes stop being sources.
- At $k = 27000001$, some nodes become new sources.
- At $k = 42000001$, some nodes become new sinks.
- At $k = 45000001$, some nodes stop being sinks.

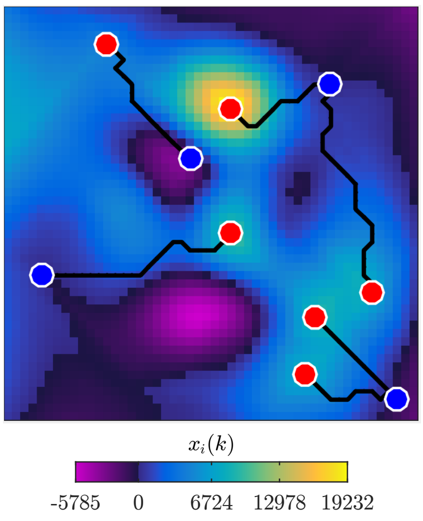
Both the stochastic and deterministic choice models are considered, for both the unconstrained and constrained system, applying the original Policies 8.1 and 9.1. The simulation settings are those from subsection 10.1.1. Metrics from Section 10.2 have been computed from each simulated case and compared.



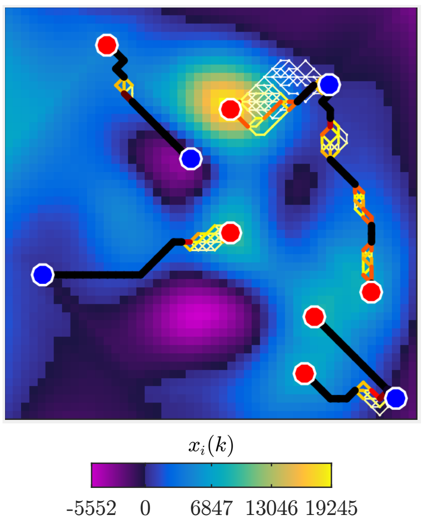
(a) Map of the rest state $\bar{x}_i(k)$ for the unconstrained system, deterministic choices.



(b) Map of the rest state $\bar{x}_i(k)$ for the unconstrained system, stochastic choices.



(c) Map of the rest state $\bar{x}_i(k)$ for the constrained system, $C_{max} = 25$, deterministic choices.



(d) Map of the global rest state $\bar{x}_i(k)$ for the constrained system, $C_{max} = 25$, stochastic choices.

Figure 11.2: (*A large grid network*). The map of the network. Red circles: source nodes; blue circles: sink nodes, colored pixels: nodes of the network. The color of each pixel represents the corresponding value of $\bar{x}_i(k)$ at steady-state (global rest state), with gray representing 0. The lines represent the paths traveled by 10000 injected tokens; from white thin lines, when few tokens traversed each arc, to thicker black lines, when almost all the tokens traversed them.

## 11.2 Results and discussion

At first, consider the static scenario in the network from Fig. 11.1. In Fig. 11.2, the steady-state of the network is represented, for both the unconstrained system and the constrained system (with $C_{max} = 25$), applying both the deterministic and stochastic choice model. The unconstrained systems, for both the choice models, reach such global rest states around time $k = 10^6$, while both the constrained systems around time $k = 1.35 \cdot 10^7$.

In the images, the color of each pixel now represents the state $x(k)$ of the nodes at steady-state (i.e., it is a global rest state $x(k) = \bar{x}$). For simplicity, in the constrained systems $\bar{x}_i = \sum_{c=0}^{C_{max}} \bar{x}_i^c$ is represented for each node $i$. The state $\bar{x}$ mostly does not depend on the applied choice model. In the unconstrained case, states are upper bounded by the maximal rest state from Fig. 11.1b, $\bar{x} \leq \bar{\bar{x}}$, as expected, and for many nodes they are even the same. Moreover, the states are way larger for the constrained system, because in that case the individual state components $\bar{x}_i^c$ for $c = 0, 1, \dots, C_{max} = 25$ are all accounted in $\bar{x}_i$.

The paths traveled by 10000 injected tokens (at steady-state) are also depicted, which are the shortest paths from each source to the closest sinks. For the unconstrained system, the global rest states $\bar{x}$ in Figs. 11.2a and 11.2b are characterized by the following states of the source nodes $s_1, \dots, s_6$: $\bar{x}_{s_1}(k) = 383$, $\bar{x}_{s_2}(k) = 1058$, $\bar{x}_{s_3}(k) = 529$, $\bar{x}_{s_4}(k) = 597$, $\bar{x}_{s_5}(k) = 635$, $\bar{x}_{s_6}(k) = 608$, corresponding to the optimal path lengths (see Table 11.1). More generally, $\bar{x}_i = \bar{\bar{x}}_i$ for all the nodes $i$ along such paths.

Instead, for the constrained system, from Table 11.1 it can be seen that the shortest outgoing paths from $s_1$,

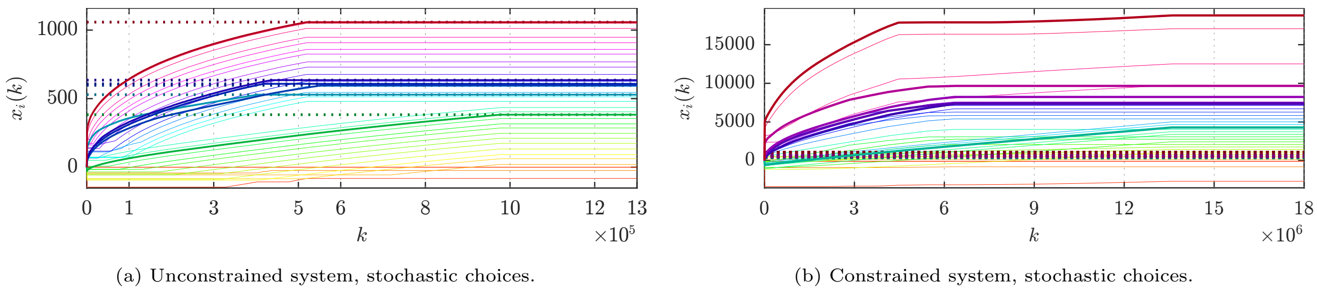
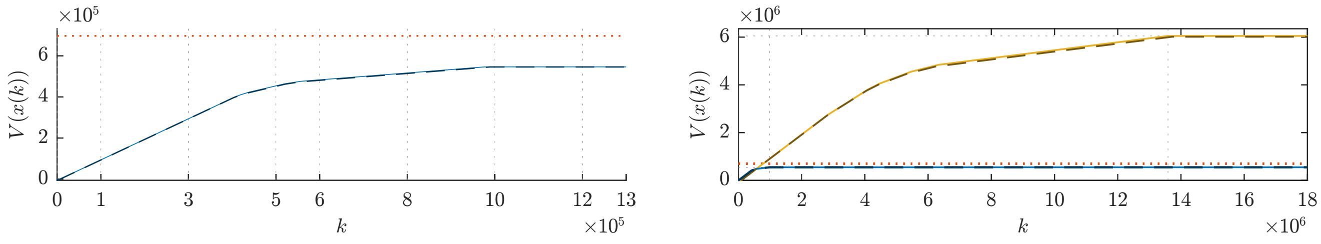


Figure 11.3: (*A large grid network*). Time profile of the states $x_i(k)$ of a subset of nodes, including the sources, which are highlighted by thick lines. Dotted lines represent $\bar{\bar{x}}_i(k)$ for the source nodes computed for the unconstrained case: the state of the source nodes converge to these values in the unconstrained case.



(a) Detail of the initial transitory for the unconstrained system. Solid lines: stochastic choice model; darker dashed lines: deterministic choice model.

(b) Detail of the initial transitory for the constrained system. Solid lines: stochastic choice model; darker dashed lines: deterministic choice model.

Figure 11.4: (*A large grid network*). Blue: $V(x(k))$ for the unconstrained system; dotted red: $V(\bar{\bar{x}}(k))$ for the unconstrained system; yellow: $V(x(k))$ for the constrained system with $C_{max} = 25$.

$s_5$ and $s_6$ to the closest sinks are not feasible when $C_{max} = 25$, then new feasible shortest paths are formed, reaching different sinks with a larger length. The shortest outgoing paths from the other three sources $s_2$, $s_3$ and $s_4$ are equal to those that emerged in the unconstrained systems, because they are feasible. The global rest states $\bar{x}$ in Figs. 11.2c and 11.2d are characterized by the following states of the source nodes $s_1, \dots, s_6$: $\bar{x}^0_{s_1}(k) = 496$, $\bar{x}^0_{s_2}(k) = 1058$, $\bar{x}^0_{s_3}(k) = 529$, $\bar{x}^0_{s_4}(k) = 597$, $\bar{x}^0_{s_5}(k) = 660$, $\bar{x}^0_{s_6}(k) = 640$, equal to the optimal feasible path lengths.

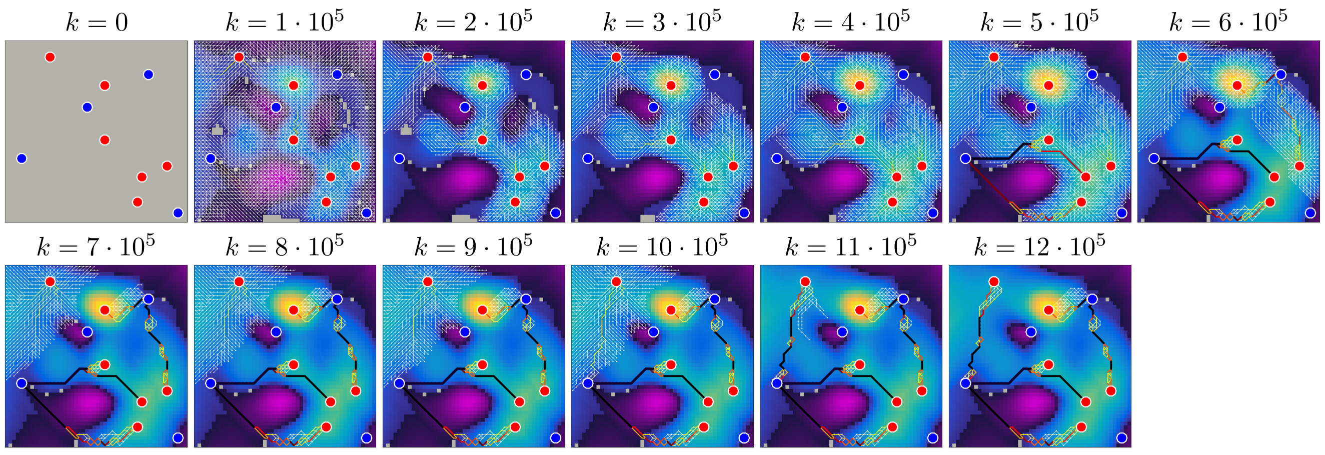
Finally, note that here multiple equivalent shortest (feasible) outgoing paths from the sources exist; they are discovered and traversed by the tokens when using a stochastic model, while only one of them is chosen when using a deterministic model. Still, as the global rest state is similar in both cases, all these paths are still potentially available if a change of choice model occurs in the latter case.

Consider now the transient to get the steady-state. The time evolution of the state of a subset of nodes (including all the sources) is shown in Fig. 11.3 for both the unconstrained and constrained system, assuming the stochastic choice model (for the deterministic one the behavior is almost identical). In both cases, the states start increasing and eventually they stabilize. In the unconstrained case, the states of the source variables reach the corresponding $\bar{\bar{x}}_i$. Note that the state of different sources converges at different times. In the constrained case, despite $x_i = \sum_{c=0}^{C_{max}} x_i^c$, is represented, generally a similar behavior can be observed.

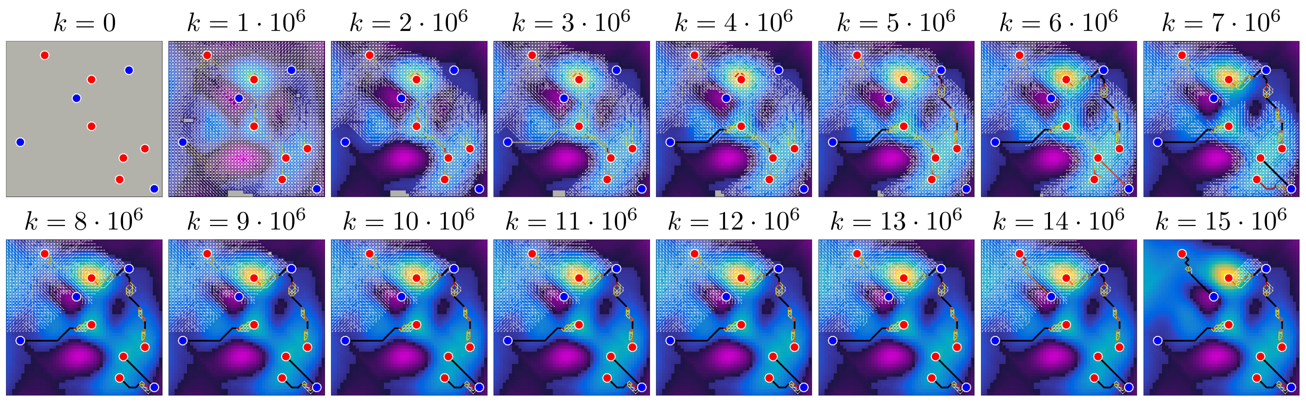
The global behavior can be observed in Fig. 11.4, where the time evolution of $V(x(k))$ is represented. Solid lines refer to the stochastic choice model, and darker dashed lines to the deterministic choice model: note that in both the unconstrained and constrained system they are about the same. Compared to the case in Chapter 10, the difference between the constrained and unconstrained system is more evident here, as now the equivalent network for the constrained system would be about $C_{max} + 1 = 26$ times larger than the original one. In the constrained system, much more tokens are required to be injected into the network and stored in the nodes. Also, the network is not *saturated*, as nodes far from the source nodes are not reached by the injected tokens.

The transient can be visualized directly in the network in Fig. 11.5 for the stochastic choice model, both for the constrained and unconstrained cases (for the deterministic model, it is similar). The color of each pixel is the state $x(k)$ at a particular time $k$; the paths of the tokens traveling in the previous $10^5$ (Fig. 11.3a) and $10^6$ (Fig. 11.3b) time units are also shown.

Consider the unconstrained system. Since there are negative cost arcs, the initial state is not admissible, so initially, there are a lot of simultaneous transitions of tokens already present in the network (such paths are included in Fig. 11.5a for the image at $k = 10^5$: note that there have been transitions in almost all the network). As such tokens move, the states of the nodes with negative gradients of $h_i$ such that $dh < h_0$ become negative, transferring tokens to nodes with smaller values of $h_i$. An admissible state is reached almost immediately, at $k = 6$, and then it remains such, where only the injected tokens move, spreading in the network and modifying its state. Their exploration area grows over time until the closest sink (or an area where the shortest outgoing



(a) Unconstrained system, stochastic choices.



(b) Constrained system, stochastic choices.

Figure 11.5: (*A large grid network*). Evolution of the nodes' state map of both the unconstrained and constrained systems, adopting both the deterministic and stochastic choice model, to get the final global rest states of Fig. 11.2. The paths traveled by the tokens in the previous $10^5$ (Fig. 11.3a) and $10^6$ (Fig. 11.3b) time units are represented.

paths have already been discovered) is reached. Then, they are routed through the shortest paths. Locally, the states converge at different times, depending on the closeness to the sinks and sources: close sources (indirectly) cooperate to make the states of the surrounding area converge earlier. Isolated sources tend to take more time to converge. In the end, 552628 injected tokens stop in the nodes and cannot reach the sinks (when using the deterministic model, they are 552561).

For the constrained system, the behavior is similar; however, now the exploration area of the tokens is limited. The state becomes admissible at $k = 50$ and, in the end, 6134934 injected tokens are deposited in the nodes without reaching the sinks, hence are lost (when using the deterministic model they are 6104811).

To summarize, three emergent behaviours can be observed. First, the reaching of an admissible state, and then of a global rest state, without the single tokens "be aware" of this. Second, the gradual growth of the exploration area from each source until some closer sinks are found; generally, this area remains localized around those nodes, without covering the entire network, especially for very large networks; the exploration area is clearly limited in the constrained case. Third, the discovery of the shortest (possibly constrained) outgoing paths from each source; in a deterministic choice model, once one of these is found from a given source, all the tokens injected next follow this same path; in a stochastic choice model, a similar phenomenon to the one observed in lightnings [105] occurs, in which the non-optimal paths are still taken by some injected tokens, but become shorter and shorter, until they "vanish" and eventually all the injected tokens take one of the existing shortest outgoing paths.

Consider now the dynamic scenario where modifications are applied to the network, as depicted in the images on the first column of Fig. 11.7. In the same figure, in the second to fourth columns, the global rest state $\bar{x}$ reached after each modification is represented, for the considered cases. In those images, the paths traveled by 40000 injected tokens at steady-state are also represented, which are the shortest feasible ones for the unconstrained and constrained system with $C_{max} = 25$, respectively.

The system adapts to the modifications on the network and new updated shortest paths emerge in all cases.

Again, note that the main difference between the deterministic and stochastic choice model is in the number of possible traveled paths followed by the newly injected tokens: in the deterministic model, only one specific path

Table 11.2: (*A large grid network*). Comparison of the simulation results at each configuration of the network in Fig. 11.1, considering the modifications from Fig. 11.7.

| Time interval | Constrained, deterministic | | | | | | Constrained, stochastic | | | | | | $C_{max}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $L_{ss}$ | $C_{ss}$ | $T_{ss}$ | $V_{ss}$ | $l_{ss}$ | $T_{na}$ | $L_{ss}$ | $C_{ss}$ | $T_{ss}$ | $V_{ss}$ | $l_{ss}$ | $T_{na}$ | |
| [0, 18000000] | 635 | 26 | 989256 | 545771 | 552561 | 5 | 635 | 25 | 1010853 | 546249 | 552628 | 5 | n/a |
| [18000001, 21000000] | 700 | 18 | 329459 | 551806 | 90274 | 0 | 700 | 18 | 329461 | 552013 | 90125 | 0 | n/a |
| [21000001, 24000000] | 654 | 22 | 35304 | 558400 | 6647 | 4 | 654 | 22 | 35128 | 557614 | 5639 | 4 | n/a |
| [24000001, 27000000] | 606 | 28 | 0 | 558400 | 0 | 0 | 606 | 28 | 0 | 557614 | 0 | 0 | n/a |
| [27000001, 42000000] | 501 | 22 | 186965 | 591611 | 33211 | 0 | 501 | 22 | 186476 | 591132 | 33518 | 0 | n/a |
| [42000001, 45000000] | 458 | 16 | 21 | 524242 | -66912 | 22 | 458 | 16 | 19 | 528926 | -61836 | 20 | n/a |
| [45000001, 66000000] | 713 | 15 | 1625841 | 1011294 | 487052 | 0 | 713 | 15 | 1633274 | 1017210 | 488284 | 0 | n/a |

| Time interval | Constrained, deterministic | | | | | | Constrained, stochastic | | | | | | $C_{max}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $L_{ss}$ | $C_{ss}$ | $T_{ss}$ | $V_{ss}$ | $l_{ss}$ | $T_{na}$ | $L_{ss}$ | $C_{ss}$ | $T_{ss}$ | $V_{ss}$ | $l_{ss}$ | $T_{na}$ | |
| [0, 18000000] | 663 | 16 | 13722936 | 6014733 | 6104811 | 47 | 663 | 16 | 13596550 | 6049551 | 6134934 | 49 | 25 |
| [18000001, 21000000] | 702 | 11 | 1515979 | 5593978 | 482348 | 0 | 702 | 12 | 1573258 | 5628983 | 488688 | 0 | 25 |
| [21000001, 24000000] | 684 | 14 | 1652689 | 6047602 | 454232 | 22 | 684 | 14 | 1823133 | 6077752 | 448890 | 13 | 25 |
| [24000001, 27000000] | 630 | 16 | 0 | 6047602 | 0 | 0 | 630 | 16 | 0 | 6077752 | 0 | 0 | 25 |
| [27000001, 42000000] | 568 | 15 | 12992321 | 8258115 | 2210513 | 0 | 568 | 15 | 12981552 | 8286966 | 2209214 | 0 | 25 |
| [42000001, 45000000] | 470 | 14 | 1649 | 7424832 | -818076 | 387 | 470 | 14 | 1409 | 7477152 | -796398 | 357 | 25 |
| [45000001, 66000000] | 713 | 15 | 19073067 | 12999899 | 5575067 | 0 | 713 | 15 | 19262010 | 13083292 | 5606140 | 0 | 25 |

*Note:* Each row refers to a different network configuration (see Figs. 11.2 and 11.7). **Metrics.** $L_{ss}$: average length $L(p)$ of the paths travelled by the tokens injected at steady-state; $C_{ss}$: the corresponding constrained cost $C(p)$, which is equal to $E_{ss}$, the corresponding number of traversed arcs; $T_{ss}$: time (number of injected tokens) needed to reach a global rest state, i.e., $k$ such that $V(x(k)) = \bar{V}$; $V_{ss}$: the corresponding value of $V(x(k)) = \bar{V}$ at global rest state; $l_{ss}$: total number of tokens lost during the transitory, because either they are stopped in a node, or their path is not feasible, or the node they are deposited has been removed; $T_{na}$: time (number of injected tokens) needed for the state of the network to become admissible, if the initial state in non-admissible.

is followed by all the tokens, while in the stochastic one, there are multiple possibilities, including the latter.

The time evolution of $V(x(t))$ over the entire time horizon for the constrained and unconstrained systems adopting the stochastic choice model is represented in Fig. 11.6 (for the deterministic choice model it is about the same). The same considerations from Section 10.3 apply here. However, here it is more evident that the unconstrained system adapts faster than the constrained one.

Finally, in Table 11.2 the considered metrics are reported for each interval occurring between two successive modifications on the network, for both the constrained and unconstrained system, and for both the stochastic and deterministic choice models. The findings discussed above are confirmed. Note that the state becomes non-admissible when at $k = 21000001$, when inserting some nodes/arcs, and $k = 42000001$, when setting some new sink nodes (other than being non-admissible at $k = 0$), but it remains such for a brief period (see the rows with $T_{ss} \neq 0$). Moreover, note that the system is adapted immediately at $k = 24000001$, i.e., when new sources are set, as expected (since the state was a global rest state just before the modification on the network), see metric $T_{ss}$. Finally, tokens might be lost because either they are forced to stop in a node or because such node fails. Recall that the final total number of tokens that are lost is equal to the sum of the values in the $l_{ss}$ column. Note that when $k = 42000001$, i.e., when new sink nodes are set, $l_{ss} < 0$: this happens because the state becomes a non-admissible state, and all the possible tokens deposited in the surrounding of the nodes move to leave the network; the number of such tokens is larger than the number of injected token in the considered interval.

Some animations of these simulations are available at `https://users.dimi.uniud.it/~franco.blanchini/examples_mkv.html`, showing the evolution over time of the nodes' states and the paths followed by the tokens.

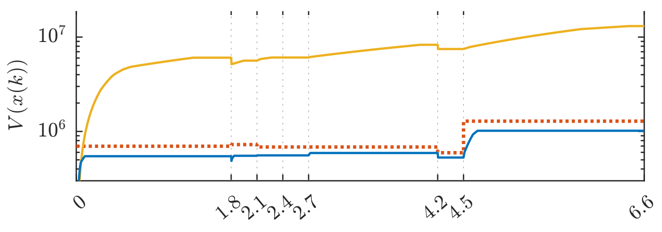


Figure 11.6: (*A large grid network*). Blue: $V(x(k))$ for the unconstrained system; dotted red: $V(\bar{\bar{x}}(k))$ for the unconstrained system; yellow: $V(x(k))$ for the constrained system with $C_{max} = 25$. The full time horizon is represented. The y-axis is in log-scale.

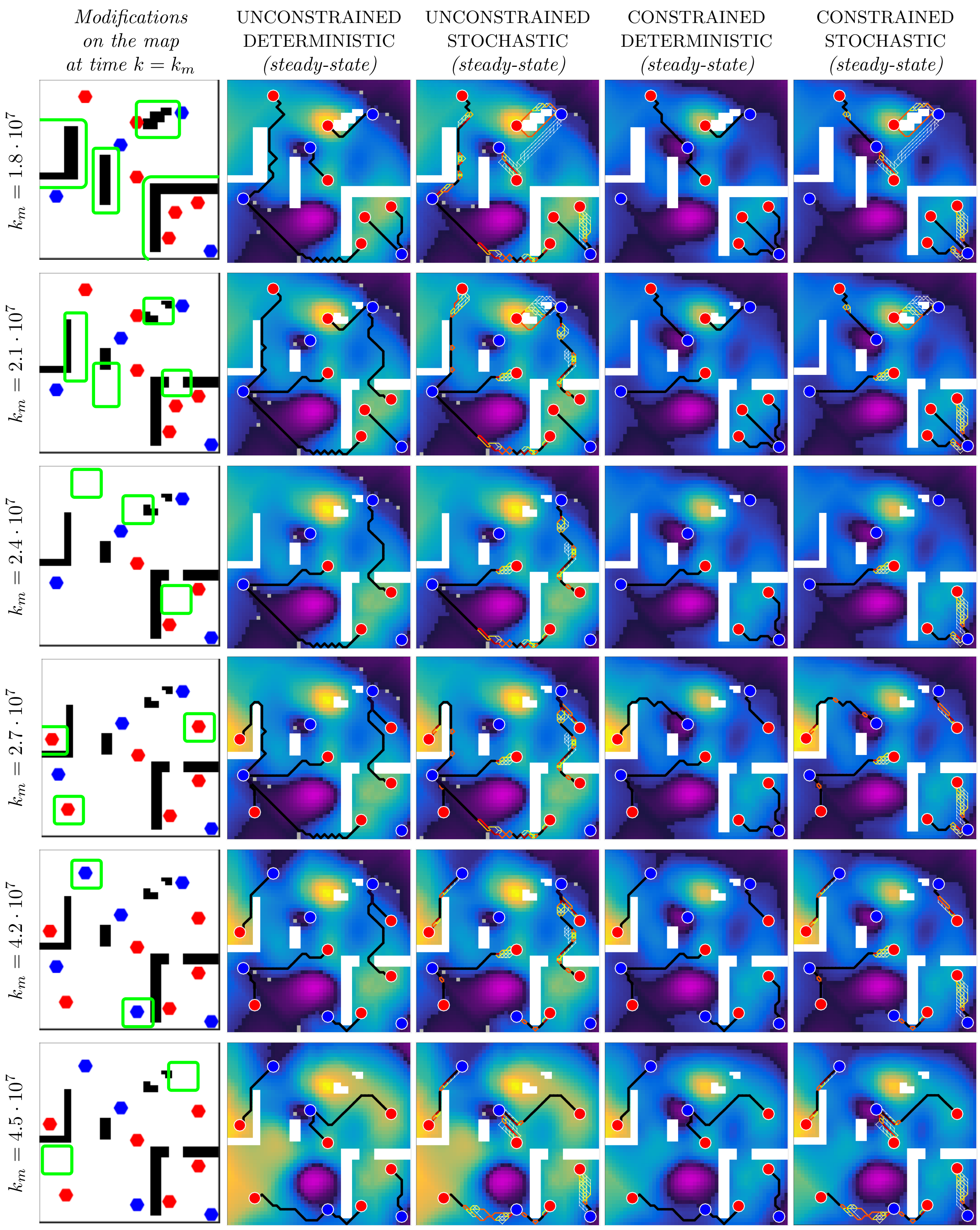


Figure 11.7: (*A large grid network*). **First column** evolution of the map of Fig. 11.2, occurring at different times $k = k_m$. Black pixels: obstacles (disabled nodes); white pixels: enabled nodes; red circles: source nodes; blue circles: sink nodes; green shapes: sources/sinks/group of nodes that have just been modified. **Second to fifth columns:** the corresponding global rest state of the network reached after the system stabilizes after each modification, for the unconstrained system with deterministic choices (evolving from Fig. 11.2a), the unconstrained system with stochastic choices (evolving from Fig. 11.2b), the constrained system with deterministic choices and $C_{max} = 25$ (evolving from Fig. 11.2c), and constrained system with stochastic choices and $C_{max} = 25$ (evolving from Fig. 11.2d). White pixels are obstacles. The paths traveled by the tokens in the previous 40000 time units are represented.

# Chapter 12

# Illustrative example in a large small-world network

In this Chapter the proposed policies Policies 8.1, 8.2, 9.1 and 9.2 are applied to a class of small-world networks. Small-world networks are networks where there is a small average path length between any pair of nodes and a high local clustering of the nodes. A base network is first generated randomly according to a given procedure; then, some variations of this network are constructed by varying, for instance, the number of nodes or the magnitude of the arc costs. The effects of these modifications on some metrics are then evaluated and compared. To focus on the performance, a static scenario is considered. Moreover, arc costs are assumed to be non-negative, so that the state is always admissible.

These proposed policies have been implemented in C and compiled as Matlab MEX functions; simulations were performed in Matlab. All the simulations were performed on a dual-core Intel Core i3 at 2.3 GHz with 8 GB of RAM.

## 12.1 Scenario and data

Some Watts-Strogatz small-world networks are considered [128], which are random graphs that have small-world network properties, such as clustering and short average path length. Each network is randomly generated according to the following procedure, exploiting a provided Matlab function [129]:

1. An undirected ring lattice with $n$ nodes of mean degree $\delta$ (assumed even) and $n\delta/2$ arcs is created, in which each node is connected to its $\delta/2$ nearest neighbors on each side.
2. The terminal node of each edge connecting nodes $i$ and $j$ is *rewired* with probability $\beta$, avoiding duplicated edges or self-loops. When $\beta = 0$, no edge is rewired and a ring lattice is obtained. When $\beta = 1$, every edge is rewired and a purely random graph is obtained.
3. A directed network of $m = n\delta$ arcs is obtained by replacing each undirected arc with two arcs $(i, j)$ and $(j, i)$ in opposite directions. Integer costs are assigned randomly to each arc, with $\gamma_{ij} \in [1, \gamma_{max}]$ and $\sigma_{ij} \in [1, \sigma_{max}]$ for some $\gamma_{max}, \sigma_{max} \in \mathbb{N}$.
4. A single source $s$ is set in node "1". A single sink $d$ is set so that the length of the shortest path from $s$ to $d$ is $\min(L(\{s, \ldots, d\})) \approx mean\{\min(L(\{s, \cdots, i\}) : i \in \mathcal{N}\}$. $C_{max}$ is chosen so that feasible paths exist.

First, a default base network is generated using $n = 1000$ (number of nodes), $\delta = 4$ (mean nodes out-degree), $\beta = 0.15$ (rewiring probability), $\gamma_{max} = 50$ (maximum cost), $\sigma_{max} = 10$ (maximum secondary cost), $C_{max} = 65$ (for the constrained system). The resulting network has $m = n\delta = 4000$ edges and is depicted in Fig. 12.1.

Then, different variants of this network are generated by varying each time one of the following characteristics, and keeping all the other parameters unchanged:

- the number of nodes, considering $n \in \{100, 1000, 10000, 50000\}$;
- the mean out-degree, considering $\delta \in \{4, 8, 16, 32\}$;
- the number $E_{sp}$ of arcs of the shortest path between the source and the sinks, by varying the location of the sink node, using $E_{sp} = \{5, 11, 15, 19\}$;

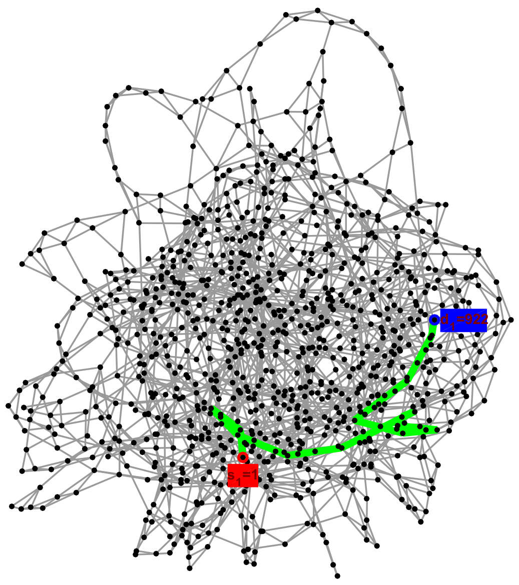


Figure 12.1: (*A large small-world network*). Directed Watts-Strogatz small-world graph with $n = 1000$ nodes, $m = 4000$ edges, $\delta = 4$ mean out-degree, and rewiring probability $\beta = 0.15$. The effects of changing some characteristics of this network will be evaluated. Arc costs are omitted. The source and sink nodes are highlighted in red and blue, respectively. The shortest path between them is highlighted in green.

- the magnitude of the costs, considering scaled costs $K_\gamma \cdot \gamma_{ij}$ with $K_\gamma \in \{1, 10, 100\}$;
- the magnitude of the constrained costs, considering scaled constrained costs $K_\sigma \cdot \sigma_{ij}$ and scaled $K_\sigma \cdot C_{max}$, with $K_\sigma \in \{1, 10, 100\}$;
- (for the constrained system only) the maximum constrained cost, $C_{max} \in \{34, 65, 95, 120\}$.

In all cases, a stochastic choice model is considered. Simulations are performed considering both the unconstrained and constrained system, applying both the original Policies 8.1 and 9.1, and the enhanced Policies 8.2 and 9.2. A static scenario is considered. Recall that, as the arc costs are all positive, the state of the system, initialized with a zero state, is always admissible and is non-negative. The simulation settings are those from subsection 10.1.1. Metrics from Section 10.2 have been computed from each simulated case and compared.

## 12.2 Results and discussion

The network configuration at steady-state is represented in Fig. 12.2, for the base network in Fig. 12.1, considering both the original and the enhanced strategies, both for the constrained and unconstrained system.

Consider the unconstrained system, first. By comparing the exact (Fig. 12.2a) and the enhanced policy (Fig. 12.2b) cases, it can be seen that in both the same shortest path emerge, which is highlighted in red and coincides with the optimal one from Fig. 12.1. In the enhanced case, nodes tend to have larger states. However, recalling that the enhanced strategy ensured that in a strongly connected network like this one, all the injected tokens are received in the sinks, it happens that in Fig. 12.2a, the states represent actual tokens that have been injected in the network and have been deposited in the nodes, while in Fig. 12.2b, the states are composed only by virtual tokens. Moreover, the spreading of the accumulation of virtual tokens over the network in the second case is due to the traveling tokens moving from node to node with no limitation, until the only sink is found.

For the constrained system, in both the exact (Fig. 12.2a) and the enhanced (Fig. 12.2b) cases the same shortest path emerges, again, but now it is not equal to the optimal (unconstrained) one, because that is unfeasible. Now the state of the nodes ($x_i = \sum_c x_i^c$) in the enhanced case in Fig. 12.2d is way more similar to that of the original case Fig. 12.2c, although it remains generally larger. Indeed, recall that in a constrained system, when paths become unfeasible, tokens are not allowed to proceed, and stop in the nodes instead. When using the enhanced policy, tokens are allowed to move for a longer time, as soon as their constrained cost $c \le C_{max}$, while using the original strategy they might stop earlier.

For comparison, numerical values of the metrics summarizing the simulation results at global rest state for the above base case and its variations are reported in Table 12.1. First, to confirm the above-mentioned findings, the metrics for the base case can be taken from the second row of such table and analyzed. The enhanced policy finds the same optimal paths (in terms of $L(p)$ and $C(p)$, see metrics $L_{ss}$, $C_{ss}$ and $E_{ss}$ in the table) as the original policy, in both the unconstrained and constrained case. The shortest path that emerges in the unconstrained system has

Table 12.1: (*A large small-world network*). Comparison of the simulation results at global rest state when varying the network characteristics. The network is built using the Watts-Strogatz small-world graph model.

| Parameter to vary | | Unconstrained, original | | | | | | Unc., enhanced | | | Constrained, original | | | | | | Constr., enhanced | | | $c_{max}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $L_{ss}$ | $C_{ss}$ | $E_{ss}$ | $T_{ss}$ | $V_{ss}$ | $l_{ss}$ | $T_{ss}$ | $V_{ss}$ | $l_{ss}$ | $L_{ss}$ | $C_{ss}$ | $E_{ss}$ | $T_{ss}$ | $V_{ss}$ | $l_{ss}$ | $T_{ss}$ | $V_{ss}$ | $l_{ss}$ | |
| $n$ | 100 | 104 | 40 | 7 | 1361 | 1344 | 1344 | 17 | 4509 | 0 | 122 | 28 | 6 | 9010 | 8886 | 8886 | 1042 | 15582 | 1018 | 32 |
| | 1000 | 167 | 95 | 14 | 16427 | 16245 | 16245 | 64 | 78522 | 0 | 175 | 64 | 10 | 281645 | 279459 | 279459 | 22717 | 494319 | 22382 | 65 |
| | 10000 | 218 | 125 | 20 | 159774 | 158746 | 158746 | 248 | 1075638 | 0 | 229 | 81 | 12 | 3024376 | 3013981 | 3013981 | 240217 | 5325649 | 239205 | 85 |
| | 50000 | 277 | 199 | 31 | 710604 | 709164 | 709164 | 341 | 6993467 | 0 | 302 | 123 | 19 | 44519968 | 43769899 | 43769899 | 2436421 | 86177388 | 2387745 | 136 |
| $\delta$ | 4 | 167 | 95 | 14 | 16427 | 16245 | 16245 | 64 | 78522 | 0 | 167 | 95 | 14 | 315195 | 309832 | 309832 | 12163 | 978229 | 11901 | 100 |
| | 8 | 69 | 52 | 7 | 6786 | 6632 | 6632 | 42 | 37282 | 0 | 69 | 52 | 10 | 42498 | 41621 | 41621 | 3819 | 359209 | 3688 | 100 |
| | 16 | 31 | 62 | 10 | 3325 | 3283 | 3283 | 36 | 17653 | 0 | 31 | 62 | 7 | 19699 | 19389 | 19389 | 3370 | 190531 | 3289 | 100 |
| | 32 | 16 | 54 | 8 | 1956 | 1895 | 1895 | 64 | 8488 | 0 | 16 | 54 | 8 | 7483 | 7301 | 7301 | 2494 | 98433 | 2370 | 100 |
| $E_{sp}$ | 5 | 94 | 31 | 5 | 1155 | 1141 | 1141 | 12 | 41419 | 0 | 94 | 31 | 5 | 6712 | 6552 | 6552 | 355 | 46851 | 332 | 100 |
| | 11 | 130 | 74 | 11 | 4699 | 4616 | 4616 | 30 | 60187 | 0 | 130 | 74 | 11 | 58029 | 56692 | 56692 | 2470 | 258335 | 2326 | 100 |
| | 15 | 201 | 102 | 15 | 39993 | 39671 | 39671 | 71 | 110753 | 0 | 206 | 89 | 13 | 1261169 | 1229403 | 1229403 | 47694 | 2455789 | 46391 | 100 |
| | 19 | 259 | 113 | 19 | 95199 | 94795 | 94795 | 110 | 157699 | 0 | 264 | 100 | 17 | 4223367 | 4174069 | 4174069 | 170082 | 5514798 | 167605 | 100 |
| $K_\gamma$ | 1 | 167 | 95 | 14 | 16427 | 16245 | 16245 | 64 | 78522 | 0 | 175 | 64 | 10 | 281645 | 279459 | 279459 | 22717 | 494319 | 22382 | 65 |
| | 10 | 1670 | 95 | 14 | 158364 | 158148 | 158148 | 64 | 785220 | 0 | 1750 | 64 | 10 | 2716498 | 2714274 | 2714274 | 172647 | 4613687 | 172217 | 65 |
| | 100 | 16700 | 95 | 14 | 1577358 | 1577178 | 1577178 | 64 | 7852200 | 0 | 17500 | 64 | 10 | 27064645 | 27062424 | 27062424 | 1675252 | 45967239 | 1674833 | 65 |
| $K_\sigma$ | 1 | - | - | - | - | - | - | - | - | - | 175 | 64 | 10 | 281645 | 279459 | 279459 | 22717 | 494319 | 22382 | 65 |
| | 10 | - | - | - | - | - | - | - | - | - | 175 | 640 | 10 | 274952 | 272982 | 272982 | 22789 | 476523 | 22386 | 645 |
| | 100 | - | - | - | - | - | - | - | - | - | 175 | 6400 | 10 | 274952 | 272982 | 272982 | 22789 | 476523 | 22386 | 6450 |
| $c_{max}$ | 34 | - | - | - | - | - | - | - | - | - | 286 | 34 | 10 | 274704 | 274324 | 274324 | 72998 | 291584 | 72813 | 34 |
| | 65 | - | - | - | - | - | - | - | - | - | 175 | 64 | 10 | 281645 | 279459 | 279459 | 22717 | 494319 | 22382 | 65 |
| | 95 | - | - | - | - | - | - | - | - | - | 167 | 95 | 14 | 306771 | 301805 | 301805 | 12381 | 889270 | 12138 | 95 |
| | 120 | - | - | - | - | - | - | - | - | - | 167 | 95 | 14 | 339537 | 333303 | 333303 | 11441 | 1342708 | 11129 | 120 |

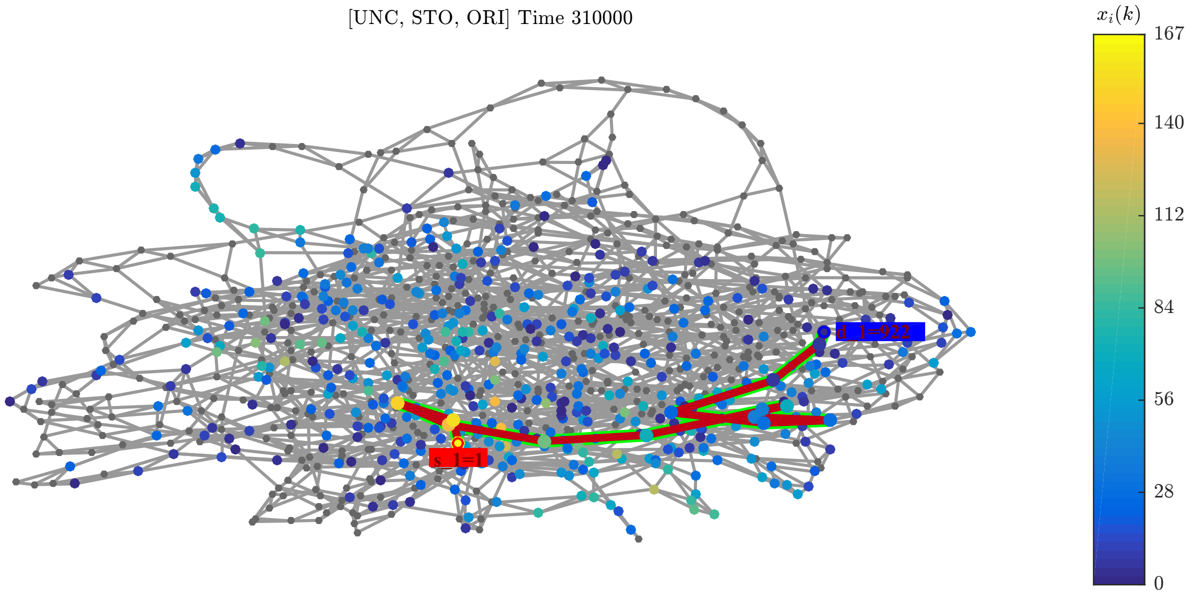
*Note:* Each row refers to a different network. Default parameters which are not specified in each row: number of nodes $n = 1000$, mean nodes out-degree $\delta = 4$, rewiring probability $\beta = 0.15$, maximum cost $\gamma_{max} = 50$, maximum secondary cost $\sigma_{max} = 10$, costs' scaling factors $K_\gamma = 1$, $K_\sigma = 1$. $E_{sp}$: number of edges in the shortest paths. $C_{max}$ is given for the constrained system. **Metrics.** $L_{ss}$: cost $L(p)$ paid by each token injected at global rest state; $C_{ss}$: the corresponding constrained cost $C(p)$; $E_{ss}$: the corresponding number of traversed arcs; $T_{ss}$: the time in which the global rest state is reached, i.e., $k$ such that $V(x(k)) = \bar{V}$; $V_{ss}$: the corresponding $\bar{V}$ at global rest state; $l_{ss}$: total number of tokens lost during the transitory, because either they are stopped in a node or their path is not feasible. $L_{ss}$, $C_{ss}$, $E_{ss}$ for the enhanced unconstrained (resp. constrained) policy are exactly the same as the ones for the original unconstrained (resp. constrained) policy, hence omitted.

constrained cost $C_{ss} > C_{max}$, while in the constrained system the shortest path is feasible $C_{ss} < C_{max}$, but has a longer length $L_{ss}$ compared to the unconstrained case.

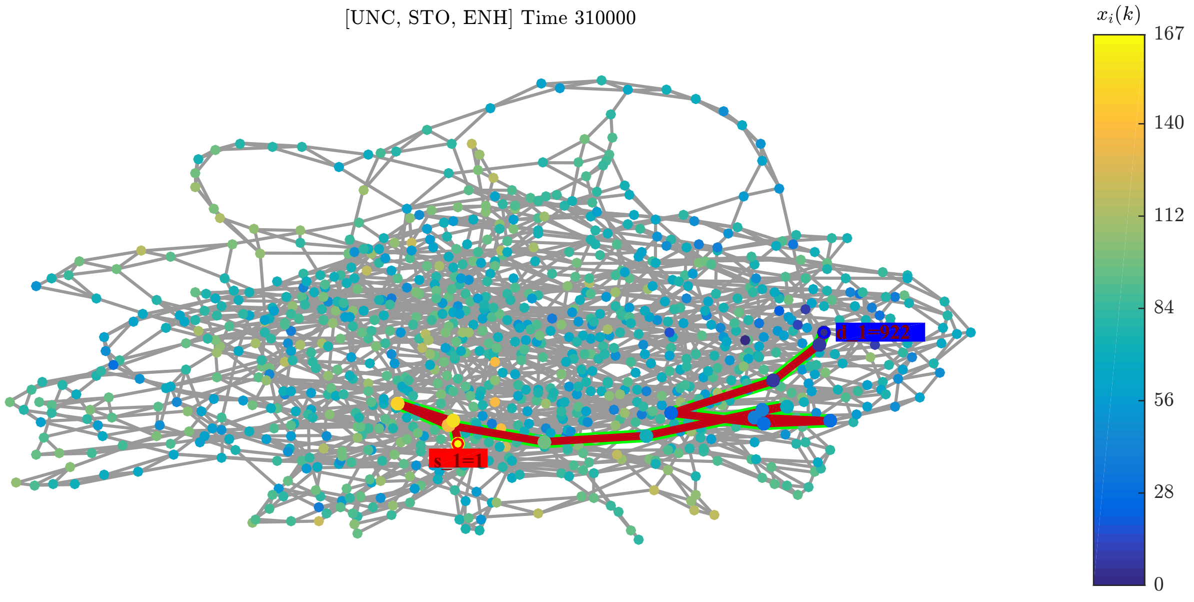
Moreover, when using the enhanced strategy, the number of tokens (the state) deposited in the nodes is generally larger compared to the case when using the original strategy, especially for the unconstrained system (see metric $V_{ss}$). When applying the original strategy, the number of lost tokens is equal to the number of tokens deposited in the nodes $V_{ss} = l_{ss}$, which means that all of these were injected and got lost (see metric $l_{ss}$). However, note that when applying the enhanced strategy in the unconstrained case, no informative injected token is lost, $l_{ss} = 0$, so the tokens deposited in the nodes are all virtual; differently, some informative tokens are asleep in the constrained system, despite most of the deposited tokens are still virtual as $l_{ss} < V_{ss}$: the number of lost tokens is however smaller compared to the case in which the original policy is applied.

Finally, the unconstrained enhanced policy takes considerably less time to reach the global rest state (see metric $T_{ss}$). While this is true also for the constrained case, it is less evident, since some tokens fall asleep and cannot continue moving in the network, filling the nodes' states.

These considerations are generally valid for the other rows of the table, which refer to the networks obtained varying a characteristic of the base one. Note that when the unconstrained shortest path is feasible ($C_{ss} < C_{max}$), the same shortest path emerges in the constrained system, too, having the same minimal length $L_{ss}$. Increasing the network size, decreasing the mean out-degree of the nodes, and increasing the number of edges in the shortest paths result in a larger number of tokens required to converge (see the rows referring to the variation of $n$, $\delta$, and $E_{sp}$). For the original policy, increasing the arcs costs $\gamma_{ij}$ worsen the performance; the unconstrained enhanced policy is unaffected, while this is not true in the constrained case as tokens might fall asleep (see the rows referring to the variation of $K_\gamma$). Finally, for the constrained system, scaling both $\sigma_{ij}$ and $C_{max}$ by the same integer factor does not vary the performance, as the corresponding expanded networks have the same topology (see the rows referring to the variation of $K_\sigma$). Instead, by increasing only $C_{max}$, the performance degrades for the original policy, while it improves for the enhanced one (see the rows referring to the variation of $C_{max}$); the reason is that in both cases, the equivalent expanded network is just larger, but for the specific enhanced case, tokens are also allowed to travel more (not necessarily modifying all the state components). As the network scales up to a larger one, the emerging behaviours that can be observed are the same as the ones from Section 11.2.

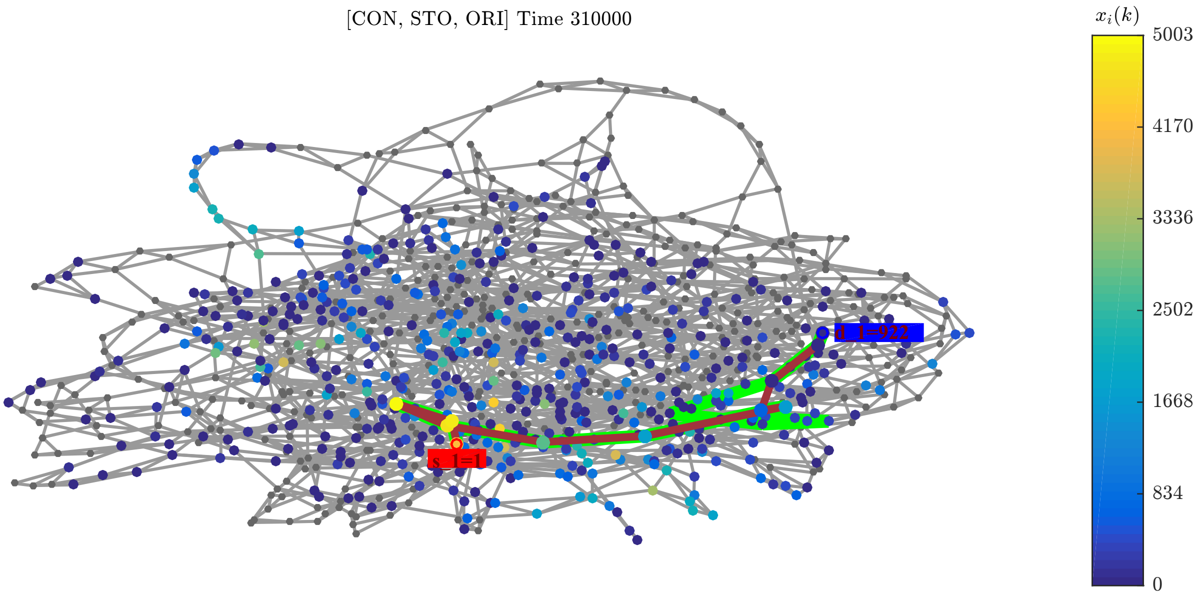


(a) Unconstrained system, stochastic choices, original strategy.

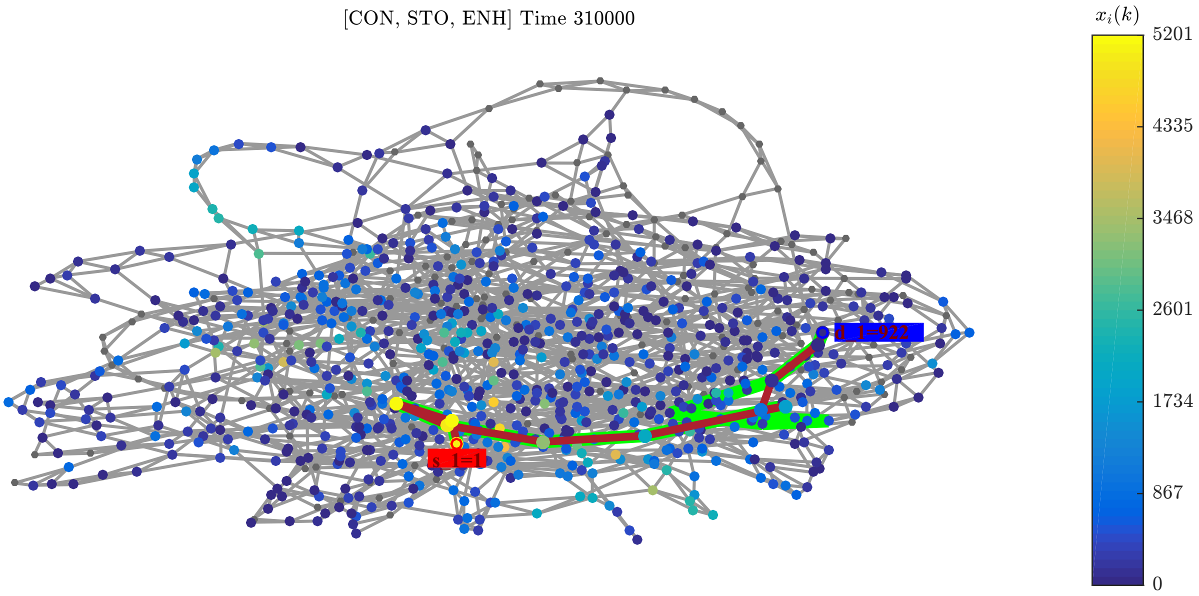


(b) Unconstrained system, stochastic choices, enhanced strategy.

Figure 12.2: (*A large small-world network*). The network of Fig. 12.1 at steady-state after having applied the proposed policies for the network in Fig. 12.1.. The color of each node represents its state $\bar{x}_i(t)$ at steady-state (global rest state). The paths followed by the tokens are highlighted in red. *(continued on the next page).*



(c) Constrained system, stochastic choices, original strategy.



(d) Constrained system, stochastic choices, enhanced strategy.

Figure 12.2: (*A large small-world network*). *(continued)* The network of Fig. 12.1 at steady-state after having applied the proposed policies for the network in Fig. 12.1. The color of each node represents its state $\bar{x}_i(t)$ at steady-state (global rest state). The paths followed by the tokens are highlighted in red.

# PART III
# Decentralized flow control for fair and sparse solutions

Based on [3], published: Franco Blanchini, Carlos Andrés Devia, Giulia Giordano, Raffaele Pesenti, and Francesca Rosset. "Fair and Sparse Solutions in Network-Decentralized Flow Control". In: *IEEE Control Systems Letters* 6 (2022). © 2022 IEEE, pp. 2984–2989. DOI: 10.1109/LCSYS.2022.3181341

# Chapter 13

# Introduction to Part III

This Part presents the work introduced in [3], which is the result of the collaboration between Franco Blanchini, Carlos Andrées Devia, Giulia Giordano, Raffaele Pesenti, and me.

Consider a flow network composed of a certain number of nodes and arcs. In each node, there is a buffer that contains a given amount of a resource. This resource can be continuously transferred between nodes along the existing arcs creating a continuous flow, and possibly modifying the buffer levels of the resource over time. This flow might be controlled by applying a given control to get some desired behavior.

Suppose that the network is connected to the external environment and assume that an uncontrolled fixed outflow demand of the resource (i.e., a flow directed toward the external environment) is required. Assume that there is also a controllable inflow (i.e., a flow coming from the external environment) that provides the required resource over time. For instance, consider a fluid network in which the nodes are some tanks containing water and the arcs are pipes connecting the tanks. Assume that from some tanks a water demand is required for usage. Some external reservoirs are connected to the network, which provide water when required.

By controlling the flow of the arcs in the network, this demand can be met. It is desirable to stabilize the system, too, to avoid fluctuations over time; indeed, when the system is stabilized, the buffer levels of the tanks are constant: the total flow that enters any given tank is compensated by the total flow leaving it. In general, however, there are many different distributions of the flows stabilizing the network and meeting the demand, so that an optimization can be performed.

In this Part, the asymptotic minimization of the $p$-norm $\sqrt[p]{\sum |f_{ij}|^p}$ of the (controllable) flows $f_{ij}$ in the network is considered. Depending on the value of $p$, the distribution of the flows in the network has some particular characteristics that might be useful. For instance, if $p = 1$, it turns out that the flow tends to be concentrated in only few arcs, while being zero in the other ones: the solution tends to be *sparse*. Considering a single (controllable) inflow and a single (uncontrollable) outflow demand, the flow distribution that meets the demand is concentrated in the arcs along the shortest paths between the corresponding nodes. If $p = 2$, the flow tends to be spread in all the arcs, and a configuration that minimizes an "energy" $\sqrt{\sum f_{ij}^2}$ is achieved. If $p = \infty$, the maximum flow is minimized, and a *fair* solution is achieved, where it is not possible to decrease the flow in a given arc, without having to increase by a larger quantity the flow in some other arc.

Getting the optimal distribution minimizing the $p$-norm (which is a global variable) and meeting the demand is certainly possible by assuming all the data about the network is known; indeed, some linear-quadratic programming problems can be formulated. However, for many reasons, including the network being too large, such information might not be available. Having a network-decentralized control is therefore advantageous, as it only uses local information, without needing to know the entire network. To this aim, a control is introduced, which is decentralized in the sense that the control to be applied to each arc might depend only on the information about the two buffers at the extremes, in particular on the difference between the two states, according to a given law depending also on the value of $p$. Note that there are some similarities between this and the decentralized policy from Part I: although the problem is certainly different, in both decisions are made at arc level, depending only on local information regarding the difference of the states of its endnodes. Remark, as in Part II, that having a decentralized policy is also advantageous as it is fault-tolerant, and it easily supports large, unknown and dynamic networks, and privacy.

When $1 < p < \infty$ it turns out the control is continuous and that the steady-state required solution asymptotically minimizing the $p$-norm of the flow, stabilizing the network, and providing the resource is actually obtained. An important aspect to be considered is that the required distribution is obtained asymptotically: an initial transient is present, in which the buffer levels vary.

When $p = 1$ or $p = +\infty$, as the cost functional is not convex anymore, the optimal solution might not be

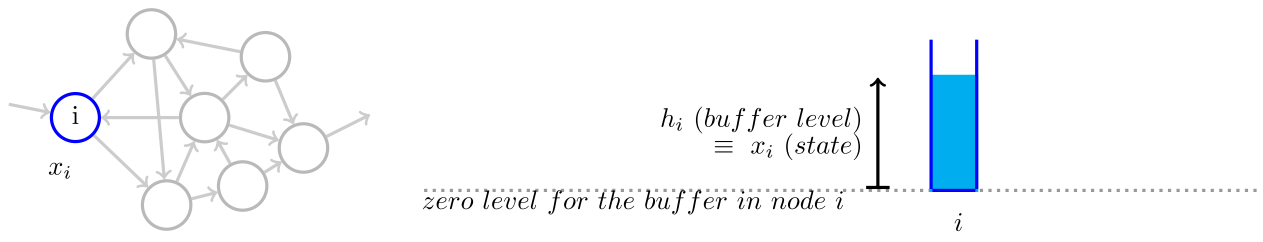


Figure 13.1: Graphical representation of the state $x_i$ of a node $i$: assuming a zero reference level, the state can be seen as the (continuous) level of resource in the node.

unique; the control, however, is no longer continuous and cannot be applied, as it introduces chattering. It will be shown that the proposed control must be applied for $p \to 1$ and $p \to +\infty$, respectively, to get suboptimal solutions arbitrarily close to the optimal 1 and $\infty$-norm solution.

Suppose now that some of the uncontrolled flows depend on the buffer levels. Unsurprisingly, the control proposed above is not effective anymore. An enhanced proportional-integral control having the same properties as the original control is introduced, which is decentralized and does not depend on the unknown dynamics. The same optimality results reported above holds also in this case.

It is interesting to notice that when applying the original control, the state components (buffer levels of the nodes) converge to the optimal Lagrange multiplier vector of the considered optimization problem, which is fixed once $p$ is fixed. In other words, the buffer levels at steady-state cannot be controlled. Instead, when applying the enhanced proportional-integral control supporting unknown dynamics, the state always converges to 0; however, now it is the integral variable vector that converges to the same optimal Lagrange multiplier vector mentioned above (assuming no unknown dynamics at zero states: as it will be seen, this is not restrictive).

The feature of the enhanced control of driving the state to zero is useful for buffer level control. Indeed, one might want to solve the considered problem while keeping the level of the buffers in the node under control, at some fixed generic set-point levels. This condition is obtained asymptotically, once the state notion is suitably redefined, see Section 13.1.

Now consider some weighted networks. So far, each arc was implicitly assumed to be given a unitary cost; here, it is assumed that arcs have real-valued costs, which might be taken into account when minimizing the flow distribution. In this regard, the minimization of the weighted $p$-norm $\sqrt[p]{\sum |f_{ij}\omega_{ij}|^p}$ of the flow could be required. The larger the arc cost, the larger the contribution to the weighted $p$-norm of the flow. Hence, intuitively, the flow tends to occupy the arcs whose cost is small. The problem can be transformed into a simple $p$-norm minimization problem *of the control*, by suitably scaling the control/flow components, see Section 13.1.

To summarize, in many different conditions, applying a local control is effective in solving the considered problem. Optimality is achieved when $1 < p < \infty$, while a suboptimal 1 or $\infty$-norm solution can be made arbitrarily close to the optimum, *at least in theory.*

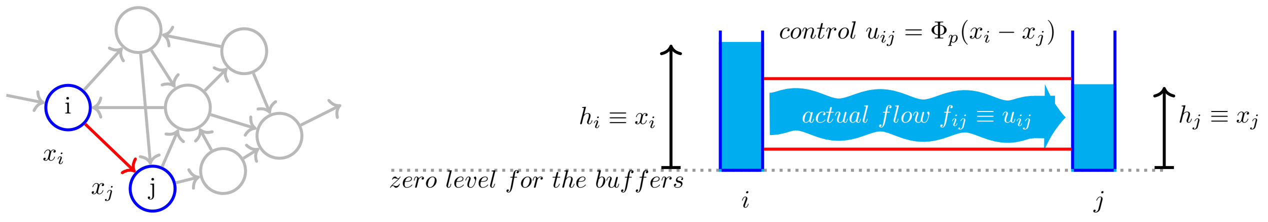
Numerical errors deriving from the implementation of the control will be investigated.

In the rest of this Chapter, first, the intuitive idea behind the proposed control is presented in Section 13.1; after that, the literature review is briefly reported in Section 13.2 and the main contributions summarized in Section 13.3. Then, in Chapter 14 the setup is described, and the considered problem is stated. In Chapter 15 networks with just a fixed uncontrolled demand that has to be met are considered and the proper decentralized control is studied, which also minimizes the $p$-norm of the flow vector. In Chapter 15 the control is suitably adapted to support networks with uncontrolled flows depending on the buffer levels, buffer level control, and weighted $p$-norm minimization. A brief discussion about the numerical issues emerging when implementing the control is also reported. Finally, in Chapter 17, the results of some simulations are reported to validate the proposed approach, and better evaluate the effects of the choices of the control and of $p$. Proofs are however omitted and can be found in [3].

## 13.1 The main idea

The main idea behind this Part is similar to that from Part II. Indeed, consider a directed network composed of some nodes and some directed arcs. It is assumed that there is a buffer in each node; the state $x_i$ of a node $i$ is defined as the amount/level $h_i$ of a given resource deposited in it with respect to an initial zero level of the node, see Fig. 13.1.

This resource is assumed "continuous" and can be transferred from node to node according to an applied decentralized control, which regulates this flow. For arc $(i, j)$, this is characterized by the control $u_{ij}$ of the arc



i
j
$x_i$
$x_j$
control $u_{ij} = \Phi_p(x_i - x_j)$
$h_i \equiv x_i$
actual flow $f_{ij} \equiv u_{ij}$
$h_j \equiv x_j$
zero level for the buffers
i
j

Figure 13.2: Graphical representation of the flow $f_{ij}$ between two nodes $i$ and $j$.

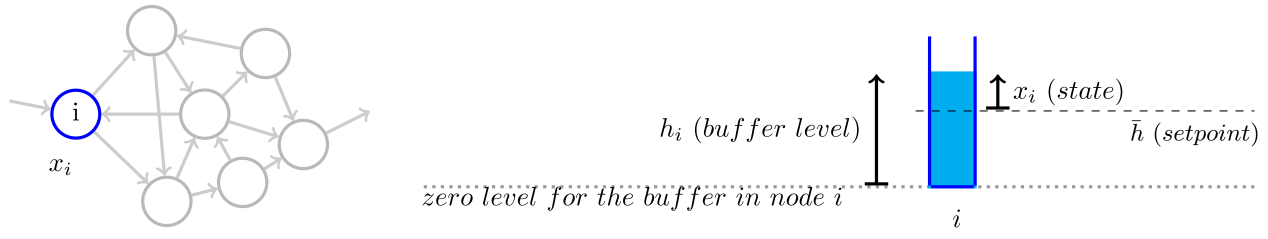
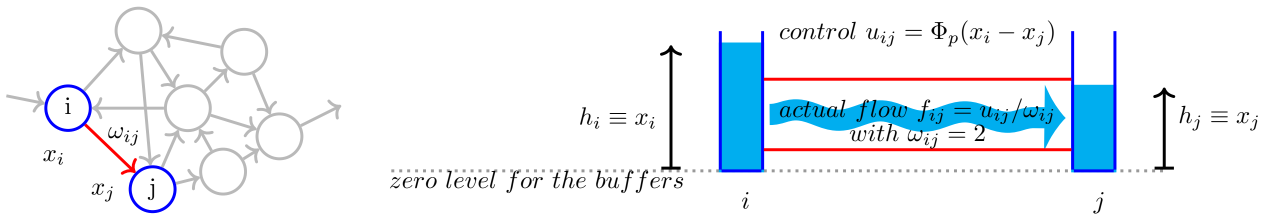


Figure 13.3: Graphical representation of the alternative definition of the state to support buffer level control.



i
j
$\omega_{ij}$
$x_i$
$x_j$
control $u_{ij} = \Phi_p(x_i - x_j)$
$h_i \equiv x_i$
actual flow $f_{ij} = u_{ij}/\omega_{ij}$ with $\omega_{ij} = 2$
$h_j \equiv x_j$
zero level for the buffers
i
j

Figure 13.4: Graphical representation of the alternative flow definition to support weighted $p$-norm minimization.

depending only on the state of the two arcs' endnodes and is given by a function $\Phi_p$ depending on some $p$. The corresponding flow is equal to the control $f_{ij} \equiv u_{ij}$, see Fig. 13.2.

In the network, there are some nodes in which a fixed flow (a demand) is imposed and must be met. Applying the proposed control under proper assumptions results in stabilizing the network, meeting the demand, and asymptotically minimizing the $p$-norm $\sqrt[p]{\sum |f_{ij}|^p}$ of the flow vector. As it will be seen, the choice of the value of $p$ influences the distribution of the flows over the arcs. Generally, large values of $p$ tend to spread the flow in all the arcs, while values of $p$ close to 1 *generally* tend to concentrate the flows in just some arcs.

Enhancements to the control include the support for unknown dynamics depending on the buffer levels. In this case, the control becomes a proportional-integral control, which still depends on local quantities: $u_{ij} = u_{ij}(p, \Phi_p, x_i - x_j)$, but not on those uncontrolled flows. This enhanced control drives the state of the nodes to zero at some steady-state. This fact can be exploited to drive the buffer levels of the nodes to some set point $\bar{h}$, by redefining the state of generic node $i$ as $x_i = h_i - \bar{h}_i$, see Fig. 13.3, which yields $h_i = \bar{h}_i$ at steady-state.

The final enhancement includes the support for the minimization of the weighted $p$-norm $\sqrt[p]{\sum |f_{ij}\omega_{ij}|^p}$, taking into account some arc costs $\omega_{ij}$. The main difference now is that the relation between the flow and the control changes as $f_{ij} = u_{ij}/\omega_{ij}$, so scaling is performed. For instance, see Fig. 13.4 and compare with Fig. 13.2: despite the same states and control, the flow is halved.

## 13.2 Literature review

The field of flow networks has been widely explored [130]; in particular, lots of research has been done regarding data transmission [131, 132], transportation networks [133, 134, 135, 136, 137], production-distribution systems [138, 139], irrigation [140], heating [141, 142], cyber-physical energy networks [143], general compartmental systems [144, 145, 146].

Here, the focus is given to the network-decentralized control of the flows in these networks. This topic was first introduced in [136, 131] and then studied in [139, 147, 148, 144, 149].

Just few works consider the asymptotic optimization of the norm of the resulting flow. In [147] a saturated network-decentralized control is presented, which asymptotically minimizes the 2-norm. The asymptotic 1-norm minimization is considered in [149], for the special case of single-source-single-sink flow network, where the flow is directed along the shortest path at steady-state. A mechanism of this kind can be used to describe natural phenomena such as lightning discharge [150], too. [151] extends this decentralized control from [147] to more general classes of smooth and *strictly convex* functionals.

Note that while the $p$-norm functional is indeed a smooth and *strictly convex* functional when $1 < p < +\infty$, the same is not true anymore for the $\infty$ and 1-norm functionals, so [151] is not applicable in these cases.

## 13.3 Contributions

To summarize, the main contribution of this Part (and [3]) is the definition of a local control that provides, among others, a global optimal result, the minimization of the $p$-norm of the flow vector; as it will be shown, this fact is possible because the considered cost function is a separable cost function [147, 151], i.e., the total cost can be written as the sum of the partial costs corresponding to the single arcs. In particular:

- the specification of a decentralized flow control strategy by means of a nonlinear function $\Phi_p$ to control the flow along the arcs, requiring only the knowledge of local information, i.e., the states of the arcs' endnodes (no information about the rest of the network is required);
- if an external flow demand is present, the proposed control ensures that this is met at steady-state;
- depending on the value of $p$, the proposed control ensures that the $p$-norm of the flow vector is minimized at steady-state, when $1 < p < +\infty$;
- when $p = 1$ or $p = \infty$ no optimality result is guaranteed: a suboptimal solution arbitrarily close to the optimum can be obtained by considering the limits of the control for $p \to 1$ and $p \to \infty$, respectively;
- the control can be enhanced to support unknown flow dynamics depending on the buffer levels (by considering a proportional-integral control), buffer level control, and weighted $p$-norm minimization (scaling the control components).

# Chapter 14
# Problem setup

In this Chapter, the considered system composed of a fluid network is first introduced. Then, the main assumptions and the main problem to be solved are stated, as well as some variations.

## 14.1 Flow networks

A flow network is considered, with $n$ nodes and $m$ arcs. The set of the nodes of the network is denoted by $\mathcal{N} = \{1, 2, \ldots, n\}$ and the set of the arcs of the network is denoted by $\mathcal{A} = \{1, 2, \ldots, m\}$.

The elements of the system are presented in detail in the following subsections.

### 14.1.1 Buffers in the nodes

A *buffer* is present in each node. The *buffer level* of the generic node $i \in \mathcal{N}$ is indicated by $h_i$. It is assumed that $h_i = 0$ is the zero reference level of the node: a negative buffer level corresponds to a physical level below this point. Moreover, a *state* is also associated with each buffer (node), and this is denoted by $x_i$ for generic node $i$.

The vectors $h(t) \in \mathbb{R}^n$ of the nodes' buffer levels at time $t$, and $x(t) \in \mathbb{R}^n$ of the nodes' states at time $t$ are introduced,

$$h(t) = \begin{bmatrix} h_1(t) \\ \vdots \\ h_n(t) \end{bmatrix}, \quad x(t) = \begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix}.$$

For simplicity, at first, it will be assumed that the states of the nodes are indeed the corresponding buffer levels,

$$x(t) \equiv h(t); \tag{14.1}$$

then, they will be both referred simply by $x(t)$. The state will be redefined in Section 16.2.

### 14.1.2 Controlled flows associated with the arcs

An *orientation* and a controllable *flow* are associated with each arc: the flow is positive if it is physically directed along the arc orientation, negative if it is directed along the opposite direction, and zero otherwise.

Some of these arcs are connected to the *external environment*: the flow associated with an arc oriented toward the external environment is called *outflow*, while the flow associated with an arc coming from the external environment is called *inflow*. The same convention for the flow sign mentioned above is applied here: for instance, a negative outflow is a flow associated with an arc orientated toward the external environment, but physically entering the network. The flows associated with arcs connecting two nodes of the network, hence not directly connected to the external environment, will be referred by *internal flows*.

The arcs associated with controllable flows will be referred to as *controllable arcs*. Such flows are decided on the basis of an applied control. A weight may also be associated with each controllable arc, and the control might take this into account.

For the generic arc $k \in \mathcal{A}$ (either joining node $i \in \mathcal{N}$ to node $j \in \mathcal{N}$, or node $i \in \mathcal{N}$ to the external environment, or the external environment to node $i \in \mathcal{N}$), this control is indicated by $u_k(t)$, the weight by $\omega_k$, and the flow by $f_k(t) = f_k(u_k(t))$.

The vectors $f \in \mathbb{R}^m$ of the arcs' flows, $u \in \mathbb{R}^m$ of the arcs' controls and $\omega \in \mathbb{R}^m$ of the arcs' weights are introduced,

$$f = \begin{bmatrix} f_1 \\ \vdots \\ f_m \end{bmatrix}, \quad u = \begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix}, \quad \omega = \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_m \end{bmatrix}.$$

For simplicity, at first, the weights of the arcs will be neglected, and it will be assumed that the flows of the arcs are equal to the corresponding applied controls,

$$u(t) \equiv f(t); \tag{14.2}$$

then, they will be both referred simply by $u(t)$. When those weights will be taken into account in Section 16.3, the flow and the control will not be coincident anymore.

### 14.1.3 The network incidence matrix

The topology of the network is specified by the (generalized) *incidence matrix* $B_I \in \mathbb{R}^{n\times m}$, whose columns describe the controllable arcs in terms of the nodes they join. In other words, its rows describe the nodes of the network in terms of the incident arcs,

$$B_I = \begin{bmatrix} \begin{bmatrix} \rotatebox{90}{$arc\ 1$} \end{bmatrix} & \dots & \begin{bmatrix} \rotatebox{90}{$arc\ m$} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} [node\ 1] \\ \vdots \\ [node\ n] \end{bmatrix}.$$

In particular, its components are defined as follows:

$$B_{I,ik} = \begin{cases} -1, & \text{if arc } k \in \mathcal{A} \text{ leaves node } i \in \mathcal{N}, \\ 1, & \text{if arc } k \in \mathcal{A} \text{ enters node } i \in \mathcal{N}, \\ 0, & \text{otherwise.} \end{cases} \tag{14.3}$$

Then, there can be three cases:

- the $k$th arc associated with an internal flow from node $i$ to node $j$ is described by the $k$th column of $B_I$, whose elements are all zero, except for $B_{ik} = -1$ and $B_{jk} = 1$;
- the $k$th arc associated with an inflow toward node $j$ is described by the $k$th column of $B_I$, whose elements are all zero, except for $B_{jk} = 1$;
- the $k$th arc associated with an outflow leaving node $i$ is described by the $k$th column of $B_I$, whose elements are all zero, except for $B_{ik} = -1$.

Note that the external environment can also be represented by an additional node.

#### The "weighted" incidence matrix

The information about the arcs' weights $\omega$ can be taken into account by computing a *"weighted" incidence matrix* $B_\omega \in \mathbb{R}^{n\times m}$. Let $\Omega \in \mathbb{R}^{m\times m}$ be defined as the diagonal matrix of the arc costs, as

$$\Omega = diag(\omega) = \begin{bmatrix} \omega_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \omega_m \end{bmatrix}. \tag{14.4}$$

Then, $B_\omega$ is defined as

$$B_\omega = B_I \Omega^{-1}. \tag{14.5}$$

Essentially, each column of $B_I$ is multiplied by the inverse of the weight of the associated arc, that is

$$B_{\omega,ik} = \begin{cases} -\frac{1}{\omega_k}, & \text{if arc } k \in \mathcal{A} \text{ leaves node } i \in \mathcal{N}, \\ \frac{1}{\omega_k}, & \text{if arc } k \in \mathcal{A} \text{ enters node } i \in \mathcal{N}, \\ 0, & \text{otherwise.} \end{cases} \tag{14.6}$$

This matrix $B_\omega$ will be exploited in Section 16.3, when the arcs' weights will be taken into account.

### 14.1.4 External, unknown, constant demand

In each node, an uncontrollable external constant flow that is unknown is imposed and must be met by setting the controllable flows of the network.

By convention, these flows are associated with some additional arcs oriented from each node toward the external environment. Essentially, they are uncontrollable constant outflows and, therefore, they are called *demand*. They follow the same sign convention presented for the controllable arcs: a *negative demand* is actually a physical request of flow from the outside to the node of the network, and a *positive demand* is actually a physical request of flow from the node of the network to the outside. The nodes with zero demand have no additional arcs connected toward the outside.

**Remark 14.1:**
The incidence matrix $B_I$ (as well as $B_\omega$) describes only the arcs whose flows can be controlled. Hence, the arcs associated with the demand are not represented in there.

For the generic node $i \in \mathcal{N}$, this demand is indicated by $d_i$. Then, the vector $d \in \mathbb{R}^n$ of the nodes' demand is introduced,

$$d = \begin{bmatrix} d_1 \\ \vdots \\ d_n \end{bmatrix}.$$

The demand is said *balanced* if there is no overall net demand, i.e., if $\bar{1}^\top d = 0$.

### 14.1.5 Unknown dynamics affecting the buffer levels

Some additional uncontrollable flows might be present, which depends on the buffer levels by an unknown dynamic. These can be internal flows, inflows (uncontrolled flow injections), or outflows (losses).

For simplicity, these are modeled by specifying the total uncontrolled flow entering each node. For the generic node $i \in \mathcal{N}$, this is modeled by the unknown function $A_i(x)$, with the convention that if the total uncontrolled flow entering a node is negative, it is physically leaving it.

The vector $A(x(t)) \in \mathbb{R}^n$ describing the unknown dynamics for the uncontrollable flows entering the nodes at time $t$ is introduced,

$$A(x(t)) = \begin{bmatrix} A_1(x(t)) \\ \vdots \\ A_n(x(t)) \end{bmatrix}.$$

For simplicity, at first the unknown dynamics $A(x)$ will not be considered,

$$A(x) \equiv \bar{0}. \tag{14.7}$$

They will be considered in Section 16.1.

**Example 14.1:**
Consider the network in Fig. 14.1.

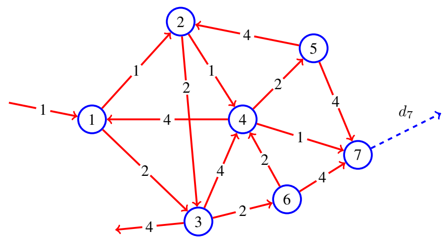


Figure 14.1: An example network. Red arcs are controllable arcs. The blue dashed arc has an uncontrollable constant demand.

The controllable arcs are labeled as follows. Arc 1: $(ext., 1)$; arc 2: $(1, 2)$; arc 3: $(1, 3)$; arc 4: $(2, 3)$; arc 5: $(2, 4)$; arc 6: $(3, ext.)$; arc 7: $(3, 4)$; arc 8: $(3, 6)$; arc 9: $(4, 1)$; arc 10: $(4, 5)$; arc 11: $(4, 7)$; arc 12: $(5, 2)$; arc 13: $(5, 7)$; arc 14: $(6, 4)$; arc 15: $(6, 7)$.
Then, the incidence matrix $B_I$, the arc costs vector $\omega$ and the weighted incidence matrix $B_\omega = B_I \Omega^{-1}$ are

$$B_I = \begin{bmatrix} 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & -1 & -1 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix},$$

$$\omega^\top = \begin{bmatrix} 1 & 1 & 2 & 2 & 1 & 4 & 4 & 2 & 4 & 2 & 1 & 4 & 4 & 2 & 4 \end{bmatrix},$$

$$B_\omega = \begin{bmatrix} 1 & -1 & -0.5 & 0 & 0 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -0.5 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & -0.25 & -0.25 & -0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0.25 & 0 & -0.25 & -0.5 & -1 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & -0.25 & -0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & -0.5 & -0.25 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0.25 & 0 & 0.25 \end{bmatrix}.$$

There is a non-zero demand only at node 7. Then, the demand vector $d$ is

$$d^\top = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & d_7 \end{bmatrix}.$$

Now assume that there are some unknown uncontrolled flows $\alpha_k(x)$ depending on the state $x$, as depicted in Fig. 14.2, left.

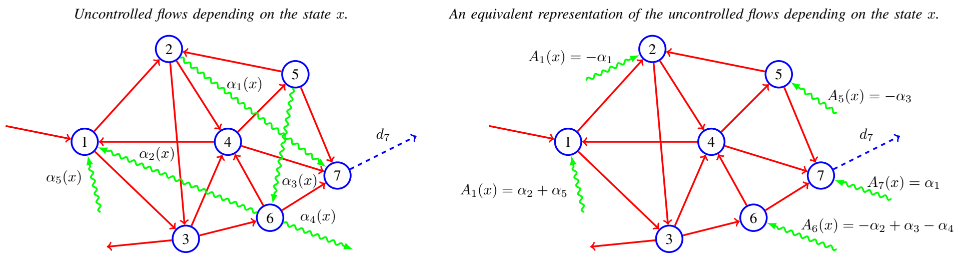

Uncontrolled flows depending on the state $x$.
An equivalent representation of the uncontrolled flows depending on the state $x$.
1
2
3
4
5
6
7
$\alpha_1(x)$
$\alpha_2(x)$
$\alpha_3(x)$
$\alpha_4(x)$
$\alpha_5(x)$
$d_7$
$A_1(x) = -\alpha_1$
$A_5(x) = -\alpha_3$
$A_1(x) = \alpha_2 + \alpha_5$
$A_7(x) = \alpha_1$
$A_6(x) = -\alpha_2 + \alpha_3 - \alpha_4$

Figure 14.2: The example network. Green arcs are uncontrollable arcs, whose flow depends on the state of the nodes. On the left, the actual uncontrolled flows $\alpha_k(x)$ are represented; on the right, the equivalent representation using functions $A_k(x)$.

The system can be modeled as in Fig. 14.2, right, by specifying the net uncontrolled flows leaving each node, which are specified by vector $A(x)$,

$$A(x)^\top = \begin{bmatrix} A_1(x) & A_2(x) & 0 & 0 & A_5(x) & A_6(x) & A_7(x) \end{bmatrix}.$$

### 14.1.6 The state equation

The buffer level of each node varies over time based on the flow in the arcs incident to the node. Taking into account the above-mentioned simplifications Eqs. (14.1), (14.2) and (14.7), a class of systems of the form

$$\dot{x}(t) = Bu(t) - d, \tag{14.8}$$

is considered, where the equality holds component-wise and $B \in \mathbb{R}^{n \times m}$ is an assigned matrix.

When $B$ is indeed the network incidence matrix $B_I$, vector $Bu$ indicates the controllable flows entering the nodes at time $t$. Recalling that $d$ is an uncontrollable constant outflow, it turns out that the state (buffer level) variation $\dot{x}(t)$ is indeed the net flow entering the nodes. Recall that if this net flow is negative for some node, a physical flow is actually leaving the node and, accordingly, its buffer level decreases. Conversely, if it is positive, a physical flow is actually entering the node, and, accordingly, its buffer level increases.

To meet the demand $d$, the stabilization of the network is also required.

**Definition 14.1:**
The network is *stabilized* if for any initial condition $x(0)$, $x(t)$ is bounded as $\|x(t)\| \leq C$, where $C > 0$ depends on $x(0)$, and it converges to a steady-state $\bar{x}$, where $\dot{x} = 0$, as $t$ goes to infinity.

Therefore, at steady-state, the following expression must hold

$$Bu(t) = d. \tag{14.9}$$

When $B \equiv B_I$, the physical flow entering a node must leave it immediately. If the demand is not balanced, $\bar{1}^\top Bu(t) = \bar{1}^\top d \neq 0$, and there might be some controlled inflows or outflows to stabilize the network.

When the above-mentioned simplifications are removed, the state equation slightly varies. In particular, the system becomes

$$\dot{x}(t) = A(x) + Bu(t) - d, \tag{14.10}$$

when Eq. (14.7) does not hold any more and the unknown dynamics $A(x)$ are present. Accordingly, at steady-state, when the network stabilizes, this becomes

$$Bu(t) = d - A(\bar{x}); \tag{14.11}$$

the unknown dynamics are still present, but they become constant and must be compensated by the control. This case will be discussed in Section 16.1.

When Eq. (14.1) does not hold anymore, the state and the buffer levels of the nodes are no more coincident. Then, one might want to specify the buffer level variation $\dot{h}(t)$. An expression for $\dot{x}(t)$ can be found, depending on the relation between the two quantities. This will be discussed in Section 16.2.

Finally, when Eq. (14.2) does not hold anymore, the flow and the control of the arcs are no more coincident. Then, the state variation might depend on $f(t)$ rather than $u(t)$. This will be discussed in Section 16.3.

## 14.2 Assumptions and requirements

No special assumption is made for the system, except for the next standing assumption, which is needed to guarantee that there exists a control $u$ stabilizing the network (i.e., such that $Bu = d$) [147, 139], when $A(x) = 0$.

**Assumption 14.1:**
Matrix $B$ has full row rank ($m \geq n$).

If $B$ is an incidence matrix, Assumption 14.1 implies that at least one column $B_k$ of $B$ must have a single non-zero entry, associated with a controlled inflow or outflow.

Moreover, it is required that the control $u(t)$ is *network-decentralized*.

**Definition 14.2:**
A state feedback control $u$ is *network-decentralized* if each component $u_k$ only depends on the buffer levels $x_i$ corresponding to nonzero entries $B_{ik}$ of the $k$th column of $B$ (and possibly on the arc cost $\omega_k$), and is independent of the demand $d$ (and of $A(x)$, if an unknown dynamic is present).

In particular, if $B$ is an incidence matrix (or a weighted incidence matrix), the control $u_k$ applied to the $k$th arc might depend only on

- the states $x_i$ and $x_j$, if the arc joins node $i \in \mathcal{N}$ to node $j \in \mathcal{N}$ (and possibly on the arc cost $\omega_k$);
- the state $x_i$, if the arc joins node $i \in \mathcal{N}$ to the external environment, or vice-versa (and possibly the arc cost $\omega_k$).

**Remark 14.2:**
The definition of network decentralized control can be extended to generic matrices $B$ requiring that control $u_j$ depends only on the node values $i$ for which $B_{ij} \neq 0$.

When the unknown dynamic $A(x)$ is present, the following Assumption is also made.

**Assumption 14.2:**
The nonlinear term $A(\cdot)$ is unknown. It is assumed that

$$A(0) = 0,$$

and that $A(x)$ is Lipschitz, namely, there exists a non-negative $L \in \mathbb{R}$, $L \geq 0$, such that, for each $x, z \in \mathbb{R}^n$,

$$\|A(z) - A(x)\|_2 \leq L\|z - x\|_2.$$

The assumption $A(0) = 0$ is not restrictive, because a nonzero term $A(0)$ could always be included in $d$, by considering $A(x)$ and $d$ as

$$\begin{aligned} A(x) \leftarrow \widetilde{A}(X) &= A(x) - A(0), \\ d \leftarrow \widetilde{d} \qquad &= d - A(0). \end{aligned}$$

Indeed, from Eq. (14.10), it holds that

$$\begin{aligned} \dot{x}(t) &= A(x) + Bu(t) - d = A(x) + Bu(t) - d + A(0) - A(0) \\ &= [A(x) - A(0)] + Bu(t) - [d - A(0)] = \widetilde{A}(x) + Bu(t) - \widetilde{d}. \end{aligned}$$

Assuming that $A(x)$ is Lipschitz is reasonable in physical systems, since the realistic dynamics of interest have a finite rate of variation in practice.

## 14.3 Minimization of the $p$-norm of the flow

The asymptotic minimization of the $p$-norm of the flow vector $f(t)$ that stabilizes the network will be considered.

**Definition 14.3: $p$-norm**
Consider a vector $f \in \mathbb{R}^m$ and $p \in \mathbb{R}, 1 \leq p \leq \infty$. The $p$-norm of vector $f$ is

$$||f||_p = \sqrt[p]{\sum_k |f_k|^p}. \tag{14.12}$$

Let $u_p^\star$ be the optimal control stabilizing the network and leading to the optimal $p$-norm flow distribution $f_p^\star$. Under the simplifications Eqs. (14.2) and (14.7), remarking that $u(t) \equiv f(t)$, i.e., $u_p^* \equiv f_p^*$, this optimal flow/control is the solution of the following problem,

$$u_p^* = \min_{Bu-d=0} ||u||_p. \tag{14.13}$$

Here, the optimal flow distribution does not depend on the buffer levels/states. When simplification Eq. (14.7) is not made, i.e., there are some uncontrolled flows governed by $A(x)$, the constraint changes, so the problem becomes

$$u_p^* = \min_{A(\bar{x})+Bu-d=0} ||u||_p. \tag{14.14}$$

Now, the optimal flow distribution depends on the states of the nodes reached at steady-state. The control that will be introduced in Section 16.1 to support this case always drives the state vector to zero, $x(t) \to \bar{x} = \bar{0}$. In fact, under Assumption 14.2, problem Eq. (14.14) reduces to Eq. (14.13).

The distribution in the network of the optimal $p$-norm flow $f_p^*$ depends on the value of $p$. On the one hand, small values of $p$ tend to concentrate the flow only along few arcs. In particular, the limit case $p = 1$ encourages *sparse solutions*, see Fig. 14.3, top-left: typically, the flow tends to be assigned only to the arcs along the shortest paths (with respect to unitary arc costs), while the others have no flow, although this is not a strict rule (see the next Example 14.2).

On the other hand, large values of $p$ tend to spread the flow among all the arcs of the network. In particular, the limit case $p = +\infty$ tends to promote *fair solutions*, see Fig. 14.3, bottom-left, where the flow is distributed in all the arcs and there exists at least one arc for which, if its flow is reduced, a larger flow must be imposed to some other arc.

### 14.3.1 Minimization of the weighted $p$-norm of the flow

Sometimes, one wants to take into account the arc costs $\omega$ when minimizing the norm of the flow vector. Then, the asymptotic minimization of the weighted $p$-norm of the flow vector $f(t)$ will also be considered.

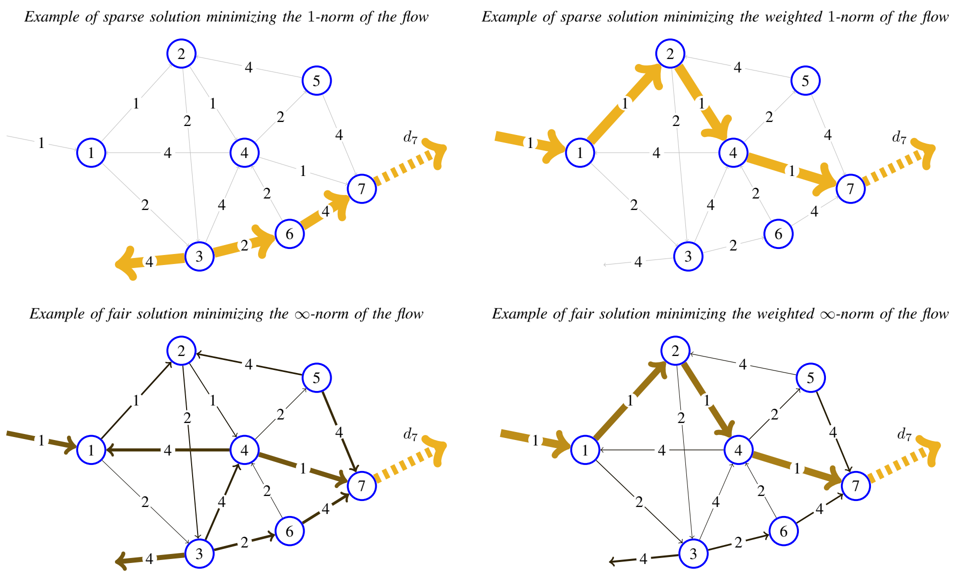

Example of sparse solution minimizing the 1-norm of the flow
Example of sparse solution minimizing the weighted 1-norm of the flow
Example of fair solution minimizing the ∞-norm of the flow
Example of fair solution minimizing the weighted ∞-norm of the flow
$d_7$

Figure 14.3: Examples of fair and sparse solutions. The colors of the arcs refer to the modulo of the corresponding flow. The total demanded flow is marked in yellow. Darker tones refer to fractions of this flow. Zero-flows are reported in gray.

**Definition 14.4: weighted $p$-norm**

Consider a vector $f \in \mathbb{R}^m$, an assigned weight vector $\omega \in \mathbb{R}^m$, and $p \in \mathbb{R}, 1 \le p \le \infty$. Let $\Omega = diag(\omega)$. The weighted $p$-norm of vector $f$ is

$$||\Omega f||_p = \sqrt[p]{\sum_k |\omega_k f_k|^p}. \tag{14.15}$$

As will be explained in Section 16.3, simplification Eq. (14.2) is not to be made when considering the weighted norm minimization: the flow $f(t)$ and the control $u(t)$ are no more coincident. Then, the optimal flow $f_p^*$ minimizing the weighted $p$-norm at steady-state is the solution to the following optimization problem:

$$f_p^* = \min_{Bf-d=0} ||\Omega f||_p. \tag{14.16}$$

It will be shown that, by setting $u = \Omega f$, this problem becomes equivalent to

$$f_p^* = \min_{B\Omega^{-1}\Omega f-d=0} ||\Omega f||_p \iff u_p^* = \min_{B\Omega^{-1}u-d=0} ||u||_p, \tag{14.17}$$

which is essentially the same as of problem Eq. (14.13), just using matrix $B\Omega^{-1}$ instead of $B$. In particular, if $B$ is the incidence matrix $B_I$, the weighted incidence matrix $B_\omega = B_I\Omega^{-1}$ is to be used.

The distribution in the network of the optimal weighted $p$-norm flow $f_p^*$ still depends on the value of $p$, as well as the values of the arc weights $\omega$. The same behavior as the one in the unweighted $p$-norm case can be observed: as $p \to 1$, the solutions tend to be sparse, see Fig. 14.3, top-right, with the flow generally assigned only to the arcs along the shortest paths, (with respect to the arc costs $\omega$), while when $p \to +\infty$, the solutions tend to be fair, see Fig. 14.3, bottom-right.

**Example 14.2:**

Consider Example 1 from [3] and, in particular, the network reported in Fig. 14.4 (left), composed by a single node (buffer), two controllable flows $u_1 \in \mathbb{R}$ and $u_2 \in \mathbb{R}$ associated with two arcs coming from the external environment, and an unknown constant demand $d \in \mathbb{R}$. The two arcs have weights $\omega_1 = 1/4$ and $\omega_2 = 1/3$. Then,

$$\omega = \begin{bmatrix} 1/4 \\ 1/3 \end{bmatrix}, \quad \Omega = \begin{bmatrix} 1/4 & 0 \\ 0 & 1/3 \end{bmatrix}, \quad B_I = \begin{bmatrix} 1 & 1 \end{bmatrix}, \quad B_\omega = B_I\Omega^{-1} = \begin{bmatrix} 4 & 3 \end{bmatrix}.$$

The steady-state equation of the system is

$$Bu - d = 0.$$

The optimal $p$-norm control, solution of Eq. (14.13) with $B = B_I$, is represented in Fig. 14.4 (center) for $p = 1, 2, \infty$; the optimal $p$-norm control, solution of Eq. (14.13) with $B = B_\omega$, is represented in Fig. 14.4 (right) for $p = 1, 2, \infty$. The latter, under $u = \Omega f$, equivalently minimizes the weighted $p$-norm of the flow vector $f$ (see Eq. (14.17)). The computations and an explanation are given next. The example from [3] is here extended, considering a more generic case.

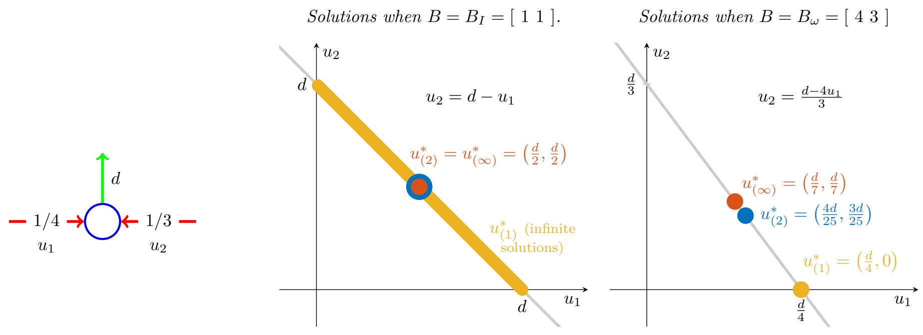

d
1/4
1/3
$u_1$
$u_2$
Solutions when $B = B_I = [\,1\ 1\,]$.
$u_2$
$d$
$u_2 = d - u_1$
$u^*_{(2)} = u^*_{(\infty)} = \left(\frac{d}{2}, \frac{d}{2}\right)$
$u^*_{(1)}$ (infinite solutions)
$d$
$u_1$
Solutions when $B = B_\omega = [\,4\ 3\,]$
$u_2$
$\frac{d}{3}$
$u_2 = \frac{d-4u_1}{3}$
$u^*_{(\infty)} = \left(\frac{d}{7}, \frac{d}{7}\right)$
$u^*_{(2)} = \left(\frac{4d}{25}, \frac{3d}{25}\right)$
$u^*_{(1)} = \left(\frac{d}{4}, 0\right)$
$\frac{d}{4}$
$u_1$

Figure 14.4: The flow problem in Example 14.2 (left) and the optimal controls that minimize the $p$-norm (center) and the weighted $p$-norm (right) of the flow vector, for $p = 1$ (yellow), $p = 2$ (cyan), $p = \infty$ (red). (*based on :* [3], © 2022 IEEE)

Consider a generic $B = \begin{bmatrix} a_1 & a_2 \end{bmatrix}$, with $a_1 \geq 0, a_2 > 0$ and a generic $d$:

$$a_1 u_1 + a_2 u_2 = d.$$

Then, setting $u_1 = r$, it holds that $u_2 = \frac{d - a_1 r}{a_2}$. The minimum $p$-norm control $u^*_{(p)}$ can be computed as follows:

- for $p = 1$, the function to minimize is

$$g(r) = |u_1| + |u_2| = |r| + \left| \frac{d - a_1 r}{a_2} \right|.$$

  $g(r)$ is a non-constant piecewise-linear non-negative function, so at least one of the knot points is a minimum point. It is immediate to see that these knot points are located at the $r$ such that $|r| = 0$ or $\left|\frac{d-a_1 r}{a_2}\right| = 0$. Then, there are two knots at

$$r_1 = 0, \quad \text{and } r_2 = \frac{d}{a_1},$$

  (which coincide when $d = 0$). Note that $g(r_1) = |d/a_2|$ and $g(r_2) = |d/a_1|$. Then, there are three possibilities:

  – if $g(r_1) > g(r_2)$, i.e., if $a_1 > a_2$, the minimum point is $r_2$ and is unique; therefore,

$$u^*_{(1)} = \begin{bmatrix} \frac{d}{a_1} \\ 0 \end{bmatrix}.$$

  See Fig. 14.5, top-left.

  – if $g(r_1) < g(r_2)$, i.e., if $a_1 < a_2$, the minimum point is $r_1$ and is unique; therefore

$$u^*_{(1)} = \begin{bmatrix} 0 \\ \frac{d}{a_2} \end{bmatrix}.$$

  See Fig. 14.5, top-right.

  – if $g(r_1) = g(r_2)$, i.e., if $a_1 = a_2$, both $r_1$ and $r_2$ are minimum points; consequently, all the points lying in the segment joining $r_1$ and $r_2$ (i.e., the $r$ such that $0 \leq r \leq d/a_1$ if $d \geq 0$, or $d/a_1 \leq r \leq 0$, if $d < 0$) are minimum points, too. Therefore, there are infinite possible solutions and there is multistability (unless $d = 0$, when $r_1$ and $r_2$ coincide and there is only one minimum point, $r = 0$). Then,

$$u^*_{(1)} \in \left\{ \begin{bmatrix} r \\ \frac{d}{a_1} - r \end{bmatrix} : \min\left(0, \frac{d}{a_1}\right) \leq r \leq \max\left(0, \frac{d}{a_1}\right) \right\}.$$

  Note that the solutions $u^*_{(1)} = \begin{bmatrix} 0 & d/a_2 \end{bmatrix}^\top$ and $u^*_{(1)} = \begin{bmatrix} d/a_1 & 0 \end{bmatrix}^\top$ are both possible. See Fig. 14.5, bottom-center.

  In the first and second cases, only one arc has a non-zero flow, and a “true” sparse solution is obtained. In the third case, however, the flow is in general partitioned in the two arcs.

If $B = B_I$, with $f(t) = u(t)$, this becomes $f^*_{(1)} = u^*_{(1)} \in \{\begin{bmatrix} r & d-r \end{bmatrix}^\top : \min(0,d) \leq r \leq \max(0,d)\}$, and there are infinite possible optimal flow distributions; for instance, $u^*_{(1)} = \begin{bmatrix} d & 0 \end{bmatrix}^\top$ or $u^*_{(1)} = \begin{bmatrix} 0 & d \end{bmatrix}^\top$, where the flow is concentrated in only one arc, or $u^*_{(1)} = \begin{bmatrix} d/2 & d/2 \end{bmatrix}^\top$ or $u^*_{(1)} = \begin{bmatrix} d/4 & 3d/4 \end{bmatrix}^\top$, where the flow is distributed in both arcs. In any case, the 1-norm of the flow/ control is minimized. The arcs are equivalent, as they have the same weight.

If $B = B_\omega$, with $u(t) = \Omega f(t)$, this becomes $u^*_{(1)} = \begin{bmatrix} d/4 & 0 \end{bmatrix}^\top$, as the arc weights are taken into account; then, the optimal flow is $f^*_{(1)} = \begin{bmatrix} d & 0 \end{bmatrix}^\top$. The 1-norm of the control and the weighted 1-norm of the flow are minimized. Note that $B_I f^*_{(1)} - d = 0$, and that the flow is concentrated in the arc with the minimum weight $\omega_1 = 1/4$.

- for $p = 2$, the (convex) function to minimize is

$$g(r) = \sqrt{u_1^2 + u_2^2} = \sqrt{r^2 + \left(\frac{d - a_1 r}{a_2}\right)^2} = \sqrt{\frac{(a_2^2 + a_1^2)\, r^2 - 2a_1 d r + d^2}{a_2^2}}.$$

Consider the equivalent problem of minimizing $h(r) = a_2^2 g^2(r) = \left(a_2^2 + a_1^2\right) r^2 - 2a_1 d r + d^2$. Then, its derivative must be zero:

$$\frac{dh(r)}{dr} = 2\left(a_2^2 + a_1^2\right) r - 2a_1 d = 0,$$

which holds for $r = \frac{a_1 d}{a_2^2 + a_1^2}$. Then,

$$u^*_{(1)} = \begin{bmatrix} \frac{a_1 d}{a_2^2 + a_1^2} \\ \frac{a_2 d}{a_2^2 + a_1^2} \end{bmatrix}.$$

Note that the solution is unique and can also be computed as $u_{(2)} = (BB^\top)^{-1} B^\top d$. The two arcs are working at the "minimum energy" $u_1^2 + u_2^2$. See Fig. 14.5, bottom-left.

If $B = B_I$, with $f(t) = u(t)$, this becomes $f^*_{(2)} = u^*_{(2)} = \begin{bmatrix} d/2 & d/2 \end{bmatrix}^\top$, and the flow is equally split in the two arcs: the 2-norm of the flow/control is minimized.

If $B = B_\omega$, with $u(t) = \Omega f(t)$, this becomes $u^*_{(2)} = \begin{bmatrix} 4d/25 & 3d/25 \end{bmatrix}^\top$, as the arc weights are taken into account; then, the optimal flow is $f^*_{(2)} = \begin{bmatrix} 16d/25 & 9d/25 \end{bmatrix}^\top$. Note that $B_I f^*_{(2)} - d = 0$. The 2-norm of the control and the weighted 2-norm of the flow are minimized.

- for $p = \infty$, the function to minimize is

$$g(r) = \max(|u_1|, |u_2|) = \max\left(|r|, \left|\frac{d - a_1 r}{a_2}\right|\right) = \begin{cases} |r|, & \text{if } |r| \geq \left|\frac{d - a_1 r}{a_2}\right|, \\ \left|\frac{d - a_1 r}{a_2}\right|, & \text{otherwise.} \end{cases}$$

Note that $g(r)$ is a piecewise-linear non-negative function. There can be two cases:

  - if $a_1 = 0$, recalling that $a_2 \neq 0$, $g(r)$ is constant and equal to $g(r) = d/a_2$ when $|r| \leq d/a_2$. Hence, there are infinite minimum points, which are given by $|r| \leq d/a_2$, and there is multistability. Then,

$$u^*_{(\infty)} \in \left\{ \begin{bmatrix} r \\ \frac{d}{a_2} \end{bmatrix} : |r| \leq \left|\frac{d}{a_2}\right| \right\}.$$

  Note that $a_1 u^*_{(\infty),1} + a_2 u^*_{(\infty),2} = a_2 u^*_{(\infty),2} = d$ regardless of $u^*_{(\infty),1}$. See Fig. 14.5, bottom-right.

  - if $a_1 \neq 0$, there is no interval of $r$ in which $g(r)$ is constant. Then, the minimum is in one of the knot points, hence, among the zeros of

$$|r| = \left|\frac{d - a_1 r}{a_2}\right|.$$

  The solutions to this equality are at most two,

$$r_1 = \frac{d}{a_1 + a_2}, \quad \text{and, if } a_1 \neq a_2,\ r_2 = \frac{d}{a_1 - a_2},$$

which coincide and are equal to 0 if $d = 0$ and $a_1 \neq a_2$. Noting that $g(r_1) = |r_1| \leq |r_2| = g(r_2)$ when $a_1 \neq a_2$, the minimum point is always $r_1$. Then,

$$u^*_{(\infty)} = \begin{bmatrix} \frac{d}{a_1+a_2} \\ \frac{d}{a_1+a_2} \end{bmatrix}.$$

Note that the flow is now the same for the two arcs, $u_1 = u_2$. See Fig. 14.5, bottom-center.

If $B = B_I$, with $f(t) = u(t)$, this becomes $f^*_{(\infty)} = u^*_{(\infty)} = \begin{bmatrix} d/2 & d/2 \end{bmatrix}^\top$, and the flow is equally split in the two arcs: the $\infty$-norm of the flow/control is minimized.

If $B = B_\omega$, with $u(t) = \Omega f(t)$, this becomes $u^*_{(\infty)} = \begin{bmatrix} d/7 & d/7 \end{bmatrix}^\top$, as the arc weights are taken into account; then, the optimal flow is $f^*_{(\infty)} = \begin{bmatrix} 4d/7 & 3d/7 \end{bmatrix}^\top$. The $\infty$-norm of the control and the weighted $\infty$-norm of the flow are minimized.

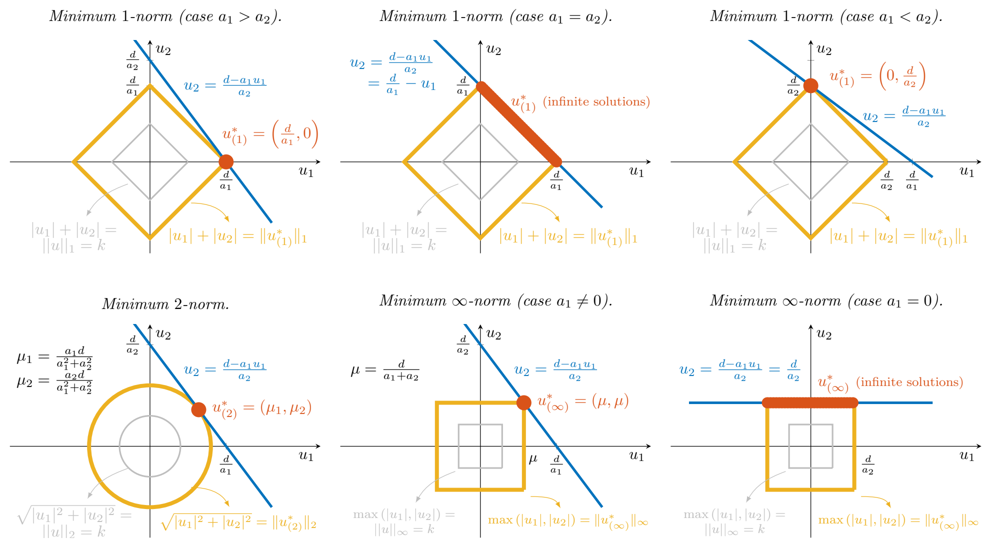

Minimum 1-norm (case $a_1 > a_2$).
Minimum 1-norm (case $a_1 = a_2$).
Minimum 1-norm (case $a_1 < a_2$).
$u_2 = \frac{d-a_1u_1}{a_2}$
$u^*_{(1)} = \left(\frac{d}{a_1}, 0\right)$
$u_2 = \frac{d-a_1u_1}{a_2} = \frac{d}{a_1} - u_1$
$u^*_{(1)}$ (infinite solutions)
$u^*_{(1)} = \left(0, \frac{d}{a_2}\right)$
$|u_1| + |u_2| = ||u||_1 = k$
$|u_1| + |u_2| = ||u^*_{(1)}||_1$
Minimum 2-norm.
Minimum $\infty$-norm (case $a_1 \neq 0$).
Minimum $\infty$-norm (case $a_1 = 0$).
$\mu_1 = \frac{a_1 d}{a_1^2+a_2^2}$
$\mu_2 = \frac{a_2 d}{a_1^2+a_2^2}$
$u^*_{(2)} = (\mu_1, \mu_2)$
$\sqrt{|u_1|^2 + |u_2|^2} = ||u||_2 = k$
$\sqrt{|u_1|^2 + |u_2|^2} = ||u^*_{(2)}||_2$
$\mu = \frac{d}{a_1+a_2}$
$u^*_{(\infty)} = (\mu, \mu)$
$\max(|u_1|, |u_2|) = ||u||_\infty = k$
$\max(|u_1|, |u_2|) = ||u^*_{(\infty)}||_\infty$
$u_2 = \frac{d-a_1u_1}{a_2} = \frac{d}{a_2}$
$u^*_{(\infty)}$ (infinite solutions)
$u_1$
$u_2$

Figure 14.5: The optimal controls $(u_1, u_2)$ minimizing the $p$-norm $||u||_p$ such that $a_1u_1 + a_2u_2 = d$, for $d \geq 0, a_1 \geq 0, a_2 > 0$, and for $p = 1$, $p = 2$, $p = \infty$. The optimal solution(s) can be obtained as the (non-empty) intersection between the line $u_2 = (d - a_1u_1)/a_2$ (blue line) and the curve given by the locus of the points for which $||u||_p = k = constant$, for the smallest possible $k$ (yellow line).

## 14.4 Problem statement for the considered model

In the sequel, the following problems are considered for the considered class of systems.

**Problem 14.1:**
Consider a network described by incidence matrix $B$. Find a network-decentralized flow control strategy $u(t)$ that stabilizes the flow network and asymptotically yields the minimum $p$-norm of the flow $f(t) \equiv u(t)$ fulfilling the demand $d$.

Some variations will also be considered, and the control will be enhanced to support them. Firstly, some uncontrollable flows depending on the buffer levels and modeled by an unknown dynamic will be introduced. The problem to consider is essentially the same as Problem 14.1: the network needs to be stabilized regardless of the presence of the unknown dynamic.

**Problem 14.2: support for unknown dynamics**
Consider a network described by incidence matrix $B$. Find a network-decentralized flow control strategy $u(t)$ that stabilizes the flow network even in the presence of some unknown dynamic $A(x)$ and asymptotically yields the minimum $p$-norm $||u(t)||_p$ of the flow $u(t)$ fulfilling the demand $d$.

A second variation considers the case in which a set-point $\bar{h}$ is set for the buffer levels.

**Problem 14.3: support for buffer level control**
Consider a network described by incidence matrix $B$. Find a network-decentralized flow control strategy $u(t)$ that stabilizes the flow network, asymptotically yields the minimum $p$-norm $||u(t)||_p$ of the flow $u(t)$ fulfilling the demand $d$, and simultaneously guarantees that $h(t) \to \bar{h}$, the reference set-point, as $t \to +\infty$.

The third and last variation considers the weights of the arcs when minimizing the $p$-norm of the flow vector.

**Problem 14.4: support for weighted $p$-norm**
Consider a network described by incidence matrix $B$ and with arc costs described by vector $\omega$. Find a network-decentralized flow control strategy $u(t)$ that stabilizes the flow network and asymptotically yields the minimum weighted $p$-norm $||\Omega^{-1}f(t)||_p$ of the flow $f(t)$ fulfilling the demand $d$.

# Chapter 15

# Networks with an uncontrolled demand

In this Chapter, Problem 14.1 is addressed. In particular, a network with an uncontrolled demand $d$ is considered. There are no uncontrolled flows depending on the buffer levels, nor set-point for the buffer levels. Also, possible arcs' weights are neglected. In other words, the simplifications Eqs. (14.1), (14.2) and (14.7) are applied. Hence, the states of the nodes are assumed equal to the corresponding buffer levels Eq. (14.1); then, both will be indicated by $x(t)$. Also, it is assumed that the flow of each arc coincides with its control Eq. (14.2); then, both will be indicated by $u(t)$. A flow control $u(t)$ must be applied to stabilize the network, meet this demand and minimize the $p$-norm of the flow vector.

## 15.1 Decentralized control for $p$-norm minimization

The following preliminary Theorem derived from [151] will be exploited to formulate the control to solve Problem 14.1.

**Theorem 15.1: Strictly convex cost**

Consider the cost

$$J(u) = \sum_{k=1}^{m} f_k(u_k),$$

where the functions $f_k : \mathbb{R} \to \mathbb{R}$ are continuously differentiable and strictly convex with strictly increasing derivatives, hence invertible. Consider the unique solution $u^*$ to the problem

$$u^* = \arg \min_{Bu-d=0} J(u), \tag{15.1}$$

as well as the strictly increasing functions

$$g_k(u_k) = \frac{d}{du_k} f_k(u_k),$$

with $g(u) = [g_1(u_1), \dots, g_m(u_m)]^\top$, and their inverse functions

$$\phi_k(\xi_k) = g_k^{-1}(\xi_k),$$

with $\phi(\xi) = [\phi_1(\xi_1), \dots, \phi_m(\xi_m)]^\top$. Then, under Assumption 14.1, the network-decentralized control

$$u(t) = \phi(-B^\top x(t))$$

ensures convergence of the trajectories of system Eq. (14.8) to the unique steady-state $\bar{x}$, whose components are equal to $\lambda^*$, the Lagrange multipliers of the optimization problem Eq. (15.1), unique solution of

$$B\phi(-B^\top \lambda^*) - d = 0, \tag{15.2}$$

that is $x(t) \to \bar{x} \equiv \lambda^*$, as well as $u(t) \to u^*$.

The following component-wise function is introduced, which depends on $p$

$$\Phi_p(\xi) = \text{sign}(\xi)\, |\xi|^{\frac{1}{p-1}}. \tag{15.3}$$

This function is visualized in Fig. 15.1 for some different values of $p$. When $p = 1$ (see the yellow line) and when $p = +\infty$ (see the red line), this function is discontinuous. For all the other values of $p$, $1 < p < \infty$, $\Phi_p$ is a symmetric, continuous and increasing function, that is also continuously differentiable and strictly convex with strictly increasing derivatives.

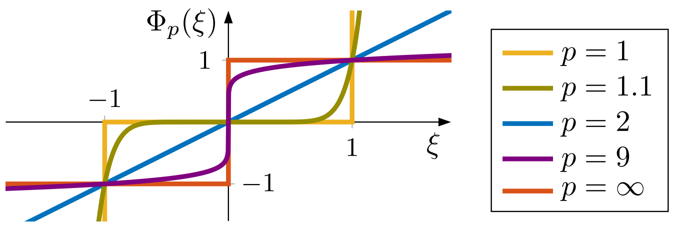


Figure 15.1: Function $\Phi_p(\xi)$ for some values of $p$. (*source:* [3], © 2022 IEEE)

Then, the following control law is proposed

$$u(t) = \Phi_p(-\gamma B^\top x(t)), \quad \gamma > 0, \tag{15.4}$$

which does not depend on the demand $d$ and is network-decentralized, as required, when $B$ is a (possibly weighted) incidence matrix. Indeed, given the definition of incidence matrix, the term $B^\top x$ is a vector whose components are associated with the controlled arcs, and each of them depends only on the state variables of the corresponding arc end nodes. Then, despite $B$ captures the network topology, only local information is used by the control of each arc, and the knowledge of the entire network is not required. There can be three cases:

- the $k$th component of Eq. (15.4) associated with an arc describing an internal flow from node $i$ to node $j$ depends only on
$$[-\gamma B^\top x]_k = \gamma(x_i - x_j);$$
- the $k$th component of Eq. (15.4) associated with an arc describing an inflow toward node $j$ depends only on
$$[-\gamma B^\top x]_k = -\gamma x_j;$$
- the $k$th component of Eq. (15.4) associated with an arc describing an outflow leaving node $i$ depends only on
$$[-\gamma B^\top x]_k = \gamma x_i.$$

**Example 15.1:**
Consider the network in Fig. 14.1 from Example 14.1, and ignore the arc costs. Let $B = B_I$ and apply control Eq. (15.4).
For brevity, consider only: arc 1, $(ext., 1)$, associated with an inflow; arc 3, $(1, 3)$, associated with an internal flow; and arc 6, $(3, ext.)$, associated with an outflow. The corresponding control components are:

$$\begin{aligned}
u_1(t) &= \Phi_p(-\gamma \left[B^\top x(t)\right]_1) = \Phi_p(-\gamma x_1) &&= \text{sign}(-x_1)\,|-\gamma x_1|^{\frac{1}{p-1}}\,,\\
u_3(t) &= \Phi_p(-\gamma \left[B^\top x(t)\right]_3) = \Phi_p(\gamma(x_1 - x_3)) &&= \text{sign}(x_1 - x_3)\,|\gamma(x_1 - x_3)|^{\frac{1}{p-1}}\,,\\
u_6(t) &= \Phi_p(-\gamma \left[B^\top x(t)\right]_6) = \Phi_p(\gamma x_3) &&= \text{sign}(x_3)\,|\gamma x_3|^{\frac{1}{p-1}}\,.
\end{aligned}$$

The other control components are similarly computed.

The proposed control is network decentralized, since its $k$th component, $u_k(t) = [\Phi_p(-\gamma \left[B^\top x(t)\right])]_k$ is a function of $B_k^\top x(t)$, where $B_k^\top$ is the $k$th row of $B^\top$, which has at most two nonzero components if $B$ is an incidence matrix. For general $B$ matrices, the control would depend exclusively on the components of the state that correspond to nonzero entries. For instance, $B_k^\top = [0\ \ -2\ \ 0\ \ -1\ \ 1\ 0\ \ 0\ \ 0]$ would mean that $u_k$ depends on $x_2$, $x_4$ and $x_5$ only, which are the only nodes that are directly affected by the $k$th flow.
The optimality of the proposed control is discussed in the following subsections.

### 15.1.1 Optimality when $1 < p < +\infty$

When $1 < p < \infty$, control Eq. (15.4) guarantees that an optimal and unique solution for Problem 14.1 is reached, as shown by the next Proposition.

**Proposition 15.1: $p$-norm minimization**

Let Assumption 14.1 be satisfied. For any real $p \in \mathbb{R}$, with $1 < p < \infty$, consider the vector $u_p^*$ as the unique solution to the problem

$$u_p^* = \arg \min_{Bu-d=0} \|u\|_p. \tag{15.5}$$

For any $\gamma > 0$, control Eq. (15.4) ensures convergence of the state of Eq. (14.8) to the equilibrium $\bar{x} = \lambda_p^*$, the Lagrange multiplier of the optimization problem Eq. (15.5), unique solution of

$$B\Phi_p(-\gamma B^\top \lambda_p^*) - d = 0. \tag{15.6}$$

The control at steady-state $u_p^* = \Phi_p(-\gamma B^\top \bar{x})$ minimizes $\|u\|_p$ under the constraint $Bu - d = 0$.

Then, for $1 < p < +\infty$,

$$u(t) \to u_p^*, \tag{15.7a}$$
$$x(t) \to \lambda_p^*, \tag{15.7b}$$

which means that the states/buffer levels at steady-state are fixed and non-controllable.

**Remark 15.1:**

The main feature of the problem that makes the decentralized control Eq. (15.4) reach a global optimality result is the fact that the cost function of the considered optimization problem Eq. (15.5) is separable [147, 151], being the $p$-norm a separable quantity; this follows from Theorem 15.1, which holds for generic separable costs under some given assumptions for the partial costs.

Moreover, when $B = B_I$ is an incidence matrix, the following property holds for the $p$-optimal control $u_p^*$.

**Proposition 15.2: No-waste at steady-state**

Let $B$ be an incidence matrix under Assumption 14.1. Then, the total controlled net inflow (i.e., the sum of the controlled inflows minus the sum of the controlled outflows) matches the total uncontrolled net outflow $\sum_k d_k$. Moreover, assume that $d_k \geq 0, \forall k$ (resp. $d_k \leq 0, \forall k$). Then, the optimal $p$-norm controlled flow $u_p^*$, $1 < p < \infty$, has no outflow (resp. no inflow) components associated with arcs to/from the external environment.

### 15.1.2 Limit cases: $p = 1$ and $p = +\infty$

Proposition 15.1 considers $1 < p < \infty$. Here, the cases in which $p = 1$ or $p = +\infty$ are analyzed. When $p = +\infty$, the control Eq. (15.4) is no longer continuous:

$$\Phi_\infty(\xi) = \operatorname{sign}(\xi),$$

while when $p = 1$, Eq. (15.4) is no longer a function:

$$\Phi_1(\xi) = \begin{cases} 0, & \text{for } |\xi| < 1, \\ -\infty, & \text{for } \xi \leq -1, \\ +\infty, & \text{for } \xi \geq 1. \end{cases}$$

Both of these cases can also be visualized in Fig. 15.1: the yellow line represents $\Phi_1(\xi)$ and the red line represents $\Phi_\infty(\xi)$.

In those two cases, the lack of strict convexity of $\Phi_p(\xi)$ makes Theorem 15.1 not applicable in proving Proposition 15.1. Indeed, the resulting controls would be discontinuous and this introduces chattering on the control, hence no asymptotic flow optimality or uniqueness can be ensured.

Still, the continuous control Eq. (15.4) for $p$ either large or close to 1 can be used to get a suboptimal solution. Indeed, in those cases, Proposition 15.1 holds. The sub-optimality of these solutions is discussed in the next Section 15.2.

## 15.2 Sub-optimality

In this Section, the limits of $\|u_p^*\|_p$ for $p \to \infty$ and $p \to 1$ are studied. It is shown that a suboptimal solution arbitrarily close to the optimum can be achieved by taking $p$ very large or close to one, respectively.

### 15.2.1 Suboptimal fair solutions: using $p \to +\infty$

The next Theorem concerns the limits of the norm $\|u_p^*\|_p$ for $p \to +\infty$.

**Theorem 15.2: $\infty$-norm case**
As $p \to \infty$, the $p$-norm optimal costs converge from above to the $\infty$-norm optimal cost:

$$\|u_p^*\|_p \to \|u_\infty^*\|_\infty.$$

A straightforward consequence of this Theorem concerns the limits of the $\infty$-norm of $u_p^*$ for $p \to +\infty$ and is stated in the following Corollary.

**Corollary 15.1: $\infty$-norm case**
As $p \to \infty$, the $\infty$-norm of the optimal $p$-norm control converge from above to the $\infty$-norm optimal cost:

$$\|u_p^*\|_\infty \to \|u_\infty^*\|_\infty.$$

In particular, letting $m$ be the size of vector $\|u_p^*\|_\infty$,

$$\|u_\infty^*\|_\infty \leq \|u_p^*\|_\infty \leq \|u_p^*\|_p \leq \|u_\infty^*\|_p \leq \sqrt[p]{m}\|u_\infty^*\|_\infty.$$

Theorem 15.2 and Corollary 15.1 guarantee that the control Eq. (15.4), applied with $p$ large enough, not only ensures the convergence of the state to the equilibrium $\bar{x}$, but also yields a steady-state flow that can be made arbitrarily close to the optimal *fair* one, getting a suboptimal solution whose $\infty$-norm can be made arbitrarily close to the optimum $\infty$-norm cost.

**Remark 15.2:**
The optimal $\infty$-norm flow $u_\infty^*$ might be not unique, as the $\infty$-norm is not strictly convex. Instead, the optimal $p$-norm flow $u_p^*$ is unique for all values $1 < p < \infty$. Then, the optimal flow $u_p^*$ converges to one of those (possibly not-unique) optimal solutions $u_\infty^*$ when $p \to \infty$. If $u_\infty^*$ is unique, the optimal flow $u_p^*$ converges to that unique optimal solution.

### 15.2.2 Suboptimal sparse solutions: using $p \to 1$

The next Theorem concerns the limits of the norm $\|u_p^*\|_p$ for $p \to +1$.

**Theorem 15.3: 1-norm case**
As $p \to 1$ from above, the $p$-norm optimal costs converge from below to the 1-norm optimal cost:

$$\|u_p^*\|_p \to \|u_1^*\|_1.$$

A straightforward consequence of this Theorem concerns the limits of the 1-norm of $u_p^*$ for $p \to 1$ and is stated in the following Corollary.

**Corollary 15.2: 1-norm case**
As $p \to 1$, the 1-norm of the optimal $p$-norm control converge from above to the 1-norm optimal cost:

$$\|u_p^*\|_1 \to \|u_1^*\|_1.$$

In particular, letting $m$ be the size of vector $\|u_p^*\|_1$,

$$\|u_p^*\|_p \leq \|u_1^*\|_p \leq \|u_1^*\|_1 \leq \|u_p^*\|_1 \leq m^{(1-\frac{1}{p})}\|u_p^*\|_p \leq m^{(\frac{p-1}{p})}\|u_1^*\|_1.$$

Theorem 15.3 and Corollary 15.2 guarantee that the control Eq. (15.4), applied with $p$ enough close to 1, not only ensures the convergence of the state to the equilibrium $\bar{x}$, but also yields a steady-state flow that can be made arbitrarily close to the optimal *sparse* one, getting a suboptimal solution whose 1-norm can be made arbitrarily close to the optimum 1-norm cost.

**Remark 15.3:**
Just like the $\infty$-norm case, the optimal 1-norm flow $u_1^*$ might be not unique, as the 1-norm is not strictly convex, too. Instead, the optimal $p$-norm flow $u_p^*$ is unique for all values $1 < p < \infty$. Then, the optimal flow $u_p^*$ converges to one of those (possibly not-unique) optimal solutions $u_1^*$ when $p \to 1$. If $u_1^*$ is unique, the optimal flow $u_p^*$ converges to that unique optimal solution.

## 15.3 Dynamic environment

So far, a static environment has been considered. In this Section, the case of a dynamic environment is briefly analyzed.

Recall that the proposed control Eq. (15.4) is independent from the network topology, the demand, and the initial state of the buffer levels. Moreover, the main optimality result in Proposition 15.1 does not make any assumption on those quantities (except for Assumption 14.1). In other words, given any network configuration, the proposed control stabilizes the network and eventually yields the optimal flow distribution (when $1 < p < \infty$).

If a modification occurs to the network changing the network topology, the demand, or the state of the buffers, the network just assumes a new configuration. If Assumption 14.1 is still satisfied, the same control guarantees that eventually a new optimal flow distribution is reached (when $1 < p < \infty$).

Changes to the network include:

- removal of an existing controllable arc (e.g., if there is a failure in such arc);
- insertion of a new controllable arc;
- removal of a node; in this case, the arcs incident to that node are removed, too (e.g., if there is a failure in such node);
- insertion of a new node; this means that new arcs connected to this node can be inserted;
- variation of the demand, which might become 0 in some nodes;
- variation of the value of $p$, i.e., of the $p$-norm to be minimized;
- (variation of the arc costs $\omega$, variation of the buffer level set point $\bar{x}$, variation of the unknown dynamics $A(x)$, see Chapter 16).

Clearly, to stabilize the network, the new configuration must be kept for a sufficient time.

**Remark 15.4:**
The insertion or removal of arcs or nodes clearly modify the incidence matrix $B$. For instance, by inserting a new controllable arc, a new column is to be inserted in $B$; by removing a node, the corresponding row is to be removed (and possibly the columns of the arcs incident to that node, too). Still, recall that the control of each arc depends only on the difference between the state of its end nodes, so the changes in the dimensions of $B$ do not affect the control.

When a generic system is considered, i.e., when matrix $B$ satisfies Assumption 14.1, but it is not necessarily an incidence matrix, the control is still effective if a modification to $B$ occurs, which makes the control adaptive.

## 15.4 Optimal solutions by linear-quadratic programming

To evaluate the network-decentralized control, it is useful to introduce some linear quadratic programming problems to efficiently compute the optimal control $u_p^\star$ which minimizes the required norm stabilizing the network Eq. (15.5). These computations are to be performed offline, and all the network data is to be known.

In particular, consider a generic linear-quadratic programming problem with $q$ variables described by vector $y \in \mathbb{R}^{q\times 1}$, $r$ inequality constraints and $s$ equality constraints, which can be written in the form

$$\begin{aligned}
\min_{u} \quad & \frac{1}{2} y^\top H y + f^\top y, && (15.8a)\\
\text{s.t.} \quad & A \cdot y \leq b, && (15.8b)\\
& A_{eq} \cdot y = b_{eq}, && (15.8c)\\
& lb \leq y \leq ub, && (15.8d)
\end{aligned}$$

where the matrices $H \in \mathbb{R}^{q\times q}$, $A \in \mathbb{R}^{r\times q}$, $A_{eq} \in \mathbb{R}^{s\times q}$ and the vectors $f, lb, ub \in \mathbb{R}^{q\times 1}$, $b \in \mathbb{R}^{r\times 1}$, $b_{eq} \in \mathbb{R}^{s\times 1}$ are to be defined.

The formulations of these linear-quadratic programming problems depend on the value of $p$. In the following, the formulations for $p = 1, 2, \infty$ are reported, which are the three main cases of interest.

### 15.4.1 Optimal 1-norm solution

The problem to consider is

$$u_1^* = \arg \min_{Bu-d=0} \|u\|_1,$$

with

$$\|u\|_1 = \sqrt[1]{\sum_k |u_k|^1} = \sum_k |u_k|.$$

Then, the following problem is considered:

$$\begin{aligned} \min_u \quad & \sum_k |u_k|, && (15.9a)\\ \text{s.t.} \quad & Bu - d = 0. && (15.9b) \end{aligned}$$

To implement this efficiently via linear programming, this problem needs to be manipulated as follows. Recalling that the absolute value can be written as

$$|u_k| = \max(-u_k, u_k),$$

vector $\sigma \in \mathbb{R}^{m\times 1}$ is introduced, whose components $\sigma_k$ satisfy $\sigma_k \geq |u_k| \geq 0$ for all $k \in \mathcal{A}$, that is

$$\sigma_k \geq -u_k, \quad \text{and} \quad \sigma_k \geq u_k.$$

Then, the following equivalent linear programming problem is to be considered, which minimizes the sum of the components of $\sigma$:

$$\begin{aligned} \min_{u,\sigma} \quad & \sum_k \sigma_k, && && (15.10a)\\ \text{s.t.} \quad & Bu - d = 0, && && (15.10b)\\ & \sigma_k \geq u_k, && \forall k \in \mathcal{A}, && (15.10c)\\ & \sigma_k \geq -u_k, && \forall k \in \mathcal{A}, && (15.10d)\\ & \sigma_k \geq 0, && \forall k \in \mathcal{A}. && (15.10e) \end{aligned}$$

This can be simply implemented in the form of problem Eq. (15.8) by taking

$$y = \begin{bmatrix} u_1 \\ \vdots \\ u_m \\ \sigma_1 \\ \vdots \\ \sigma_m \end{bmatrix}, \quad H = \begin{bmatrix} 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & \dots & 0 \end{bmatrix}, \quad f = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \quad lb = \begin{bmatrix} -\infty \\ \vdots \\ -\infty \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad ub = \begin{bmatrix} +\infty \\ \vdots \\ +\infty \\ +\infty \\ \vdots \\ +\infty \end{bmatrix},$$

$$A = \begin{bmatrix} -1 & \dots & 0 & -1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & -1 & 0 & \dots & -1 \\ 1 & \dots & 0 & -1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 1 & 0 & \dots & -1 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad A_{eq} = \left[\begin{array}{c|ccc} B & 0 & \dots & 0 \\ & \vdots & \ddots & \vdots \\ & 0 & \dots & 0 \end{array}\right], \quad b_{eq} = d.$$

**Remark 15.5:**
The 1-norm optimal flow, solution of Eq. (15.9), might be not unique. Problem Eq. (15.10) provides a possible $u_1^*$ minimizing the 1-norm, so that the minimal $\|u_1^*\|_1$ is obtained.

### 15.4.2 Optimal 2-norm solution

The problem to consider is

$$u_2^* = \arg \min_{Bu-d=0} \|u\|_2,$$

with

$$\|u\|_2 = \sqrt[2]{\sum_k |u_k|^2} = \sqrt[2]{\sum_k u_k^2}.$$

Firstly, minimizing $\|u\|_2$ is equivalent to minimizing $\|u\|_2^2$. Then, the following quadratic programming problem is presented:

$$\min_u \quad \sum_k u_k^2, \tag{15.11a}$$

$$\text{s.t.} \quad Bu - d = 0. \tag{15.11b}$$

This can be simply implemented in the form of problem Eq. (15.8) by taking

$$y = \begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix}, \qquad H = \begin{bmatrix} 2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 2 \end{bmatrix}, \qquad f = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \qquad lb = \begin{bmatrix} -\infty \\ \vdots \\ -\infty \end{bmatrix}, \qquad ub = \begin{bmatrix} +\infty \\ \vdots \\ +\infty \end{bmatrix},$$

$$A, b = \emptyset, \qquad A_{eq} = B, \qquad b_{eq} = d.$$

**Remark 15.6:**
Problem Eq. (15.11) is convex and has a unique optimal solution $u_2^*$, which provides the optimal 2-norm cost $\|u_2^*\|_2$.

### 15.4.3 Optimal $\infty$-norm solution

The problem to consider is

$$u_\infty^* = \arg \min_{Bu-d=0} \|u\|_\infty,$$

with

$$\|u\|_\infty = \sqrt[\infty]{\sum_k |u_k|^\infty} = \max(|u_k|).$$

Then, the following problem is presented:

$$\min_u \quad \max(|u_k|), \tag{15.12a}$$

$$\text{s.t.} \quad Bu - d = 0. \tag{15.12b}$$

As in the case of the 1-norm, to implement this efficiently, the modulo in the objective function needs to be linearized. However, now only a single variable $\pi \in \mathbb{R}$ needs to be introduced, which satisfies $\pi \geq \max(|u_k|) \geq |u_k| \geq 0$ for all $k \in \mathcal{A}$, that is

$$\pi \geq -u_k, \quad \text{and} \quad \pi \geq u_k.$$

Then, the following equivalent linear programming problem is to be considered, which minimizes $\pi$:

$$\min_{u,\pi} \quad \pi, \tag{15.13a}$$

$$\text{s.t.} \quad Bu - d = 0, \tag{15.13b}$$

$$\pi \geq -u_k, \qquad \text{for } k \in \mathcal{A}, \tag{15.13c}$$

$$\pi \geq u_k, \qquad \text{for } k \in \mathcal{A}, \tag{15.13d}$$

$$\pi \geq 0. \qquad . \tag{15.13e}$$

This can be simply implemented in the form of problem Eq. (15.8) by taking

$$y = \begin{bmatrix} u_1 \\ \vdots \\ u_m \\ \pi \end{bmatrix}, \qquad H = \begin{bmatrix} 0 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & 0 \\ 0 & \dots & 0 & 0 \end{bmatrix}, \qquad f = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \qquad lb = \begin{bmatrix} -\infty \\ \vdots \\ -\infty \\ 0 \end{bmatrix}, \qquad ub = \begin{bmatrix} +\infty \\ \vdots \\ +\infty \\ +\infty \end{bmatrix},$$

$$A = \begin{bmatrix} -1 & \dots & 0 & -1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & -1 & -1 \\ 1 & \dots & 0 & -1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 1 & -1 \end{bmatrix}, \qquad b = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \qquad A_{eq} = \left[\begin{array}{c|c} B & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \end{array}\right], \qquad b_{eq} = d.$$

**Remark 15.7:**
The $\infty$-norm optimal flow, solution of Eq. (15.12), might be not unique. Problem Eq. (15.13) provides a possible $u^*_\infty$ minimizing the $\infty$-norm, so that the minimal $\|u^*_\infty\|_\infty$ is obtained.

# Chapter 16

# Enhancements to the control

In this Chapter, some enhancements to the control proposed in Chapter 15 are presented.

In particular, at first Problems 14.2 to 14.4 are addressed, where the simplifications Eqs. (14.1), (14.2) and (14.7) are removed.

Then, the problems due to the numeric implementation of the control are discussed, and some possible solutions to reduce their effects are also presented.

## 16.1 Handling networks with unknown dynamics

In this Section, Problem 14.2 is addressed, where the simplification Eq. (14.7) is not made, but simplifications Eqs. (14.1) and (14.2) are kept. In particular, a network with an uncontrolled demand $d$ and some uncontrolled flows depending on the buffer levels described by the dynamic $A(x)$ (see subsection 14.1.5) is considered. There is no set-point for the buffer levels, and possible arcs' weights are neglected.

Then, a network-decentralized flow control strategy $u(t)$ must be applied to stabilize the network, meet the demand $d$ regardless of the unknown dynamics, and asymptotically minimize the $p$-norm of the flow vector.

As discussed in subsection 14.1.6, the system state equation becomes Eq. (14.8), that is

$$\dot{x}(t) = A(x) + Bu(t) - d.$$

Due to the term $A(x)$, the control Eq. (15.4) presented in Chapter 15 cannot ensure that at steady-state $x = 0$ (i.e., under Assumption 14.2, $A(x) = 0$). Therefore, it has to be adapted for this purpose.

To this aim, an integral variable $\xi_i \in \mathbb{R}$ is introduced for each node $i$, which is proportional to the corresponding state $x_i$. Vector $\xi \in \mathbb{R}^n$ is therefore defined, as

$$\xi = \begin{bmatrix} \xi_1 \\ \vdots \\ \xi_n \end{bmatrix}.$$

Then, the following proportional-integral control is proposed, whose specification depends on the value of $p$:

$$u(t) = \begin{cases} -\gamma B^\top x + \Phi_p(-\gamma B^\top \xi), & \text{if } 1 < p \le 2, \\ \Phi_p(-\gamma B^\top (x + \xi)), & \text{if } 2 \le p < +\infty, \end{cases} \tag{16.1a}$$

$$\dot{\xi}(t) = \alpha x, \quad \xi(0) = 0, \tag{16.1b}$$

with $\alpha > 0$ arbitrarily given.

This control does not depend on the demand $d$, nor the unknown dynamic $A(x)$. When $2 \le p < \infty$, the integral variable is added to the state variable in the nonlinear function $\Phi_p$ from Eq. (15.3). When $1 < p \le 2$ the system is stabilized by a linear term depending on the state and the integral variable is in $\Phi_p$. Note that, when $p = 2$, the two cases of the formulation of the control are the same:

$$-\gamma B^\top x + \Phi_2(-\gamma B^\top \xi) = \Phi_2(-\gamma B^\top (x + \xi)) = -\gamma B^\top (x + \xi).$$

The control remains decentralized. Indeed, each control component $u_k$ depends only on state variables *and their integrals* corresponding to nonzero entries of the $k$th column of $B$ (i.e., the $k$th row of $B^\top$).

**Example 16.1:**
Consider the network in Fig. 14.1 from Example 14.1, and ignore the arc costs. Let $B = B_I$ and apply control Eq. (16.1).
For brevity, consider only: arc 1, $(ext., 1)$, associated with an inflow; arc 3, $(1,3)$, associated with an internal flow; and arc 6, $(3, ext.)$, associated with an outflow. The corresponding control components are:

$$u_1(t) = \begin{cases} -\gamma[B^\top x]_1 + \Phi_p(-\gamma[B^\top \xi]_1) = -\gamma x_1 + \Phi_p(-\gamma \xi_1), & \text{if } 1 < p \leq 2, \\ \Phi_p(-\gamma[B^\top(x+\xi)]_1) \qquad = \Phi_p(-\gamma(x_1+\xi_1)), & \text{if } 2 \leq p < +\infty, \end{cases}$$
$$u_3(t) = \begin{cases} -\gamma[B^\top x]_3 + \Phi_p(-\gamma[B^\top \xi]_3) = \gamma(x_1 - x_3) + \Phi_p(\gamma(\xi_1 - \xi_3)), & \text{if } 1 < p \leq 2, \\ \Phi_p(-\gamma[B^\top(x+\xi)]_3) \qquad = \Phi_p(\gamma(x_1+\xi_1 - x_3 - \xi_3)), & \text{if } 2 \leq p < +\infty, \end{cases}$$
$$u_6(t) = \begin{cases} -\gamma[B^\top x]_6 + \Phi_p(-\gamma[B^\top \xi]_6) = \gamma x_3 + \Phi_p(\gamma \xi_3), & \text{if } 1 < p \leq 2, \\ \Phi_p(-\gamma[B^\top(x+\xi)]_6) \qquad = \Phi_p(\gamma(x_3+\xi_3)), & \text{if } 2 \leq p < +\infty. \end{cases}$$

The other control components are similarly computed.

### 16.1.1 Optimality when $1 < p < \infty$

The next Theorem claims the optimality of control Eq. (16.1) when $2 \leq p < \infty$.

**Theorem 16.1: Dynamic network-decentralized control**
For $2 \leq p < \infty$, under Assumptions 14.1 and 14.2, consider the proportional-integral control

$$u = \Phi_p\left(-\gamma B^\top(x+\xi)\right), \tag{16.2a}$$
$$\dot{\xi} = \alpha x, \quad \xi(0) = 0, \tag{16.2b}$$

with $\alpha > 0$ arbitrarily given. Consider the initial domain

$$x(0) \in \mathcal{X}_0 = \left\{ x : \|x\|^2 \leq \rho_0^2 = \frac{\rho^2}{8} - \frac{3}{2}\|\bar{\xi}\|^2 \right\},$$

with given $\rho^2 > 12\|\bar{\xi}\|^2$, where $\bar{\xi} = \lambda_p^*$, the Lagrange multiplier of the optimization problem Eq. (15.5), is the unique vector that solves

$$B\Phi_p\left(-\gamma B^\top \bar{\xi}\right) - d = 0.$$

Then, there exist $\gamma > 0$ such that $x(t) \to 0$, (and then $A(x(t)) \to 0$, by Assumption 14.2), $u(t) \to u_p^*$ and $\xi(t) \to \bar{\xi} = \lambda_p^*$.

Note that, by Theorem 16.1, when $2 \leq p < \infty$ control Eq. (16.1) makes the state converge to zero, $x \to \bar{x} = \bar{0}$. By Assumption 14.2, this means that at steady-state $A(\bar{x}) = A(\bar{0}) = \bar{0}$ and that the steady-state condition Eq. (14.11) coincides to Eq. (14.9). In other words, the optimal flow stabilizing the network to some steady-state $\bar{x}$ in the presence of an unknown dynamic $A(x)$ is

$$u_p^* = \min_{A(\bar{x})+Bu-d=0} ||u||_p = \min_{Bu-d=0} ||u||_p. \tag{16.3}$$

This justifies the reference to Eq. (15.5) in Theorem 16.1: under Assumption 14.2, $\lambda_p^*$ is indeed also the optimal Lagrange multiplier vector of problem Eq. (16.3).

Theorem 16.1 cannot hold when considering $1 < p < 2$, see [3]. Indeed, the control is defined differently in Eq. (16.1) for $1 < p < 2$. The following Assumption is needed to prove the next Theorem 16.2.

**Assumption 16.1:**
The optimal flow $u_p^*$ corresponding to $d$ has at least $n$ nonzero components. The corresponding columns of $B$ have rank $n$.

Then, the optimality of the control Eq. (16.1) when $1 < p \leq 2$ follows from the following Theorem:

**Theorem 16.2:**
Let $1 < p \leq 2$ and Assumptions 14.1, 14.2 and 16.1 be satisfied with $A(x)$ smooth. The closed-loop system admits the unique steady-state $x = 0$ and $\xi = \bar{\xi}$ and $u$ is the optimal $u_p^*$. The steady-state is locally stable for

$\gamma > 0$ large enough (which exists because $BB^\top$ is positive definite), such that

$$[\bar{A} - \gamma BB^\top]^\top + [\bar{A} - \gamma BB^\top] = -Q \prec 0,$$

where $\bar{A}$ is the Jacobian of $A(x)$ evaluated at 0.

To summarize, the control Eq. (16.1) guarantees that, when $1 < p < +\infty$, at steady-state the optimal and unique solution of the optimization problem Eq. (16.3) is reached. Also, the state is always driven to zero, regardless of the values of $p, B, d, A(x)$: this property will be exploited in the next Section 16.2 to support buffer level control. Then,

$$u(t) \to u_p^*, \tag{16.4a}$$
$$x(t) \to \bar{0}, \tag{16.4b}$$
$$\xi(t) \to \lambda_p^*. \tag{16.4c}$$

**Remark 16.1:**
Proposition 15.2 continues to hold if there is an uncontrolled dynamic $A(x)$ with $A(x) \to 0$, even if the external uncontrolled demand also takes into account the effect of the dynamics: $\hat{d} = d - A(0)$.

Consider now the limit cases of $p = 1$ and $p = +\infty$. As $\Phi_p$ is discontinuous for $p = 1$ and $p = +\infty$, no asymptotic flow optimality or uniqueness can be ensured for the control Eq. (16.1). As in Chapter 15, sub-optimal fairer solutions arbitrarily close to the optimum $\infty$-norm flow can be achieved by applying the control Eq. (16.1) for $p \to \infty$, and sub-optimal sparser solutions arbitrarily close to the optimum 1-norm flow can be achieved by applying the same control Eq. (16.1) for $p \to 1$ (see Section 15.2).

**Remark 16.2: Dynamic environment**
For the same reasons presented in Section 15.3, the control Eq. (16.1) is still adaptive. If a modification occurs to the network and the new configuration is kept for a sufficient time, the network stabilizes and eventually, the flow distribution still minimizes the required norm, provided that Assumptions 14.1 and 14.2 (and Assumption 16.1 when $1 < p < 2$) are still satisfied.

### 16.1.2 Optimal solutions by linear-quadratic programming

To evaluate the network-decentralized control in the presence of an unknown dynamic, some linear quadratic programming problems can be formulated to efficiently compute the optimal control which minimizes the required norm stabilizing the network.

The optimization problem to consider is indeed Eq. (16.3), as under Assumption 14.2, control Eq. (16.1) makes $x(t) \to \bar{0}$ with $A(\bar{0}) = \bar{0}$. That is exactly the same problem as Eq. (15.5), the one considered in Chapter 15. Hence, to find optimal solutions $u_p^*$ offline, the linear-quadratic programming problem formulations from Section 15.4 can be used.

Recall that if $A(\bar{0}) = \bar{0}$ is removed from Assumption 14.2, the following transformation must be applied,

$$d \to A(\bar{0}) + d.$$

## 16.2 Support for buffer level control

In this Section, Problem 14.3 is addressed, where a set-point is imposed for the buffer levels: the support for buffer level control is therefore introduced. A network with just an uncontrolled demand $d$ is considered: there are no uncontrolled flows depending on the buffer levels, and possible arcs' weights are neglected, i.e., simplifications Eqs. (14.2) and (14.7) are kept.

Then, a network-decentralized flow control strategy $u(t)$ must be applied to stabilize the network, meet the demand $d$, asymptotically minimize the $p$-norm of the flow vector and simultaneously guarantee that the set-point of the buffer levels is reached at steady-state.

To this aim, simplification Eq. (14.1) is removed, and the state is redefined as follows. Recall that $h(t)$ is the vector describing the buffer levels of the nodes, and $\bar{h}$ is the vector of the corresponding set points describing some desired buffer levels.

Suppose that at steady-state, the buffer level $h_i$ of each node $i$ must be driven to its set point $\bar{h}_i$. Then, the following must be enforced

$$h(t) \to \bar{h}. \tag{16.5}$$

So far, the state of every node has been essentially set as its buffer level, as of simplification Eq. (14.1). Now, that this simplification is removed, the state of each node $i$ is redefined as the difference between its buffer level $h_i$ and its set point $\bar{h}_i$. Then, the state becomes

$$x(t) = h(t) - \bar{h}. \tag{16.6}$$

By this definition, by Eq. (16.5), it follows that the state necessarily has to converge to zero, that is

$$x(t) \to \bar{h} - \bar{h} = \bar{0}. \tag{16.7}$$

Then, the control Eq. (16.1) can be exploited to support buffer level control. Indeed, recall that this control guarantees that at steady-state the state becomes zero, as of Theorems 16.1 and 16.2. Thus, an optimal and unique solution $u_p^*$ is obtained when $1 < p < +\infty$ even if a set point is imposed for the buffer levels: the integral variables $\xi(t)$ converge to the optimal Lagrange multiplier vector $\lambda_p^*$ of the optimization problem Eq. (16.3). Then,

$$\begin{align} u(t) &\to u_p^*, \tag{16.8a}\\ x(t) &\to \bar{0}, \tag{16.8b}\\ h(t) &\to \bar{h}, \tag{16.8c}\\ \xi(t) &\to \lambda_p^*. \tag{16.8d}\end{align}$$

Note that this control Eq. (16.1) can be used even if there are no uncontrolled flows due to some unknown dynamics: if $A(x) \equiv 0$, the state is still driven to zero. The support for handling possible unknown dynamics is also automatically guaranteed, under the proper assumptions. The same properties regarding the adaptability to dynamic environments, and the possibility to get suboptimal 1 and $\infty$-norm solutions apply.

**Remark 16.3:**
Instead of specifying the dynamic of the state $x(t)$, the dynamics of the buffer levels can be specified

$$\dot{h}(t) = A(h) + Bu(t) - d, \tag{16.9}$$

where the unknown dynamics depend on the buffer levels, too. By Eq. (16.6), it holds that $\dot{h}(t) \equiv \dot{x}(t)$. Then, the corresponding state equation is

$$\dot{x}(t) = A(x + \bar{h}) + Bu(t) - d. \tag{16.10}$$

If $\bar{h} \neq \bar{0}$, the equation can be rewritten as

$$\begin{aligned} \dot{x}(t) &= A(x + \bar{h}) + Bu(t) - d + A(\bar{h}) - A(\bar{h})\\ &= [A(x + \bar{h}) - A(\bar{h})] + Bu(t) - [d - A(\bar{h})]\\ &= \hat{A}(x) + Bu(t) - \hat{d}, \end{aligned}$$

where the following quantities have been defined,

$$\begin{aligned} \hat{A}(x) &= A(x + \bar{h}) - A(\bar{h}),\\ \hat{d} &= d - A(\bar{h}). \end{aligned}$$

Note that if the state is driven to zero, $\hat{A}(\bar{x}) = \hat{A}(\bar{0}) = \bar{0}$, under Assumption 14.2.

### 16.2.1 Optimal solutions by linear-quadratic programming

The optimal flow distribution does not depend on the buffer levels. Hence, when the buffer level control is required, the optimal solution $u_p^*$ is computed as in Section 15.4, even if there is an unknown dynamic, see subsection 16.1.2.

## 16.3 Support for weighted norm minimization

In this Section, Problem 14.4 is addressed, where arcs' weights are taken into account when minimizing the norm of the flow vector: the support for weighted $p$-norm minimization is therefore introduced. A network with just an uncontrolled demand $d$ is considered: for simplicity, suppose that there are no uncontrolled flows depending on the buffer levels, nor set-points for the buffer levels, i.e., simplifications Eqs. (14.1) and (14.7) are kept.

Recall that $u(t)$ is the vector describing the control applied to the arcs, $f(t)$ is the vector of the corresponding flows, and $\omega$ is the vector of the corresponding arcs' weights. Then, a network-decentralized flow control strategy $u(t)$ must be applied to stabilize the network, meet the demand $d$, and asymptotically minimize the weighted $p$-norm of the flow vector $f(t)$, that is $\|\Omega f(t)\|_p$, with $\Omega = diag(\omega)$.

To this aim, simplification Eq. (14.2) is removed, and the relation between the flow and the control is redefined as follows.

Assume that the state variation depends on the flows $f(t)$, rather than the controls, and some matrix $C \in \mathbb{R}^{n\times m}$ satisfying Assumption 14.1, that is

$$\dot{x}(t) = Cf(t) - d. \tag{16.11}$$

Then, the relation between the flow and the control is set so that for arc $k \in \mathcal{A}$, the flow $f_k$ is equal to the control $u_k$ multiplied by the arc weight $\omega_k$, that is $u_k(t) = \omega_k f_k(t)$. Equivalently,

$$u(t) = \Omega f(t).$$

Moreover, matrix $B \in \mathbb{R}^{n\times m}$ is defined as

$$B = C\Omega^{-1}.$$

As $\Omega$ is a diagonal matrix, if $B$ satisfies Assumption 14.1, so does $C$ and vice-versa.

On the one hand, a reason for imposing these relations is that the weighted p-norm of the generic flow $f(t)$, which is to be minimized, is equal to the p-norm of the generic control $u(t)$ for all $t$, by construction:

$$\|\Omega f(t)\|_p \equiv \|u(t)\|_p.$$

On the other hand, another reason is that the state variation $Cf(t)$ due to the flow is equal to the state variation $Bu(t)$ due to the control, that is,

$$\dot{x}(t) = Cf(t) - d = C\Omega^{-1}\Omega f(t) - d = Bu(t) - d.$$

This means that the problem of finding a stabilizing control $u(t)$ minimizing the weighted $p$-norm of the flow vector $f(t)$ in a weighted network under Eq. (16.11) becomes equivalent to the problem of finding a stabilizing control $u(t)$ minimizing the $p$-norm of $u(t) \equiv f(t)$ in a network under Eq. (14.8). This latter case is the one considered in Chapter 15. Hence, the control Eq. (15.4),

$$u(t) = \Phi_p(-\gamma B^\top x) = \Phi_p(-\gamma \left[\Omega^{-1}\right]^\top C^\top x)$$

can be successfully applied to solve Problem 14.4. Then, the actual flow can be easily computed as

$$f(t) = \Omega^{-1}\Phi_p(-\gamma B^\top x) = \Omega^{-1}\Phi_p(-\gamma \left[\Omega^{-1}\right]^\top C^\top x).$$

Note that if $C$ is the network incidence matrix, $C = B_I$, the matrix $B$ to be used in the control is the corresponding weighted incidence matrix $B = B_\omega = B_I\Omega^{-1}$, see subsection 14.1.3. The control remains decentralized. Indeed, there can be three cases:

- the $k$th component of Eq. (15.4) associated with an arc with cost $\omega_k$ describing an internal flow from node $i$ to node $j$ depends only on the term $[-\gamma B^\top x]_k = \frac{\gamma}{\omega_k}(x_i - x_j)$;
- the $k$th component of Eq. (15.4) associated with an arc with cost $\omega_k$ describing an inflow toward node $j$ depends only on the term $[-\gamma B^\top x]_k = -\frac{\gamma}{\omega_k}x_j$;
- the $k$th component of Eq. (15.4) associated with an arc with cost $\omega_k$ describing an outflow leaving node $i$ depends only on the term $[-\gamma B^\top x]_k = \frac{\gamma}{\omega_k}x_i$.

**Example 16.2:**
Consider the network in Fig. 14.1 from Example 14.1, and take into account the arc costs. Let $B = B_\omega$ and apply control Eq. (15.4).
For brevity, consider only: arc 1, $(ext., 1)$, associated with an inflow with $\omega_1 = 1$; arc 3, $(1, 3)$, associated with an internal flow with $\omega_3 = 2$; and arc 6, $(3, ext.)$, associated with an outflow with $\omega_6 = 4$. The corresponding control components are:

$$u_1(t) = \Phi_p\left(-\gamma\left[B^\top x(t)\right]_1\right) = \Phi_p\left(-\frac{\gamma}{\omega_1}x_1\right) \qquad = \text{sign}\left(-x_1\right)\left|-\frac{\gamma x_1}{1}\right|^{\frac{1}{p-1}},$$

$$u_3(t) = \Phi_p\left(-\gamma\left[B^\top x(t)\right]_3\right) = \Phi_p\left(\frac{\gamma}{\omega_3}(x_1 - x_3)\right) = \text{sign}\,(x_1 - x_3)\left|\frac{\gamma(x_1 - x_3)}{2}\right|^{\frac{1}{p-1}},$$
$$u_6(t) = \Phi_p\left(-\gamma\left[B^\top x(t)\right]_6\right) = \Phi_p\left(\frac{\gamma}{\omega_6}x_3\right) = \text{sign}\,(x_3)\left|\frac{\gamma x_3}{4}\right|^{\frac{1}{p-1}}.$$

The other control components are similarly computed.

**Remark 16.4:**
Under $B = C\Omega^{-1}$ and $u(t) = \Omega f(t)$, if there is an uncontrolled unknown dynamic $A(x)$, i.e., Eq. (16.11) is modified into

$$\dot{x}(t) = A(x) + Cf(t) - d = A(x) + Bu(t) - d,$$

or a set-point is imposed for the buffer levels, the optimal control stabilizing the network and asymptotically minimizing the weighted $p$-norm of the flow vector $f(t)$ is Eq. (16.1), i.e.,

$$u(t) = \begin{cases} -\gamma B^\top x + \Phi_p(-\gamma B^\top \xi), & \text{if } 1 < p \le 2, \\ \Phi_p(-\gamma B^\top (x + \xi)), & \text{if } 2 \le p < +\infty. \end{cases}$$
$$\dot{\xi}(t) = \alpha x, \quad \xi(0) = 0.$$

The corresponding flow is

$$f(t) = \Omega^{-1}u(t) = \begin{cases} -\gamma\Omega^{-1}B^\top x + \Omega^{-1}\Phi_p(-\gamma B^\top \xi), & \text{if } 1 < p \le 2, \\ \Omega^{-1}\Phi_p(-\gamma B^\top (x + \xi)), & \text{if } 2 \le p < +\infty. \end{cases}$$

### 16.3.1 Optimal solutions by linear-quadratic programming

As already explained, to find the optimal control $u_p^*$ minimizing the weighted $p$-norm of the flow $f(t)$ under Eq. (16.11), the equivalent problem of minimizing the $p$-norm of the control $u(t) = \Omega f(t)$ under Eq. (14.8) can be considered instead, considering matrix $B = C\Omega^{-1}$.

Then, the optimal control vector $u_p^*$ can be found as in Section 15.4, even if there is an unknown dynamic, see subsection 16.1.2, or buffer level control, see subsection 16.2.1. The only difference is that the transformation $B \to C\Omega^{-1}$ must be applied.

The corresponding optimal flow distribution $f_p^*$ is simply $f_p^* = \Omega^{-1}u_p^*$.

## 16.4 Numerical issues

Despite the controls Eq. (15.4) and Eq. (16.1) are theoretically continuous for any $1 < p < +\infty$, in practice, when implementing them as numerical controls, due to numerical errors and finite precision, the values of $p$ cannot be chosen arbitrarily large or close to 1.

Indeed, to simulate the system or implement the control, it must be discretized. A well-known procedure for numerical integration of ordinary differential equations is the *Euler method*. Consider discrete time instants $t_0 = 0, t_1, \dots, t_k, t_{k+1}, \dots$, and, for generic variable $\sigma(t)$, denote

$$\sigma(k) := \sigma(t_k).$$

Let $\delta \in \mathbb{R}^+$ be a given sampling period such that $t_{k+1} - t_k = \delta$. The state equations Eq. (14.8) and Eq. (14.10) become

$$x(k+1) = x(k) + \delta(Bu(k) - d),$$

and

$$x(k+1) = x(k) + \delta(A(x(k)) + Bu(k) - d),$$

respectively. The appropriate control to be used, either Eq. (15.4) or Eq. (16.1), must be discretized, as well, becoming, respectively,

$$u(k) = \Phi_p(-\gamma B^\top x(k)),$$

and

$$u(k) = \begin{cases} -\gamma B^\top x(k) + \Phi_p(-\gamma B^\top \xi(k)), & \text{if } 1 < p \leq 2, \\ \Phi_p(-\gamma B^\top (x(k) + \xi(k))), & \text{if } 2 \leq p < +\infty, \end{cases}$$
$$\xi(k+1) = \xi(k) + \delta\alpha x(k), \quad \xi(0) = 0.$$

Also, to perform computations in a finite-precision machine, variables have to be quantized.

Two problems arise:

- the first one is due to how numbers are represented in computers (*roundoff errors*);
- the second one is due to the approximations needed to perform math operations (like derivatives) in computers, in particular, because $\delta$ cannot be chosen arbitrarily close to 0 (*truncation errors*).

Then, solutions arbitrarily close to the optimum cannot be achieved: the range of the values of $p$ that can be successfully used is limited.

### 16.4.1 Roundoff errors

Consider the first problem and let $\delta \to 0$, i.e., neglect truncation errors. Then, assume that the control is implemented using double-precision data type according to the *IEEE Standard 754 for double precision*, using 64 bits; for instance, this is the default for Matlab variables.

For simplicity, take the function $\Phi_p(\xi)$, defined in Eq. (15.3), for $\xi \geq 0$, which becomes

$$\Phi_p(\xi) = \xi^{\frac{1}{p-1}}.$$

Let $realmax = (2 - 2^{-52}) \cdot 2^{1023} \approx 1.7977 \cdot 10^{+308}$ be the largest finite floating-point number. Whenever $\Phi_p(\xi) = \xi^{\frac{1}{p-1}} > realmax$, $\Phi_p(\xi)$ is treated as $+\infty$ in the computations. Infinities propagate through computations; then, both the control and the state become $+\infty$ and diverge. This occurs when

$$\xi > (realmax)^{p-1}.$$

A similar phenomenon is the following. Let $\epsilon_0 = 4.9407 \cdot 10^{-324}$ be the distance from 0 to the next larger double-precision number. Whenever $\Phi_p(\xi) = \xi^{\frac{1}{p-1}} < \epsilon_0$, $\Phi_p(\xi)$ is treated as 0 in the computations. This occurs when

$$\xi < (\epsilon_0)^{p-1}.$$

As $p \to 1$, function $\Phi_p$ to be used in the computations becomes more and more similar to $\Phi_1$. The interval of $\xi$ for which $\Phi_p(\xi) \neq \Phi_1(\xi)$ increases as $p$ increases.

Then, to implement the control to avoid making it (and hence the state) diverge or make it discontinuous, values of $p$ not too close to 1 and proper values of $\gamma$ must be used, so that

$$(\epsilon_0)^{p-1} < |\gamma B^\top x| < (realmax)^{p-1}$$

(if control Eq. (15.4) is to be applied) is not too tight.

Moreover, consider $\Phi_p(\xi)$ for $\xi$ close to 0. In particular, take $\xi = \epsilon_0$, i.e., the smallest positive floating-point value for $\xi$; the corresponding value assumed by $\Phi_p(\xi)$ is

$$\Phi_p(\epsilon_0) = \epsilon_0^{\frac{1}{p-1}}.$$

As $\Phi_p(0) = 0$, around $\xi = 0$ there is a discontinuity of size $\epsilon_0^{\frac{1}{p-1}}$, whose gap increases as $p$ increases. Then, as $p \to +\infty$, the function becomes more and more similar to $\Phi_1(\xi)$.

Then, to implement the control to avoid making it (and hence the state) discontinuous, values of $p$ not too large and proper values of $\gamma$ must be used, so that the resulting gap is not too large, and can therefore be considered negligible.

**Example 16.3:**
Assume that the control is implemented using double-precision data type according to the *IEEE Standard 754 for double precision*, using 64 bits.
In Table 16.1, the range of values of $\xi$ for which $\Phi_p(\xi) \neq \Phi_0(\xi)$ is reported for some values of $p$. Outside this

interval, $\Phi_p(\xi) \equiv \Phi_0(\xi)$, i.e., it is either $\Phi_p(\xi) \equiv 0$ or $\Phi_p(\xi) \equiv +\infty$.

Table 16.1: Range of values of $\xi$ for which $\Phi_p(\xi) \neq \Phi_1(\xi)$ when variables are represented according to the *IEEE Standard 754 for double precision.* Outside this range, $\Phi_p(\xi)$ is either 0 or $+\infty$.

| $p$ | range |
|---|---|
| 1.0001 | $0.9283 < \lvert\xi\rvert < 1.0736$ |
| 1.001 | $0.4750 < \lvert\xi\rvert < 2.0335$ |
| 1.01 | $5.8471 \cdot 10^{-4} < \lvert\xi\rvert < 1.2093 \cdot 10^{3}$ |
| 1.1 | $4.6707 \cdot 10^{-33} < \lvert\xi\rvert < 6.6907 \cdot 10^{30}$ |

In Table 16.2, the gap of the discontinuity of $\Phi_p(\xi)$ at $\xi = 0$, i.e., $\Phi_p(\epsilon_0) - \Phi_p(0) = \Phi_p(\epsilon_0)$, where $\epsilon_0$ is the minimum non-zero value for $\xi$, is reported for some values of $p$.

Table 16.2: The gap of the discontinuity of $\Phi_p(\xi)$ at $\xi = 0$, i.e., $\Phi_p(\epsilon_0) - \Phi_p(0) = \Phi_p(\epsilon_0)$, where $\epsilon_0$ is the minimum non-zero value for $\xi$, when variables are represented according to the *IEEE Standard 754 for double precision.*

| $p$ | gap |
|---|---|
| 10 | $1.1942 \cdot 10^{-36}$ |
| 100 | $5.4235 \cdot 10^{-4}$ |
| 1000 | 0.4746 |
| 10000 | 0.9283 |

The range of values of $p$ that can be used to get fair and sparse solutions, which are obtained for $p \to \infty$ and $p \to 1$, respectively, increases as the number of bits representing variables increases. Still, as shown in the simulations in Chapter 17, a solution very close to the optimum is eventually achieved, even for non-extreme values of $p$, e.g., $p = 1.1$ and $p = 9$, respectively; if these are implemented using double precision with 64 bits, the above-mentioned problems can be certainly neglected, see Example 16.3.

### 16.4.2 Truncation errors

Consider now the second problem, which results in truncation errors. The cause of this is that the derivatives $\dot{x}(t)$ and possibly $\dot{\xi}(t)$ must be discretized to be implemented.

As already mentioned, the simplest way to do this is approximating the derivative through the (forward) Euler method as

$$\dot{x}(t) \approx \frac{x(k) - x(k-1)}{\delta},$$

where $\delta$ is the sampling period, and similarly for $\xi(k)$.

As $\delta$ increases, the approximation error for the derivative increases, and this produces chattering or diverging when $p \to 1$ or $p \to \infty$. Indeed, this error produces larger variations on the argument of function $\Phi_p(\xi)$ to be used in the control. Recalling that as $p \to 1$, $\Phi_p(\xi)$ becomes steeper around $\xi = 1$, and that as $p \to \infty$, $\Phi_p(\xi)$ becomes steeper around $\xi = 0$, large variations on the control may result in even larger variations on the flow and numerical instability. This occurs even for values of $p$ not too close to 1 or not too large, see Fig. 15.1.

A trivial way to reduce these errors is by reducing the sampling period $\delta$. However, adopting a very small sampling period might slow down the computations. Also, due to limited precision, a very small $\delta$ might also worsen the numerical errors.

Another possible solution to reduce the chattering when $p \to \infty$ and the diverging of the control when $p \to 1$, is applying a low-pass (LP) filter with time constant $\tau_f > 0$ to the control $u(k)$, like

$$u(k) = u(k-1) + \frac{\delta}{\tau_f}(u(k) - u(k-1)).$$

As $u(0) = 0$, this prevents the control to diverge when $p \to 1$, and the on-off behavior due to chattering is smoothed.

While this proposed solution does not solve all the above-mentioned problems, the range of values of $p$ that can be successfully used is generally larger compared to the case in which this filter is not present, see Chapter 17.

# CHAPTER 17

# Illustrative example in a system of interconnected tanks

In this Chapter an example of application of the proposed control is presented for a simple fluid network representing a system of interconnected tanks.

Several simulations are presented, showing the effectiveness of the proposed control both in stabilizing the network and minimizing the $p$-norm of the control network, even in the presence of an unknown dynamic depending on the buffer levels, buffer level control, and weighted $p$-norm minimization.

These proposed controls have been implemented Matlab. All the simulations were performed on a dual-core Intel Core i3 at 2.3 GHz with 8 GB of RAM.

## 17.1 Scenario and data

Consider the fluid network represented in Fig. 17.1, where there are $n = 9$ nodes representing some tanks, whose levels are $h \in \mathbb{R}^9$ and whose states are $x \in \mathbb{R}^9$, and $m = 19$ controllable arcs connecting the tanks, associated with the controls $u \in \mathbb{R}^{19}$ and the controllable flows $f \in \mathbb{R}^{19}$.

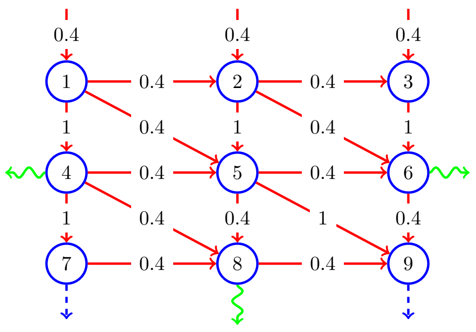


Figure 17.1: Fluid network: controlled arcs (red arrows), with weight $\omega_k$ for each controlled arc $k$; losses (green arrows); demands (blue arrows). (*source:* [3], © 2022 IEEE)

The $9 \times 19$ incidence matrix $B_I$ of the network, describing the controllable arcs, is

$$B_I = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Moreover, a weight is associated with each arc (see Fig. 17.1). Then, given the vector of the arc weights,

$$\omega^\top = \begin{bmatrix} 1 & 0.4 & 0.4 & 1 & 0.4 & 0.4 & 1 & 0.4 & 0.4 & 1 & 0.4 & 0.4 & 0.4 & 1 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 \end{bmatrix},$$

recalling that $\Omega = \mathrm{diag}(\omega)$, the corresponding weighted incidence matrix is

$$B_\omega = B_I \Omega^{-1} = \begin{bmatrix} -1 & -\nu & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\nu & 0 & 0 & 0 & \nu & 0 & 0 \\ 0 & \nu & -\nu & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -\nu & 0 & 0 & \nu & 0 \\ 0 & 0 & \nu & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \nu \\ 1 & 0 & 0 & -1 & -\nu & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \nu & -\nu & 0 & 0 & -\nu & 1 & 0 & 0 & \nu & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \nu & 1 & -\nu & 0 & 0 & 0 & 0 & 0 & 0 & \nu & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -\nu & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \nu & 0 & \nu & -\nu & 0 & 0 & 0 & \nu & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \nu & 0 & 0 & 0 & \nu & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

where $\nu = 1/0.4 = 2.5$.

There is fixed uncontrolled demand from nodes 7 and 9; the demand vector is

$$d^\top = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0 & 0.3 \end{bmatrix}.$$

Also, there might be some uncontrolled losses from nodes 4, 6 and 8, which depend on the corresponding buffer levels. In particular, these unknown dynamics are modeled *for numerical purposes* by function

$$A(h) = b - \sqrt{b^2 + Hh},$$

where $H \in \mathbb{R}^{n \times n}$ is defined as

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.002 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

and $b \in \mathbb{R}^n$ as

$$b^\top = \begin{bmatrix} 0 & 0 & 0 & 0.002 & 0 & 0.003 & 0 & 0.003 & 0 \end{bmatrix}.$$

For some of the performed simulations, no unknown dynamic is assumed; hence, $A(h) \equiv 0$.

Moreover, for some simulations, buffer level control to the non-zero set point $\bar{h} \in \mathbb{R}^n$ is imposed, where

$$\bar{h}^\top = \begin{bmatrix} 17.69 & 20.37 & 22.70 & 16.59 & 22.42 & 17.93 & 19.54 & 20.68 & 15.66 \end{bmatrix}.$$

The problem of minimizing the weighted $p$-norm of the flow $||\Omega f(t)||_p$ is considered; indeed, recall that this requires only the scaling of the control components as

$$u(t) = \Omega f(t),$$

so that $||u(t)||_p \equiv ||\Omega f(t)||_p$ is equivalently to be minimized.

Finally, letting $B = B_\omega$, the dynamics of the buffer levels are given by

$$\dot{h}(t) = A(h) + B_I f(t) - d = A(h) + Bu(t) - d,$$

with initial conditions $h(0) = h_0$ given by

$$h_0^\top = \begin{bmatrix} 15.51 & 18.41 & 19.01 & 18.80 & 17.34 & 18.36 & 19.63 & 18.12 & 19.77 \end{bmatrix}.$$

Then, recalling that $x = h - \bar{h}$ when $A(h) \equiv 0$ does not hold, the system evolves according to (see Remark 16.3)

$$\dot{x}(t) = A(x + \bar{h}) + Bu(t) - d.$$

This system is studied in three different conditions:

- *case (A)*, in which there is no unknown dynamic (i.e., $A(h) \equiv 0$) and no buffer level control is imposed. In this case, the state is defined as $x(t) = h(t)$. The original control Eq. (15.4) is applied.
- *case (B)*, in which there is no unknown dynamic (i.e., $A(h) \equiv 0$), but buffer level control to $\bar{h}$ is imposed. In this case, the state is defined as $x(t) = h(t) - \bar{h}$. The enhanced control Eq. (16.1) is applied.
- *case (C)*, in which there is the non-zero unknown dynamic $A(h)$ and buffer level control to $\bar{h}$ is also imposed. In this case, the state is defined as $x(t) = h(t) - \bar{h}$. The enhanced control Eq. (16.1) is applied. Note that, at steady-state, $x \to 0$: the losses $A(\bar{h})$ are non-zero and must be compensated by the control. Still, recall that the control is independent from $A(h)$.

### 17.1.1 Simulations' setup

Three different aspects are tested, applying the proposed control to the network described above.

For evaluation purposes, the optimal controls $u_1^*, u_2^*$ and $u_\infty^*$ can be computed offline, using the linear-quadratic problems formulated in Section 15.4 and subsections 16.1.2, 16.2.1 and 16.3.1.

#### Test 1: analysis of the system evolution

Firstly, the evolution of the system over time is analyzed. A scenario in which the $p$-norm to be minimized varies over time is considered.

In particular, three consecutive time intervals are considered:

- for $0 \leq t < 600$, the 2-norm is to be minimized. Hence, the proper control is applied for $p = 2$, which guarantees that the unique optimal solution is eventually reached;
- for $600 \leq t < 1200$, the $\infty$-norm is to be minimized. As the control with $p = \infty$ does not ensure optimality, the proper control is applied for $p = 9$, getting a suboptimal solution whose norm is very close to the optimal one;
- for $1200 \leq t < 1800$, the 1-norm is to be minimized. Again, the control with $p = 1$ does not ensure optimality. Then, the proper control is applied for $p = 1.1$, getting a suboptimal solution whose norm is very close to the optimal one.

All the three cases (A), (B) and (C) are studied.

The following parameters are adopted: sampling time $\delta = 0.0001, \alpha = 0.05, \gamma = 0.03$ (when $p = 2$), $\gamma = 0.000001$ (for $p = 9$), $\gamma = 0.06$ (for $p = 1.1$).

#### Test 2: the problem of the chattering

Then, the problem of the chattering of the control due to numerical errors when using $p \to \infty$ is analyzed. Only the specific case (C), in which there is the unknown dynamic $A(h)$ and buffer level control to $\bar{h}$ is also imposed, is considered, where control Eq. (16.1) is applied. First, the system is simulated four times over a horizon of length 700, each one using a different value of $p$, specifically $p = 9, 10, 11, 12$. Severe chattering emerges as $p$ increases.

To mitigate this, the same simulations are repeated applying a real-time low-pass filter with time constant $\tau_f$ to the control. Recall that at steady-state, the low pass filter has no effects.

The following parameters are adopted: $\delta = 0.001, \tau_f = 0.01, \alpha = 0.05, \gamma = 000001$.

#### Test 3: the effects of the choice of $p$

Finally, the effects of the choice of the value of $p$ on the steady-state solution are analyzed. Again, only the specific case (C), in which there is the unknown dynamic $A(h)$ and buffer level control to $\bar{h}$ is also imposed, is considered, where control Eq. (16.1) is applied. To avoid chattering issues, the control is filtered in real-time with a low-pass filter with time constant $\tau_f$. Several simulations are performed: each one considering a specific value of $p$:

$$p \in \{1.01, 1.02, \ldots, 1.28, 1.29, 1.3, 1.4, 1.5, \ldots, 3.8, 3.9, 4, 5, 6, 7, 8, 9, 10\}.$$

For each value of $p$, the system is simulated long enough to reach the steady-state. Then, the components of $u^*(t)$ and some of its norms are compared for the range of considered values of $p$.

The following parameters are adopted: $\delta = 0.001, \tau_f = 0.01, \alpha = 0.05, \gamma = 0.06$ (for $p < 1.1$), $\gamma = 0.03$ (for $1.1 \leq p \leq 7$), $\gamma = 0.0000001$ (for $p > 7$). However, recall that the steady-state does not depend on the specific parameters $\gamma, \alpha$ which are used in the control. Also, recall that the final flow vector $u^*(t)$ that is achieved is unique for $1 < p < \infty$.

## 17.2 Results and discussion

For *Test 1*, the results are reported in Figs. 17.2 to 17.4, for the three cases (A), (B) and (C), respectively, which show the evolution of the components of the vectors related to the nodes (subfigures on the left) and to the arcs (subfigures on the right). In particular, the evolution of the components of the following quantities is shown: the buffer level vector $h(t)$, the corresponding state vector $x(t)$, the corresponding integral variable vector $\xi(t)$ (if control Eq. (16.1) is applied), the control vector $u(t)$; the corresponding actual flow vector $f(t) = \Omega^{-1}u(t)$. Moreover, also the time evolution of the norms $\|u(t)\|_1, \|u(t)\|_2, \|u(t)\|_{+\infty}, \|u(t)\|_{1.1}, \|u(t)\|_9$, is reported, as well as the optimal $\|u_1^*\|_1, \|u_2^*\|_2, \|u_\infty^*\|_{+\infty}$. In such graphs (except the ones reporting the norms), each color refers to a specific node or arc; the same color in different graphs refers to the same node or arc. Numerical values for some quantities reached at steady-state are reported in Tables 17.1 to 17.3, for the three cases (A), (B) and (C).

For the variables related to the nodes, in case (A), where $x(t) \equiv h(t)$, the buffer level/state components reach some negative values $\bar{h}$ and stabilize; recall that in this case no set-point was imposed for the buffer levels. Instead, in both cases (B) and (C), the buffer levels reach the desired levels given by $\bar{h}$. Consequently, now that the state is defined as $x(t) = h(t) - \bar{h}$, the state $x(t)$ converges to zero in all the three intervals.

Moreover, the variables $\xi(t)$, which have been introduced for control Eq. (16.1) and are proportional to the integral of $x(t)$, converge to some negative values $\bar{\xi}$, too. Comparing the numeric values of the components of $\bar{\xi}$ from case (B) with the corresponding components $\bar{x}$ from case (A), the two vectors match. This confirms the findings from Proposition 15.1 and Theorem 16.1; note that these correspond to the optimal Lagrange multipliers of the optimization problem considered in case (A) and (B), which depend on the $p$-norm to be minimized but is independent from the fact that a set point is imposed. Instead, comparing the numeric values of the components of $\bar{\xi}$ from case (B) with those from case (C), slightly different values are achieved, because the corresponding optimal problems of which they are Lagrange multipliers are different (due to the presence of the losses at steady-state due to the unknown dynamics $A(\bar{h})$ in case (C)).

For the variables related to the arcs, in each interval, for all the three cases (A), (B) and (C), the control converges to some $u_p^*$ depending on the value of $p$. In particular, in the first interval, the 2-norm optimal control $u_2^*$ is reached. In the second interval, the control vector $\bar{u}_p^*$ converges to a control that is slightly different from the optimal $u_\infty^*$, because the optimal $\infty$-norm distribution is not unique, and the value of $p$ is not very large. In the third interval, the steady-state solution is very close to the optimal 1-norm one $u_1^*$. The same applies to the corresponding flows, which are slightly different from the corresponding controls, as they are weighted by the inverse of the arcs' costs. Note that some chattering emerges when using $p = 9$ (see orange lines, in the second interval), but this is just temporary.

Comparing the numeric values of the components of $u_p^*$ from cases (A) and (B), the values are the same at steady-state, since the optimal flow does not depend on the fact that a set point is set. The corresponding components from case (C) are slightly different, because the steady-state losses $A(\bar{h})$ are to be taken into account.

In any case, from Tables 17.1 to 17.3, it can also be seen that, for any $p$, at steady-state the total controlled inflow matches the total uncontrolled outflow, taking into account both the demand and the losses modeled by the nonlinear dynamics, if present, that is:

$$\frac{u_{p,17}^*}{\omega_{17}} + \frac{u_{p,18}^*}{\omega_{18}} + \frac{u_{p,19}^*}{\omega_{19}} = \sum_k \left[d - A(\bar{h})\right]_k .$$

Regarding the corresponding norms, in each interval, the 1, 2 and $\infty$-norms of the control vector, which are equal to the corresponding weighted norms of the flow vector, approach the required optimal ones: in particular, in the first interval, at steady-state, the optimal 2-norm is reached, i.e., $\|u(t)\|_2 \rightarrow \|u_2^*\|_2$, in all the three cases (A), (B) and (C), confirming the optimality of the control. In the second and third intervals the $\infty$ and 1-norm of the control, respectively, are very close to the optimal ones, i.e., $\|u(t)\|_\infty \rightarrow \|u_\infty^*\|_\infty$ in the second interval, and $\|u(t)\|_2 \rightarrow \|u_1^*\|_1$ in the third interval; indeed, both results were expected to be suboptimal.

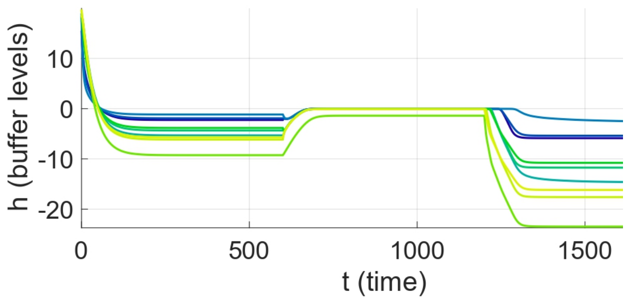
Table 17.1: (*Test 1, case (A)*). The steady-state vectors for the case considering a network with no unknown dynamics ($(A(x(t)) \equiv 0)$) and no buffer level control, minimizing the weighted norm of the flow vector.

(a) The steady-state solution $u_p^*$, the optimal $u_q^*$ and their norms.

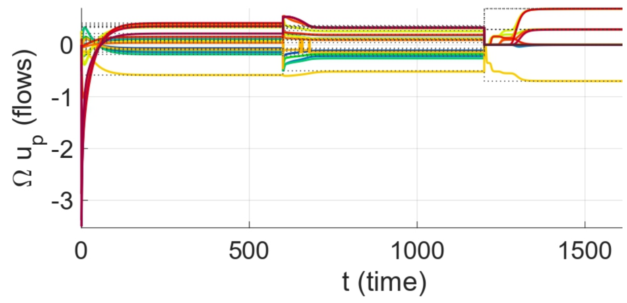
| $p$ | $q$ | $u_p^*$ *(steady-state control)* | $u_q^*$ *(computed via linear/quadratic programming)* | $\lVert u_p^*\rVert_p$ | $\lVert u_p^*\rVert_q$ | $\lVert u_q^*\rVert_q$ |
|---|---|---|---|---|---|---|
| 2 | 2 | $[0.092, -0.025, -0.056, 0.118, -0.073, -0.038, 0.080, 0.139, 0.136, 0.073, -0.233, -0.035, 0.157, 0.040, 0.145, 0.063, 0.169, 0.144, 0.088]^\top$ | $[0.092, -0.025, -0.056, 0.118, -0.073, -0.038, 0.080, 0.139, 0.136, 0.073, -0.233, -0.035, 0.157, 0.040, 0.145, 0.063, 0.169, 0.144, 0.088]^\top$ | 0.496 | 0.496 | 0.496 |
| 9 | $\infty$ | $[0.118, -0.044, -0.086, 0.184, -0.104, -0.071, 0.116, 0.106, 0.105, 0.116, -0.206, -0.054, 0.130, 0.094, 0.130, 0.078, 0.134, 0.134, 0.133]^\top$ | $[0.104, -0.036, -0.072, 0.200, -0.085, -0.050, 0.097, 0.128, 0.127, 0.091, -0.200, -0.026, 0.145, 0.045, 0.139, 0.047, 0.150, 0.139, 0.111]^\top$ | 0.216 | 0.207 | 0.200 |
| 1.1 | 1 | $[0.002, -0.000, -0.000, 0.002, -0.000, -0.000, 0.001, 0.120, 0.279, 0.000, -0.279, -0.000, 0.279, 0.000, 0.120, 0.000, 0.280, 0.120, 0.000]^\top$ | $[-0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.120, 0.280, 0.000, -0.280, 0.000, 0.280, 0.000, 0.120, 0.000, 0.280, 0.120, -0.000]^\top$ | 1.249 | 1.482 | 1.480 |

(b) The steady-state vector $\bar{x}$ and integral vector $\bar{\xi}$.

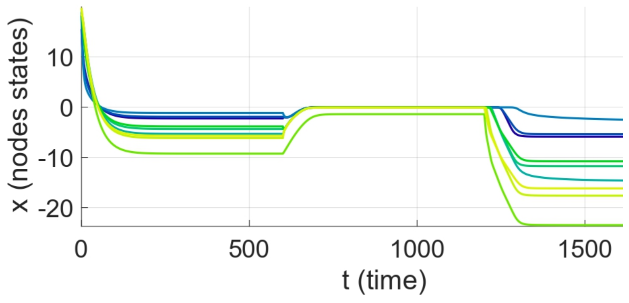
| $p$ | $q$ | $\bar{x}$ | $\bar{\xi}$ |
|---|---|---|---|
| 2 | 2 | $[-2.251, -1.913, -1.170, -5.320, -4.348, -3.841, -9.257, -6.154, -5.690]^\top$ | *n/a* |
| 9 | $\infty$ | $[-0.040, -0.040, -0.039, -0.079, -0.073, -0.073, -1.395, -0.079, -0.079]^\top$ | *n/a* |
| 1.1 | 1 | $[-5.870, -5.391, -2.682, -14.656, -11.738, -10.783, -23.474, -17.606, -16.176]^\top$ | *n/a* |



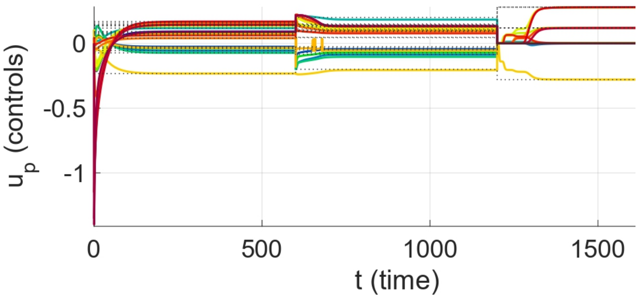
(a) The components of the buffer level vector $h(t)$ over time. Each color refer to a specific node.



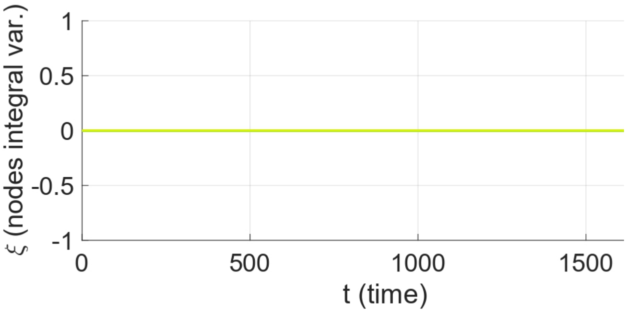
(b) The components of the flow vector $f(t) = \Omega^{-1}u(t)$ over time. Each color refer to a specific arc. Grey dotted lines: optimal values.



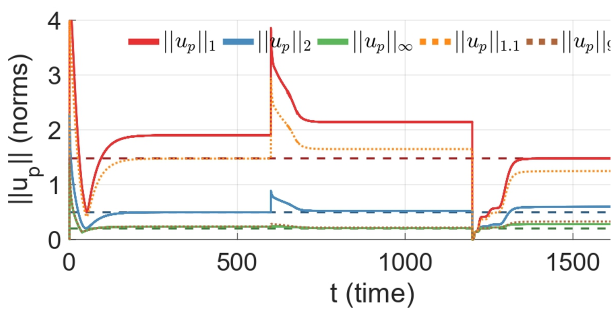
(c) The components of the state vector $x(t)$ over time. Each color refer to a specific node.



(d) The components of the control vector $u(t)$ over time. Each color refer to a specific arc. Grey dotted lines: optimal values.



(e) The components of the integral vector $\xi(t)$ over time. Not applicable for this case ($\alpha = 0$).



(f) Solid lines represent the norms $\|u_p\|1$ (red), $\|u_p\|2$ (blue) and $\|u_p\|_\infty$ (green), which respectively get close to the optimal $\|u_1^*\|_1$, $\|u_2^*\|_2$ and$\|u_\infty^*\|_\infty$ (dashed lines) in the first ($p = 1.1$), second ($p = 2$), and third ($p = 9$) intervals.

Figure 17.2: (*Test 1, case (A)*). Time evolution profiles for the case considering a network with no unknown dynamics ($(A(x(t)) \equiv 0)$) and no buffer level control, minimizing the weighted norm of the flow vector.

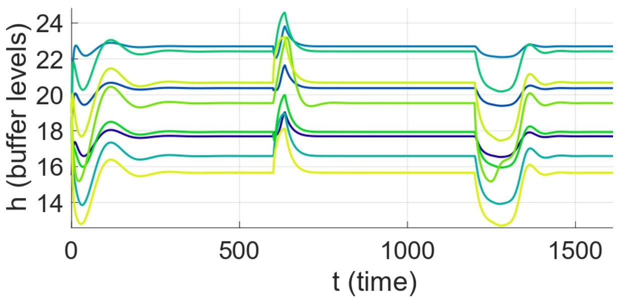
Table 17.2: (*Test 1, case (B)*). The steady-state vectors for the case considering a network with no unknown dynamics ($A(x(t)) \equiv 0$), but with buffer level control to $\bar{h}$, minimizing the weighted norm of the flow vector.

(a) The steady-state solution $u_p^*$, the optimal $u_q^*$ and their norms.

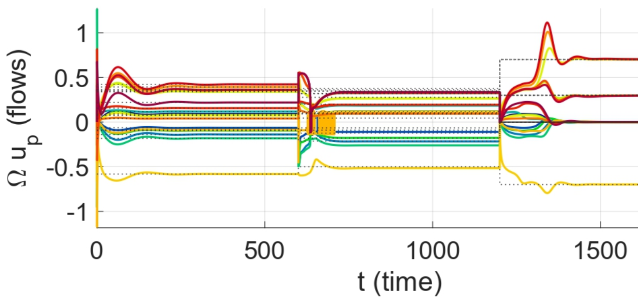
| $p$ | $q$ | $u_p^*$ *(steady-state control)* | $u_q^*$ *(computed via linear/quadratic programming)* | $\lVert u_p^*\rVert_p$ | $\lVert u_p^*\rVert_q$ | $\lVert u_q^*\rVert_q$ |
|---|---|---|---|---|---|---|
| 2 | 2 | $[0.092, -0.025, -0.056, 0.118, -0.073, -0.038, 0.080, 0.139, 0.136, 0.073, -0.233, -0.035, 0.157, 0.040, 0.145, 0.063, 0.169, 0.144, 0.088]^\top$ | $[0.092, -0.025, -0.056, 0.118, -0.073, -0.038, 0.080, 0.139, 0.136, 0.073, -0.233, -0.035, 0.157, 0.040, 0.145, 0.063, 0.169, 0.144, 0.088]^\top$ | 0.496 | 0.496 | 0.496 |
| 9 | $\infty$ | $[0.118, -0.044, -0.086, 0.184, -0.104, -0.071, 0.116, 0.106, 0.105, 0.116, -0.206, 0.042, 0.130, 0.094, 0.130, 0.078, 0.134, 0.134, 0.133]^\top$ | $[0.104, -0.036, -0.072, 0.200, -0.085, -0.050, 0.097, 0.128, 0.127, 0.091, -0.200, -0.026, 0.145, 0.045, 0.139, 0.047, 0.150, 0.139, 0.111]^\top$ | 0.216 | 0.206 | 0.200 |
| 1.1 | 1 | $[0.002, 0.000, -0.000, 0.002, -0.000, 0.000, 0.001, 0.120, 0.279, 0.000, -0.279, 0.000, 0.280, 0.000, 0.120, 0.000, 0.280, 0.119, 0.001]^\top$ | $[-0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.120, 0.280, 0.000, -0.280, 0.000, 0.280, 0.000, 0.120, 0.000, 0.280, 0.120, -0.000]^\top$ | 1.249 | 1.483 | 1.480 |

(b) The steady-state vector $\bar{x}$ and integral vector $\bar{\xi}$.

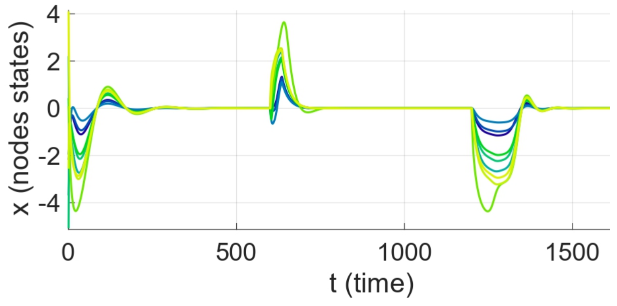
| $p$ | $q$ | $\bar{x}$ | $\bar{\xi}$ |
|---|---|---|---|
| 2 | 2 | $[-0.000, -0.000, -0.000, -0.000, -0.000, -0.000, -0.000, -0.000, -0.000]^\top$ | $[-2.251, -1.913, -1.170, -5.321, -4.348, -3.841, -9.258, -6.154, -5.690]^\top$ |
| 9 | $\infty$ | $[0.000, 0.000, 0.000, 0.000, 0.000, -0.000, 0.000, -0.000, 0.000]^\top$ | $[-0.040, -0.040, -0.039, -0.079, -0.073, -0.073, -1.395, -0.079, -0.079]^\top$ |
| 1.1 | 1 | $[0.000, -0.000, 0.003, -0.000, 0.000, -0.002, 0.000, 0.000, -0.002]^\top$ | $[-5.871, -5.389, -3.259, -14.607, -11.740, -10.780, -23.476, -17.608, -16.172]^\top$ |



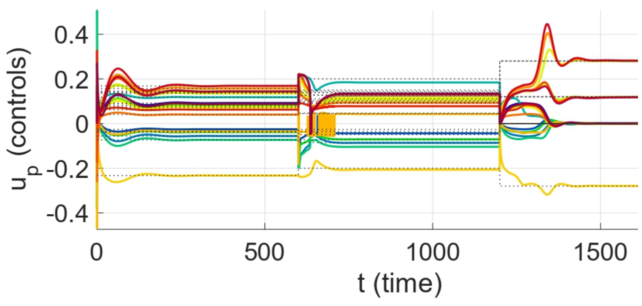
(a) The components of the buffer level vector $h(t)$ over time. Each color refer to a specific node.



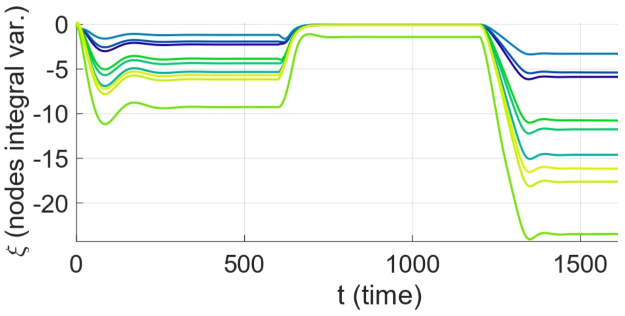
(b) The components of the flow vector $f(t) = \Omega^{-1}u(t)$ over time. Each color refer to a specific arc. Grey dotted lines: optimal values.



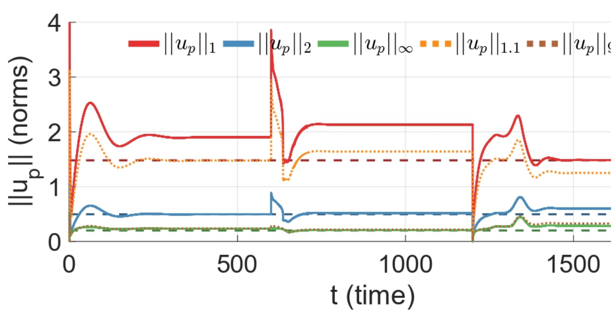
(c) The components of the state vector $x(t)$ over time. Each color refer to a specific node.



(d) The components of the control vector $u(t)$ over time. Each color refer to a specific arc. Grey dotted lines: optimal values.



(e) The components of the integral vector $\xi(t)$ over time. Each color refer to a specific node.



(f) Solid lines represent the norms $\|u_p\|1$ (red), $\|u_p\|2$ (blue) and $\|u_p\|_\infty$ (green), which respectively get close to the optimal $\|u_1^*\|_1$, $\|u_2^*\|_2$ and$\|u_\infty^*\|_\infty$ (dashed lines) in the first ($p = 1.1$), second ($p = 2$), and third ($p = 9$) intervals.

Figure 17.3: (*Test 1, case (B)*). Time evolution profiles for the case considering a network with no unknown dynamics ($A(x(t)) \equiv 0$), but buffer level control to $\bar{h}$, minimizing the weighted norm of the flow vector.

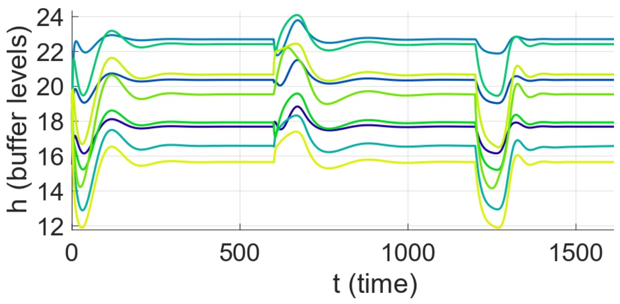
Table 17.3: (*Test 1, case (C)*). The steady-state vectors for the case considering a network with unknown dynamics $A(x(t))$ and buffer level control to $\bar{h}$, minimizing the weighted norm of the flow vector.

(a) The steady-state solution $u_p^*$, the optimal $u_q^*$ and their norms (*source:* [3], © 2022 IEEE).

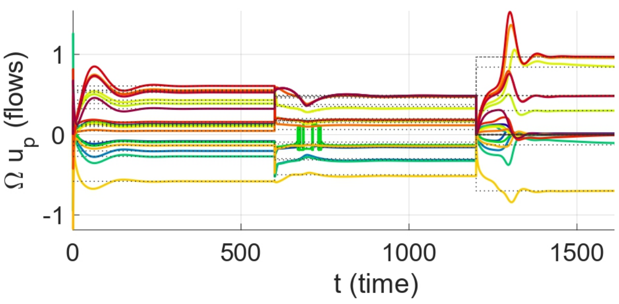
| $p$ | $q$ | $u_p^*$ *(steady-state control)* | $u_q^*$ *(computed via linear/quadratic programming)* | $\|u_p^*\|_p$ | $\|u_p^*\|_q$ | $\|u_q^*\|_q$ |
|---|---|---|---|---|---|---|
| 2 | 2 | $[\ 0.132, -0.033, -0.081, 0.118, -0.109, -0.037, 0.120, 0.155, 0.172, 0.102, -0.233, -0.054, 0.222, 0.047, 0.218, 0.063, 0.243, 0.210, 0.129]^\top$ | $[\ 0.132, -0.033, -0.081, 0.118, -0.109, -0.037, 0.120, 0.155, 0.172, 0.102, -0.233, -0.054, 0.222, 0.047, 0.218, 0.063, 0.243, 0.210, 0.129]^\top$ | 0.641 | 0.641 | 0.641 |
| 9 | $\infty$ | $[\ 0.170, -0.063, -0.126, 0.184, -0.131, -0.052, 0.170, 0.131, 0.131, 0.169, -0.206, -0.058, 0.189, 0.117, 0.189, 0.074, 0.194, 0.194, 0.193]^\top$ | $[\ 0.178, -0.065, -0.126, 0.200, -0.119, -0.037, 0.173, 0.141, 0.150, 0.173, -0.200, -0.046, 0.189, 0.064, 0.183, 0.060, 0.196, 0.191, 0.195]^\top$ | 0.244 | 0.206 | 0.200 |
| 1.1 | 1 | $[\ 0.015, 0.000, -0.001, 0.000, -0.045, 0.000, 0.001, 0.120, 0.337, 0.000, -0.280, 0.000, 0.381, 0.000, 0.194, -0.000, 0.387, 0.193, 0.001]^\top$ | $[\ 0.000, 0.0000, 0.000, -0.000, -0.051, 0.000, 0.000, 0.120, 0.336, 0.000, -0.280, 0.000, 0.387, 0.000, 0.195, 0.000, 0.387, 0.195, 0.000]^\top$ | 1.634 | 1.955 | 1.950 |

(b) The steady-state vector $\bar{x}$ and integral vector $\bar{\xi}$.

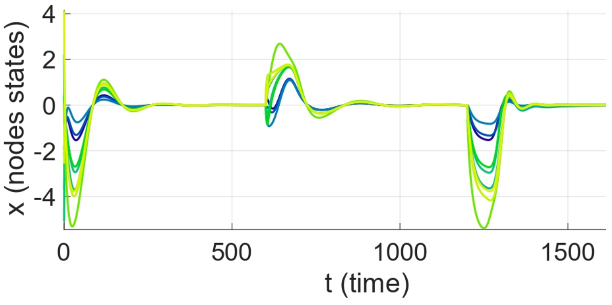
| $p$ | $q$ | $\bar{x}$ | $\bar{\xi}$ |
|---|---|---|---|
| 2 | 2 | $[-0.000, -0.000, -0.000, -0.000, -0.000, -0.000, -0.000, -0.000, -0.000]^\top$ | $[-3.234, -2.801, -1.720, -7.643, -6.196, -5.708, -11.590, -8.488, -7.772]^\top$ |
| 9 | $\infty$ | $[-0.003, -0.003, -0.002, -0.004, -0.004, -0.004, -0.006, -0.004, -0.004]^\top$ | $[-0.799, -0.799, -0.774, -1.485, -1.451, -1.451, -2.797, -1.485, -1.485]^\top$ |
| 1.1 | 1 | $[\ 0.000, -0.000, 0.005, -0.001, 0.000, -0.000, 0.000, 0.000, -0.000]^\top$ | $[-6.063, -5.656, -3.570, -16.999, -12.117, -11.315, -23.966, -18.096, -16.707]^\top$ |



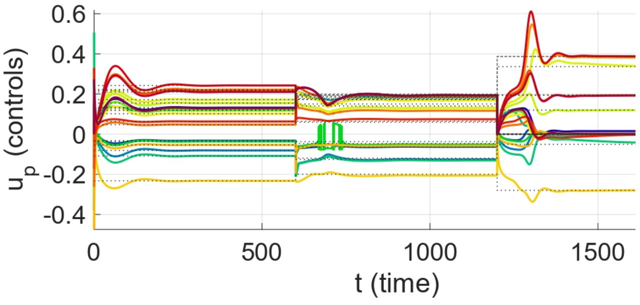
(a) The components of the buffer level vector $h(t)$ over time. Each color refer to a specific node.



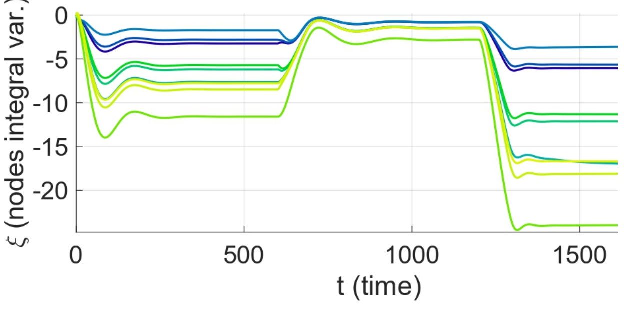
(b) The components of the flow vector $f(t) = \Omega^{-1}u(t)$ over time. Each color refer to a specific arc. Grey dotted lines: optimal values.



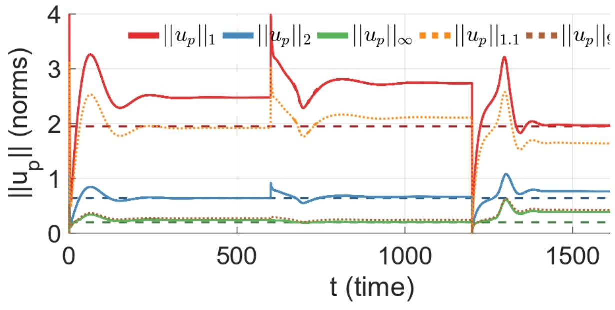
(c) The components of the state vector $x(t)$ over time. Each color refer to a specific node.



(d) The components of the control vector $u(t)$ over time. Each color refer to a specific arc. Grey dotted lines: optimal values.



(e) The components of the integral vector $\xi(t)$ over time. Each color refer to a specific node.



(f) Solid lines represent the norms $\|u_p\|1$ (red), $\|u_p\|2$ (blue) and $\|u_p\|_\infty$ (green), which respectively get close to the optimal $\|u_1^*\|_1$, $\|u_2^*\|_2$ and $\|u_\infty^*\|_\infty$ (dashed lines) in the first ($p = 1.1$), second ($p = 2$), and third ($p = 9$) intervals. (*source:* [3], © 2022 IEEE)

Figure 17.4: (*Test 1, case (C)*). Time evolution profiles for the case considering a network with unknown dynamics $A(x(t))$ and buffer level control to $\bar{h}$, minimizing the weighted norm of the flow vector.

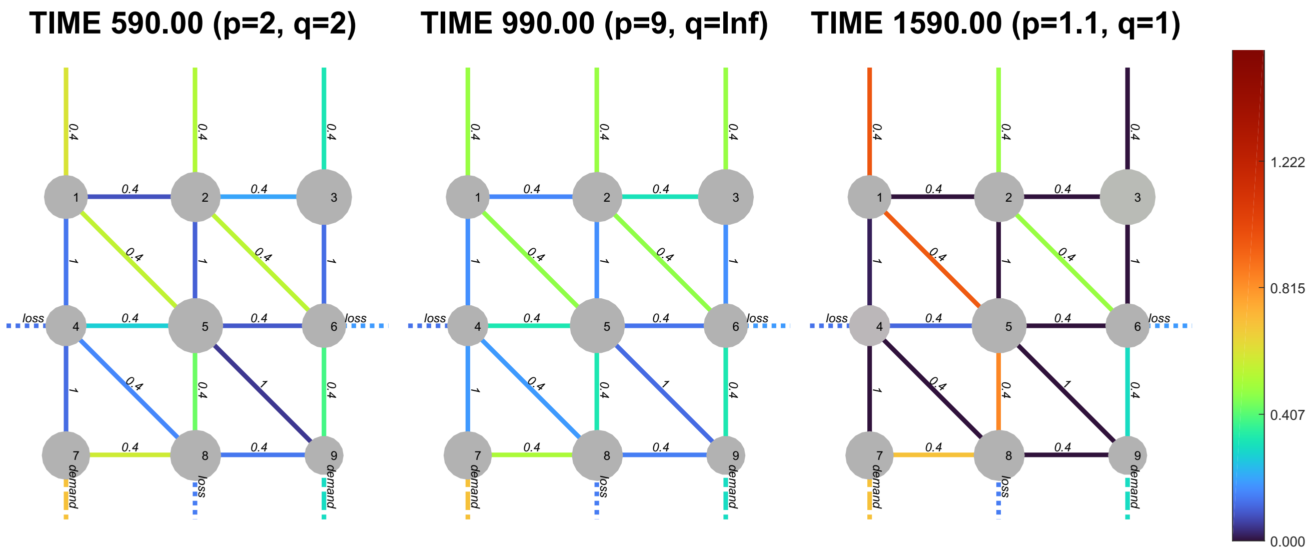


Figure 17.5: (*Test 1, case (C)*). The network at steady-state for each possible value of $p$, trying to minimize the $q$-norm. The color of each arc $h$ represents the modulo of the flow $f_h = u_h/\omega_h$.

An animation of the evolution of the network for case (C) is available at `https://users.dimi.uniud.it/~franco.blanchini/oneinf.html`. A representation of the actual distribution of the flows $\Omega^{-1}u(t)$ in the network at steady-state, for the three time intervals in which $p$ is set, is shown in Fig. 17.5, again for case (C).

It can be seen that when trying to minimize the 2 or $\infty$-norm (using $p = 2$ and $p = 9$, respectively), the flow is distributed in the arcs of the network. In particular, in the latter case, the maximum flow is minimal. Instead, the steady-state distribution in the third interval, when trying to minimize the 1-norm using $p = 1.1$, clearly shows that some shortest paths emerge, making the flow concentrated just in some arcs (the arcs represented in dark blue have no flow). In particular, two main independent paths are formed to meet the demand, whose flows are represented in red and green; note that some ramifications of the flows from such paths are present, to compensate for the steady-state losses.

For *Test 2*, the results are summarized in Fig. 17.6. In particular, on the subplots on the left, the evolution of the control components $u_i(t)$ over time is plotted, when $p = 9, 10, 11, 12$ without filtering the result. Chattering clearly emerges as $p$ increases, even with small values of $p$, e.g., for $p = 10$. A better tuning of the control parameters $\alpha, \gamma$ and a smaller sampling period $\delta$ might reduce these unwanted issues. Still, using a smaller value of $p$ like 9 produces no apparent/negligible chattering.

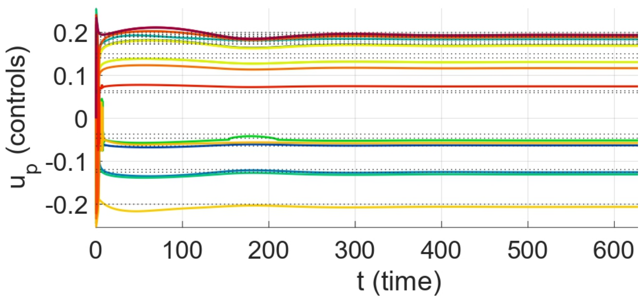
On the subplots on the right, the evolution of the control components over time is plotted, when applying a low-pass filter to the control. This alternative allows using some larger $p$ and is also effective in reducing chattering a lot, although the phenomenon starts to emerge too, as $p$ increases. Again, better tuning of the parameter $\tau_f$ might lead to better results.

For *Test 3*, the results are summarized in Fig. 17.7. In particular, in Fig. 17.7a, the trend of the optimal control components $u_p^*$ are plotted, as a function of $p$. In Fig. 17.7b, the trends of the corresponding $1, 2, \infty, p$-norms are plotted. Finally, in Fig. 17.7c the corresponding trend of the level of sub-optimality is plotted.

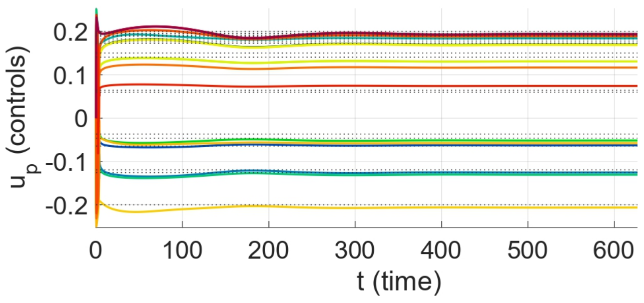
It actually turns out that there is no need to use extreme values of $p$ to minimize the 1 and $\infty$-norms. All the reported quantities vary continuously as $p$ increases. When minimizing the 2-norm, the exact solution is obtained exactly.

When trying to minimize the 1-norm, values of $p$ in the range $[1.01, 1.1]$ produce suboptimal solutions which are very close to the 1-optimal ones, with a level of sub-optimality $\|u_p^*\|_1/\|u_1^*\|_1$ being very close to 1, so there is no need to use smaller values of $p$, which might introduce more numerical errors. Note that the optimal 1.01-norm is also very close to the optimal 1-norm, i.e., $\|u_{1.01}^*\|_{1.01} \approx \|u_1^*\|_1$.

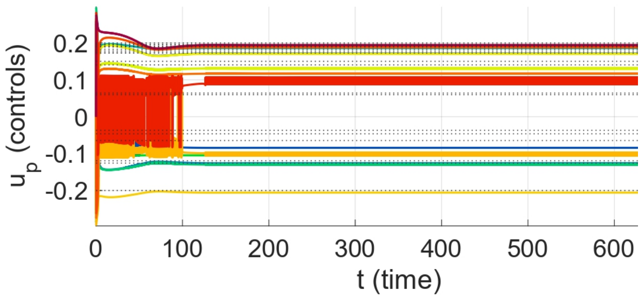
When trying to minimize the $\infty$-norm, values of $p$ greater than 8 produce suboptimal solutions which are close to the $\infty$-optimal ones (recall this is not unique), with a level of sub-optimality $\|u_p^*\|_\infty/\|u_\infty^*\|_\infty$ being below 1.05. Using larger values of $p$ produce results which are closer to the optimum, however, the problem of the chattering is to be taken into account. Note that also the optimal 10-norm is close to the optimal $\infty$-norm, i.e., $\|u_{10}^*\|_{10} \sim \|u_\infty^*\|_\infty$.



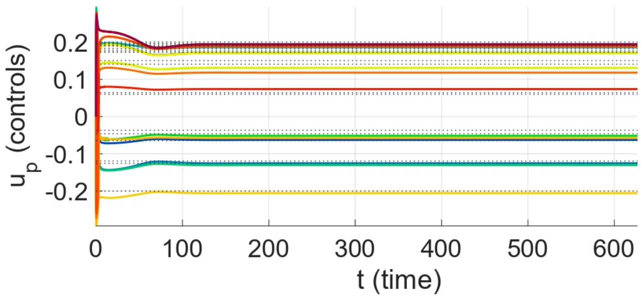
(a) $p = 9$, no low-pass filter applied.
($\|u_9^*\|_9$= 0.2442, $\|u_9^*\|_\infty$= 0.2064)



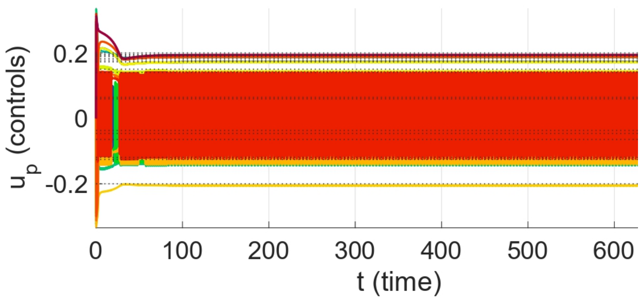
(b) $p = 9$, with low-pass filter applied.
($\|u_9^*\|_9$= 0.2442, $\|u_9^*\|_\infty$= 0.2064)



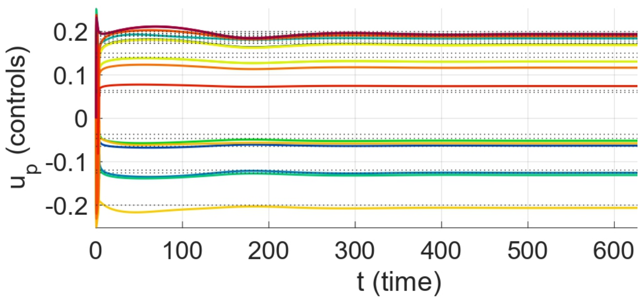
(c) $p = 10$, no low-pass filter applied.



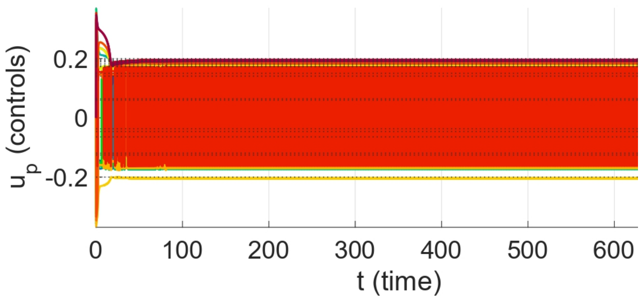
(d) $p = 10$, with low-pass filter applied.
($\|u_{10}^*\|_{10}$= 0.2383, $\|u_{10}^*\|_\infty$= 0.2057)



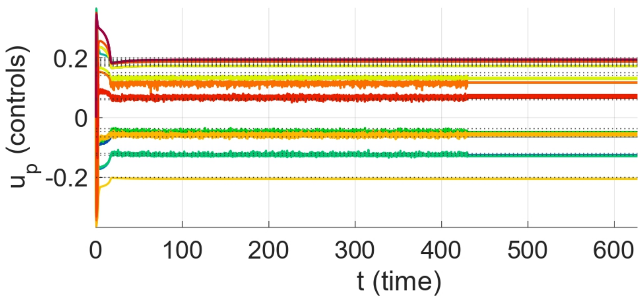
(e) $p = 11$, no low-pass filter applied.



(f) $p = 11$, with low-pass filter applied.
($\|u_{11}^*\|_{11}$= 0.2336, $\|u_{11}^*\|_\infty$= 0.2051)

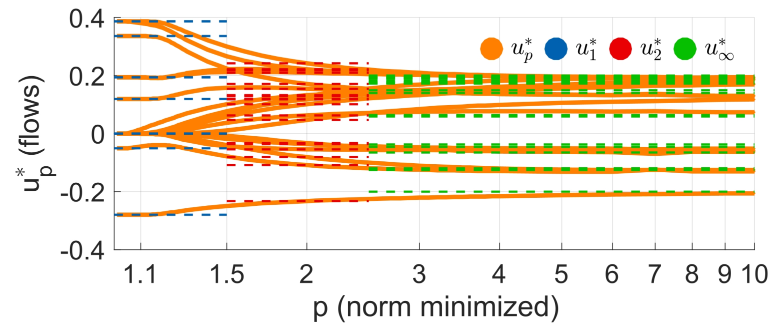


(g) $p = 12$, no low-pass filter applied.

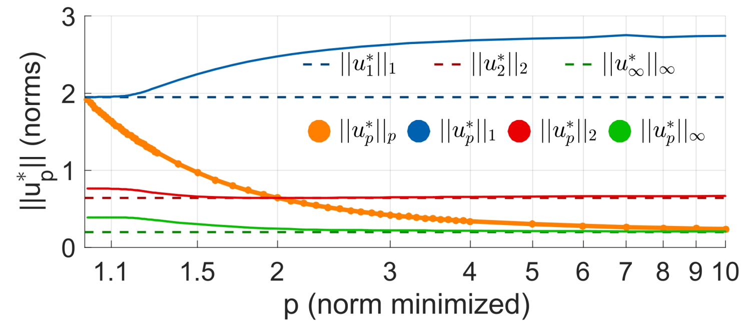


(h) $p = 12$, with low-pass filter applied.
($\|u_{12}^*\|_{12}$= 0.2299, $\|u_{12}^*\|_\infty$= 0.2047)

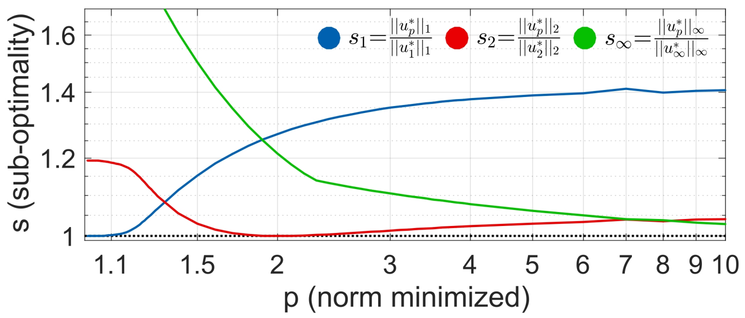
Figure 17.6: (*Test 2*). Evolution over time of the components of the control applied to the arcs. Each color refers to a different arc. Dashed lines are the control components of the optimal $\infty$-norm control computed via linear quadratic programming. For some values of $p$ chattering emerges, but its effects are clearly reduced by applying a low-pass filter.



(a) The components of $u_p^*$ (orange lines). Red, blue and green dashed lines represent the components of the optimal 1,2 and $\infty$ control components, respectively, computed by linear-quadratic programming, and are reported as a reference.



(b) The $p$-norm of $u_p^*$ (orange line). Red, blue and green lines are the corresponding 1,2 and $\infty$ norms of $u_p^*$, respectively. Red, blue and green dashed lines represent the optimal norms $\|u_1^*\|_1$,$\|u_2^*\|_2$,$\|u_\infty^*\|_\infty$, respectively, computed by linear-quadratic programming, and are reported as a reference.



(c) The sub-optimality of the control over time. The suboptimality of the $q$-norm is defined as $s_q = \|u_p^*\|_q/\|u_q^*\|_q$.

Figure 17.7: (*Test 3*). Comparison of the (interpolated) trend of the steady-state (optimal) control $u_p^*$ and its norms, for different values of $p$.

# Conclusion

In this thesis, the main results achieved during my Ph.D. have been presented.

In Part I the problem of scheduling a certain number of requests requiring a given amount of a limited resource, minimizing the average waiting time has been considered. The maximum rate of supply is limited, and if the system is congested, the timing and rate of the supply are to be modified to avoid overloading the system.

Firstly, batch solutions have been taken into account: an optimal control framework has been proposed, which is capable of solving different scheduling problems for waiting time minimization supporting three supply strategies: interruptible, non–interruptible, and variable rate. The support for these supplies is enforced by means of some additional constraints.

The exact optimal problems result in Mixed-Integer Linear Programming problems: then, finding a solution might be hard, depending on the size of the problem. A relaxed version of the problem resulting in a Linear Programming problem, for which a solution can be computed very efficiently even for large instances, has been studied. This can be seen both as a heuristic for the variable rate case, and as a way to find some lower bound to the optimal average waiting time.

Since in practice the data of the requests are possibly not known in advance, some online heuristics have been considered, both centralized and decentralized. While the centralized heuristics tend to perform better, the knowledge of all the active unfulfilled requests is required to solve the problem. The decentralized ones have the advantage that they just require the knowledge of local data: each request is scheduled only on the basis of the amount of resource that it requests, as well as the aggregate information of the total supplied resource at a given time. This makes them fault-tolerant and easily scalable.

The presented methodology could be applied in a hierarchical way as follows. The real–time scheduling of the requests is decided on the basis of an online heuristic, which in general is based on some variables or specifications to be tuned. Periodically, e.g., once every day, a global optimization is performed based on the data recorded in the previous period: the optimal cost (or a lower bound) is used to evaluate the performance and efficacy of the on–line part, and possibly re–tune the parameters of the heuristic.

In Part II the problem of finding a decentralized policy to route some tokens injected in the source nodes of an unknown integer-weighted network with the aim of finding a way to leave it has been addressed. The proposed decentralized policy is specified as a threshold mechanism based on tokens accumulating in the nodes. The state of each node has been defined as the number of tokens deposited in each node. Then, the transition rule is very simple and depends only on local information: a token can move from a node to another adjacent one only if the "above threshold" condition holds, that is: the difference between the states of the corresponding nodes is greater than the connecting arc cost, which is the threshold. Otherwise, if the above threshold condition does not hold for any arc leaving the current node, the token stops and is deposited in there.

Initially, most tokens fail to leave the network, because they are stopped in the nodes; while this influences the choices of the tokens injected later on, these tokens are "lost". However, it has been shown that, in the long run, not only the newly injected tokens eventually find a way to leave the network, but also they follow the shortest paths to the closest sink nodes.

An enhanced version of the policy has been presented: now, if a token located in a given node could not proceed to any other connected node, instead of being deposited in there, the state of the current node is virtually increased until the above threshold condition is met, making the token eventually leave. If the network is strongly connected, all the tokens can eventually find a way to leave the network, so no token is lost, although initially the followed route is not necessarily the shortest one. With this enhanced version, the number of tokens needed to reach the steady-state is also reduced.

Then, it has been shown that the policy can be extended to the case in which the possible paths that each token can traverse are constrained: considering a secondary cost in each arc, a path is feasible only if its secondary cost is bounded. While the unconstrained problem could be solved in polynomial time by means of a centralized algorithm (Dijkstra, Bellman), the introduction of such constraints makes the problem NP-hard. To support this case, multi-component states are considered in each node: every component refers to the number of tokens that have paid a specific secondary cost to get in there. Then, the decentralized policy uses such state components; a transition is denied also if the path would become infeasible by it. In this case, the tokens which travel too much are indefinitely lost, even if the enhanced policy is applied. Still, in the long run, all the newly injected tokens leave the network through the shortest feasible path to the closest sink.

In both the constrained and unconstrained case, it has been demonstrated that, by construction, the policy also supports negative arc costs, multiple sources, multiple sinks, and dynamic time-varying networks, which makes the policy adaptive. Also, as the policy is decentralized, knowledge of the whole network is not required, so that the policy is easily scalable and fault-tolerant. Fractional arc costs could be supported too, but the performance worsens as the precision increases.

The main result from this Part is that from local decisions, global optimality results emerge in the long run, even in the presence of additional constraints that make the problem hard to be solved.

In Part III the problem of finding an online network-decentralized flow control stabilizing a flow network and based on node buffers to asymptotically minimize the $p$-norm of the controlled flow meeting a given demand has been addressed. The proposed control is network-decentralized in the sense that the controlled flow of each arc depends only on the state of its two extreme nodes, as well as the value of $p$. The control is also independent from the demand. Therefore, knowledge of the whole network is not required: this makes the proposed control adaptive, easily scalable and fault-tolerant.

The control has also been enhanced to support uncontrollable unknown dynamics depending on the buffer levels, buffer level control and weighted norms.

Both the proposed control and its enhanced versions guarantee that, when $1 < p < +\infty$, the unique optimal solution minimizing the $p$-norm and stabilizing the network is eventually reached at steady-state. This is not true anymore when $p = 1$ or $p = +\infty$: the control becomes discontinuous and there can be multiple optimal solutions. Still, as $p \to 1$, suboptimal sparser solutions arbitrarily closer to the optimal solution (minimizing the 1-norm) can be reached. Similarly, as $p \to \infty$, suboptimal fairer solutions arbitrarily closer to the optimal solution (minimizing the $\infty$-norm) can be reached.

As in Part II, the main result from this Part is that from local decisions, global optimality results emerge in the long run, even in the presence of unknown demands and unknown uncontrolled flows.

In all the Parts I to III, several simulations have been performed to validate the proposed approaches in different conditions.

# Appendices

# Appendix A
# Proofs from Part I

In this Appendix, the proofs of the theorems and propositions from Part I are reported. Most of these proofs are taken from [1] and are reported almost integrally, for completeness.

## Proof of Theorem 2.1.

By the definition of $z_i(t)$ (see Eq. (2.11)),

$$\int_0^{\infty} z_i(t)\ dt = \int_{t_i}^{t_i+\delta_i} 1\ dt = \delta_i.$$

Moreover, as the supplied energy must be equal to the requested one, it follows that

$$\int_{t_i}^{t_i+\delta_i} u_i(t)dt = \int_{t_i}^{t_i+\delta_i} d_i(t)dt = \tau_i\,,$$

where the equivalence between the two integrals comes from the identity $x_i(t_i) = x_i(t_i + \delta_i) = 0$.

Then, Eq. (2.13) is obtained by subtracting the two expression, recalling that $\delta_i = \omega_i + \tau_i$.

## Proof of Proposition 2.1.

For the equivalence to Eq. (2.6a), note that $d_i(t) = 0$ for $t < t_i$. Then, Eq. (2.14) becomes

$$x_i(t) = \frac{1}{\tau_i}\int_0^t [-u_i(\xi)]d\xi\,, \quad t < t_i.$$

As $\tau_i > 0$ and $u_i(t) \geq 0$ for all $t$, it follows that $x_i(t) \leq 0$ for $t < t_i$. Then, by imposing Eq. (2.16), it must be $x_i(t) = 0$ before $t_i$, which can be achieved only if $u_i(t) = 0$, too.

For the equivalence to Eq. (2.6a), note that for $t > t_i + \delta_i$, Eq. (2.14) can be written as

$$\begin{aligned}
x_i(t) &= \frac{1}{\tau_i}\int_0^t [d_i(\xi) - u_i(\xi)]d\xi \\
&= \frac{1}{\tau_i}\int_0^{t_i+\delta_i} [d_i(\xi) - u_i(\xi)]d\xi + \frac{1}{\tau_i}\int_{t_i+\delta_i}^t [d_i(\xi) - u_i(\xi)]d\xi \\
&= \frac{1}{\tau_i}\int_0^{t_i+\delta_i} [d_i(\xi)]d\xi - \frac{1}{\tau_i}\int_0^{t_i+\delta_i} [-u_i(\xi)]d\xi + \frac{1}{\tau_i}\int_{t_i+\delta_i}^t [-u_i(\xi)]d\xi \\
&= \frac{1}{\tau_i}\tau_i - \frac{1}{\tau_i}\tau_i + \frac{1}{\tau_i}\int_{t_i+\delta_i}^t [-u_i(\xi)]d\xi \\
&= \frac{1}{\tau_i}\int_{t_i+\delta_i}^t [-u_i(\xi)]d\xi\,, \quad t > t_i + \delta_i,
\end{aligned}$$

where the definition of completion time $c_i = t_i + \delta_i$, and the specification of $d(t)$ have been exploited. Again, as $\tau_i > 0$ and $u_i(t) \geq 0$ for all $t$, it follows that $x_i(t) \leq 0$ for $t > t_i + \delta_i$. Then, by imposing Eq. (2.16), it must be

$x_i(t) = 0$ after $t_i$, which can be achieved only if $u_i(t) = 0$, too.

## Proof of Proposition 2.2.

Assume that the request is fulfilled at time $t^*$. For all $t \geq t^* \geq t_i + \tau_i$, Eq. (2.14) becomes

$$\begin{aligned}
x_i(t) &= \frac{1}{\tau_i}\int_0^t \; [d_i(\xi) - u_i(\xi)]d\xi \\
&= \frac{1}{\tau_i}\int_0^{t_i+\tau_i} \; [d_i(\xi)]d\xi + \frac{1}{\tau_i}\int_0^{t^*} \; [-u_i(\xi)]d\xi + \frac{1}{\tau_i}\int_{t^*}^{t} \; [-u_i(\xi)]d\xi \\
&= \frac{1}{\tau_i}\tau_i - \frac{1}{\tau_i}\tau_i + \frac{1}{\tau_i}\int_{t^*}^{t} \; [-u_i(\xi)]d\xi \\
&= \frac{1}{\tau_i}\int_{t^*}^{t} \; [-u_i(\xi)]d\xi \,, \quad \text{for all } t \geq t^* \geq t_i + \tau_i,
\end{aligned}$$

where the specification of $d(t)$ and the definition of fulfilled request,

$$\int_0^{t^*} \; u_i(t)\; dt = \tau_i,$$

have been exploited. As $\tau_i > 0$ and $u_i(t) \geq 0$ for all $t$, it follows that $x_i(t) \leq 0$ for all $t \geq t^*$. Then, by imposing Eq. (2.16), it must be $x_i(t) = 0$ for all $t \geq t^* \geq t_i + \tau_i$.

To prove the contrary, now assume $x_i(t) = 0$ for all $t \geq t^* \geq t_i + \tau_i$. Then, Eq. (2.14) becomes

$$\begin{aligned}
0 &= \frac{1}{\tau_i}\int_0^t \; [d_i(\xi) - u_i(\xi)]d\xi \\
&= \frac{1}{\tau_i}\int_0^{t_i+\tau_i} \; [d_i(\xi)]d\xi + \frac{1}{\tau_i}\int_0^{t^*} \; [-u_i(\xi)]d\xi + \frac{1}{\tau_i}\int_{t^*}^{t} \; [-u_i(\xi)]d\xi \\
&= \frac{1}{\tau_i}\tau_i + \frac{1}{\tau_i}\int_0^{t^*} \; [-u_i(\xi)]d\xi + \frac{1}{\tau_i}\int_{t^*}^{t} \; [-u_i(\xi)]d\xi \,, \quad \text{for all } t \geq t^* \geq t_i + \tau_i.
\end{aligned}$$

The term $\frac{1}{\tau_i}\int_0^{t^*} \; [-u_i(\xi)]d\xi$ does not depend on $t$, therefore it is a constant. Then, the latter equation holds for all $t \geq t^* \geq t_i + \tau_i$ only if $\frac{1}{\tau_i}\int_{t^*}^{t} \; [-u_i(\xi)]d\xi$ is a constant too, say equal to $K$. In particular, this is true even for $t = t^*$, so that the value of this constant is

$$K = \frac{1}{\tau_i}\int_{t^*}^{t} \; [-u_i(\xi)]d\xi = \frac{1}{\tau_i}\int_{t^*}^{t^*} \; [-u_i(\xi)]d\xi = 0.$$

Then, it must be

$$\tau_i = \int_0^{t^*} \; u_i(\xi)d\xi,$$

which means that the request is fulfilled.

This proves the first part of the proposition. The second claim is true because the completion time is the first time instant in which the request becomes fulfilled.

## Proof of Theorem 2.2.

The inequality $\omega_i < \delta_i$ is true by construction (see Eq. (2.5)). To prove the other two inequalities, note that $x_i(t) = 0$ outside the interval $[t_i, t_i + \delta_i]$; therefore, the integral can be restricted to this interval. Then, consider the two following optimal control problems

$$\begin{aligned}
\mu_{min}\ (\mu_{max}) \quad &= \quad \min_{u_i}\ (\max_{u_i}) \int_{t_i}^{t_i+\delta_i} \; x_i(t)dt, \\
\text{s.t.} \quad & \quad \dot{x}_i(t) = [d_i(t) - u_i(t)]/\tau_i\,, \\
& \quad u_i(t) \in [0,1], \quad t \in [t_i + \Delta_i, t_i + \delta_i]\,, \\
& \quad u_i(t) = 0, \quad t \notin [t_i + \Delta_i, t_i + \delta_i]\,,
\end{aligned}$$

$$x_i(t_i) = x_i(t_i + \delta_i) = 0\,.$$

It will be proven that $\mu_{min} = \Delta_i$ and $\mu_{max} = \delta_i - \tau_i = \omega_i$. For a simple notation, drop all indices and assume $t_i = 0$. Then, the optimal control formulated on the interval $[0, \delta]$ becomes

$$\begin{aligned} \mu_{min}\ (\mu_{max}) \quad &= \quad \min_u\ (\max_u) \int_0^\delta x(t)dt, \\ \text{s.t.} \quad & \quad \dot{x}(t) = [d(t) - u(t)]/\tau, \\ & \quad u(t) \in [0,1],\ \text{for all}\ t \in\ [\Delta, \delta], \\ & \quad u(t) = 0\,,\ \ \text{for all}\ t \in\ [0, \Delta], \\ & \quad x(0) = x(\delta) = 0. \end{aligned}$$

The associated Lagrangian function is

$$L(x, u, \lambda) = x + \lambda(d - u)\,,$$

where the parameter $\tau$ has been included in the multiplier $\lambda$. The optimal control is the minimizer $u^*$ of $L(x, u, \lambda)$,

$$u^*(t) = \begin{cases} 1 & \text{if}\ \ \lambda(t) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

The dynamic equation for $\lambda$ is

$$\dot{\lambda}(t) = -\frac{\partial L(x, u, \lambda)}{\partial x} = -1\,. \tag{A.1}$$

Eq. (A.1) implies that $\lambda$ is a linear function of the time $t$, $\lambda(t) = -t$, and, consequently, $u^*$ switches (at most) once in the interval $[\Delta, \delta]$ passing from 1 to 0. Let $t'$ be the time instant in which this commutation occurs. Since $x(0) = 0$ and $x(\delta) = 0$, then $\int_0^\delta \dot{x}(t)dt = 0$, which implies

$$\int_\Delta^{t'} u(t)dt = (t' - \Delta) = \int_0^\delta d(t) = \tau,$$

so that $t' = \Delta + \tau$. The control is 1 on $[\Delta, \Delta + \tau]$ and zero thereafter. This corresponds to the *NI* case. The integral

$$\int_0^\delta x(t)dt = \mu_{min} = \Delta$$

is immediately computed as follows.
**Case (a)** If the $i$-th demand is satisfied immediately, then $u_i(t) = d_i(t) = 1$ for all $t \in [t_i, t_i + \tau_i]$, while $u_i(t) = d_i(t) = 0$ elsewhere. Hence, $d(t) \equiv u(t)$ and so $x_i(t) = 0$ for all $t$. Then, $\int_0^\delta x(t)dt = \mu_{min} = \Delta$, which means that $i$-th user experiences no delay in the supply.
Instead, if the demand is not immediately satisfied, but delayed, there exists $t''$ such that, for all $t_i < t < t''$, $d_i(t) = 1$ and $u_i(t) = 0$, which means that $x_i(t)$ increases (linearly) in time. At time $t_i + \Delta_i$ the user is eventually admitted to the use of the resource; two more cases are possible.
**Case (b)** If $\Delta_i \leq \tau_i$, then $t'' = t_i + \Delta_i$ and

$$\dot{x}(t) = \begin{cases} 1/\tau_i\,, & \text{for}\ t \in [t_i, t_i + \Delta_i]\,, \\ 0\,, & \text{for}\ t \in [t_i + \Delta_i, t_i + \tau_i]\,, \\ -1/\tau_i\,, & \text{for}\ t \in [t_i + \tau_i, t_i + \tau_i + \Delta_i]\,. \end{cases} \tag{A.2}$$

See the plot on the left of Fig. 2.3 for a pictorial representation of $x_i(t)$ and $d_i(t)$.

Eventually, at $t_i + \tau_i + \Delta_i$, $x_i$ reaches zero (again). A the generic time $t$ the value of the state variable $x_i(t)$ represents the time, evaluated as a fraction of $\tau_i$, needed by the user $i$ to reach the state it would have had if it had been admitted immediately at time $t_i$. Consider, for instance, an instant $t_i + \widetilde{t}_1 \in [t_i, t_i + \Delta_i]$. In the case of immediate admittance, the user $i$ would have completed, at time $t_i + \widetilde{t}_1$, a fraction $\widetilde{t}_1/\tau_i$ of its task. Correspondingly, $x_i(t_i + \widetilde{t}_1) = \widetilde{t}_1/\tau_i$ represents the delay between the actual user, which has been waiting up to $t_i + \widetilde{t}_1$, to the case of immediate admittance. This delay grows until the actual user $i$ is admitted to the resource, which occurs at time $t_i + \Delta_i$. After that, the delay with respect to the case of immediate admittance stops growing, as $u_i(t) = d_i(t) = 1$ for $f \in [t_i + \Delta, t_i + \tau_i]$. Finally, at time $t_i + \tau_i$ the user would have been fulfilled in the case of immediate admittance, while the actual user still needs time to complete it. In particular, if $\widetilde{t}_2$ is such that $t_i + \widetilde{t}_2 \in [t_i + \tau_i, t_i + \tau_i + \Delta_i]$, then $x_i(t_i + \widetilde{t}_2)$ represents the time (as a fraction of $\tau_i$) needed to complete the task.

**Case (c)** Instead, if $\Delta_i \geq \tau_i$, then $t'' = t_i + \tau_i$ and

$$\dot{x}(t) = \begin{cases} 1/\tau_i\,, & \text{for } t \in [t_i, t_i + \tau_i]\,, \\ 0\,, & \text{for } t \in [t_i + \tau_i, t_i + \Delta_i]\,, \\ -1/\tau_i\,, & \text{for } t \in [t_i + \Delta_i, t_i + \tau_i + \Delta_i]\,. \end{cases} \tag{A.3}$$

See Fig. 2.3, right for a pictorial representation. As before, at $t_i + \tau_i + \Delta_i$, $x_i$ is zero. In this case, the delay with respect to the case of immediate admittance grows up to 1, without exceeding this number, as the time needed to complete the task is, at most, $\tau_i$.

In all cases (a), (b) and (c) the area is equal to $\Delta_i$. In case (a), this is trivial, as $\Delta_i = 0$ and $d_i - u_i \equiv 0$, so Eq. (2.18) is satisfied. For cases (b) and (c), note that $x(t)$ has a trapezoidal profile, so that its integral can be simply computed.
In case (b), where $0 < \Delta_i < \tau_i$, from Fig. 2.3 (left) the area of the trapezoid is

$$\frac{\Delta_i}{\tau_i}\,\frac{(\tau_i + \Delta_i) + (\tau_i - \Delta_i)}{2} = \Delta_i\,.$$

In case (c), where $\Delta_i > \tau_i$, from Fig. 2.3 (right) the area of the trapezoid is

$$1\,\frac{(\tau_i + \Delta_i) + (\Delta_i - \tau_i)}{2} = \Delta_i\,.$$

To prove that the maximum is $\omega$, reconsider the Lagrangian function for maximization

$$L(x, u, \lambda) = -x + \lambda(d - u)$$

to get the new adjoint equation

$$\dot{\lambda}(t) = -\frac{\partial L(x, u, \lambda)}{\partial x} = 1.$$

The control $u$ switches from 0 to 1 (only) once at some time $t''$. To find the precise value of $t''$, note that, since the supplied energy must be the same as the requested one,

$$\int_{t''}^{\delta} u(t)dt = (\delta - t'') = \int_0^{\delta} d(t) = \tau,$$

so that $t'' = \delta - \tau$. The maximizer control is 1 on $[\delta - \tau, \delta]$ and zero elsewhere, while $d(t)$ is 1 over $[0, \tau]$ and zero elsewhere. Consequently,

$$\int_0^{\delta} x(t)dt \;=\; \frac{1}{\tau}\int_0^{\delta}\int_0^{t} d(\sigma)d\sigma dt - \frac{1}{\tau}\int_0^{\delta}\int_0^{t} u(\sigma)d\sigma dt = \left[\frac{\tau^2}{2} + \tau(\delta - \tau)\right]\cdot\frac{1}{\tau} - \left[\frac{\tau^2}{2}\right]\cdot\frac{1}{\tau} = \delta - \tau = \omega\,.$$

## Proof of Theorem 3.1.

It follows immediately from Theorem 2.2 by the fact that, for the *NI* case, $\Delta_i = \int_0^{\infty} x_i(t)dt = \omega_i$, $\delta_i = \Delta_i + \tau_i$, and each $\tau_i$ is constant.

## Proof of Proposition 3.1.

Let $u_i^*(t)$ be the optimal *NI* solution for user $i$ and assume it should be admitted at time $t_i + \Delta_i$. When sampling, $u_i$ must be delayed to the next sampled time $k\theta \geq t_i + \Delta_i$. The delay introduced is smaller than $\theta$.

Consider the first request with $i = 1$; sampling introduces the delay $k\theta - (t_1 + \Delta_1) < \theta$. If all the other requests $j = 2, 3, ..., n$ are assumed to be delayed of this same amount, the cumulative delay will be at most $n\theta$. For the second request $i = 2$, sampling will introduce a cumulative delay up to $(n-1)\theta$, because there are $n - 1$ requests after it. Iterating, for $j = 3, 4, ..., n$, the cumulative delay will be

$$[n + (n-1) + (n-2) + \cdots + 1]\theta = \theta\frac{n(n+1)}{2}.$$

## Proof of Proposition 3.2.

Let $u_i^* : \mathbb{R}^+ \to \{0,1\}$, for $i = 1,\dots,n$, be the optimal solution for the *NI* case. On the one hand, in view of Theorem 2.2 and Theorem 3.1, $u_1^*,\dots,u_n^*$ minimize $\sum_{i=1}^n \omega_i = \int_0^\infty \sum_{i=1}^n x_i(t)dt$ .

On the other hand, $u_1^*,\dots,u_n^*$ give a feasible solution for problem Eq. (3.18), hence the optimal value of the cost for the latter, denoted as $X_{rlx}^*$, must be less than or equal to $J_{rlx}(u_1^*,\dots,u_n^*)$, which is the minimizer, in general.

Note that $X_{rlx}^*$ is the minimum value that Eq. (3.3) can assume in the *VR* class. Now, let $v_i^* : \mathbb{R}^+ \to [0,1]$, for $i = 1,\dots,n$, be the optimal solution for the *VR* case minimizing $\sum_{i=1}^n \omega_i$. As of Theorem 2.2, considering the (non-optimal) $X_{VR} = \int_0^\infty \sum_{i=1}^n x_i(t)dt$ for solution $v_i^*$, it follows that $X_{rlx}^* \leq X_{VR} \leq J_{VR}(v_1^*,\dots,v_n^*)$, which means that $X_{rlx}^*$ is a lower bound for the *VR* case.

From Eq. (3.2), it follows that $X_{rlx}^*$ is a lower bound for $\sum_i \omega_i$ for all the *NI*, *IT* and *VR* cases.

## Proof of Proposition 3.3.

In [55] the optimality of the greedy strategy is proven, but that result cannot be applied here directly, because no positivity constraint was considered for $x_i$ in there, while this is a requirement in the considered framework. To solve the problem, the constraints $x_i(t) \geq 0$ can be *pretended to be neglected*, so that [55] can be applied and optimality results. To do so, it is to be proven that if $\mu > 0$ in Eq. (3.20), $x_i(t)$ cannot become negative, so the constraint is automatically satisfied. In this regard, if by contradiction $x_i(t) < 0$, the value of $u_i$ minimizing the derivative of $g(x_i)$, namely,

$$\dot{g}(x_i(t)) = -\mu(d_i(t) - u_i(t))$$

would be $u_i^*(t) = 0$, hence $\dot{x}_i = d_i - u_i^* = d_i \geq 0$. Then, $x_i(t) \geq 0$ is always satisfied, making $J_{pen} = \int_0^\infty \sum_i x_i(t)dt$. This means that $u_i^*$ is optimal for the original problem, too.

# Appendix B
# Proofs from Part II

In this Appendix, the proofs of the theorems, and lemmas from Part II are reported. Most of these proofs are taken from [2] and are reported almost integrally, for completeness.

## Proof of Lemma 8.1.

*If part*) If $x$ is admissible, $x_{h_s} - x_{h_{s+1}} \leq \gamma_{h_s,h_{s+1}}$ for each pair $h_s, h_{s+1} \in p$. Let $r$ be the number of (non-repeated) nodes in path $p$. Then,

$$\sum_{s=1}^{r-1} x_{h_s} - x_{h_{s+1}} \leq \sum_{s=1}^{r-1} \gamma_{h_s,h_{s+1}} = L(p),$$

where the first term does not depend on the actual path, but just from the starting and the terminal node:

$$\sum_{s=1}^{r-1} x_{h_s} - x_{h_{s+1}} = x_{h_1} - x_{h_r} = x_i - x_j.$$

*Only if part*) If $x$ is not admissible, there exists $(i,j) \in \mathcal{A}$ such that $x_i - x_j > \gamma_{ij}$. For the trivial path $p = \{i,j\}$ made of the arc $(i,j)$, it holds that $L(p) = \gamma_{ij} < x_i - x_j$, which contradicts the above relation and completes the proof.

## Proof of Lemma 8.2.

Consider a network $\mathcal{G}$ with a single sink node $j \in \mathcal{T}$ and the admissible state $x(k) \in \mathcal{O}$, for some $k \geq 0$. If no token is injected, $v(k) = 0$, and then $x(k+1) = x(k)$ must hold; the state of the sink node, that is 0 by definition, remains $x_j(k) = 0$. Also, for all $i \in \mathcal{N}$, $x_i(k) \leq L(p)$ for all paths $p$ joining $i$ with such sink node $j$, by the above argument and Lemma 8.1.

Now consider any anti-arborescence that joins all the nodes in $\mathcal{N}$ to the sink node, and let $l_i$ be the number of arcs of the path $q$ from the generic node $i$ to $j$, whose length fulfills $L(q) \leq l_i\bar{\gamma}$, with $\bar{\gamma}$ being the maximum arc cost.

By the two observations reported above, it holds that $x_i(k) \leq l_i\bar{\gamma}$, and hence

$$V(x(k)) = \sum_{i \in \mathcal{N}} x_i(k) \leq \sum_{i \in \mathcal{N}} l_i\bar{\gamma}.$$

To complete the proof, recall that the following inequality holds for all the anti-arborescences:

$$\sum_{i \in \mathcal{N}} l_i\bar{\gamma} \leq \bar{\gamma}\frac{|\mathcal{N}|(|\mathcal{N}|-1)}{2};$$

in this expression, the equality holds if the anti-arborescence is a directed path.

When there are multiple sinks, the proof is an easy but a cumbersome generalization of this one; as $x_j = 0$ for all $j \in \mathcal{T}$, the above argument holds for each anti-arborescence joining all the nodes to each sink node. Also, recall that a multiple-sink network can be transformed into an equivalent single-sink network, connecting all the former sinks to a new sink with arcs of zero costs.

## Proof of Theorem 8.1.

This Theorem is proven by showing that if $x(k) \in \mathcal{O}$ at a time $k$, then $x(k+1) \in \mathcal{O}$.

If at $k$ no token is injected in $\mathcal{G}$, then $x(k+1) = x(k) \in \mathcal{O}$ by definition of $\mathcal{O}$.
If at $k$ a token is injected in a source node $i \in \mathcal{S}$, note that Policy 8.1 imposes that only the injected token may move through the network. Hence, two cases may occur.

*Case $x(k)+e_i \in \mathcal{O}$).* By Condition Eq. (8.4c) on $u()$ and Eq. (8.5), it follows that the state equation Eq. (8.2) imposes $x(k+1) = x(k) + e_i$ with $x(k+1) \in \mathcal{O}$.

*Case $x(k)+e_i \notin \mathcal{O}$).* Assume by contradiction that a state $x(k+1) \in \mathcal{O}$ is not eventually reached. This means that the token makes a walk of an infinite number of elementary transitions. Since the number of nodes is finite, the token walk must include at least a circuit. Assume that the circuit is defined by the sequence of nodes

$$p = \{i_h \in \mathcal{N}, h = 1, 2 \ldots r\} \text{ with } i_1 = i_r.$$

Since $u_{i_h i_{h+1}} = 1$, for $h = 1, \ldots, r-1$, it implies

$$x_{i_h} - x_{i_{h+1}} \geq \gamma_{i_h, i_{h+1}}.$$

Summing up for all the considered values of $h$, the following is obtained

$$\sum_{h=1}^{r-1} x_{i_h} - x_{i_{h+1}} = x_{i_1} - x_{i_r} = x_{i_1} - x_{i_1} = 0 \geq \sum_{h=1}^{r-1} \gamma_{i_h, i_{h+1}}. \tag{B.1}$$

Hence, the length $L(p)$ of the circuit $p$ is non-positive, in contradiction with Assumption 7.1.c on the absence of non-positive circuits. As the walk of the injected token has no circuit, it is a path from the source node $i$ to the destination node $j$. Hence, $x(k+1) = x(k) \in \mathcal{O}$ if $j$ is a sink node, and $x(k+1) = x(k) + e_j$ otherwise. In the latter case $x(k+1) \in \mathcal{O}$ since $x_r(k+1) = x_r(k)$ for all nodes $r \neq j$ and because Policy 8.1 imposes that if the token stops in $j$ then $x_j(k) + 1 - x_r(k) = x_j(k+1) - x_r(k) \leq \gamma_{jk}$ for all $r \in \mathcal{N}_j$.

## Proof of Theorem 8.2.

An immediate consequence of Theorem 8.1 is that the number of tokens $V(x)$ in the network is non–decreasing, i.e, $V(x(k+1)) \geq V(x(k))$ when the state is admissible. Hence, $V(x(k))$ either converges to a finite value or diverges to $+\infty$. Lemma 8.2 excludes the latter possibility. Hence, $V(x(k))$ reaches a finite limit $\bar{V}$ from below. Since $V$ is integer-valued, this limit is reached in a finite time $\bar{k}$:

$$V(x(k)) = \bar{V} \leq \bar{\gamma} \frac{|\mathcal{N}|(|\mathcal{N}|-1)}{2}, \quad k \geq \bar{k}. \tag{B.2}$$

First, assume that at some time $k'$ a condition is reached in which whenever a new token is injected in a *non-empty subset* of the sources nodes this triggers the expulsion of a token from a sink in $\mathcal{T}$ without increasing the state; by Theorem 8.1, it necessarily holds that $x(k'+1) = x(k')$, and then $V(x(k'+1)) = V(x(k'))$, which means that $x(\bar{k})$ is a rest state. By Condition Eq. (B.2), $k'$ is finite: indeed, $V(x(k')) \leq \bar{V}$ and $k' \leq \bar{k}$.

Now assume that Condition Eq. (B.2) is reached (i.e., that $V(x(k))$ has converged to $\bar{V}$) by persistently injecting at least a token in *each* source node. Then, for $k \geq \bar{k}$, the injection of a new token in *any* source node triggers the expulsion of a token from a sink in $\mathcal{T}$, hence $x(k+1) = x(k) = x(\bar{k})$, which means that $x(\bar{k})$ is a global rest state.

In the special case of single-source network, initially, every token inserted in the network increases $V(x(k))$ by 1. Once a new token reaches a sink, all the successive ones do the same and $V(x(k)) = \bar{V}$ is reached. Then, the maximum number of tokens that needs to be inserted is equal to the upper bound of $\bar{V}$, minus the number of tokens that are initially already present in the buffers of the network nodes, i.e., $V(x(0))$.

## Proof of Theorem 8.3.

Under the theorem hypotheses, $V(x(k))$ is non-increasing, since tokens can only leave the network. Then, either $V(x(k))$ converges to $\underline{V}$ from above or to $-\infty$. The latter situation may not occur. Indeed, Policy 8.1 imposes that a token in node $i \in \mathcal{N}$ may leave the network only along a path $p = \{h_s \in \mathcal{N}, s = 1, \ldots r\}$, with $h_1 = i$ and $h_r$ a sink node in $\mathcal{T}$, such that condition $x_{h_s} - x_{h_{s+1}} \geq \gamma_{h_s, h_{s+1}}$ holds for $s = 1, \ldots r-1$. Summing up the inequalities for all $s = 1, \ldots r-1$, it is obtained that also $x_i > L(p)$ hold and, being the number of nodes finite, the value of $V(x(k))$ cannot keep on decreasing. Differently, $V(x(k))$ converges in finite time to $\underline{V}$ as $V(x(k))$ can assume solely integer values.

Following the same reasoning in the proof of Theorem 2, the subnetwork of $\mathcal{G}$ induced by the arcs $(i, j)$ traversed by a moving token consists of a forest of non-intersecting paths and includes no circuits. Hence, if $V(x(\underline{k})) = \underline{V}$ for $\underline{k} \geq 0$ then $x(k) = x(\underline{k})$ for all $k > \underline{k}$ if no tokens are injected. Consequently, $x(\underline{k}) \in \mathcal{O}$.

## Proof of Theorem 8.4.

Assume that at time $k$ the state $x(k)$ is admissible and that a new token is injected in node $i$. This token reaches its destination node $j$ along a path $p = \{i = h_0, h_1, \ldots, h_r = j\}$. Following the same arguments from the proof of Theorem 8.2, it follows that

$$x_i(k) - x_j(k) = \sum_{s=0}^{r-1} \gamma_{h_s, h_{s+1}} = L(p).$$

Now assume that an alternative path $p' = \{i = h'_0, h'_1, \ldots, h'_r = j\}$ shorter than $p$ exists. Then,is holds that

$$L(p') < L(p) = x_i(k) - x_j(k).$$

To complete the proof, observe that the above inequality implies that, by Lemma 8.1, the state $x$ is not admissible in contradiction with the hypothesis $x(k) \in \mathcal{O}$, as Eq. (8.1) does not hold for $p'$.

## Proof of Corollary 8.1.

The fact that $p$ is the shortest path from $i$ to $j$ follows from Theorem 8.4.

The first and the second points follow immediately from the proof of Theorem 8.4, recalling that the state of nodes $j, j' \in \mathcal{T}$ is zero, $x_j \equiv x_{j'} \equiv 0$, by definition of sink.

For the third point, the discussion reported above can be repeated, considering $i = h$.

## Proof of Theorem 9.1.

Suppose all the circuits $\varphi$ in $\mathcal{G}$ fulfill $C(\varphi) = \sum_{\varphi} \sigma_{ij} > 0$. A token with constrained cost $c$ in node $i \in \varphi$ which travels this circuit will reach node $i$ again with cost $c + C(\varphi) > c$: this means that in $\mathcal{G}_E$ the token, initially in node $i^c$, would reach a different node $i^{c+C(\varphi)}$ (which is closer to $i^{C_{max}}$). So, any walk in $\mathcal{G}$ becomes a path in $\mathcal{G}_E$ when there are no non-positive circuits for the costs $\sigma_{ij}$.

Now assume that a circuit $\varphi$ with $C(\varphi) = 0$ and, by Assumption 7.1, $L(\varphi) > 0$, exists in $\mathcal{G}$. Now, the token traveling the circuit would reach again node $i$ with the same constrained cost $c$ in $\mathcal{G}$, equivalently the token would reach the same node $i^c$ in $\mathcal{G}_E$. Hence, any circuit with $C(\varphi) = 0$ in $\mathcal{G}$ corresponds to at most $C = |\mathcal{C}|$ circuits in $\mathcal{G}_E$, each one with a positive length $L(\varphi_E) = L(\varphi)$.

Conversely, suppose there exists a circuit $\varphi_E$ in $\mathcal{G}_E$. A token in node $i^c \in \varphi_E$ traveling this circuit would reach node $i^c$ again. This means that the token travels a circuit $\varphi$ in $\mathcal{G}$, too: starting in node $i$ with constrained cost $c$, it returns to $i$ with the same $c$, and this is possible only if $C(\varphi) = 0$. The costs $L(\varphi)$ and $L(\varphi_E)$ paid by the token are the same in both networks by construction, and positive by Assumption 7.1.

## Proof of Theorem 9.2.

Let $p = \{h_1^{c_1}, \ldots, h_r^{c_r} : h_s^{c_s} \in \mathcal{N}_E\}$ be a shortest path in $\mathcal{G}_E$ from the source $h_1^{c_1}$ to a sink. By Corollary 8.1, there cannot exist another path $p'$ with $L(p') < L(p)$ that joins $h_1^{c_1}$ with any other sink; by construction, $p$ corresponds either to a feasible walk or a path in $\mathcal{G}$, in particular the shortest one. It will be shown that the former alternative is not possible.

Assume, by contradiction, that $p$ corresponds to a walk $\omega$ in $\mathcal{G}$. Because of Theorem 9.1, there must exist two values $s', s''$, $1 \leq s' < s'' \leq r$, such that the corresponding nodes $h_{s'}^{c_{s'}}$, $h_{s''}^{c_{s''}}$ in the path

$$p = \{h_1^{c_1}, \ldots, h_{s'}^{c_{s'}}, \ldots, h_{s''}^{c_{s''}}, \ldots, h_r^{c_r}\}$$

are associated with a same node $i \in \mathcal{N}$, i.e., $i = h_{s'} = h_{s''}$, so that the subpath

$$p' = \{h_{s'}^{c_{s'}}, \ldots, h_{s''}^{c_{s''}}\}$$

corresponds to a circuit $\varphi$ of the walk $\omega$ in $\mathcal{G}$ with $C(\varphi) = C(p') > 0$ and $L(\varphi) = L(p') > 0$ by Assumption 7.1. Consider now the new path

$$p'' = \{h_1^{c_1}, \ldots, h_{s'}^{c_{s'}}, h_{s''+1}^{c_{s''+1}-L(p')} \ldots, h_r^{c_r - L(p')}\}$$

of $\mathcal{G}_E$ that, in $\mathcal{G}$, corresponds to the walk $\omega'$ obtained by removing the circuit $\varphi$ from the walk $\omega$, passing from $h_{s'}^{c_{s'}}$ to $h_{s''+1}^{c_{s''+1}-L(p')}$ directly. It holds that $L(p'') = L(p) - L(\varphi) < L(p)$ and $C(p'') = C(p) - C(\varphi) < C(p)$. So $\omega'$ remains feasible for $\mathcal{G}$, and $p''$ is a path of $\mathcal{G}_E$, shorter than $p$, which contradicts the fact that path $p''$ with $L(p'') < L(p)$ that joins $h_1^{c_1}$ with any sink cannot exist. Therefore, $p$ must correspond to a shortest feasible path of $\mathcal{G}$.

# Bibliography

[1] Francesca Rosset, Daniele Casagrande, Babak Jafarpisheh, Pier Luca Montessoro, and Franco Blanchini. "Optimal Control Approach to Scheduling Power Supply Facilities: Theory and Heuristics". In: *IEEE Transactions on Control of Network Systems* 9.4 (2022). © 2022 IEEE, pp. 1679–1691. DOI: 10.1109/TCNS.2022.3165019.

[2] Francesca Rosset, Franco Blanchini, and Raffaele Pesenti. "An agent–based decentralized threshold policy finding the constrained shortest paths". In: (2022). Submitted.

[3] Franco Blanchini, Carlos Andrés Devia, Giulia Giordano, Raffaele Pesenti, and Francesca Rosset. "Fair and Sparse Solutions in Network-Decentralized Flow Control". In: *IEEE Control Systems Letters* 6 (2022). © 2022 IEEE, pp. 2984–2989. DOI: 10.1109/LCSYS.2022.3181341.

[4] Georgios C Chasparis. "Measurement-based efficient resource allocation with demand-side adjustments". In: *Automatica* 106 (2019), pp. 274–283.

[5] John S Vardakas, Nizar Zorba, and Christos V Verikoukis. "A survey on demand response programs in smart grids: Pricing methods and optimization algorithms". In: *IEEE Communications Surveys & Tutorials* 17.1 (2015), pp. 152–178.

[6] Tariq Samad, Edward Koch, and Petr Stluka. "Automated demand response for smart buildings and microgrids: The state of the practice and research challenges". In: *Proceedings of the IEEE* 104.4 (2016), pp. 726–744.

[7] Sijie Chen and Chen-Ching Liu. "From demand response to transactive energy: state of the art". In: *Journal of Modern Power Systems and Clean Energy* 5.1 (2017), pp. 10–19.

[8] Miadreza Shafie-khah, Pierluigi Siano, Jamshid Aghaei, Mohammad AS Masoum, Fangxing Li, and João PS Catalão. "Comprehensive review of the recent advances in industrial and commercial DR". In: *IEEE Transactions on Industrial Informatics* 15.7 (2019), pp. 3757–3771.

[9] Dario Bauso. "Dynamic demand and mean-field games". In: *IEEE Transactions on Automatic Control* 62.12 (2017), pp. 6310–6323.

[10] Maria Lorena Tuballa and Michael Lochinvar Abundo. "A review of the development of Smart Grid technologies". In: *Renewable and Sustainable Energy Reviews* 59 (2016), pp. 710–725.

[11] Daniel E Olivares, Ali Mehrizi-Sani, Amir H Etemadi, Claudio A Cañizares, Reza Iravani, Mehrdad Kazerani, Amir H Hajimiragha, Oriol Gomis-Bellmunt, Maryam Saeedifard, Rodrigo Palma-Behnke, et al. "Trends in microgrid control". In: *IEEE Transactions on smart grid* 5.4 (2014), pp. 1905–1919.

[12] Marc Beaudin and Hamidreza Zareipour. "Home energy management systems: A review of modelling and complexity". In: *Renewable and sustainable energy reviews* 45 (2015), pp. 318–335.

[13] AM Vega, F Santamaria, and E Rivas. "Modeling for home electric energy management: A review". In: *Renewable and Sustainable Energy Reviews* 52 (2015), pp. 948–959.

[14] Hussain Shareef, Maytham S Ahmed, Azah Mohamed, and Eslam Al Hassan. "Review on home energy management system considering demand responses, smart technologies, and intelligent controllers". In: *Ieee Access* 6 (2018), pp. 24498–24509.

[15] Usman Zafar, Sertac Bayhan, and Antonio Sanfilippo. "Home energy management system concepts, configurations, and technologies for the smart grid". In: *IEEE access* 8 (2020), pp. 119271–119286.

[16] Joy Chandra Mukherjee and Arobinda Gupta. "A review of charge scheduling of electric vehicles in smart grid". In: *IEEE Systems Journal* 9.4 (2015), pp. 1541–1553.

[17] Qinglong Wang, Xue Liu, Jian Du, and Fanxin Kong. "Smart charging for electric vehicles: A survey from the algorithmic perspective". In: *IEEE Communications Surveys & Tutorials* 18.2 (2016), pp. 1500–1517.

[18] Himadry Shekhar Das, Mohammad Mominur Rahman, S Li, and CW Tan. "Electric vehicles standards, charging infrastructure, and impact on grid integration: A technological review". In: *Renewable and Sustainable Energy Reviews* 120 (2020), p. 109618.

[19] Nanduni I Nimalsiri, Chathurika P Mediwaththe, Elizabeth L Ratnam, Marnie Shaw, David B Smith, and Saman K Halgamuge. “A survey of algorithms for distributed charging control of electric vehicles in smart grid”. In: *IEEE Transactions on Intelligent Transportation Systems* 21.11 (2020), pp. 4497–4515.

[20] Julian Barreiro-Gomez, Hamidou Tembine, Leonardo Stella, Dario Bauso, and Patrizio Colaneri. “Risk-Aware Control and Games in Engineering”. In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE. 2020, pp. 3860–3870.

[21] Zachary J Lee, Daniel Chang, Cheng Jin, George S Lee, Rand Lee, Ted Lee, and Steven H Low. “Large-scale adaptive electric vehicle charging”. In: *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE. 2018, pp. 1–7.

[22] Zachary J Lee, George Lee, Ted Lee, Cheng Jin, Rand Lee, Zhi Low, Daniel Chang, Christine Ortega, and Steven H Low. “Adaptive charging networks: A framework for smart electric vehicle charging”. In: *IEEE Transactions on Smart Grid* 12.5 (2021), pp. 4339–4350.

[23] Franco Blanchini, Franca Rinaldi, and Walter Ukovich. “A network design problem for a distribution system with uncertain demands”. In: *SIAM Journal on optimization* 7.2 (1997), pp. 560–578.

[24] Franco Blanchini, Raffaele Pesenti, Franca Rinaldi, and Walter Ukovich. “Feedback control of production-distribution systems with unknown demand and delays”. In: *IEEE Transactions on Robotics and Automation* 16.3 (2000), pp. 313–317.

[25] Zachary J Lee, Tongxin Li, and Steven H Low. “ACN-data: Analysis and applications of an open EV charging dataset”. In: *Proceedings of the Tenth ACM International Conference on Future Energy Systems*. 2019, pp. 139–149.

[26] Jean-Yves Le Boudec and Dan-Cristian Tomozei. “Worst-Case Optimal Battery Filling Policies With Constrained Adjustable Service”. In: *IEEE Transactions on Automatic Control* 60.10 (2015), pp. 2650–2660.

[27] Yunjian Xu, Feng Pan, and Lang Tong. “Dynamic scheduling for charging electric vehicles: A priority rule”. In: *IEEE Transactions on Automatic Control* 61.12 (2016), pp. 4094–4099.

[28] Jiangliang Jin, Yunjian Xu, and Zaiyue Yang. “Optimal deadline scheduling for electric vehicle charging with energy storage and random supply”. In: *Automatica* 119 (2020), p. 109096.

[29] Dhaou Said, Soumaya Cherkaoui, and Lyes Khoukhi. “Queuing model for EVs charging at public supply stations”. In: *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE. 2013, pp. 65–70.

[30] Tian Zhang, Wei Chen, Zhu Han, and Zhigang Cao. “Charging scheduling of electric vehicles with local renewable energy under uncertain electric vehicle arrival and grid power price”. In: *IEEE Transactions on Vehicular Technology* 63.6 (2014), pp. 2600–2612.

[31] Krishnan Muralitharan, Rathinasamy Sakthivel, and Yan Shi. “Multiobjective optimization technique for demand side management with load balancing approach in smart grid”. In: *Neurocomputing* 177 (2016), pp. 110–119.

[32] Zahoor Ali Khan, Adia Khalid, Nadeem Javaid, Abdul Haseeb, Tanzila Saba, and Muhammad Shafiq. “Exploiting nature-inspired-based artificial intelligence techniques for coordinated day-ahead scheduling to efficiently manage energy in smart grid”. In: *IEEE Access* 7 (2019), pp. 140102–140125.

[33] Shaojie Tang, Qiuyuan Huang, Xiang-Yang Li, and Dapeng Wu. “Smoothing the energy consumption: Peak demand reduction in smart grid”. In: *2013 Proceedings IEEE INFOCOM*. IEEE. 2013, pp. 1133–1141.

[34] Qiang Tang, Kezhi Wang, Yun Song, Feng Li, and Jong Hyuk Park. “Waiting time minimized charging and discharging strategy based on mobile edge computing supported by software-defined network”. In: *IEEE Internet of Things Journal* 7.7 (2020), pp. 6088–6101.

[35] Angelos Aveklouris, Maria Vlasiou, and Bert Zwart. “A stochastic resource-sharing network for electric vehicle charging”. In: *IEEE Transactions on Control of Network Systems* 6.3 (2019), pp. 1050–1061.

[36] Leehter Yao, Zolboo Damiran, and Wei Hong Lim. “Energy management optimization scheme for smart home considering different types of appliances”. In: *2017 IEEE international conference on environment and electrical engineering and 2017 IEEE industrial and commercial power systems Europe (EEEIC/I&CPS Europe)*. IEEE. 2017, pp. 1–6.

[37] Mahnoosh Alizadeh, Hoi-To Wai, Mainak Chowdhury, Andrea Goldsmith, Anna Scaglione, and Tara Javidi. “Optimal pricing to manage electric vehicles in coupled power and transportation networks”. In: *IEEE Transactions on control of network systems* 4.4 (2017), pp. 863–875.

[38] Zeinab Moghaddam, Iftekhar Ahmad, Daryoush Habibi, and Quoc Viet Phung. “Smart charging strategy for electric vehicle charging stations”. In: *IEEE Transactions on transportation electrification* 4.1 (2018), pp. 76–88.

[39] Mahnoosh Alizadeh, Hoi-To Wai, Andrea Goldsmith, and Anna Scaglione. "Retail and wholesale electricity pricing considering electric vehicle mobility". In: *IEEE Transactions on Control of Network Systems* 6.1 (2019), pp. 249–260.

[40] S Rasoul Etesami, Walid Saad, Narayan B Mandayam, and H Vincent Poor. "Smart routing of electric vehicles for load balancing in smart grids". In: *Automatica* 120 (2020), p. 109148.

[41] Giulio Ferro, Massimo Paolucci, and Michela Robba. "Optimal charging and routing of electric vehicles with power constraints and time-of-use energy prices". In: *IEEE Transactions on Vehicular Technology* 69.12 (2020), pp. 14436–14447.

[42] Peter Brucker. "Scheduling algorithms". In: Fifth Edition. Springer, 2007.

[43] Michael Pinedo. "Scheduling: Theory, Algorithms, And Systems". In: Fifth Edition. Springer, 2016.

[44] Konstantin Kogan and Eugene Khmelnitsky. *Scheduling: control-based theory and polynomial-time algorithms.* Vol. 43. Springer Science & Business Media, 2013.

[45] Mathijs M. de Weerdt, Michael Albert, Vincent Conitzer, and Koos van der Linden. "Complexity of Scheduling Charging in the Smart Grid". In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems.* AAMAS '18. Stockholm, Sweden, 2018, pp. 1924–1926.

[46] Ronald Lewis Graham, Eugene Leighton Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan. "Optimization and approximation in deterministic sequencing and scheduling: a survey". In: *Annals of discrete mathematics.* Vol. 5. Elsevier, 1979, pp. 287–326.

[47] Linus Schrage. "A proof of the optimality of the shortest remaining processing time discipline". In: *Operations Research* 16.3 (1968), pp. 687–690.

[48] Saeed Rubaiee and Mehmet Bayram Yildirim. "An energy-aware multiobjective ant colony algorithm to minimize total completion time and energy cost on a single-machine preemptive scheduling". In: *Computers & Industrial Engineering* 127 (2019), pp. 240–252.

[49] Jacek Blazewicz, Jan Karel Lenstra, and AHG Rinnooy Kan. "Scheduling subject to resource constraints: classification and complexity". In: *Discrete applied mathematics* 5.1 (1983), pp. 11–24.

[50] Jan Karel Lenstra. "Sequencing by enumerative methods". In: (1977).

[51] Jianzhong Du, Joseph Y-T Leung, and Gilbert H Young. "Minimizing mean flow time with release time constraint". In: *Theoretical Computer Science* 75.3 (1990), pp. 347–355.

[52] Nhan-Quy Nguyen, Farouk Yalaoui, Lionel Amodeo, Hicham Chehade, and Pascal Toggenburger. "Total completion time minimization for machine scheduling problem under time windows constraints with jobs' linear processing rate function". In: *Computers & Operations Research* 90 (2018), pp. 110–124.

[53] E. Kalvelagen. *Modeling number of machine start-ups.* 2014. URL: https://yetanothermathprogrammingconsultant.blogspot.com/2014/12/modeling-number-of-machine-start-ups.html.

[54] Vadim Utkin. "Variable structure systems with sliding modes". In: *IEEE Transactions on Automatic control* 22.2 (1977), pp. 212–222.

[55] Francesco Martinelli, Chang Shu, and James R Perkins. "On the optimality of myopic production controls for single-server, continuous-flow manufacturing systems". In: *IEEE Transactions on Automatic Control* 46.8 (2001), pp. 1269–1273.

[56] Pooya Rezaei, Jeff Frolik, and Paul DH Hines. "Packetized plug-in electric vehicle charge management". In: *IEEE Transactions on Smart Grid* 5.2 (2014), pp. 642–650.

[57] Zachary J. Lee, Sunash Sharma, Daniel Johansson, and Steven H. Low. "ACN-Sim: An Open-Source Simulator for Data-Driven Electric Vehicle Charging Research". In: *IEEE Transactions on Smart Grid* 12.6 (2021), pp. 5113–5123.

[58] Franco Blanchini, Daniele De Caneva, Pier Luca Montessoro, and Davide Pierattoni. "Control-based p-persistent adaptive communication protocol". In: *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 7.2 (2012), pp. 1–18.

[59] Franco Blanchini, Daniele Casagrande, Giulia Giordano, and Pier Luca Montessoro. "A robust decentralized control for channel sharing communication". In: *IEEE Transactions on Control of Network Systems* 4.2 (2017), pp. 336–346.

[60] Francesca Rosset. "Agent-based control for distributed energy management: study and evaluation". MA thesis. University of Udine, 2019.

[61] Babak Jafarpisheh, Francesca Rosset, Franco Blanchini, and Pier Luca Montessoro. "Agent-Based Distributed Control for Schedulable Appliances in Home Energy Management Systems". In: (). In preparation.

[62] Jeff Frolik and P Hines. "Urgency-driven, plug-in electric vehicle charging". In: *2012 3rd IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*. IEEE. 2012, pp. 1–5.

[63] Jonathan Currie, David I Wilson, Nick Sahinidis, and Jose Pinto. "OPTI: Lowering the barrier between open source optimizers and the industrial MATLAB user". In: *Foundations of computer-aided process operations* 24 (2012), p. 32.

[64] Tobias Achterberg. "SCIP: solving constraint integer programs". In: *Mathematical Programming Computation* 1.1 (2009), pp. 1–41.

[65] Francesco Marra, Guang Ya Yang, Chresten Træholt, Esben Larsen, Claus Nygaard Rasmussen, and Shi You. "Demand profile study of battery electric vehicle under different charging options". In: *2012 IEEE power and energy society general meeting*. IEEE. 2012, pp. 1–7.

[66] Chengxiu Chen, Fei Shang, Mohamad Salameh, and Mahesh Krishnamurthy. "Challenges and advancements in fast charging solutions for EVs: A technological review". In: *2018 IEEE Transportation Electrification Conference and Expo (ITEC)*. IEEE. 2018, pp. 695–701.

[67] Elpiniki Apostolaki-Iosifidou, Paul Codani, and Willett Kempton. "Measurement of power loss during electric vehicle charging and discharging". In: *Energy* 127 (2017), pp. 730–742.

[68] Caltech. *ACN-Data – A Public EV Charging Dataset*. URL: https://ev.caltech.edu/dataset (visited on 06/07/2020).

[69] DATA.GOV. *Electric Vehicle Population Data*. URL: https://catalog.data.gov/dataset/electric-vehicle-population-data (visited on 06/07/2020).

[70] EVCompare.io. *All Electric Passenger Cars*. URL: https://evcompare.io/cars/ (visited on 06/07/2020).

[71] ClipperCreek. *EVSE Selector Tool*. URL: https://www.clippercreek.com/charging-station-selector-tool/ (visited on 06/07/2020).

[72] Franco Blanchini, Daniele Casagrande, Filippo Fabiani, Giulia Giordano, and Raffaele Pesenti. "A network-decentralised strategy for shortest-path-flow routing". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE. 2019, pp. 1126–1131.

[73] Daniel Ratton Figueiredo and Michele Garetto. "On the emergence of shortest paths by reinforced random walks". In: *IEEE Transactions on Network Science and Engineering* 4.1 (2016), pp. 55–69.

[74] Sheng Yu, Baoxian Zhang, Cheng Li, and Hussein T Mouftah. "Routing protocols for wireless sensor networks with mobile sinks: A survey". In: *IEEE Communications Magazine* 52.7 (2014), pp. 150–157.

[75] Zaher Al Aghbari, Ahmed M Khedr, Walid Osamy, Ifra Arif, and Dharma P Agrawal. "Routing in wireless sensor networks using optimization techniques: A survey". In: *Wireless Personal Communications* 111.4 (2020), pp. 2407–2434.

[76] James Evans. *Optimization Algorithms for Networks and Graphs: Revised and Expanded*. CRC Press, 2017.

[77] Amgad Madkour, Walid G Aref, Faizan Ur Rehman, Mohamed Abdur Rahman, and Saleh Basalamah. "A survey of shortest-path algorithms". In: *arXiv preprint arXiv:1705.02044* (2017).

[78] Edsger W Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische Mathematik* 1 (1959), pp. 269–271.

[79] Richard Bellman. "On a routing problem". In: *Quarterly of applied mathematics* 16.1 (1958), pp. 87–90.

[80] Lester R Ford Jr. *Network flow theory*. Tech. rep. Rand Corp Santa Monica Ca, 1956.

[81] Stefan Funke and Sabine Storandt. "Consistent rounding of edge weights in graphs". In: *Ninth Annual Symposium on Combinatorial Search*. 2016.

[82] Herman Haverkort, David Kübel, and Elmar Langetepe. "Shortest-path-preserving rounding". In: *International Workshop on Combinatorial Algorithms*. Springer. 2019, pp. 265–277.

[83] Christos H Papadimitriou and Mihalis Yannakakis. "Shortest paths without a map". In: *Theoretical Computer Science* 84.1 (1991), pp. 127–150.

[84] Ariel Felner, Roni Stern, Asaph Ben-Yair, Sarit Kraus, and Nathan Netanyahu. "PHA*: Finding the shortest path with A* in an unknown physical environment". In: *Journal of Artificial Intelligence Research* 21 (2004), pp. 631–670.

[85] Daniele Ferone, Paola Festa, Antonio Napoletano, and Tommaso Pastore. "Shortest paths on dynamic graphs: a survey". In: *Pesquisa Operacional* 37 (2017), pp. 487–508.

[86] Sunita and Garg Deepak. "Dynamizing Dijkstra: A solution to dynamic shortest path problem through retroactive priority queue". In: *Journal of King Saud University-Computer and Information Sciences* 33.3 (2021), pp. 364–373.

[87] Alireza Tahbaz-Salehi and Ali Jadbabaie. “A one-parameter family of distributed consensus algorithms with boundary: From shortest paths to mean hitting times”. In: *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE. 2006, pp. 4664–4669.

[88] Yinyan Zhang and Shuai Li. “Distributed biased min-consensus with applications to shortest path planning”. In: *IEEE Transactions on Automatic Control* 62.10 (2017), pp. 5429–5436.

[89] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. “Ant colony optimization”. In: *IEEE computational intelligence magazine* 1.4 (2006), pp. 28–39.

[90] Feng Yu, Yanjun Li, and Tie-Jun Wu. “A temporal ant colony optimization approach to the shortest path problem in dynamic scale-free networks”. In: *Physica A: Statistical Mechanics and its Applications* 389.3 (2010), pp. 629–636.

[91] Andrei Lissovoi and Carsten Witt. “Runtime analysis of ant colony optimization on dynamic shortest path problems”. In: *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. 2015, pp. 1605–1612.

[92] Pablo Rabanal, Ismael Rodríguez, and Fernando Rubio. “Applications of river formation dynamics”. In: *Journal of computational science* 22 (2017), pp. 26–35.

[93] Grzegorz Redlarski, Mariusz Dabkowski, and Aleksander Palkowski. “Generating optimal paths in dynamic environments using River Formation Dynamics algorithm”. In: *Journal of Computational Science* 20 (2017), pp. 8–16.

[94] Xiaoge Zhang, Qing Wang, Andrew Adamatzky, Felix TS Chan, Sankaran Mahadevan, and Yong Deng. “An improved physarum polycephalum algorithm for the shortest path problem”. In: *The Scientific World Journal* 2014 (2014).

[95] Xiaoge Zhang, Felix TS Chan, Hai Yang, and Yong Deng. “An adaptive amoeba algorithm for shortest path tree computation in dynamic graphs”. In: *Information Sciences* 405 (2017), pp. 123–140.

[96] Xuezhi Zhu, Wenjian Luo, and Tao Zhu. “An improved genetic algorithm for dynamic shortest path problems”. In: *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2014, pp. 2093–2100.

[97] Ammar W Mohemmed, Nirod Chandra Sahoo, and Tan Kim Geok. “Solving shortest path problem using particle swarm optimization”. In: *Applied Soft Computing* 8.4 (2008), pp. 1643–1653.

[98] Katja Verbeeck and Ann Nowe. “Colonies of learning automata”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 32.6 (2002), pp. 772–780.

[99] Sudip Misra and B John Oommen. “Dynamic algorithms for the shortest path routing problem: learning automata-based solutions”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 35.6 (2005), pp. 1179–1192.

[100] Hamid Beigy and Mohammad Reza Meybodi. “A sampling method based on distributed learning automata for solving stochastic shortest path problem”. In: *Knowledge-Based Systems* 212 (2021), p. 106638.

[101] William T Zaumen and JJ Garcia-Luna Aceves. “Dynamics of distributed shortest-path routing algorithms”. In: *Proceedings of the conference on Communications architecture & protocols*. 1991, pp. 31–42.

[102] Horst F Wedde and Muddassar Farooq. “A comprehensive review of nature inspired routing algorithms for fixed telecommunication networks”. In: *Journal of Systems Architecture* 52.8-9 (2006), pp. 461–484.

[103] Azzedine Boukerche, Begumhan Turgut, Nevin Aydin, Mohammad Z Ahmad, Ladislau Bölöni, and Damla Turgut. “Routing protocols in ad hoc networks: A survey”. In: *Computer networks* 55.13 (2011), pp. 3032–3080.

[104] Eiman Alotaibi and Biswanath Mukherjee. “A survey on routing algorithms for wireless ad-hoc and mesh networks”. In: *Computer networks* 56.2 (2012), pp. 940–965.

[105] Franco Blanchini, Daniele Casagrande, Filippo Fabiani, Giulia Giordano, David Palma, and Raffaele Pesenti. “A threshold mechanism ensures minimum-path flow in lightning discharge”. In: *Scientific reports* 11.1 (2021), pp. 1–9.

[106] Frederick Bock and Scott Cameron. “Allocation of network traffic demand by instant determination of optimum paths”. In: *Operations Research*. Vol. 6. 4. INST OPERATIONS RESEARCH MANAGEMENT SCIENCES 901 ELKRIDGE LANDING RD, STE 400, LINTHICUM HTS, MD 21090-2909. 1958, pp. 633–634.

[107] Robert M. Peart, Paul H. Randolph, and Thomas E. Bartlett. “Letter to the Editor—The Shortest-Route Problem”. In: *Operations Research* 8 (1960), pp. 866–868.

[108] George J Minty. “A comment on the shortest-route problem”. In: *Operations Research* 5.5 (1957), pp. 724–724.

[109] Victor Klee. “A “string algorithm” for shortest path in directed networks”. In: *Operations Research* 12.3 (1964), pp. 428–432.

[110] Linkai Bu and Tzi-Dar Chiueh. “Solving the shortest path problem using an analog network”. In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 46.11 (1999), pp. 1360–1363.

[111] Eleonora Papadimitriou, George Yannis, and John Golias. “A critical assessment of pedestrian behaviour models”. In: *Transportation research part F: traffic psychology and behaviour* 12.3 (2009), pp. 242–255.

[112] Dario Bauso, Xuan Zhang, and Antonis Papachristodoulou. “Density Flow in Dynamical Networks via Mean-Field Games”. In: *IEEE Transactions on Automatic Control* 62.3 (2017), pp. 1342–1355. DOI: 10.1109/TAC.2016.2584979.

[113] Hans C Joksch. “The shortest route problem with constraints”. In: *Journal of Mathematical analysis and applications* 14.2 (1966), pp. 191–197.

[114] Michael R Garey. “A Guide to the Theory of NP-Completeness”. In: *Computers and intractability* (1979).

[115] Stefan Irnich and Guy Desaulniers. “Shortest path problems with resource constraints”. In: *Column generation*. Springer, 2005, pp. 33–65.

[116] Luigi Di Puglia Pugliese and Francesca Guerriero. “A survey of resource constrained shortest path problems: Exact solution approaches”. In: *Networks* 62.3 (2013), pp. 183–200.

[117] Irina Dumitrescu and Natashia Boland. “Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem”. In: *Networks: An International Journal* 42.3 (2003), pp. 135–153.

[118] Xiaoyan Zhu and Wilbert E Wilhelm. “Three-stage approaches for optimizing some variations of the resource constrained shortest-path sub-problem in a column generation context”. In: *European journal of operational research* 183.2 (2007), pp. 564–577.

[119] Xiaoyan Zhu and Wilbert E Wilhelm. “A three-stage approach for the resource-constrained shortest path as a sub-problem in column generation”. In: *Computers & Operations Research* 39.2 (2012), pp. 164–178.

[120] Gabriel Y Handler and Israel Zang. “A dual algorithm for the constrained shortest path problem”. In: *Networks* 10.4 (1980), pp. 293–309.

[121] Leonardo Lozano and Andrés L Medaglia. “On an exact method for the constrained shortest path problem”. In: *Computers & Operations Research* 40.1 (2013), pp. 378–384.

[122] Xiaoge Zhang, Yajuan Zhang, Yong Hu, Yong Deng, and Sankaran Mahadevan. “An adaptive amoeba algorithm for constrained shortest paths”. In: *Expert Systems with Applications* 40.18 (2013), pp. 7607–7616.

[123] Hongping Wang, Xi Lu, Xiaoge Zhang, Qing Wang, and Yong Deng. “A bio-inspired method for the constrained shortest path problem”. In: *The Scientific World Journal* 2014 (2014).

[124] Yannis Marinakis, Athanasios Migdalas, and Angelo Sifaleras. “A hybrid particle swarm optimization–variable neighborhood search algorithm for constrained shortest path problems”. In: *European Journal of Operational Research* 261.3 (2017), pp. 819–834.

[125] Yong Zheng, Min Han, Liuting He, Ya Li, Guanglin Xing, and Rui Hou. “A QoS-supported Multi-constrained Routing Strategy Based on Ant-Colony Optimization for Named Data Networking”. In: *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*. IEEE. 2018, pp. 18–23.

[126] KePing Li, ZiYou Gao, Tao Tang, and LiXing Yang. “Solving the constrained shortest path problem using random search strategy”. In: *Science China Technological Sciences* 53.12 (2010), pp. 3258–3263.

[127] Franco Blanchini, Franca Rinaldi, and Walter Ukovich. “A network design problem for a distribution system with uncertain demands”. In: *SIAM Journal on optimization* 7.2 (1997), pp. 560–578.

[128] Duncan J Watts and Steven H Strogatz. “Collective dynamics of ‘small-world’ networks”. In: *Nature* 393.6684 (1998), pp. 440–442.

[129] MathWorks documentation for MATLAB. *Build Watts-Strogatz Small World Graph Model*. 2015. URL: https://www.mathworks.com/help/matlab/math/build-watts-strogatz-small-world-graph-model.html (visited on 07/18/2022).

[130] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. “Network flows”. In: (1988).

[131] Altuğ İftar and Edward J. Davison. “Decentralized control strategies for dynamic routing”. In: *Optimal Control Applications and Methods* 23.6 (2002), pp. 329–355.

[132] Franklin Moss and Adrian Segall. “An optimal control approach to dynamic routing in networks”. In: *IEEE Transactions on Automatic Control* 27.2 (1982), pp. 329–339.

[133] Mathias Bürger and Claudio De Persis. “Dynamic coupling design for nonlinear output agreement and time-varying flow control”. In: *Automatica* 51 (2015), pp. 210–222.

[134] Giacomo Como. “On resilient control of dynamical flow networks”. In: *Annual Reviews in Control* 43 (2017), pp. 80–90.

[135] Samuel Coogan and Murat Arcak. “A compartmental model for traffic networks and its dynamical behavior”. In: *IEEE Transactions on Automatic Control* 60.10 (2015), pp. 2698–2703.

[136] Altuğ İftar. “A linear programming based decentralized routing controller for congested highways”. In: *Automatica* 35.2 (1999), pp. 279–292.

[137] Gustav Nilsson and Giacomo Como. “Generalized proportional allocation policies for robust control of dynamical flow networks”. In: *IEEE Transactions on Automatic Control* (2022).

[138] Angelo Alessandri, Mauro Gaggero, and Flavio Tonelli. “Robust predictive control for the management of multi-echelon distribution chains”. In: *53rd IEEE Conference on Decision and Control*. IEEE. 2014, pp. 6459–6464.

[139] Franco Blanchini, Stefano Miani, and Walter Ukovich. “Control of production-distribution systems with unknown inputs and system failures”. In: *IEEE Transactions on Automatic Control* 45.6 (2000), pp. 1072–1081.

[140] Michael Cantoni, Erik Weyer, Yuping Li, Su Ki Ooi, Iven Mareels, and Matthew Ryan. “Control of large-scale irrigation networks”. In: *Proceedings of the IEEE* 95.1 (2007), pp. 75–91.

[141] Tjardo Scholten, Claudio De Persis, and Pietro Tesi. “Optimal steady state regulation of distribution networks with input and flow constraints”. In: *2016 American Control Conference (ACC)*. IEEE. 2016, pp. 6953–6958.

[142] Sebastian Trip, Tjardo Scholten, and Claudio De Persis. “Optimal regulation of flow networks with transient constraints”. In: *Automatica* 104 (2019), pp. 141–153.

[143] Hyo-Sung Ahn, Byeong-Yeon Kim, Young-Hun Lim, Byung-Hun Lee, and Kwang-Kyo Oh. “Distributed coordination for optimal energy generation and distribution in cyber-physical energy networks”. In: *IEEE transactions on cybernetics* 48.3 (2018), pp. 941–954.

[144] Franco Blanchini, Elisa Franco, Giulia Giordano, Vahid Mardanlou, and Pier Luca Montessoro. “Compartmental flow control: Decentralization, robustness and optimality”. In: *Automatica* 64 (2016), pp. 18–28.

[145] John A Jacquez and Carl P Simon. “Qualitative theory of compartmental systems”. In: *Siam Review* 35.1 (1993), pp. 43–79.

[146] Jieqiang Wei and Arjan J van der Schaft. “Load balancing of dynamical distribution networks with flow constraints and unknown in/outflows”. In: *Systems & Control Letters* 62.11 (2013), pp. 1001–1008.

[147] Dario Bauso, Franco Blanchini, Laura Giarré, and Raffaele Pesenti. “The linear saturated decentralized strategy for constrained flow control is asymptotically optimal”. In: *Automatica* 49.7 (2013), pp. 2206–2212.

[148] Franco Blanchini, Elisa Franco, and Giulia Giordano. “Network-decentralized control strategies for stabilization”. In: *IEEE Transactions on Automatic Control* 60.2 (2015), pp. 491–496.

[149] Franco Blanchini, Daniele Casagrande, Filippo Fabiani, Giulia Giordano, and Raffaele Pesenti. “A network-decentralised strategy for shortest-path-flow routing”. In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE. 2019, pp. 1126–1131.

[150] Franco Blanchini, Daniele Casagrande, Filippo Fabiani, Giulia Giordano, David Palma, and Raffaele Pesenti. “A threshold mechanism ensures minimum-path flow in lightning discharge”. In: *Scientific reports* 11.1 (2021), pp. 1–9.

[151] Franco Blanchini, Daniele Casagrande, Filippo Fabiani, Giulia Giordano, and Raffaele Pesenti. “Network-decentralised optimisation and control: An explicit saturated solution”. In: *Automatica* 103 (2019), pp. 379–389.

[152] Simone Milanesi, Francesca Rosset, Marta Colaneri, Giulia Giordano, Kenneth Pesenti, Franco Blanchini, Paolo Bolzern, Patrizio Colaneri, Paolo Sacchi, Giuseppe De Nicolao, and Raffaele Bruno. “Early detection of variants of concern via funnel plots of regional reproduction numbers”. In: *Scientific Reports* 13.1 (2023), p. 1052. DOI: 10.1038/s41598-022-27116-8.

[153] Franco Blanchini, Lorenzo Brunato, Carlo Drioli, Raffaele Pesenti, and Francesca Rosset. “Optimal trajectory generation in corridors via quadratic programming and its receding horizon implementation”. In: (2022). Submitted.

# List of publications and submitted manuscripts

The main results I've worked on during the three years of my Ph.D. are presented in the following papers. Parts I to III of this thesis are based on them.

- **Francesca Rosset, Daniele Casagrande, Babak Jafarpisheh, Pier Luca Montessoro, and Franco Blanchini. "Optimal Control Approach to Scheduling Power Supply Facilities: Theory and Heuristics". In: *IEEE Transactions on Control of Network Systems* 9.4 (2022). © 2022 IEEE, pp. 1679–1691. DOI:** 10.1109/TCNS.2022.3165019
- **Francesca Rosset, Franco Blanchini, and Raffaele Pesenti. "An agent–based decentralized threshold policy finding the constrained shortest paths". In: (2022). Submitted**
- **Franco Blanchini, Carlos Andrés Devia, Giulia Giordano, Raffaele Pesenti, and Francesca Rosset. "Fair and Sparse Solutions in Network-Decentralized Flow Control". In: *IEEE Control Systems Letters* 6 (2022). © 2022 IEEE, pp. 2984–2989. DOI:** 10.1109/LCSYS.2022.3181341

During these three years of my Ph.D., I've also had the opportunity to work in some other topics, which are not presented in this thesis. The two main resulting works are just briefly described below.

- **Simone Milanesi, Francesca Rosset, Marta Colaneri, Giulia Giordano, Kenneth Pesenti, Franco Blanchini, Paolo Bolzern, Patrizio Colaneri, Paolo Sacchi, Giuseppe De Nicolao, and Raffaele Bruno. "Early detection of variants of concern via funnel plots of regional reproduction numbers". In: *Scientific Reports* 13.1 (2023), p. 1052. DOI:** 10.1038/s41598-022-27116-8

  The first work [152] comes from the collaboration between Simone Milanesi, Marta Colaneri, Giulia Giordano, Kenneth Pesenti, Franco Blanchini, Paolo Bolzern, Patrizio Colaneri, Paolo Sacchi, Giuseppe De Nicolao, Raffaele Bruno, and me, and is inspired by the recent emergence and spread of the COVID-19 pandemic. It is well known that the outbreaks of new variants of concern can have severe health-economic-social consequences: the early detection of these events certainly help to contain the spread, or at least possibly mitigate the effects.

  A statistical tool based on funnel plots has been presented to monitor the regional reproduction numbers $R_t$'s of a country and identify the regions whose $R_t$'s behave anomalously with respect to the average national one, for instance, because a new variant has started spreading in there.

  There are some advantages for applying the proposed methodology. The first one is that the method is based on epidemiological data that is published daily, hence it is faster and requires a lower cost compared to massive genomic sequencing, which is used to identify variants of concern. Note that funnel plots detect anomalous regions, but not the causes of the anomaly: a specific further investigation is required, possibly through genomic sequencing, which can be concentrated in the abnormal regions, instead of being performed in the whole nation. Moreover, anomalies due to other causes, like for instance malfunctioning of the testing infrastructure, can be detected. Another advantage is that funnel plots are designed to reduce the number of false alarms due to the sample size, which is the number of infectious cases. Indeed, it is expected that if this is low, the variability of the $R_t$ is larger.

  As an additional result, plotting the temporal evolution of the $R_t$'s as a function of the number of infectious cases results in a clockwise spiral trend: indeed, when the $R_t$ is above 1 the trajectories go rightwards, because infectious cases tend to increase, while when the $R_t$ is below 1 the trajectories go leftwards when Rt is less than one, because the infectious cases tend to decrease.

  The proposed methodology has been applied to five case studies that have really occurred recently: the first emergence of the Delta variant in India in February 2021, the first emergence of the Omicron variant in

South Africa in November 2021, the initial spread of the Omicron variant in England in December 2021, the initial spread of the Omicron variant in Italy in December 2021, and a malfunctioning episode of a diagnostic infrastructure in England in September 2021, resulting in a large number of false positive cases reported. In all cases, an early warning is raised by applying the proposed tool, which anticipates the official detection that has actually occurred. This demonstrates the efficacy of the proposed approach in the early detection of the anomalies regarding the COVID-19 spread.

- **Franco Blanchini, Lorenzo Brunato, Carlo Drioli, Raffaele Pesenti, and Francesca Rosset. "Optimal trajectory generation in corridors via quadratic programming and its receding horizon implementation". In: (2022). Submitted**

  The second work [153] comes from the collaboration between Franco Blanchini, Lorenzo Brunato, Carlo Drioli, Raffaele Pesenti, and me. In this paper, the problem of generating a trajectory within a corridor formed by a sequence of convex contiguous regions is considered, under the condition that the union of each pair of consecutive regions is convex, too.

  A model of the trajectory has been formulated as a finite-dimensional discrete-time linear system. In particular, a specific node is to be determined within each region and the trajectory must pass through all these points. Each node is characterized by a state, which describes the position and velocity components of the moving point that reaches it. It is imposed that the time to move between two consecutive nodes is given and that in each of these intervals, the curve components have to be polynomials. Specifically, the components of the acceleration are forced to be first-order polynomials, whose coefficients are the decision variables.

  The trajectory must be the optimal one according to a given objective function to be minimized: the integral of the square of the acceleration; the integral of the square of the speed; the integral of the square of the derivative of the acceleration (jerk); the integral of the squared distance from the middle line; the sum of squared distances between consecutive nodes. All of these objectives can be expressed as quadratic integral cost functions of the state of the node points and the decision variables.

  Also, other than imposing that the curve must lie within the corridor, several other constraints can be imposed, too: continuity of the acceleration components, no retrograde movement of the moving point, minimum (maximum) speed along a direction, maximum angle with respect to a fixed direction, maximum (approximate) curvature. In any case, while these constraints must be fulfilled for all the time instants, instead of imposing infinitely many constraints, it is possible to express them as linear functions of the state of the node points and the decision variables.

  Then, the resulting formulation reduces to a linear-quadratic programming problem with a finite number of variables and constraints. An optimal trajectory can be easily computed even for large instances where a very long sequence of regions is to be considered. Moreover, the problem can be efficiently implemented using a receding horizon approach and, by introducing suitable conditions, recursive feasibility is ensured. This allows for online control, capable of taking into account possible modifications of the corridor over time; the support to unknown sequences of regions is also achieved.

  Applications of this technique include car trajectory generation in a track and ship trajectory navigation problems inside channels.